PROBABILISTIC APPROACHES TO PROTEIN-PROTEIN DOCKING

by

Irina Hashmi
A Dissertation
Submitted to the
Graduate Faculty
of
George Mason University
In Partial fulfillment of
The Requirements for the Degree
of
Doctor of Philosophy
Computer Science

Committee:

_____ Dr. Amarda Shehu, Dissertation Director

_____ Dr. Kenneth De Jong, Committee Member

_____ Dr. Huzefa Rangwala, Committee Member

_____ Dr. Daniel Barbará, Committee Member

_____ Dr. Nadine Kabbani, Committee Member

_____ Dr. Sanjeev Setia, Department Chair

_____ Dr. Kenneth S. Ball, Dean, Volgenau School of Engineering

Date: _____ Summer Semester 2015
George Mason University
Fairfax, VA

Probabilistic Approaches to Protein-protein Docking

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University

By

Irina Hashmi
Bachelor of Science
University of Dhaka, 2008

Director: Dr. Amarda Shehu, Associate Professor
Department of Computer Science

Summer Semester 2015
George Mason University
Fairfax, VA

Acknowledgments

First of all I would like to thank Almighty for making this journey easier and possible for me. The progression of this work could not possibly be carried out without the help of several people who, directly or indirectly, are responsible for the this work. I would like to thank my advisor Prof. Amarda Shehu for all her guidance, continuous support and help in every aspect of my research and graduate studies. She has taught me (and I am still learning) how to be thoughtful and well-organized in research and be a better writer. Her insightful and detailed comments about my work are always immensely helpful. I must thank all the members of the Shehu lab for their thoughtful discussion and feedback on this work. I would like to thank all the researchers within this community whose profound papers, reports and journals have helped cultivate my interest in computational biology. I am very much grateful to all my committee members. I would like to thank Dr. Kenneth A. De Jong for introducing me to the filed of evolutionary computation through his class. I would like to thank Dr. Huzefa Rangwala for introducing me to the field of data mining and machine learning through his class and projects. I would like to thank Dr. Barbará for his valuable advice and continuous feedback on my research of special protein class. I like to thank Dr. Nadine Kabbani for her collaboration and insight help on GPCR project. I would also like to thank Jayshree Sarma and Alastair Neil for all their quick help and support on running Argo and Hydra clusters.

Finally, I cannot thank enough my husband and my parents, my brother, sister-in-law and my in-laws for their unconditional love and support no matter what I do and how far I am. I owe special thanks to my uncle and specially my aunt for their love and care during my stay in USA. I would also like to thank all my family in Bangladesh and here in USA. I would like to thank my best friends for their continuous communication and suggestions on uncountable things. Thanks to you all!

Table of Contents

List of Tables

List of Figures

Abstract

PROBABILISTIC APPROACHES TO PROTEIN-PROTEIN DOCKING

Irina Hashmi, PhD

George Mason University, 2015

Dissertation Director: Dr. Amarda Shehu

Characterizing the three-dimensional structures of protein-protein assemblies, a problem known as protein-protein docking, is central to understanding the physical and structural bases of molecular interactions in cellular processes. Doing so can also provide useful insights in structure-function studies and the design of effective drugs. Despite significant contributions from wet-laboratory techniques, the number of high-resolution structures of protein assemblies characterized in the wet laboratory cover only a small fraction of possible interactions.

Research in dry laboratories is vibrant but challenged by the complexity of molecular interactions. Predominantly, methods based on stochastic optimization are employed to handle the size and complexity of the space of possible placements of units in an assembly. Despite significant work showing that knowledge of interaction interfaces can be valuable to guide docking methods, very few methods incorporate such information. Those that do are restricted to the setting of directly incorporating wet-lab macroscopic measurements, such as chemical shifts, which are hard to obtain on a variety of systems.

Moreover, currently no stochastic optimization methods integrate machine learning models in their search for functionally-relevant structures. The contribution of this thesis is the proposal of hybrid probabilistic approaches that integrate domain-specific insight into powerful stochastic optimization algorithms for the pairwise protein docking problem. Various sources of domain-specific insight are integrated and tested for how they guide a docking algorithm towards the true, native structure. Specifically, these sources consist of information stored in the sequences of evolutionary related proteins regarding the location of possible interaction interfaces, qualitative information provided from wet-laboratory experts, and information provided from machine learning models trained on known interaction interfaces. On the latter, several such models are considered and integrated in a powerful algorithm that approaches stochastic optimization under the umbrella of evolutionary computation. Our work shows that hybrid approaches such as those proposed in this thesis provide a good balance between computational efficiency and accuracy in protein-protein docking.

The work in this thesis incorporates powerful techniques and concepts from evolutionary computation, machine learning, and molecular biology. In addition to pointing towards several directions of promise in further improving pairwise docking, this thesis opens up several novel research directions, such as function-specific machine learning models, and the employment of evolutionary algorithms for the general, multimeric protein-protein docking problem where the number of units is arbitrary.

Chapter 1: Introduction

Proteins are the main workforce in cells, participating in almost all major biochemical processes that make up the cell machinery [11]. Proteins are comprised of amino acids, which link to one another in a serial fashion to form a polypeptide chain. These chains fold into unique three-dimensional (3d) structures under physiologic conditions [12]. What is typically referred to as a protein molecule may consist of one or more polypeptide chains. To make the distinction clear, in this thesis we will reserve the word protein for a single polypeptide chain and will instead refer to the assembly of several, folded chains as a protein-protein assembly. Proteins are indispensable molecules in the cell, serving as principal catalytic agents, key structural elements, transporters, signal transmitters, and building blocks of molecular machines. Proteins carry their functions in the cell through their folded structures, essentially employing their structures to form sticky interactions and form stable or passing assemblies with other molecules.

In this thesis we focus on a specific class of protein-participating assemblies, those that consist of only polypeptide chains. We do so for several reasons. First, it has been estimated that among all structures deposited in Protein Data Bank (PDB) [13], 52% are composed of two or more chains. Despite the ubiquity of protein-protein assemblies in the cell, the 3d structures of only a small percentage of such assemblies are currently solved [14]. High-throughput techniques, such as yeast two hybrid (Y2H) [15], co-immunoprecipitation [16], and mass spectrometry [17], provide evidence of putative interactions but offer little insight into the structure or physical basis of such interactions. Characterizing 3d structures of protein-protein assemblies remains challenging in the wet laboratory. Techniques, such as Nuclear Magnetic Resonance (NMR), X-ray crystallography, and cryo-electron microscopy (cryoEM) remain laborious, limited by the number of participating units in an assembly, or

restricted to too low resolutions to be useful [18]. Challenges in the wet laboratory provide opportunities for dry-laboratory investigations to make contributions, particularly considering that knowledge of 3d structures of protein-protein assemblies can provide valuable information for computer-aided drug design studies [19].

Computational methods can in principle assist wet-laboratory investigations, but they also face challenges. By now, protein-protein docking is recognized to be a computationally hard problem [20] for two major reasons: (i) There is great diversity among types of protein-protein interactions. Even proteins central to a known biological process may interact with other proteins not involved in that particular process. There seem to be no universal rules to predicting functional interactions. (ii) The number of possible, different placements of units relative to one another in an assembly is too large to be amenable for enumeration; that is, the configuration search is vast.

Computationally, finding the naturally-occurring, native structure of a protein-protein assembly involves searching in a high-dimensional search space. For instance, when considering an assembly of k units, the total dimension of the problem is $\sum_{i=1}^{i=k} N_i + 6^{k-1}$, where each N_i is the number of parameters to represent the unbound unit i, and 6 is the number of parameters to represent the spatial arrangement of one unit on top of another (3 rotation followed by 3 translation). Considering parameters N_i for each unbound unit allows it to possibly undergo full configurational changes upon docking. Though not universal this has been observed in protein-based assemblies, and the model that is able to account for flexible docking is known as the induced-fit model. An illustration is provided in Figure 1.1.

Accounting for full flexibility in silico is currently impractical due to the resulting high-dimensionality of the configuration search space. Fortunately, in a large number of interactions, flexibility can be modeled after docking [21, 22]. This alternative model, known as lock-and-key model, is illustrated in Figure 1.2. In response, many computational methods are justified to pursue rigid-body docking, where the units undergo no/very small configurational changes upon docking. This effectively removes the parameter N_i from the solution space leaving only the spatial parameters of 6.

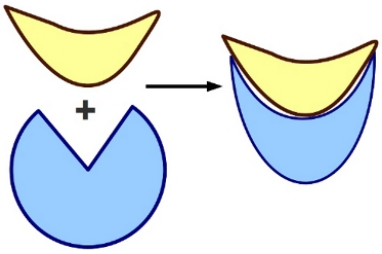


Figure 1.1: Induced-fit model of flexible docking. The shapes of the molecules may change upon docking.

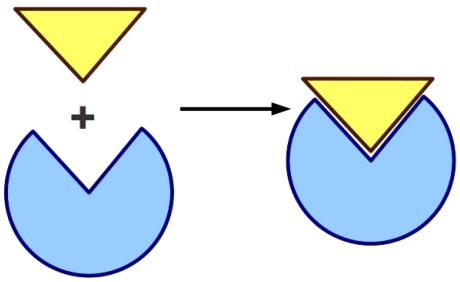


Figure 1.2: Lock-key model of rigid body docking. The shape of the molecules remain largely unchanged upon docking.

Even after simplifying protein-protein docking to rigid-body docking, the problem remains challenging. The reason is broadly two-fold. First, as the number of units grows, the problem becomes intractable because of the combinatorial complexity of the solution space [23]. Therefore, work on multimeric docking, where $k > 2$, is very limited. Most methods address the problem of dimeric docking or pairwise docking, where $k = 2$. In addition to the high-dimensionality of the space, the computed interaction energy associated with bound configurations often leads the search towards non-native interactions [24]. This is partly due to the fact that most of the energy functions are an approximation model of the true one. Moreover, there is inherent bias in them because they are trained to fit with particular sets of data. Even though the docked structure that one expects to find in the cell has lowest energy among alternative docked configurations, finding it in a high-dimensional, multi-modal search space is challenging. The difficulty for dimeric docking can be summarized by the statistics of the community-wide experiment Critical Assessment of PRedicted Interactions (CAPRI) that only $30-58\%$ of the interaction interface are predicted correctly for given target proteins [25].

Given outstanding challenges in rigid-body pairwise protein-protein docking, this thesis contributes several novel algorithms to address this problem. In particular, this thesis takes a deliberate approach and contributes several integrative algorithms that elucidate promising, novel directions for protein-protein docking.

The current approaches to pairwise protein-protein docking can be categorized into systematic versus stochastic. Systematic approaches that discretize the configuration space in protein-protein docking are limited in the sizes of configuration spaces they can address and the accuracy of docked configurations they report. This category includes the more recent geometry-driven methods that indirectly discretize configuration space by pre-processing molecular surfaces into regions of interest for docking [23, 26]. Geometry-driven methods are computationally efficient but largely recognized as less accurate than a second category of methods that address protein-protein docking under the umbrella of stochastic optimization [19]. These methods have higher exploration capabilities. The core functional unit in

them consists of a biased random walk in the configuration space of possible, docked configurations, guided by an objective function tallying up interatomic interactions among units in interaction energy scores. These methods are often referred to as energy-driven, and state-of-the-art ones in this category include ClusPro [27], RosettaDock [28], SKE-DOCK [29], GRAMM-X [30], PIPER [31], and others [32].

Energy-driven methods are typically more computationally demanding than geometry-driven ones. Sufficient time needs to be allocated to explore the breadth of the dimeric configuration space in order not to miss configurations near the native structure. In addition, optimizations of molecular energy functions are expensive mainly due to the pairwise interaction terms in these functions. In addition to the computational demands, energy-driven methods may miss the native structure entirely. If the exploration does not draw a configuration sufficiently similar to the native structure, energy optimization, which is essentially a local improvement technique, is not guaranteed to make sufficient structural improvements and reach the native structure. In many cases, the optimization itself may modify the configuration away from the native structure. Most energy functions linearly combine different weighted energy terms, where the weights are optimized for a particular dataset. For these reasons, even energy-driven methods can lead the search towards non-native structures [24, 33].

One of the main lessons in computational protein-protein docking is that optimization of an energy function is at best a local technique to improve a configuration that is in the vicinity of the native structure in the configuration space. The consensus in protein-protein docking is that, when provided enough constraints to focus their search around the true interaction interfaces, energy-driven methods can be effective. The difficulty remains on where to obtain such constraints and how to efficiently incorporate them in energy-driven methods. There are several sources of constraints for docking algorithms. The first source of constraints can be from domain experts. Haddock [34] follows this approach and restricts sampling of docked configurations to those that agree with NMR chemical shift data. However, such data are not readily available on all systems. Having access to such

data is restricted to protein assemblies studied extensively in the wet laboratory.

One of the contributions of this thesis is the integration of qualitative knowledge from wet-laboratory experts on the possible location on the true interaction interface to restrict sampling in an energy-driven method for protein-protein docking. Chapter 4 details our work in this direction.

In the absence of domain-expert data, the focus turns to extracting information on the location of interaction interfaces from available protein data, which include sequence and structure data. Several studies have shown that sequence analysis of evolutionarily-related proteins can yield valuable information on the location of interaction interfaces. In particular, work in [2] is one of the first to operationalize such findings by devising an objective function rewarding evolutionary conservation and low interaction energies to guide energy-driven docking methods.

Another contribution of this thesis is the direct integration of evolutionary conservation in the search protocol itself. In Chapter 3 such information is integrated in geometry-based methods, and in Chapter 5 geometry- and energy-driven methods are combined in an integrative, evolutionary algorithm that approaches stochastic optimization under the umbrella of evolutionary computation.

While analysis of sequence and structure data can reveal important characteristics or features of true interaction interfaces, such features by themselves may not be sufficient to discriminate between native and non-native interaction interfaces. A growing direction of research in protein docking consists of machine learning methods that forego the traditional setting of predicting a native structure but instead focus on learning what makes true native interaction interfaces [35–40]. However, to our knowledge, machine learning models are not integrated in search algorithms that compute docked protein configurations.

A central contribution of this thesis is the integration of machine learning models in stochastic optimization methods to address shortcomings of energy functions. Chapter 6 details how a trained model is integrated in an evolutionary algorithm that combines ingredients from geometry- and energy-driven approaches.

The direction of integrating machine learning in stochastic optimization methods for protein-protein docking is novel, exciting, and results in what we refer to as a hybrid algorithm that opens up a new line of research in direction. In Chapter 7 we investigate a new direction in function-specific models and put forth an important criterion for machine learning models to be successful in guiding stochastic optimization methods. In Chapter 8 we estimate the ability of evolutionary algorithms to address docking beyond the dimeric setting, opening the way for further research in this direction.

The rest of the thesis is organized as follows: Chapter 2 provides some background on protein-protein docking and summarizes current methods. Chapters 3-8 describe methods and results obtained from the different lines of investigations summarized above. Chapter 9 concludes this thesis with an overview and future prospects.

Chapter 2: Background and Related Work

This chapter provides some useful concepts and related work on protein-protein docking. Section 2.1.1 summarizes rigid-body protein docking. Section 2.2 summarizes related work. Evaluation measures are defined in Section 2.3. Section 2.4 describes the systems in the testing dataset we employ for gaging the performance of the algorithms described in this thesis in a comparative setting against other state-of-the art methods. Section 2.5 details the computational resources employed for the experiments reported in this thesis.

2.1 Background

2.1.1 Docking as a Search over Rigid-body Transformation

In rigid-body docking, one unit, chosen arbitrarily, say A, remains static and is the reference unit. A rigid-body, SE(3), transformation is applied on the other, moving unit B. The transformation can be defined as $T =< r_x, r_y, r_z, t_x, t_y, t_z >$ where r_x, r_y and r_z are the rotation and t_x, t_y and t_z are the translation along x, y and z axis respectively. Applying T onto B yields a new placement for B in 3d space.

In geometry-driven methods, a new docked configuration is obtained by computing transformations T that align triplets of points on the molecular surface of the moving unit with triplets of points on the molecular surface of the reference unit. Since triplets define coordinate frames, let us refer to an arbitrary pair as tr_A and tr_B, respectively. The corresponding transformation that aligns tr_B to tr_A is $T = tr_A \times tr_B^{-1}$, where tr_B^{-1} is the inverse of tr_B.

In geometry-driven methods, pairs tr_A and tr_B are selected to be geometrically-complementary. We describe how such pairs are obtained by first introducing the concept of

molecular surface, critical point, triangle, and geometric complementary.

2.1.2 Interaction Interface and Molecular Surface

The Molecular Surface (MS) or Connolly surface, detailed in [3], is a dense dot representation of the solvent excluded surface of a molecule (illustrated in Figure 2.1). Each dot corresponds to one of the three principal shapes: convex, concave or saddle. Each dot is represented through its corresponding atom number(s), 3D coordinates, a unit normal vector pointing outward of the surface, and the total area of the surface.

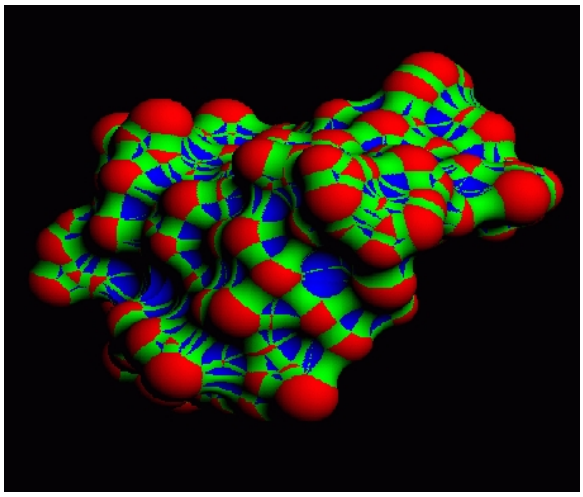


Figure 2.1: Connolly molecular surface representation. Red represents the convex surface, green is the saddle-shaped surface and blue stands for concave surface. ©Michael Connolly [3]

A second numerical method provides a sparser representation known as the Shuo representation [4]. The method processes the Connolly surface representation and extracts critical points. Each critical point represents a local maximum or minimum of a Connolly face. Three types of points obtained this way are nicknamed as “caps”, “pits” and “belts” and correspond to Connolly’s convex, concave and saddle faces, respectively. The Shuo

representation is less dense. It is thus memory efficient, while at the same time sufficient to cover the important locations of a molecular surface.

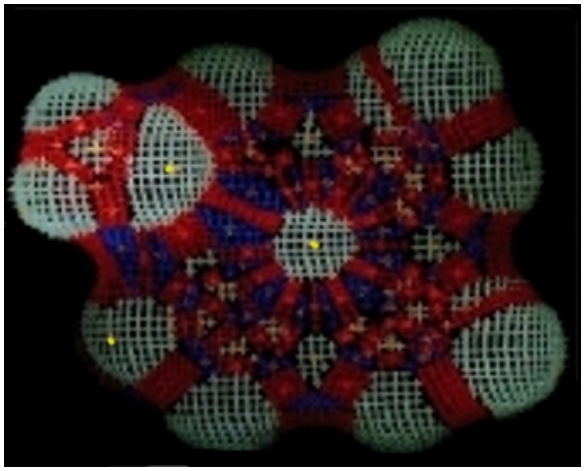


Figure 2.2: Sparse critical point representation of the Connolly surface. The critical points are shown as yellow dots. ©Shuo Lin [4]

2.2 Related Work

Methods for protein-protein docking can largely be divided into two categories, geometry-driven and energy-driven. We summarize each category below.

2.2.1 Geometry-driven Approach

Geometry-driven methods [26, 41] use geometric information about the molecular surface as described above. These methods are systematic. They organize the critical points into concave or convex triplets (or triangles) to match geometrically-complementary regions of molecular surfaces. There are two main subcategories of such methods.

Fast Fourier Transform (FFT)-based Search

Traditional brute force matching methods try to exhaustively match every pair of points of one unit with every pair of points from another unit. The complexity is very high and in order of n^5 [42], where n is the number of points in a unit. FFT approaches perform the search over the full 3D translational space with one FFT calculation and thus reduce the complexity to the order of $O(n^3 log(n^3))$. State-of-the-art methods based on FFT as a first stage of docking include ZDOCK [43], F2Dock [44], PIPER [31], and Hex [45].

Geometric Hashing

An efficient computer vision based technique, Geometric Hashing [42], has been employed in rigid-body pairwise docking [23, 26, 41]. Geometric Hashing reduces the time complexity of the previous approaches significantly. This technique uses a transform-invariant representation of the molecular surface which allows a hash-based matching between two units. The algorithm consists of two stages: a preprocessing stage and a matching stage. During the preprocessing stage, some features of critical points of the reference unit are extracted and hashed into a table. The matching stage similarly extracts the same features from the moving unit and then matches them to those stored for the reference unit in the hash table. The preprocessing stage takes $O(m^3)$, where m is the number of points in the reference unit, and matching stage takes $O(n^3)$ time, where n is the number of points in the moving unit. The result is an overall time complexity of the order $O(n^3)$.

2.2.2 Energy-driven Approach

Geometry-based methods are less accurate due to the inherent discretization of the search space. Energy-based methods, such as ClusPro [27], RosettaDock [28], SKE-DOCK [29], GRAMM-X [30], PIPER [31], and others [32], operate over a continuous search space. They are computationally demanding. Traditional energy-based methods build over Monte Carlo optimization (MC), simulated annealing (SA), or evolutionary computation (EA). Each of

these sub-categories are discussed below:

Monte Carlo (MC)

MC-based methods, such as [28,46], work as follows: the search starts from a random docked configuration and a perturbation operator is applied on the current configuration to obtain a new one. The result of the operator is accepted based on the Metropolis criterion[47], which aims to drive the MC search towards a local minimum of the energy surface while occasionally allowing for small energetic increases so as to cross energy barriers.

Simulated Annealing (SA)

In simulated annealing (SA), the Metropolis criterion is gradually varied from relaxed to conservative in order to gradually settle the search onto local minima. In the beginning, higher energetic increases are allowed. As the search progresses, the algorithm becomes more conservative, focusing more on exploitation of local minima. SA has been widely used for protein-protein docking [46, 48].

Evolutionary Algorithms (EA)

EAs have been demonstrated effective for docking [2, 49], and some are widely used in protein-ligand binding for drug-design [48]. In EAs, configurations are treated as individuals of a population that evolves over a number of generations. Reproductive operators are employed to obtain offspring from individuals in a population. Rules based on individual fitness determine survival in a population and result in the EA settling to local minima of the configuration space. Many docking methods employ Genetic Algorithms (GA), which are a specific class of EAs. Canonical GA provides higher exploration capability through multi-parent reproductive operators. In Lamarckian GA (LGA) [48], each individual inherits the good characteristics from its parent through a local search. The sustainable GA (SGA) [50] introduces an age-based scheme to delay premature convergence. Some methods combine traditional optimization techniques such as MC and SA with GA [2].

2.2.3 Hybrid Approaches

Both geometry-driven and energy-driven methods have their unique set of difficulties in finding the true native interaction interface. A third group of methods combine search and a-priori knowledge, whether obtained from experimental data or from learning, to improve accuracy in protein-protein docking. Many methods currently exist that analyze the known native structures of protein-based assemblies for finding the characteristic features [51] important for building an interaction. For instance, native interfaces are generally found to contain residues of high evolutionary conservation [52], [53]. Currently, two docking methods exploit knowledge of interaction interfaces in guiding the search towards native-like configurations [2, 34]. While the method in [34] integrates data obtained from NMR, the method in [2] scores configurations based evolutionary conservation prior to conducting energy optimizations.

2.2.4 Approaches for Multimeric Protein Docking

Most of the work discussed so far address the problem of dimeric docking. This section briefly discusses two methods proposed for multimeric docking, CombDock [23] and Multi-LZerD [54].

CombDock is a geometry-driven docking algorithm which relies on the geometric hashing technique. At each stage, the method hierarchically adds one unit to the current configuration and greedily selects top ones based on shape complimentarity to control the combinatorial explosion in the number of possible configurations. Multi-LZerD is a GA-based method. It employs an expensive physics-based scoring scheme as the fitness function. Though Multi-LZerD is more detailed and shows higher accuracy than CombDock while dealing with complex protein cases, its computational demand is very high.

2.3 Evaluation Measures for the Performance of Docking Methods

To evaluate the quality of a generated configuration, one of the widely-used measurements is least-root-mean-square-deviation (lRMSD). lRMSD measures the average atomic displacement between two configurations. Given the 3D atomic coordinates of two configurations x and y of N points each, the RMSD is calculated as follows:

$$\sqrt{\frac{1}{N}\sum_{l=1}^{N}\|x_l - y_l\|^2} \tag{2.1}$$

An optimum rotation matrix is calculated which minimizes the RMSD distance between two structures and hence the name is least-RMSD. The most commonly used unit for RMSD is Angstrom (Å) which is equal to 10^{-10}m. A configuration is compared through lRMSD to the known native structure. An lRMSD measure $< 2\mathring{A}$ is considered to be near-native and a distance between $2 - 5\mathring{A}$ is considered to be good.

2.4 Target Systems

In this thesis we employ a carefully constructed testing dataset of 15 protein systems. These systems have native structures deposited in the PDB. They are selected because they vary in size, functional classification, have been used by other different docking methods, and some are actual CAPRI targets. Table 2.1 lists these systems.

2.5 Computational Resources Employed for Production Runs

We have used several computing clusters provided by George Mason University for the production runs analyzed and reported in this thesis.

Table 2.1: Systems used in our testing dataset are listed here. The PDB ID of the native structure of each of these systems is shown in the second column. The chains considered are shown in parentheses. The number of atoms in each chain is shown in the third column. The fourth column lists the known functional classification. An asterik marks the CAPRI targets.

Nr.	PDB ID (Chains)	Size(Nr. of Atoms)	Functional classification
1	1C1Y (A,B)	1376, 658	Signaling Protein
2	1DS6 (A,B)	1413, 1426	Signaling Protein
3	1TX4 (A,B)	1579, 1378	Complex(gtpase Activation/proto Oncogene)
4	1WWW (W,Y)	862, 782	Nerve Growth Factor/trka Complex
5	1FLT (V,Y)	770, 758	Complex (growth Factor/transferase)
6	1IKN (C,D)	916, 1589	Transcription Factor
7	1VCB (A,B)	755, 692	Transcription
8	1VCB (B,C)	692, 1154	Transcription
9	1OHZ* (A,B)	1027, 416	Cell Adhesion
10	1ZHI* (A,B)	1597, 1036	Transcription/replication
11	2HQS* (A,C)	3127, 856	Transport Protein/lipoprotein
12	1QAV (A,B)	663, 840	Membrane Protein/oxidoreductase
13	1G4Y (B,R)	682, 1156	Signaling Protein
14	1CSE (E,I)	1920, 522	Complex(serine Proteinase Inhibitor)
15	1G4U (R,S)	1398, 2790	Signaling Protein

2.5.1 Argo

Argo is a research computing cluster provided by the Office of Research Computing at George Mason University (`http://orc.gmu.edu`). The Cluster comprises of 41 nodes with a total of 688 cores and 3.5 TB of RAM. 37 nodes are reserved for computing with a specification of Dual 8 core with 64 GB of RAM. Rest 4 nodes are reserved for head and storage. All the compute nodes are Intel Xeon E5-2670 CPU with 2.6GHz base processing speed.

2.5.2 Hydra

The Hydra Cluster consists of sixty dual- and quad-core processors of Linux machines. These are connected to a server at `hydra.cs.gmu.edu`. These machines are intended to support research experiments for the department of computer science.

Chapter 3: evoDock: Integrating Evolutionary Conservation in a Geometry-driven Approach

Our research on protein docking begins by integrating evolutionary conservation in a geometry-driven, exhaustive method. The integration of evolutionary conservation is based on findings that molecular surface regions participating in molecular interactions are under higher evolutionary pressure than other surface regions to maintain their functional integrity [52]. Thus, amino acids with higher evolutionary conservation are more likely to be part of an interaction interface than others. In this chapter we first analyze the rationale behind this statement in Section 3.1 and then detail the proposed evoDock algorithm in Section 3.2. Section 3.3 provides analysis of the performance of evoDock.

3.1 Evolutionary Conservation and True Interaction Interfaces

We analyze here the extent to which true interaction interfaces are evolutionary conserved compared to the rest of the molecular surface.

The Joint Evolutionary Trace (JET) method [53] is employed to associate with each amino acid a conservation score ranging from 0.0 (least conserved) to 1.0 (most conserved). JET analyzes a multiple sequence alignment of evolutionary-related proteins to determine the conservation of each amino acid in a protein sequence of interest. JET obtains a set of sequences that are homologous to the query sequence using PSI-BLAST [55]. From this set of sequences JET recovers another subset of sequences that uniformly represents a broad range of sequence similarity. These sequences are then employed to construct small distance trees that are analyzed to determine the importance of the residues in the

query sequence. The residues are then ranked and clustered together to identify the most conserved ones. Due to the method employing Gibbs sampling to reduce errors in multiple sequence alignment, slightly different results can be obtained from each run. Hence, we have employed its iterative version, iJET, which essentially runs JET for 50 times and provides averaged scores over the runs for each amino acid.

The interaction interface is defined as a list of pairs of atoms. We rely on the concept of contact. We determine that two atoms are in contact, and so part of the interaction interface, if their Euclidean distance is less than 5 Å. This distance threshold is commonly employed in other works [34, 43].

We measure two different ratios, $R_{interface}$ and R_{rest} . $R_{interface}$ is the ratio between the number of conserved critical points on the known interaction interface and the total number of conserved critical points on the molecular surface. R_{rest} is the same ratio measure for rest of the surface, which can be computed as $1 - R_{interface}$. Three different thresholds of conservation are employed to determine whether a critical point is conserved or not.

We have calculated $R_{rest} - R_{interface}$ and plotted the results in a bar diagram format in Figure 3.1. The results are presented for 3 different conservation threshold values $\{0.25, 0.50, 0.75\}$ in different colors. Figure 3.1 shows that on 10 of the systems conserved critical points are concentrated more on the interface than the rest of the surface. Among these 10 systems, on 7 of the systems the bar with the 0.5 threshold is the most negative. On the rest of the proteins, the difference is a low positive number. This analysis shows that focusing on evolutionary conservation score along with geometric information can be a good predictor for defining interaction interface. Based on this analysis, we have employed a conservation threshold score of $conservation_{th} = 0.5$ for all our experiments.

3.2 Methods

This section described a novel algorithm that aligns geometrically-complementary *and evolutionary conserved* regions of molecular surfaces for rigid-body pairwise protein-protein

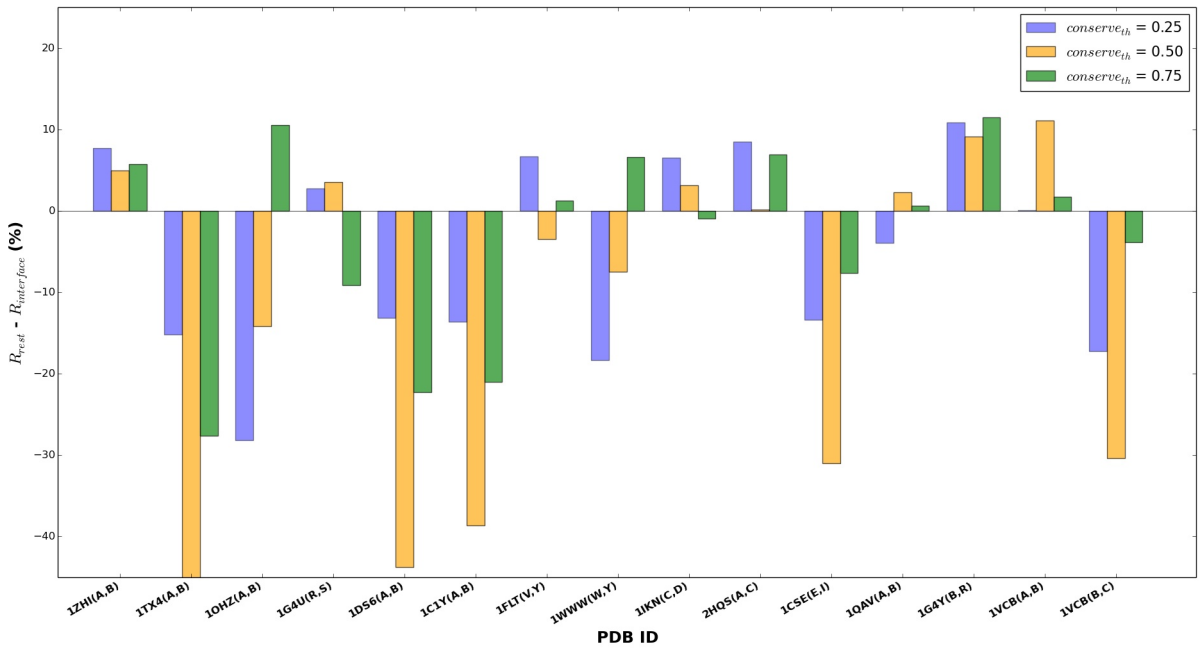


Figure 3.1: Bar diagram showing the difference between $R_{interface}$ and R_{rest} on 15 dimers on three different conservation threshold denoted as $conserve_{th}$. A negative difference indicates that the interacting interface is more conserved than the rest of the surface.

docking.

3.2.1 From Conserved Amino Acids to Active Critical Points

As described above, iJET is employed to obtain a conservation score with each amino acid. The score of the amino acid closest to a critical point is then transferred to the point itself. A critical point with evolutionary score >= $conservation_{th}$ is deemed as "active" as opposed to "passive". The definition of active/passive is inspired from [34].

3.2.2 From Active Critical Points to Active Triangles

As described in Chapter 2, two triangles (triplets of critical points), one from each unit, can be employed to define a rigid-body transformation that aligns one triangle on top of the other. Here we restrict our focus to active triangles, defined as follows: an active critical point with conservation score >= $conservation_{th}$ is considered first from the molecular surface. Let us refer this as p_1. Two more critical points p_2 and p_3 are obtained, which lie within a neighborhood distance of p_1. p_2 and p_3 are not necessarily active critical points. The selection of these points satisfies shape, distance and angle constraints. p_2 and p_3 must reside on the same Connolly face as $p1$. They also must be no closer than 2Åand no further than 5Åfrom p_1. The minimum distance of 2Åmakes sure that no two points lie on the same atom, since the van der Waals radius of a C_α atom is 1.94 Å. The maximum distance of 5Åensures that the triangle cover a larger area. An angle constraint ensures that the points are not collinear. Values for the angle and distance constraints are taken from [56].

3.2.3 From Active Triangles to Rigid-body Transformation

Once two active, geometrically-complementary triangles are obtained, one from each unit, the rigid-body transformation aligning that of the moving unit onto that of the reference unit is trivially computed as described in Chapter 2.

A simple hashing scheme is used to avoid extracting duplicate active triangles (hence,

the same transformation). The hashing scheme is designed as follows: triangle vertices are ordered lexicographically so that no two triangles share the first vertex in the ordering. The hashing key employs the center of mass of a triangle to ensure uniqueness.

The evoDock algorithm consists of repeatedly applying n independent transformations to obtain n dimeric configurations.

3.3 Applications of evoDock

3.3.1 Experimental Setup

evoDock is implemented in C/C++, and the experiments are carried out on Argo research computing cluster. Results are obtained over 5 independent runs of evoDock to encounter any variance in the results. At each run the number of configurations generated vary from $100,000$ to $800,000$ for each system depending on the size of the system.

3.3.2 Comparison of evoDock to Other Methods

The methods that are considered for comparison are BUDDA [1] and the method in [2]. BUDDA is a complete geometry-driven method that builds upon the concept of Geometric Hashing as described in section 2.2.1. It scores each configuration based on shape-complementarity. At the very last stage, BUDDA applies a hydrophobic filter to extract top-scoring configuration. On the other hand, the method in [2] is an energy-driven one that employs evolutionary conservation score in addition to the energy function to score sampled configurations. Though the method is not available as an executable or a web server, we report here its published performance on select systems. The second and third columns shows the lowest lRMSD to the known native structure obtained by BUDDA for its top-scoring configuration and reported by the method in[2]. The last three columns summarize the performance of evoDock on its 5 independent runs in terms of the lowest lRMSD over configurations in a run from the native structure. Column 4 reports the lowest (η) lRMSD over all 5 runs; column 5 reports the average (μ) over 5 runs of the lowest

Table 3.1: Lowest lRMSDs by evoDock are compared to those published in [1] and [2]. NA indicates data is not available. The results of evoDock are reported over 5 runs and shown in terms of lowest (η), average (μ) and variance (σ^2) lRMSD. evoDock μ lRMSDs are shown in bold if better than at least one of the other methods.

PDB ID (Chains)	BUDDA [1] (Å)	Kanamori [2] (Å)	evoDock η (Å)	evoDock μ(Å)	evoDock σ^2 ($Å^2$)
1C1Y (A, B)	1.2	NA	0.7	**0.9**	0.07
1DS6 (A, B)	1.2	NA	2.1	2.3	0.01
1TX4 (A, B)	1.4	NA	1.4	1.4	0.001
1WWW (W, Y)	11.4	NA	1.2	**1.5**	0.02
1FLT (V, Y)	1.5	NA	1.9	2.0	0.14
1IKN (C, D)	2.0	NA	1.0	**1.2**	0.02
1VCB (A, B)	0.7	NA	0.8	1.4	0.11
1VCB (B, C)	1.3	NA	0.5	**1.1**	0.21
1OHZ (A, B)	1.8	0.66	0.7	**1.0**	0.06
1ZHI (A, B)	25.3	3.4	1.2	**1.8**	0.22
2HQS (A, C)	29.1	2.55	1.9	**2.5**	0.12
1QAV (A, B)	1.4	NA	0.4	**0.5**	0.001
1G4Y (B, R)	0.8	NA	1.3	1.4	0.02
1CSE (E, I)	0.7	NA	0.9	1.1	0.05
1G4U (R, S)	1.0	NA	0.9	1.2	0.08

lRMSD in each run, and column 6 reports the variance (σ^2) over the 5 runs of the lowest lRMSD in each run.

On most systems, evoDock is able to obtain near-native configurations and reach comparable low lRMSDs to BUDDA and the method in [2]. On 8/15 of the systems, evoDock performs better than one or both of the other methods. This result indicates that the incorporation of evolutionary conservation constraints search to regions of molecular surfaces relevant for docking.

Chapter 4: Incorporating Information from Domain Experts: A Proposed Protocol for the Dimerization of GPCR

In this chapter we show a specific case study on how to incorporate constraints obtained from domain experts in a simple protocol. For this work we have chosen G protein-coupled receptors (GPCRs). GPCRs are responsible for different neurophychotic disease and possibly affect the drug-efficacy. GPCRs are the most common targets in modern pharmaceuticals. Wet-lab experiments suggests that GPCRs undergo assembly [57]. However, structural models are difficult to obtain.

We investigate the suspected dimerization of a specific GPCR, the Dopamine D2 receptor (D2R). The dimerization of D2R has been suggested in the wet-laboratory and possibly pay a role in the efficacy of anti-psychotic drugs [7], [8]. Figure 4.1 shows the best predicted model for the D2R-D2R dimer by a thorough list of methods. Juxtaposition of these models shows that there is no consensus among them on even the overall structural characteristics of the dimer, which is known by wet-lab studies to have fewer options due to the presence of the membrane.

4.1 Structural Characteristics of GPCRs

GPCRs are highly similar in their structural characteristics and share common motifs for binding with other molecules [61]. GPCRs are more than 400 amino acids long. All GPCRs are composed of 7 transmembrane regions, referred to as $TM1-7$ connected by 3 intracellular and 3 extracellular loops shown in figure 4.2(a)). The TMs are arranges in a way so that they form a characteristic hollow cylinder inside the cellular membrane. Many recent wet-laboratory techniques point out the involvement of these regions to form dimeric

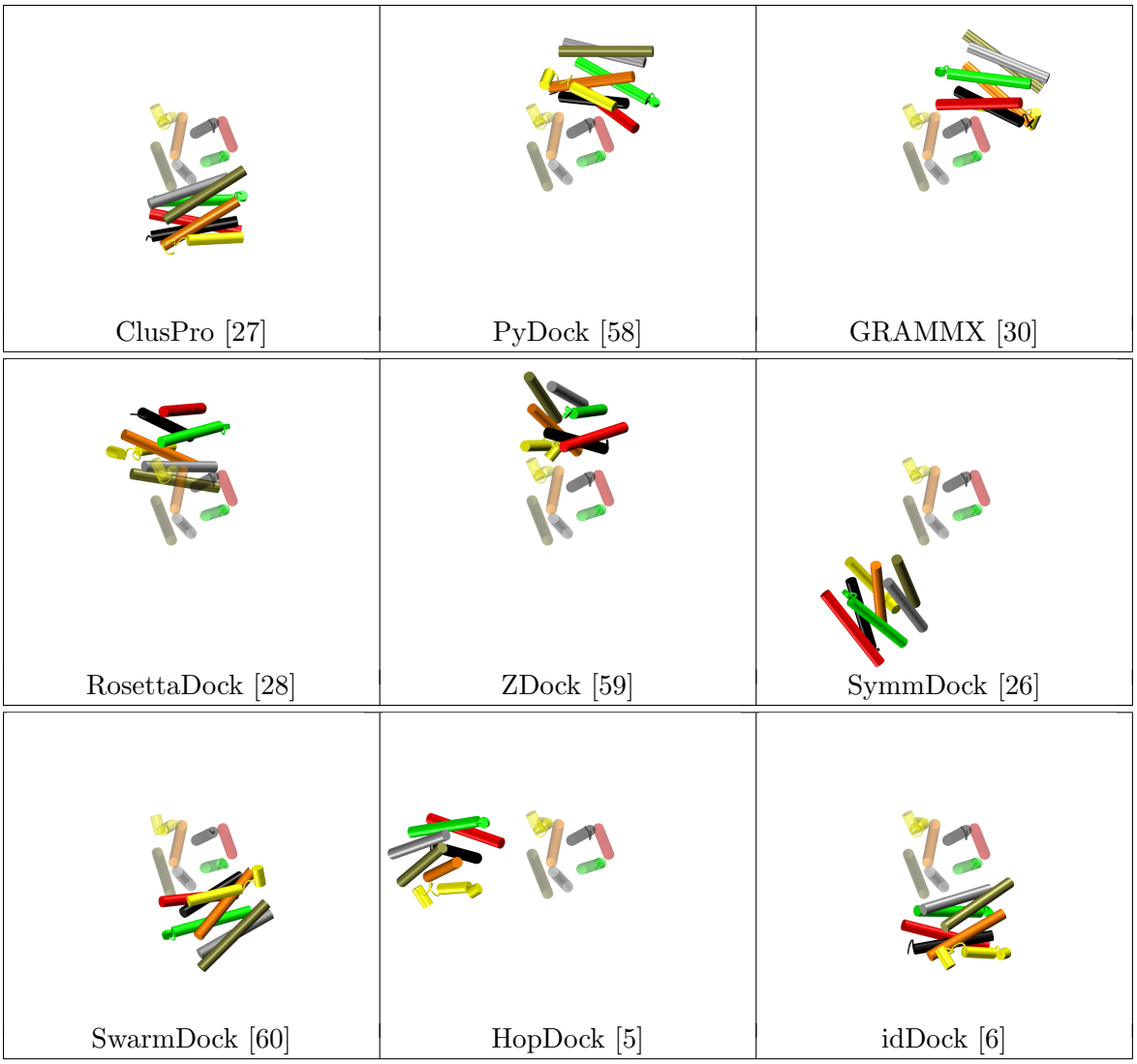


Figure 4.1: Several top methods, available as web servers, in pairwise docking (including our recent HopdDock [5] and idDock [6] methods) have been applied to obtain best models for a D2R-D2R assembly indicated to form in the living cell by wet-lab studies [7, 8]. The transmembrane regions (TMs) are color coded in each of the two chains (TM1 red, TM2 dark grey, TM3 orange, TM4 yellow, TM5 tan, TM6 light grey, TM7 green.) To show that there is no consensus among these models, all models are superimposed via least Root-Mean-Squared-Deviation (lRMSD) on unit/chain A, which is used as the base/reference unit. This unit is drawn in transparent and appears at the same position and orientation on all the subfigures. Due to the differences among the models, unit B, drawn in opaque, occupies various placements in space, clearly showing the lack of consensus among the models. Not only does the interaction interface not seem to involve TM 4, in contrast to wet-lab evidence [8], but in some models the actual contact interface is between the extra-cellular regions not drawn here in the interest of visibility.

and oligomeric complexes [62–65]. We propose a general docking protocol that is capable of taking into account the experimental knowledge of potential interaction interfaces, some environmental constraints that these receptors operate on, and energetic constraints to compute stable dimers of GPCRs.

4.2 Methods

4.2.1 Decoy Sampling

The method samples dimeric configurations. It is geometry-driven, since geometrically-complementary regions are employed to dock between two units. The novel component in it is to exploit the experimental knowledge to guide sampling. In essence, the method samples triangles from putative regions and computes rigid-body transformation that align such regions, resulting in dimeric configurations. Since a GPCR unit is large, we have also employed an axis representation which runs from the top to the bottom of the cylinder formed by the TM regions and hence rapidly determines whether a generated configuration complies with the environmental constraints that these receptors operate on. Once these test has been passed, another filter is applied to account for any big steric clashes between any two pairs of atoms. Once a generated configuration passes both filters, it is added to a growing trajectory of credible dimers. This protocol is summarized in pseudocode in algorithm 1. We now relate each of the main steps and then conclude with the various analyses of the configuration ensemble to select credible dimeric models.

Decoy Sampling through Rigid-body Transformation

Let us consider two GPCR units, A and B that are subject for docking. As before we randomly consider A as base unit and B to be the moving unit and let us denote their cartesian coordinates as C_A and C_B. As part of preprocessing, first we generate set of critical points CP_A and CP_B containing desired interaction information from both the

Algo. 1 GPCR-GPCR Docking

Input :
3D Cartesian coordinates of base unit C_A and moving unit C_B
List of residues in desired interaction interface
Maximum relative orientation angle Θ
Maximum LJ potential value $E_{\max}$
Target number of models N

Output: Ensemble of dimeric configurations $\Omega = \{C_{AB}\}$

Preprocessing:
Generate Connolly Surface representation MS_A for C_A and MS_B for C_B
Sample Critical Point representation CP_A and CP_B from regions of MS_A and MS_B containing desired interface
Generate Triangular representation Δ_A from CP_A and Δ_B from CP_B
Define the axis vector v_A for C_A and v_B for C_B

while $|\Omega| \leq N$ **do**
    Sample $\delta_A \in \Delta_A$ and $\delta_B \in \Delta_B$ ▷ δ_A and δ_B are geometrically-complementary triangles
    Compute $T \in \mathrm{SE}(3)$ that superimposes δ_B on δ_A ▷ T: rigid-body transformation
    $v_B^{'} \leftarrow T(v_B)$ ▷ Apply T on v_B to move only axis vector
    $\theta \leftarrow \langle(v_A, v_B^{'})$ ▷ Compute angle between axis vectors
    if $\theta \in [0, \Theta]$ **then**
        $C_{AB} \leftarrow T(C_B)$ ▷ Apply T on C_B to obtain a dimeric configuration C_{AB}
        Compute atoms in contact in C_{AB} and measure LJ potential e_{AB} over them
        if $e_{AB} \leq E_{\max}$ **then** ▷ C_{AB} meets both geometric and energetic constraints
            $\Omega \leftarrow \Omega \cup C_{AB}$ ▷ Add new configuration to Ω
        end if
    end if
end while

molecular surface MS_A and MS_B from units C_A and C_B. Then we generate sets of triangles ΔA and ΔB from these desired lists of critical points. Sampling critical points from potential interaction interfaces allows incorporating experimental knowledge in docking. Two geometrically-complementary triangles δA and δB are sampled as usual to define a rigid-body transformation T aligning two units to generate a new configuration C_{AB}. The rest of the section details the two filters employed to efficiently obtain biologically feasible dimeric models.

Geometric Filter

The geometric filter is designed to ensure that all generated dimeric configurations satisfy the environmental constraints posed by GPCRs. The constraints are as follows:

- All TM regions need to be inside the membrane.
- All extracellular parts need to be outside of the membrane (with some flexibility)
- The top and the bottom parts of the TMs are need to touch the edge of the membrane regions.

To satisfy these constraints we have designed a geometric filter which works as follows: An axis-vector representation has been employed to determine the valid placement of any unit with respect to the membrane. We illustrate the scenario in Figure 4.2(a). Given a global coordinate system, we define the y-axis to be aligned perpendicularly with the membrane and z-axis is the lateral axis. As shown, an ideal placement of GPCR is the main axis, which runs from top to bottom of the hollow cylinder formed by the TM regions. Let us refer to this main axis as v_A and v_B for units A and B, respectively.

Given C_A, the top residues of a TM region are those just before the chain becomes extracellular, and the bottom residues are defined as the last residues just before the chain becomes intracellular. The center of mass of the bottom residues, $cm_{A,\text{bottom}}$, and that of the top residues, $cm_{A,\text{top}}$, are calculated. The axis vector $v_A = cm_{A,\text{top}} - cm_{A,\text{bottom}}$. Similarly, an axis vector v_B is defined for unit B to be docked onto unit A.

This minimal yet effective representation allows us to quickly reject a dimeric models by computing the invalid placement of unit B relative to the membrane. As shown in Algorithm 1, we calculate the angle between the two vectors θ which needs to be within a specific segment of $[0, 2\pi]$. In an ideal placement, as shown in Figure 4.2(b), where two units in a dimer are perfectly aligned with the global y axis, the angle $\theta = 0^\circ$ determines the threshold. This is determined as follows: if we rotate a GPCR unit 5° increments before its TM regions exit the membrane, gives the maximum misalignment allowed for a GPCR unit relative to the membrane. The maximum allowed value between two units, defined as θ is twice this value, Lines $9 - 10$ show that if $0 \leq \theta \leq \Theta$, only then is the rigid-body transformation T applied to the entire unit C_B. The resulting dimeric model C_{AB} is next subjected to an energetic filter.

Energetic Filter

Configurations passing through the geometric filter are then subjected to energetic filter to determine any big steric clashes that cannot be removed by energetic minimization or slight fluctuations of the structure. Rather that applying any intensive energy function, here we consider only the Lennard-Jones potential (LJ) on atoms in contact. An atom from unit A is said to be in contact with an atom of transformed unit B if they are within d_{contact}Å of each-other. We implement the LJ potential based on the CHARMM22 force field [66]. We have down-weighted the contribution from the repulsion. To determine the energetic threshold E_{max} and down-weight the repulsion term, we generate 1000 configurations that pass the geometric filter and record their LJ potential values. Then we select three different configurations with low, medium, and high energetic values and then feed them to a leading refinement protocol, Firedock[67]. Configurations where steric collisions could not be removed upon slight backbone and side-chain fluctuations by Firedock are recorded in order to determine both E_{max} and a weight for the repulsion term. As shown in lines $11-12$ in Algorithm 1, a configuration is considered valid and finally added to the ensemble Ω only if its LJ value is below our defined E_{max}.

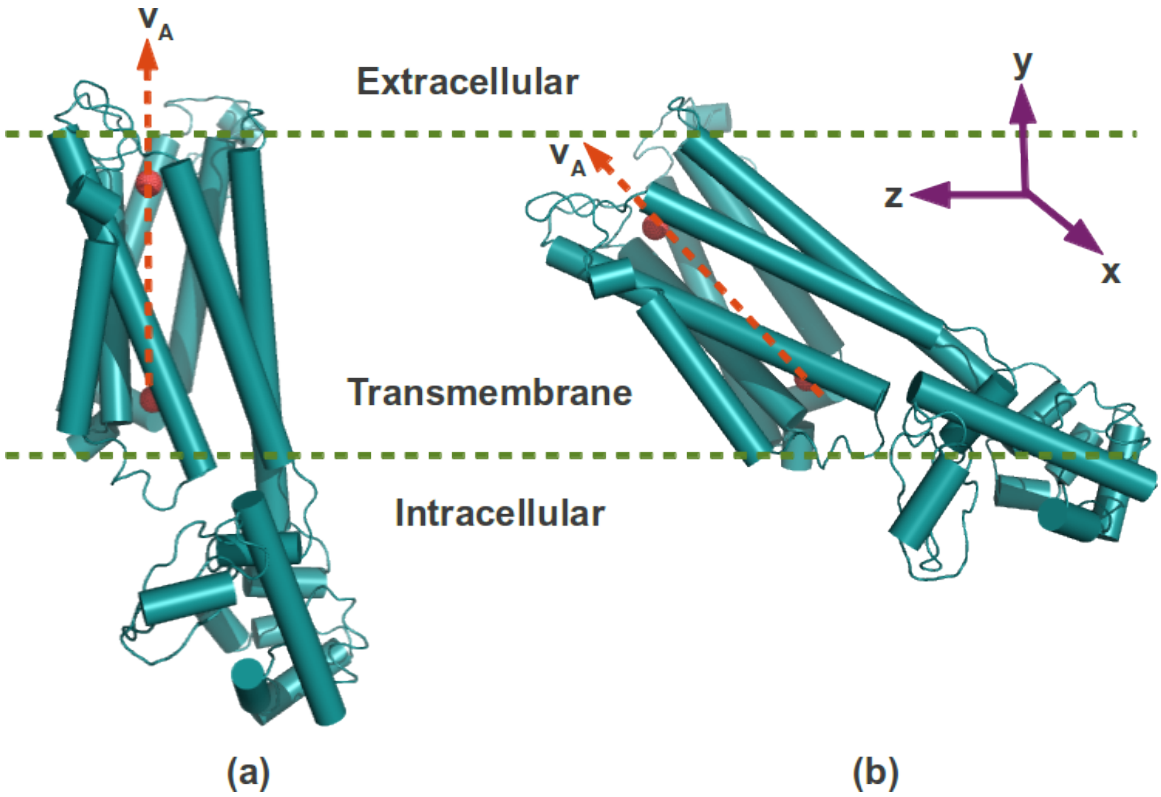


Figure 4.2: (a) An axis vector v_A is defined on a unit A to track the unit placement relative to the membrane (y is chosen to be normal to the lipid bilayer). (b) The axis vector allows determining the maximum allowed unit misplacement relative to the membrane, prior to TM regions exiting it. An angular constraint can be defined to force specific relative placements of two units in a dimer.

4.2.2 Decoy Selection

Decoy selection refers to the process of selecting a subset of generated configurations most likely to capture the native structure. We pursue decoy selection here on different mechanisms. We take an energy landscape approach where each generated interaction interface involving two GPCR units corresponds to an energy basin. More specifically, we organize the interaction based on TMs involved in the interface. Given x TMs, potentially important for interaction, we have a total of $\binom{x}{2}$ states corresponding to an energy basin. We measure density to compare states. Three different energetic criteria have been employed: total interaction energy, a new scoring scheme referred to as combined score, and multi-objective analysis.

Selection Based on Total Interaction Energy

A detailed energetic evaluations can be performed on the ensemble Ω to select decoys based on total interaction energy. We employ FoldX [68] (details can be found in 6). Using FoldX, one can focus only on the lowest-energy decoys. Distributing these decoys into the TM-based states described above allows then comparing basins in the energy landscape. Various statistics can be measured, the most basic of which is number of decoys in a basin, or density of state. Such comparison allows determining which are the most populated basins to determine whether certain TM pairings in the interaction interface are more thermodynamically-stable (energetically-favored) over others.

Selection Based on Combined Score (CS)

The density of the states can also be measured on decoys not based on lowest energy but based on new scoring scheme. This type of analysis helps to encounter any outlier produced by energy functions, as most of the energy functions have inherent errors. We design a combined scoring scheme shown in Equation 4.1, where D_{state} is the lowest energy, and Z-score $Z_{\text{state}} = (D_{\text{state}} - \mu_{\text{state}})/\delta_{\text{state}}$, where μ_{state} and δ_{state} are the mean and standard

deviation over energy values of decoys in a given state. The penalty term, C_{state} estimates high structural diversity via the maximum least root mean square deviation (lRMSD) [69] between any two decoys in a state.

$$\text{CS}_{\text{state}} = \frac{\text{D}_{\text{state}}}{\text{Z}_{\text{state}} \cdot \text{C}_{\text{state}}} \tag{4.1}$$

Diversity in a state is penalized for the following reason: if a specific TM pairing is observed among low-energy dimers, that pairing is likely to be native if there is consistency among the obtained decoys; that is, the sampling algorithm repeatedly reproduces it.

Selection Based on Multi-objective Analysis

Terms employed in the energy functions are sometimes conflicting optimization criteria. Our analysis on the correlations between the energy terms in FoldX reveals which terms are conflicting and need to be grouped together. According to our analysis, we see that hydrogen bond and solvation have a strong positive correlation. So we group together these two terms, leaving the van der Waals (LJ) interaction by itself in a second group, and electrostatics in the third group. Instead of treating all the terms in one group as single optimization criteria, these three groups are defined so that each group works as an optimization objective and decoys can be selected and sampled across all these groups.

We employ Pareto dominance to compare decoys.

Pareto Dominance: A configuration C_i is said to dominate another configuration C_j, if every energy term of C_i is lower than that of C_j and C_j is said to be dominated by C_i. This is also noted as strong dominance, and is what we employ here.

Pareto Count and Pareto Rank: The Pareto count PC_i of a configuration C_i denotes the total number of configurations that C_i dominates. The Pareto rank PR_i of a configuration C_i is the total number of configurations that dominates C_i. If C_i is not dominated by any configuration in the ensemble Ω, then C_i has Pareto rank 0. The set of all such configurations are also referred to as the Pareto front.

Beside energy, Pareto rank and count can be applied to compare generated decoys and to propose a subset of configurations. For example, given two decoys, the one with lower Pareto rank may be preferred in a selection procedure, since it is the dominating one. If two decoys have the same Pareto rank, the one with lower Pareto count may be preferred, as lower Pareto count ensures energetic diversity[70].

4.3 Applications of the Protocol

We demonstrate the proposed protocol and its usefulness on offering credible models of dimerization of D2R, a central GPCR. A consensus is emerging from wet-lab studies that interaction interfaces for higher-order assembly of D2Rs involve a subset of TMs, namely, $TM1$, $TM4$ and $TM5$[63–65]. We emphasize that the application setup of the proposed protocol is a blind prediction setting, as no structural models exist for GPCR dimerization. In addition, no X-ray structure of a D2R unit exists. However, we exploit the high structural similarity among GPCRs to obtain a credible model for a D2R unit. We employ I-TASSER [71], a top performer in the community-wide **C**ritical **A**ssessment of Protein **S**tructure **P**rediction (CASP) [72] and the human X-ray structure of D3R, the closest GPCR to D2R available to date, as a template model for I-TASSER to build a model for a D2R unit.

4.3.1 Implementation Details

Parameter values used in application of our protocol to D2R are shown in Table 4.1. Row 1 shows the size of a D2R unit (number of atoms). Row 2 shows that, in keeping with experimental knowledge [8, 63–65], the active regions used are those on TM1, TM4, and TM5. Row 3 shows that d_{contact} to determine amino acids in contact is set to 6.5Å, as in [73]. Row 4 shows the value for the angular threshold Θ used by the geometric filter. This value was obtained by observing the angle, in 5°, at which TMs in a D2R exit the membrane. Beyond 40°, a significant portion of regions in TMs exit the membrane. So, Θ is

Table 4.1: Parameter Settings

Parameter	Values
Size (Nr. of Atoms)	7198
Active TMs	$1, 4, 5$
d_{contact} (Å)	6.5
Θ(degree)	$[0, 80]$
E_{max} (kcal/mol)	$100,000$
w_{attr}	1.0
w_{repul}	0.1
Ω	10,000

set to twice this value, 80°. Row 5 shows the $E_{\max}$ value used by the energetic filter. Rows 6−7 show the weights used for the attraction and repulsion terms in the LJ potential. We recall that the reason for downplaying the contribution from the repulsion is to allow slight inter-unit penetrations that can be resolved with short energetic refinement protocols and anticipated structural changes, particularly in the possibly highly-flexible loop regions. The results presented here for D2R dimerization are those with an ensemble Ω of $10,000$ sampled decoys (last row in Table 4.1). A second ensemble of $50,000$ decoys has been generated. The same results have been obtained, which allows concluding that an ensemble of $10,000$ decoys is sufficient to reach conclusions on D2R dimerization.

4.3.2 Distribution of Lowest-energy Decoys

This experiment was performed based on our first selection criteria in Section 4.2.2 to examine the distribution of negative interaction energy configurations of different TM-based states. Figure 4.3 shows the distribution of configurations with negative FoldX interaction energy for all TM combinations. The x-axis shows $E[TM_i{-}TM_j]$, which is the actual energy score obtained from FoldX, and the y-axis shows the frequency of the configurations through

$n(E)$. Figure 4.4 summarizes the density of state comparison in a heatmap color-coded by density. Both of the analyses show that the three most populous states are TM1-TM4, TM4-TM4, with TM4-TM5 following third.

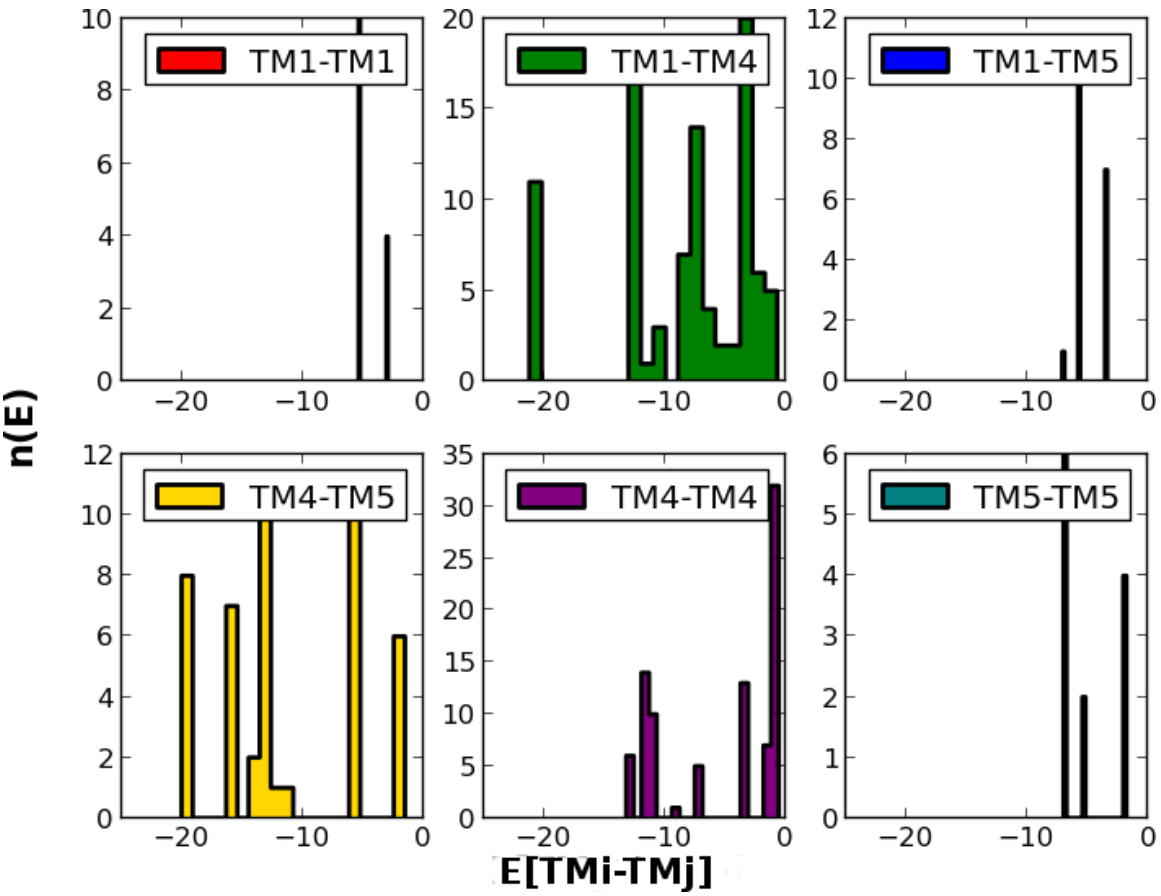


Figure 4.3: Histogram of lowest-energy decoys grouped in the different TMi-TMj states.

These results strongly agree with cross-linking wet-lab studies in [8, 63–65], which propose a central role for TM4 in the dimerization process of D2R. Work in [63] proposes that TM4-TM4 is the specific interface, while others [64, 65] suggest that TM4 may interface with TM1 and TM5, as well, and form the basis for higher-order assemblies. Our results

in Figures 4.3 and 4.4 indicate that all interfaces where TM4 is involved are energetically-favorable. This suggests that TM4 may be involved in both stable and transient interfaces which possibly provide flexibility to generate oligomeric states.

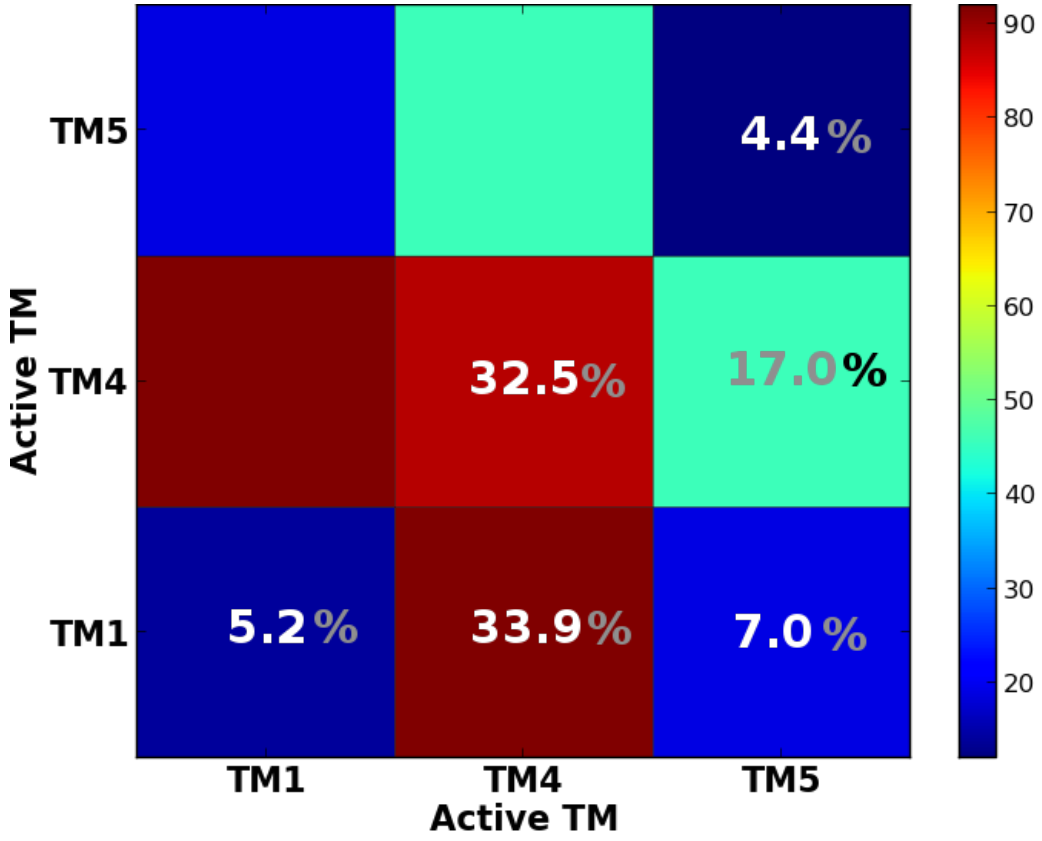


Figure 4.4: States are color-coded based on their density (focusing only on negative-energy decoys in Ω). Actual distribution of population is indicated over each cell/state in the heatmap.

4.3.3 Distribution of Highest-CS Decoys

This experiment is based on our selection mechanism Combined Score (CS). Figure 4.4 (b) shows the bar diagram where the y-axis represents the CS-score and the x-axis presents

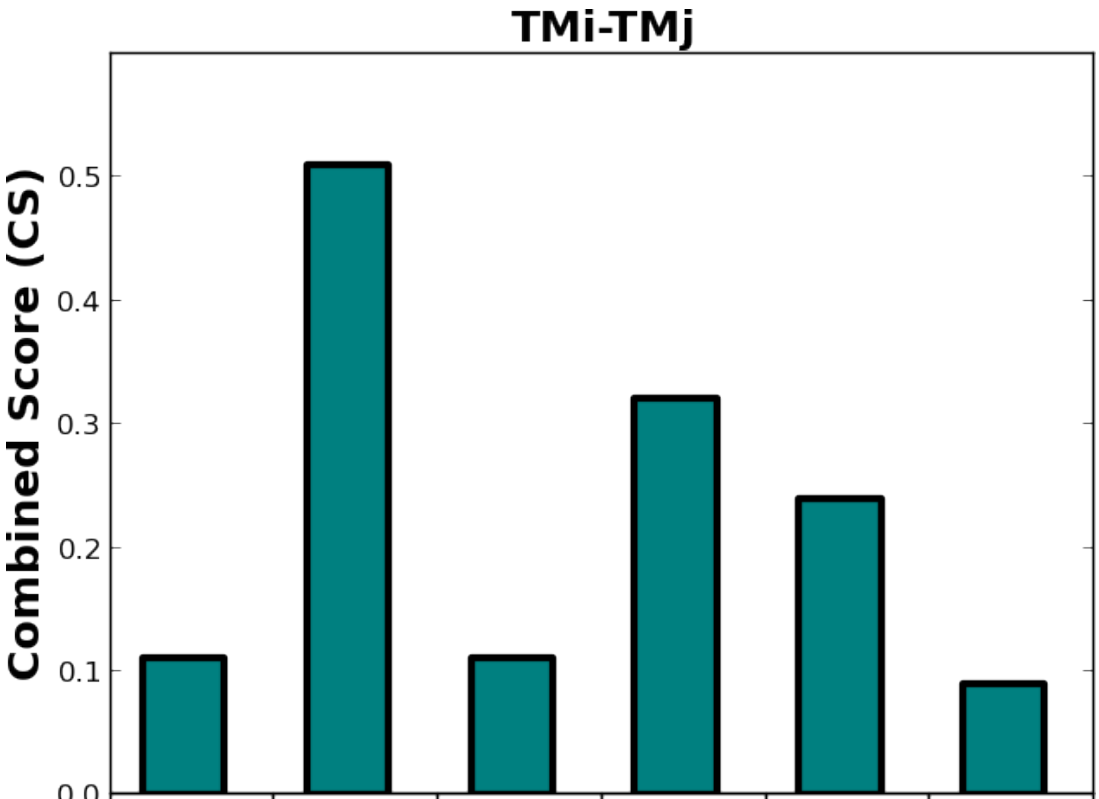
each type of TM-based states. A breakdown of each of the terms in CS are shown in a table at the bottom of the figure. The column labels show each interface type and the row labels are the individual terms employed in equation 4.1 to calculate CS. While the structural diversity in each of these three states is similar, TM1-TM4 has more configurations with low energies (as evidenced by the mean energy value shown in the table below the bar diagram). The score also slightly favors TM4-TM5 over TM4-TM4. These results support those shown above, that it is perhaps TM1-TM4 that promotes a stable dimer, but other TM4-based interfaces may be transient dimers possibly giving D2R the flexibility to form a rich set of higher-order assemblies.

4.3.4 Distribution of Decoys Based on Multi-objective Analysis

Two types of experiments have been done in this section based on multiobjective analysis based on Pareto rank and Pareto count. The first analysis is performed as follows: the Ω configurations are sorted according to the Pareto rank in ascending order, and then within each rank the configurations have been sorted according to their total FoldX interaction energy and we take top $p\%$. Let us denote this whole process as $\omega_{PR:E_{p\%}}$. The decoys in this ensemble are grouped into the possible 6 TM-based states, and the density of state analysis is conducted.

The next set of experiment was conducted first by sorting the configurations by their Pareto rank in ascending order, then within each rank the configurations were first sorted according to their Pareto count in ascending order and then by total interaction energy in ascending order. Lets denote this process as $\omega_{PR:PC:E_{p\%}}$ over entire ensemble of Ω, where p denotes the top $p\%$ of configurations. The bar diagram in Figure 4.6 shows the density of state analysis for each sorted ordering for $p \in \{5, 10\}$.

Figure 4.6 shows that two states, TM1-TM4 and TM4-TM5, are well populated with changing p and the Pareto metrics. The additional consideration of Pareto count in the decoy selection favors TM5, with TM4-TM5 ranking higher than TM1-TM4. These results indicate that TM5 may add specific energetic contributions to the interaction interface,



	TM1-TM1	TM1-TM4	TM1-TM5	TM4-TM5	TM4-TM4	TM5-TM5
D_{state}	-5.35	-21.22	-7.12	-20.01	-13.21	-7.01
μ_{state}	205.15	228.62	304.44	345.16	205.83	377.52
σ_{state}	164.6	245.84	195.31	244.65	175.64	210.09
C_{state}	37.37	40.52	40.11	41.41	42.43	39.69

Figure 4.5: CS is shown for each state. Values of various metrics used in CS are shown in the table.

however they may not result in the lowest interaction energy.

4.3.5 Proposed Models for D2R Dimerization

After taking the consensus of varied selection mechanism, the results demonstrate the involvement of TM4 as part of the interaction interface in D2R dimerization. Selection based on interaction energy and CS show that the possible models are $TM1-TM4$ and/or $TM4-TM5$ and/or $TM4-TM4$. A detailed multiobjective selection on different energy terms favor $TM4-TM5$ and $TM1-TM4$ over $TM4-TM4$. Taken all these analysis together, we propose TM1-TM4 as the core interface for a stable dimer whereas dimers as TM4-TM4 and TM4-TM5, is proposed here to be seminal for transient, shorter-lived dimers, which may be less stable than TM1-TM4, but may be employed as alternative ways to link D2R units in higher-order assemblies.

In figure 4.7 we show the structural details of some of the D2R-D2R models that we propose to occur in the living cells. Figures 4.7(a)-(b) show the representative model/decoy in the TM1-TM4 state proposed here for a stable D2R-D2R dimer. It is also interesting to note that a TM1-TM4 interface promotes a secondary symmetric TM4-TM1 interface (see top view in Figure 4.7(a)), which additionally lowers the potential energy and further explains why a TM1-TM4 interface may be the most stable in the cell. Moreover, a side view in Figure 4.7(b) shows that this placement of the TM-states allow enough room for the intracellular loops to move around without energetic penalty. In contrast, Figure 4.7(c)-(d) shows a representative model for the TM4-TM4, and Figure 4.7(e)-(f) shows a representative model for the TM4-TM5 interface, which we predict as short-lived. The intracellular loop regions are closer to each-other than in the TM1-TM4 state. While this may promote secondary interactions between the loop regions, the entropic cost may be too high in these alternative dimeric states.

It is worth noting that the criteria employed for decoy selection here are not readily available for general application for several reasons. The criteria here are designed to discriminate states, which are specific subgroups of topologically-similar configurations based

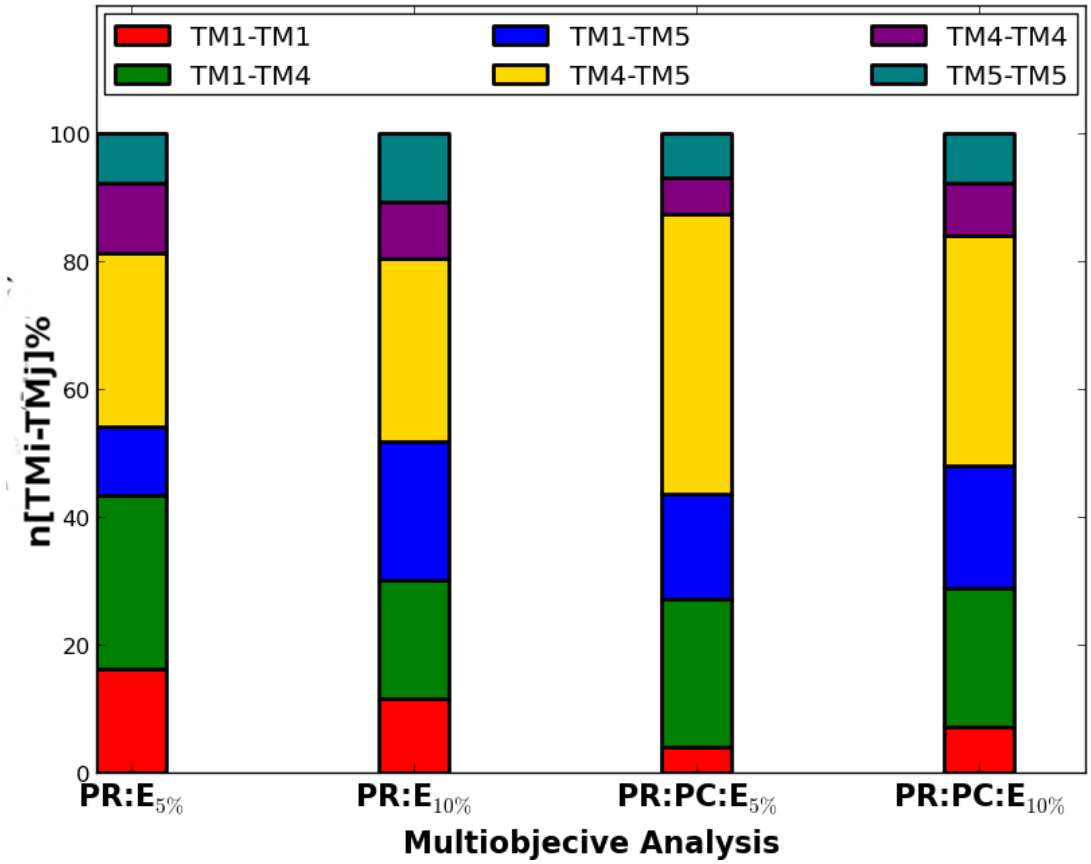
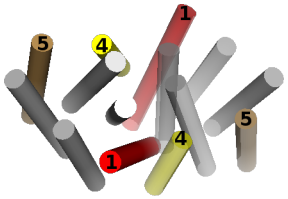
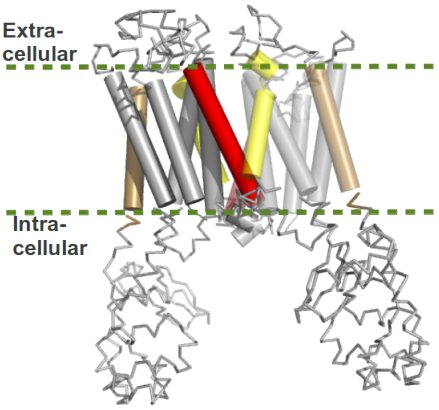


Figure 4.6: Decoys in the $\Omega_{PR:E_{p\%}}$ ensemble are organized into the 6 possible states, and the density analysis is repeated in the first two bar diagrams for $p = 5\%$ and $p = 10\%$. The density analysis is repeated on decoys in the $\Omega_{PR:PC:E_{5\%}}$ and $\Omega_{PR:PC:E_{10\%}}$ ensemble in the next two bars.

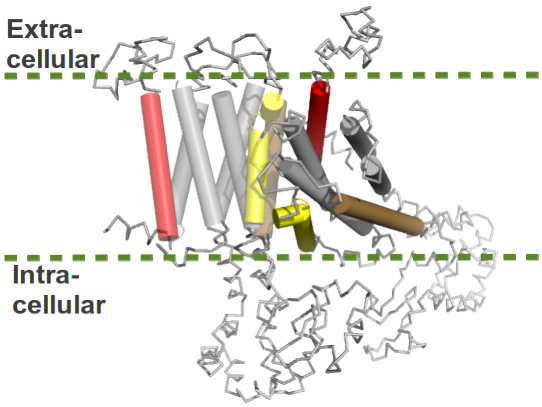


(a) TM1-TM4 top view

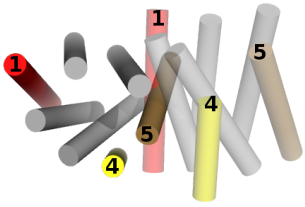


(b) TM1-TM4 side view

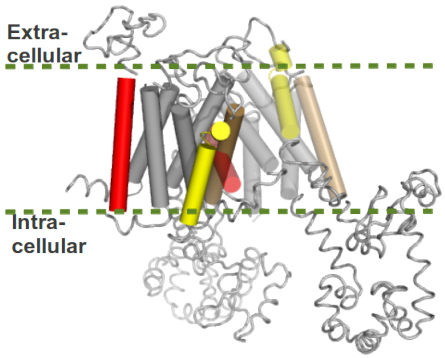
(c) TM4-TM4 top view



(d) TM4-TM4 side view



(e) TM4-TM5 top view



(f) TM4-TM5 side view

Figure 4.7: (a)-(f) Representative models are drawn with pyMol [9], with the TM regions shown as cylinders. TM1 is drawn in red, TM4 in yellow, and TM5 in tan. For ease of visualization, the base unit is drawn in transparent, while the other one in opaque. (a)-(b) show a representative model of the TM1-TM4 state, whereas (c)-(d) show one such model of the TM4-TM4 state, and (e)-(f) show one such model of the TM4-TM5 state. Top views are shown in (a), (c) and (e), and side views in (b), (d) and (f)

on pairings of TM regions. The latter are not applicable to other protein assemblies. An interesting direction of research in decoy selection can consider incorporating some of the Pareto-based analysis proposed here in clustering-based decoy selection methods.

Chapter 5: HopDock: Combining Energy and Geometry under a Probabilistic Framework

The presence of sufficient constraints may make enumeration practical and simple geometry-driven methods reasonable approaches for docking of specific proteins. In the absence of prior knowledge on the location of the true interaction interface, it becomes critical to have a stochastic optimization algorithm with high exploration capability. In this chapter, we propose HopDock, a novel evolutionary algorithm that integrates geometric complementarity in an energy-driven framework. Unlike evoDock, the proposed algorithm offers a smaller, yet informative set of dimeric configurations corresponding to local minima of a dimeric energy surface.

5.1 Methods

HopDock is an adaptation of an evolutionary algorithm known as Basin Hopping (BH) (illustrated in Figure 5.1). BH is an effective sampling technique which has been used in other fields of computational biology [74]. The algorithm explicitly samples a trajectory of n dimeric low-energy local minima configurations from a chosen energy function. The hopping between two local minima C_i and C_{i+1} is performed through an intermediate perturbed configuration $C_{\text{perturb,i}}$. The perturbation operator thus try to modify a local minima C_i to obtain a configurations $C_{\text{perturb,i}}$ by escaping the current minima energy barrier. On the other hand, the minimization operator takes a perturbed configuration $C_{\text{perturb,i}}$ and by making some smaller moves help it to reach a nearby local minimum C_{i+1}. C_{i+1} is added to a growing trajectory according to a Metropolis Criterion $e^{-\Delta E/T_e}$ where ΔE is the energetic difference between C_i and C_{i+1} and T_e is the effective temperature that allows a certain

energy jump with a probability.

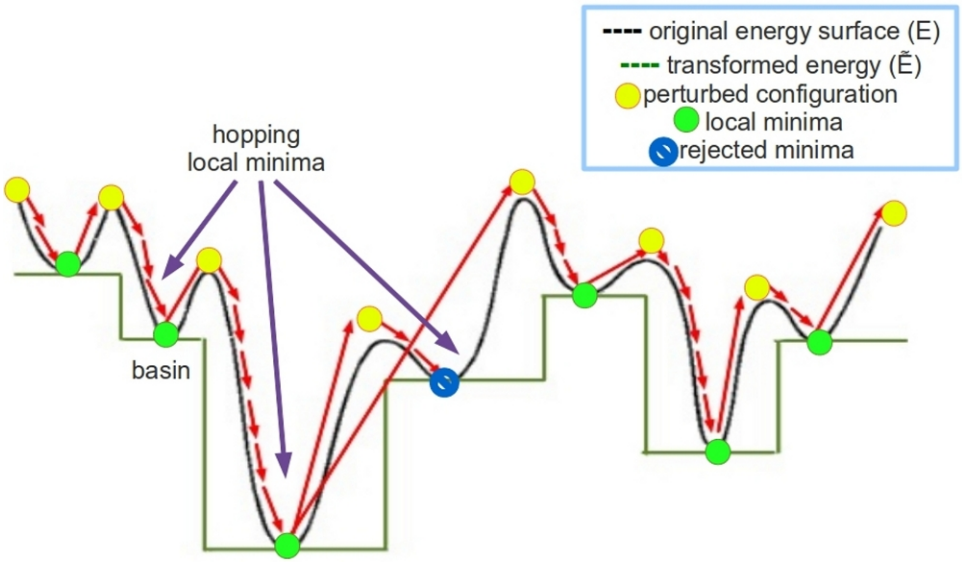


Figure 5.1: Under the BH framework, the energy surface is transformed into a collection of interpenetrating staircases. A trajectory of local minima is obtained consecutively, through iterated applications of a structural perturbation to jump out of a current local minimum and an ensuing local optimization to map to another nearby local minimum.

HopDock consists of two main operators, a structural perturbation and a local optimization/improvement or minimization operator. We detail each next.

5.1.1 Perturbation Operator

The perturbation operator takes a local minimum configuration C_i as input to obtain a perturbed configuration $C_{\text{perturb,i}}$. Let us refer to the two active triangles used to obtain C_i as tr_A and tr_B (one for each unit). The perturbation operator then samples a new active triangle $tr_A^{'}$ over the molecular surface of unit A uniformly at random in a d-neighborhood of tr_A. d is defined as distance between the center of mass of tr_A and $tr_A^{'}$ and expressed in Å. Given

a newly sampled triangle $tr_A^{'}$, the operator seeks another geometrically-complimentary triangle $tr_B^{'}$ sampled again in a d neighborhood radius distance of tr_B. Employing these two triangles, a new rigid-body transformation is obtained, whose application on the moving unit B results in $C_{perturb,i}$.

Smaller values of d ensure the adjacency between C_i and $C_{\text{perturb,i}}$ so that some structural features of C_i are preserved in the newly sampled configuration $C_{\text{perturb,i}}$. Our analysis shows that small values can result in no geometrically-complementary triangles being sampled. On the other hand, a large value of d will throw the new configuration far away in the search space, resulting in a weak relationship between C_i and $C_{\text{perturb,i}}$. Comparison of the performance of the algorithm under different values of d shows that HopDock is successful when the magnitude of the perturbation jump is not too small or not too large, as also demonstrated in other domains [74, 75]. An analysis of different values of d has been performed in 5.2. We also demonstrate that controlling the distance d yields better results than random restarts (where d is essentially infinite).

5.1.2 Local Improvement Operator: Energy Minimization

The improvement operator tries to modify the perturbed configuration $C_{perturb,i}$ and map it to a nearby local minimum C_{i+1} through a simple minimization process. A perturbed configuration $C_{\text{perturb,i}}$, sampled from a rigid-body motion can be represented as $\langle u, \theta, t \rangle$, where $\langle u, \theta \rangle$ refers to the orientation operator in axis-angle representation and t refers to the translation operator. At each step of the minimization process, a new random rotation is obtained by sampling a new axis $u^{'}$ rotating around the axis u by a predefined angle value, and a new angle is obtained by sampling in a neighborhood around θ. A new translation operator is then sampled in a predefined neighborhood of t. The minimization is carried out for at most m consecutive steps until k consecutive modifications fail to lower the energy. A newly generated local minimum is not directly added to the growing trajectory of local minima. A Metropolis criterion determines acceptance. If rejected, the process begins

again, with another perturbed configuration. A detailed analysis of the role of different values of the effective temperature in the Metropolis criterion is provided below.

Our implementation of the local improvement operator does not seek to identify the true basin of a local minimum. The depth of the exploration is determined by the parameter m in the minimization. Given that the sampled configurations need to be low-energy but can be refined in greater detail at a later stage in a protein-protein docking protocol, this working definition of a local minimum is sufficient.

For this reason, the minimization operator employs a simple energy function, designed as:

$$E = E_{vdW} + E_{electrostatic} + E_{hydrogen-bonding}. \tag{5.1}$$

The first two terms, capturing van der Waals and electrostatic interactions, are implemented as in the CHARMM22 force field [66]. To capture the van der Waals energy we have used the standard 6-12 Lennard-Jones potential as follows:

$$E_{vdW} = \sum_{atompairs} \epsilon[(\frac{r_{ij}}{d_{ij}})^{12} - 2 \times (\frac{r_{ij}}{d_{ij}})^{6}] \tag{5.2}$$

where r_{ij} is the atomic radii sum, ϵ is the energy well depth derived from CHARM22 [66], and d_{ij} is the distance between atoms i and j.

The electrostatic term is computed based on Coulomb's law:

$$E_{electrostatic} = \sum_{atompairs} \frac{q_i \times q_j}{e \times {d_{ij}}^2} \tag{5.3}$$

where q_i and q_j are the electrostatic charges of atoms i and j obtained from CHARM22 [66], e is the dielectric constant (vacuum constant 1 is used for this paper), and d_{ij} is the distance between atoms i and j.

The hydrogen-bonding term is calculated through the 12-10 hydrogen potential [76] as follows:

$$E_{hydrogen-bonding} = 5 \times (\frac{r_0}{d_{ij}})^{12} - 6 \times (\frac{r_0}{d_{ij}})^{10} \tag{5.4}$$

where d_{ij} is the distance between the interface acceptor and donor atoms i and j, and $r_0 = 2.9$ Å is the optimal distance for hydrogen bonding.

5.2 Applications of HopDock

We first provide a detailed analysis on key parameter settings in the various algorithmic components in HopDock before relating summary performance results.

5.2.1 Experimental Setup

HopDock is implemented in C/C++. The experiments are carried out on the Argo research computing cluster provided by George Mason University. We select same 15 different dimers with known native structures as in evoDock. Results are obtained over 5 independent runs of HopDock to account for any variance in the results. For each case, $10,000$ configurations are generated.

5.2.2 Analysis of Effective Temperature

A high temperature in the Metropolis criterion degenerates HopDock to random search. On the other hand, a low temperature hinders crossing the current energy barrier to converge to another local minimum. We compare here the effect of two temperatures, T_0, a higher temperature that allows an energy increase of 2 kcal/mol with a probability of 0.39, and T_1, a lower temperature that allows accepting an energy increase with a lower probability of 0.16.

Table 5.1 summarizes the effect of T_0 and T_1 on the sampling of local minima by HopDock

Table 5.1: Analysis of effective temperature T_e on 8 representative systems. The first two columns provide details on the protein systems selected for this analysis. The third column shows the lowest lRMSD achieved for each system. The next two columns show the lowest energy reached and the percentage of configurations with lRMSD less than $5\mathring{A}$ from the native structure.

PDB ID	T_e	lRMSD (Å)	Energy (kcal/mol)	$< 5\mathring{A}$ (%)
1FLT	T_0	1.69	-1.73	0.21
	T_1	2.10	-0.67	0.09
1WWW	T_0	2.98	-0.88	0.14
	T_1	2.16	-21.39	0.12
1C1Y	T_0	1.05	-0.83	0.78
	T_1	1.42	-10.50	0.80
1QAV	T_0	2.57	-0.87	0.07
	T_1	2.59	-0.52	0.11
1DS6	T_0	3.40	-0.63	0.10
	T_1	3.52	-0.86	0.16
1OHZ	T_0	2.73	-1.07	0.76
	T_1	1.48	-1.76	0.80
1VCB (A,B)	T_0	3.44	-0.63	0.11
	T_1	3.27	-0.69	0.12
1VCB (B, C)	T_0	2.68	-0.80	0.16
	T_1	2.91	-1.63	0.16

on 8 of the 15 dimers. The last three columns show lowest lRMSD to the native structure (in $\mathring{A}$), lowest energy reached (in kcal/mol), and the percentage of generated configurations with lRMSD less than $5\mathring{A}$ to the native structure. On 5 systems, T_0 allows HopDock to get closer to the native structure. Column 4 shows that T_1 drives the search to the lower energy region. However, lower energy does not always drive closer to the native structure. The last column shows comparable results in terms of achieving near-native configurations. Taken all together, this analysis suggests using T_0 for the rest of the production runs of HopDock.

5.2.3 Analysis of Perturbation Distance

In this section, we compare different magnitudes d in the perturbation operator. For this experiment, we have chosen 3 different settings. The first one is essentially the random restart, where a geometrically-complementary triangle is sampled randomly with $d = \infty$. The next two setting employ $d = 5$ and $d = 7\mathring{A}$. Anything smaller than $5\mathring{A}$ makes it difficult to sample geometrically-complimentary triangles on the molecular surfaces.

We show results on three representative systems (with PDB IDs 1FLT, 1WWW, and 1C1Y). We measure 5 different statistics, shown in Table 5.2. The first measures l, the distance between two consecutive perturbed configurations, $C_{\text{perturb,i}}$ and $C_{\text{perturb,i+1}}$, measured in lRMSD. We report the median of l, l_m, over all perturbed configurations obtained by HopDock. The next two columns provide more detailed statistics of l by showing the percentage of distances in the $0-5$Å, and $5-10$Årange, respectively. The last two columns show the least lRMSD in Å and the percentage of configurations with lRMSD to the native structure less than 5Å.

The results shown in Table 5.2 suggest that controlling d allows reducing l_m. Lower values of d result in lower values of l_m. The number of perturbed configuration with lRMSD < 5 and < 10Åalso goes up with lower values of d. Lower d also generally increases the number of minima configurations with lRMSD less than $5\mathring{A}$. Taken all together, the production runs reported next employ a perturbation operator with $d = 5$Å.

5.2.4 Analysis of Improvement Operator

The implementation of the improvement operator employs thresholds of $\delta_\phi = 10^\circ$ and $\delta_\theta = 30^\circ$. We analyze here the variation in the translation operator t, which can take three different values $\{1.5, 2.0, 2.5\}$Å. The goal of this experiment is to determine whether smaller moves increase the probability of generating a local minimum nearest to the $C_{\text{perturb,i}}$, and whether it has any effect on the overall quality of the the sampled configurations.

We measure the distance between $C_{\text{perturb,i}}$ to C_i sampled by HopDock. Let us refer this

Table 5.2: Effect of d in the perturbation operator on representative systems.

PDB ID	d (Å)	l_{m} (Å)	$l <$ 5Å(%)	$l <$ 10Å(%)	min lRMSD(Å)	lRMSD< 5Å(%)
1FLT	$d = \infty$	16.73	0.29	3.82	3.37	0.11
	$d = 7$	14.76	2.09	16.57	2.48	0.12
	$d = 5$	13.48	4.73	18.20	1.69	0.23
1WWW	$d = \infty$	17.83	0.27	2.86	2.33	0.12
	$d = 7$	14.62	2.19	13.32	3.63	0.08
	$d = 5$	14.22	4.86	14.91	2.89	0.08
1C1Y	$d = \infty$	14.31	0.51	9.87	2.17	0.70
	$d = 7$	11.23	4.86	30.02	2.09	0.71
	$d = 5$	11.06	6.49	31.00	1.05	0.86

distance as i. In Table 5.3 we report the median i_m over all i values obtained in a HopDock trajectory. This value is shown in column 3. The next two columns show the percentage of distances in the $0 - 5$Å, and $5 - 10$Å range, respectively. Finally, columns 6 and 7 show the effect of i on the sampled minima in terms of the lowest lRMSD to the native structure and the percentage of minima with lRMSD to the native structure less than 5Å.

The results shown in Table 5.3 suggest that varying t does not have a large effect on most of the statistics. By comparing the number of local minima less than 5Å, we elect to employ a translation threshold of $t = 1.5$Å for the rest of the analysis on the performance of HopDock.

5.2.5 Comparison of HopDock to Other State-of-the-art Methods

In this section we perform a comprehensive comparative analysis of HopDock in terms of two different measures: lRMSD and normalized rank. The normalized rank is calculated as follows: we sort the computed configurations according to their energy values in ascending order. Then we report the position of the lowest lRMSD configuration in that sorted

Table 5.3: Effect of translation threshold t in the improvement operator on three representative systems.

PDB ID	t (Å)	i_m (Å)	i < 5Å(%)	i < 10Å(%)	min lRMSD (Å)	lRMSD< 5Å(%)
1FLT	1.5	10.14	19.30	29.69	2.70	0.07
	2.0	9.81	20.56	30.66	1.90	0.15
	2.5	9.98	20.26	29.82	1.69	0.23
1WWW	1.5	9.74	21.12	30.30	2.63	0.21
	2.0	9.68	21.50	30.55	3.88	0.12
	2.5	9.50	21.85	31.43	2.98	0.08
1C1Y	1.5	9.04	24.41	31.94	1.79	0.92
	2.0	9.40	21.99	31.78	2.14	0.51
	2.5	9.56	21.62	31.71	1.05	0.86

ordering. The normalized rank reports the percentage positioning of the lowest lRMSD structure, since different methods generate different numbers of configurations. If a method is unable to find a structure with lRMSD less than $5\mathring{A}$ to native, then a rank of 100 is reported, whereas a rank of 0 means that the lowest energy configuration is indeed the lowest lRMSD one.

For comparison, in addition to BUDDA [1] (a geometry-driven method), we have also chosen three leading energy-driven methods, pyDock [77], ClusPro [27] and ZDock [59]. These three methods are available as web servers. For this set of experiments, each method was run 5 independent times on each system to account for variance between runs. It is worth noting that ZDock and ClusPro are deterministic methods.

Table 5.4 compares HopDock to these four methods on the 15 systems in our testing data set. Column 2 shows the lowest lRMSD achieved by BUDDA. The third, fourth and fifth columns show the average lRMSD obtained by pyDock, ClusPro and ZDock respectively. Each of the entry is followed by the average normalized rank (ANR) in parentheses. The last four columns show the statistics obtained by HopDock. The sixth and seventh columns

show the lowest lRMSD and average lRMSD achieved by HopDock over 5 different runs. The next column shows the variance in the results. The very last column provides the average normalized rank of the lowest lRMSD structure obtained over 5 different runs of HopDock.

Table 5.4 allows drawing a few conclusions: on all of the cases, HopDock is able to generate near-native configurations and performs comparatively or better in terms of lowest lRMSD. We have highlighted in gray rows where HopDock indeed performs better than at least two of the other methods in terms of the average lowest lRMSD. Ranks obtained by HopDock are usually higher than the other energy-driven approaches, since HopDock employs a very simple energy function. However, on 9 out of 15 systems, the average normalized rank of HopDock is within 20% of any two other methods (these entries are in bold).

Table 5.4: Comparative analysis of HopDock to other state-of-the-art methods on 15 different systems. Results of HopDock are reported over 5 runs and shown in terms of lowest (η), average (μ), variance (σ^2) lRMSD, and Average Normalized Rank (ANR) on the last 4 columns. The average lRMSD to the native structure and ANR (in parentheses) are shown for BUDDA, pyDock, ClusPro and ZDock on columns 3−7.

PDB ID (Chains)	Budda [1] (Å)	pyDock [77](Å) (ANR)	ClusPro [27](Å) (ANR)	ZDock [59](Å) (ANR)	HopDock η(Å)	HopDock μ(Å)	HopDock σ^2(Å^2)	HopDock ANR
1C1Y (A,B)	1.2	3.2(4.3)	1.7(13)	0.7(0.3)	1.5	2.1	0.1	26.2
1DS6 (A,B)	1.2	0.7(0)	1.8(0)	1.2(0.05)	1.7	2.6	0.4	31.4
1TX4 (A,B)	1.4	0.4(0.1)	2.1(1)	0.7(0.1)	1.2	1.9	0.3	**4.4**
1WWW (W,Y)	11.4	9.3(100)	10.0(100)	8.3(100)	1.9	2.8	0.8	**47.8**
1FLT (V,Y)	1.5	0.5(0.7)	2.1(10)	1.1 (0.1)	1.4	2.4	0.3	35.4
1IKN (C,D)	2.0	5.0(0.3)	4.2(27)	1.8(42.85)	2.0	2.9	0.4	**45.7**
1VCB (A,B)	0.7	0.5(0.05)	1.5(49)	1.0(0.0)	1.7	2.1	0.1	**11.4**
1VCB (B,C)	1.3	0.55(0)	1.9(0)	0.7(0.0)	1.2	1.8	0.1	**10.1**
1OHZ (A,B)	1.8	1.2(0.08)	1.2(52)	0.7(1.95)	0.9	1.5	0.1	**20.0**
1ZHI (A,B)	25.3	3.2(0.9)	1.4(7)	1.2(0.1)	2.9	3.1	0.1	30.8
2HQS (A,C)	29.1	9.5(100)	9.7(100)	7.0(100)	2.3	3.2	0.3	**53.6**
1QAV (A,B)	1.4	1.2(0.8)	1.7(0)	0.9(1.7)	1.7	2.7	1.2	32.0
1G4Y (B,R)	0.8	18.3(100)	18.2(100)	0.7(3.7)	2.7	3.4	0.3	**33.4**
1CSE (E,I)	0.7	0.8(0)	1.1(0)	0.5(2.4)	0.7	1.6	0.2	27.4
1G4U (R,S)	1.0	15.2(100)	16.1(100)	1.3(0.05)	3.1	4.1	0.8	**51.1**

Chapter 6: idDock: Integration of a Predictive Machine Learning Model within HopDock

Integration of sophisticated energy functions is computationally-demanding. In addition, many such functions may lead the search towards non-native configurations. Hence, in this chapter we propose a novel algorithm that adapts the underlying BH sampling framework in HopDock to integrate a predictive machine learning model. idDock, which stands for (**i**nformatics-**d**riven protein-protein **dock**ing) method [78], applies improvement via a sophisticated energy function only on sampled configuration classified as native-like by a machine learning model trained a priori on true interaction interfaces.

6.1 Overview of idDock

idDock builds over HopDock. Instead of the simple energy function, it now uses a new, sophisticated one, FoldX [68] in the local improvement operator. The energy terms in FoldX are weighted using experimental data obtained from protein engineering experiments. The terms include solvation energy, van der Waals, hydrogen bond potential, electrostatic energy, entropic and clash penalty.

Unlike HopDock, after obtaining a perturbed configuration, idDock does not send the configuration to the improvement operator. Instead, the configuration is first sent to a machine learning classification model to determine whether the contact interface in the configuration is native-like or not. If the interface predicted by the model is deemed as positive/native, then it is sent to the local improvement operator; otherwise the configuration is discarded, and a new one is obtained from the perturbation operator. In this way, only promising configurations are subject to intensive energetic minimizations.

Since idDock builds over HopDock, in the following we describe only the novel components.

6.1.1 FoldX

FoldX is specially designed for protein interaction and the terms are weighted using protein engineering experiments using empirical data. The free energy of a target protein is calculated using following equation:

$$\Delta G = W_{vdw} * \Delta G_{vdw} + W_{solv_H} * \Delta G_{solvH} + W_{solvP} * \Delta G_{solvP} + \Delta G_{wb} + \Delta G hbond + others \tag{6.1}$$

where ΔG_{vdw} is the sum of the van der Waals contributions of all atoms to that with the solvent. ΔG_{solvH} and ΔG_{solvP} are the difference terms in solvation energy for apolar and polar groups respectively when changing from the unfolded to the folded state. ΔG_{hbond} is the free energy difference between the formation of an intra-molecular hydrogen bond and to that of inter-molecular hydrogen-bond with solvent. ΔG_{wb} is the stabilizing free energy provided by a water molecule. Others include electrostatic contribution of charged groups, entropy cost of fixing the backbone, entropic cost of fixing a side chain, effect of electrostatic interactions on the association constant and loss of translational and rotational entropy. The last two terms are specially designed for a protein complex. The energy values of ΔG_{vdw}, ΔG_{solvH}, ΔG_{solvP} and ΔG_{hbond} have been derived from a set of experimental data, and others have been obtained from theoretical estimates. The W_{vdw}, W_{solv_H}, W_{solvP} are the weighting factors for each of the terms associated with it.

6.1.2 Machine Learning Model to Evaluate Perturbed Configurations

We employ a large and diverse protein-protein interaction training dataset and then identify an optimal model to evaluate an interaction interface. We first describe the training dataset, then provide analysis of various features employed for this work, and compare different state-of-the-art machine learning models to determine an optimal one for integration in idDock.

Training Dataset Construction

Positive Dataset:

The positive dataset consists of 1071 true/native interaction found on experimentally-obtained assemblies extracted from PDBbind, a protein-protein dataset [79]. To obtain the dataset Protein Data Bank (PDB) [13] database is scanned, for protein-protein complexes, and are checked for the interface to assure at least 10 interacting residues on each chain of the two proteins. The minimum chain length of each protein unit is longer than 20 amino acids per chain. The dataset consists of non-redundant complexes. The non-redundancy implies the amino acid sequence of two complexes have no more than 90% sequence similarity. For example: suppose one complex C_1 is composed of chain A and chain B and another complex C_2 is composed of chain X and chain Y. Complex C_1 and C_2 are considered as redundant if and only if chain A is 90% similar to chain X and chain B is 90% similar to Y or chain A is 90% similar to chain Y and chain B is 90% similar to X.

Negative Dataset:

The negative datasets of 991 instances is obtained in three different ways. The first is constructed by randomizing the positive dataset. Units selected randomly from two different complexes from positive dataset are docked with a random rigid-body motion. A total of 456 dimers are obtained by repeated randomization. A detailed analysis on sequence similarity between chains in randomized complexes is provided in the appendix. The second set consists of 76 crystal packing structures obtained from [37]. The crystal packing structures are artificial contact structures that are generated as part of X-ray crystallography. These sort of interactions are not biologically viable. The third set consists of 459 dimeric structures ranging from $5-12$Å away in RMSD to native. This set is generated by pyDOCK [77].

Feature Vector

Each training instance is converted into a feature vector of 7 dimension. So, machine learning models discriminate between native/positive and non-native/negative instances in

a 7-dimensional space.

Each instance is expressed through an interaction interface. An amino acid is said to be on the interaction interface if its solvent accessible surface area (SASA) decreases by $> 1\text{Å}^2$ upon the formation of a complex. This definition is taken from [37]. The interaction interface is defined by combining the interacting amino acids from both of the units and then converted to a feature vector. The features are: interface area, interface area ratio, four different types of amino acid composition and conservation score.

The first entry of a feature vector is the interface area which is defined as in [37]:

$$\text{InterfaceArea}_{\text{A+B}} = 0.5 \cdot (\text{SASA}_{\text{A}} + \text{SASA}_{\text{B}} - \text{SASA}_{\text{A+B}}), \tag{6.2}$$

where $SASA_A$ is the SASA of reference unit A, $SASA_B$ is the SASA of moving unit B, and $SASA_{A+B}$ is the SASA of the dimer.

The second entry is the interface area ratio obtained defined as in [37]:

$$\frac{\text{InterfaceArea}_{\text{A+B}}}{min(\text{SASA}_{\text{A}}, \text{SASA}_{\text{B}})}, \tag{6.3}$$

The next 4 entries measure the ratio of the number of amino acid types on an interaction interface to the surface as follows:

$$\frac{NAA_k^{interface}}{NAA_k^{surface}}, \tag{6.4}$$

where, $NAA_k^{interface}$ is the number of amino acids of type k on an interaction interface, and $NAA_k^{surface}$ is the number of amino acids of the same type k the surface. As before, an amino acid is determined to be on the surface if it loses about $< 1\text{Å}^2$ in its SASA upon the formation of the complex [37]. Type k includes hydrophobic, hydrophilic, basic, and acidic.

The last entry in the feature vector is the average conservation score of the interaction

interface, measured over conservation scores of amino acids on the interface. The conservation score of an amino acid is measured through iJET, as in evoDock.

Analysis of Interaction Properties

Figure 6.-1 shows the distribution of the 7 interaction properties for both the negative and positive datasets. The x-axis shows the value range for a particular feature while the y-axis represents the percentage of the structures that fall within that particular range. Figure 6.-1 shows that the interface area of positive instances is overall higher than that of negative instances; however no conclusive observations can be made regarding the interface area ratio. Positive instances tend to have more hydrophobic atoms than non-native ones as obvious, and a similar observation can be drawn regarding the ratio of hydrophobic amino acids. Acidic and basic compositions do not seem to be a discriminating feature between native and non-native instances. More conserved amino acids are seem to be on the positive instance than on the negative with a threshold value > 0.6.

Selection of Optimal Machine Learning Model

The interaction interface of a training instance is converted to a 7 dimensional feature vector and labeled as 1 as positive and 0 being negative. Various supervised classification models are trained through machine learning tools weka [80] and the performance of each of the model is recorded in a 10-fold cross validation setting for comparison.

The model trained here are entropy-based Decision Tree (J48 implementation in weka), Random Forest, Bagging, and Support Vector machines. Though the performance of each trained model is measured in terms of F-measure, precision, recall and area under Receiver Operating Curve (ROC) curves, we only show here the ROC curves of each trained model in Figure 6.0. An ROC curve is a graphical plot that illustrates the performance of a binary classifier by plotting the true positive rate against the false positive rate at different threshold settings. Here to mention that an area under the curve (AUC) of 1 represents a perfect prediction, while an AUC of 0.5 represents a random prediction.

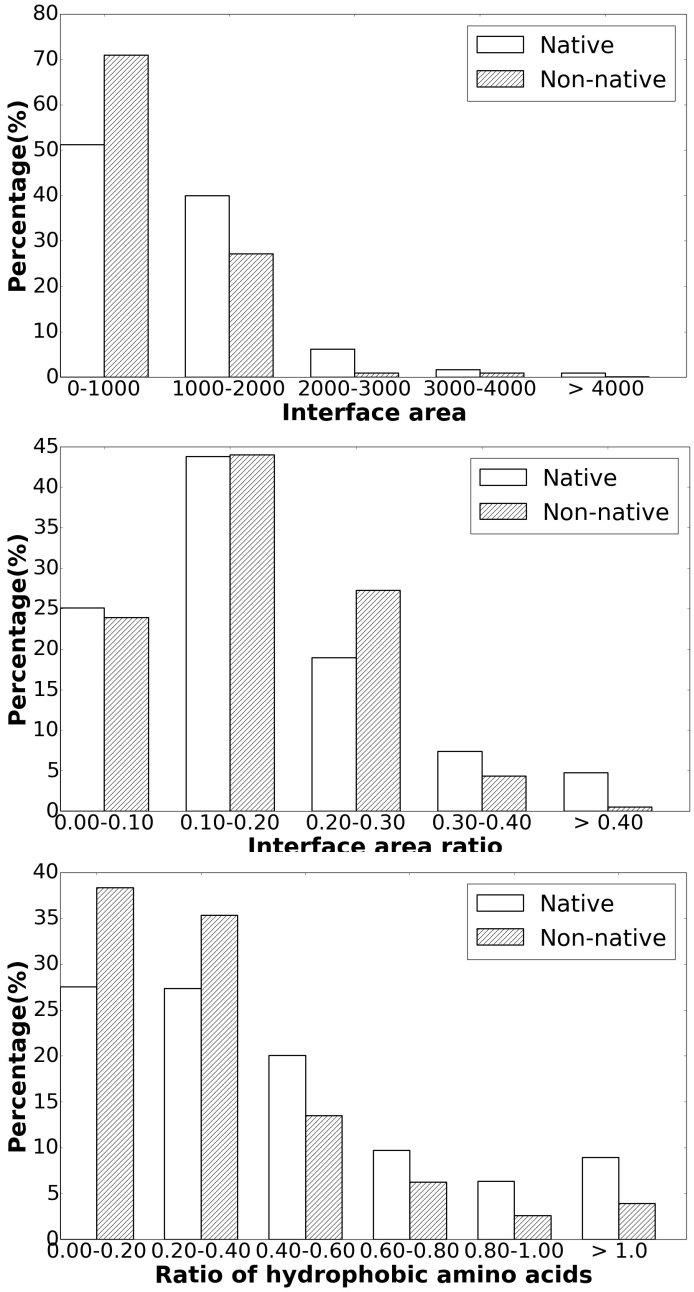
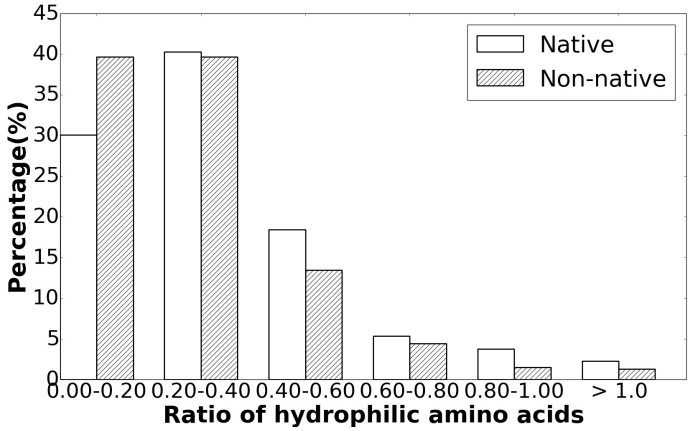
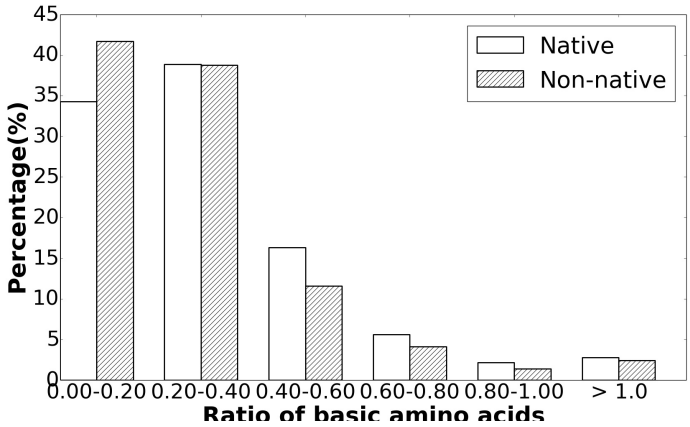


Figure 6.1: Figure shows the distribution of the first 3/7 features/interaction properties computed over positive/native and negative/non-native instances in bar diagrams. The x axis represents the value range for the properties, and the y axis shows the percentage of instances that fall within a particular value range. Positive and negative instances are shown in white and diagonal lined bars, respectively .





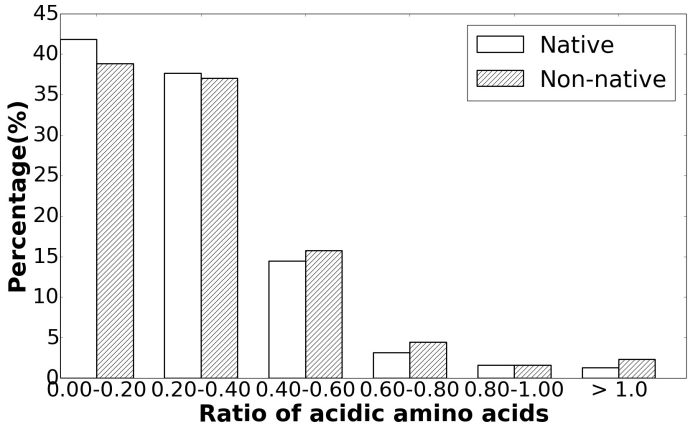


Figure 6.0: (Continued) Figure shows the distribution of the other 3/7 features.

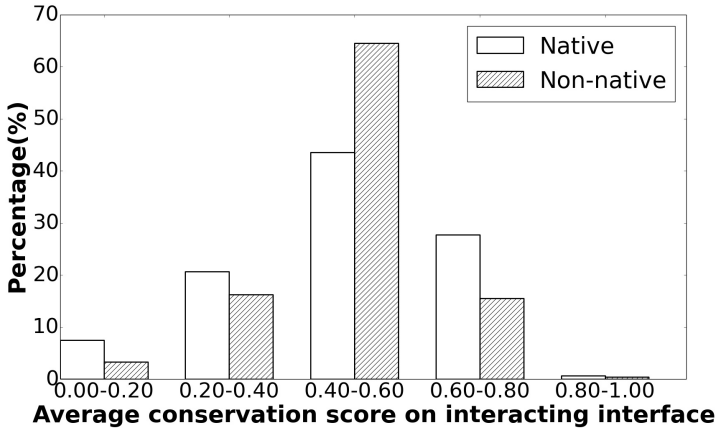
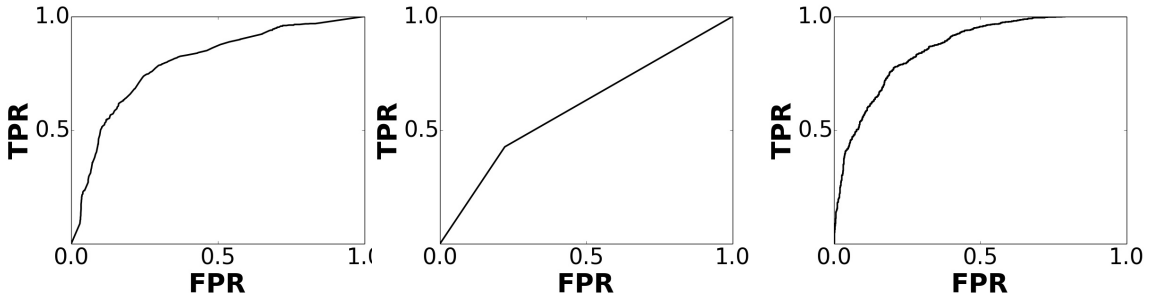


Figure 6.-1: (Continued) Figure shows the distribution of the last of the 7 features.

As shown in Figure 6.0, the J48 tree and SVM with polynomial kernel perform worse among all models. It is worth noting that we have tuned the parameters of each of the model to obtain the best performance. The performance of bagging and random forest tree models grow as the number of iterations I and number of trees T increase. However, the model complexity also grows with for increase of each and hence raises a possibility of model overfitting. Overfitting is an undesirable scenario in any machine learning model where the training error minimizes with the increase of the generalization error.

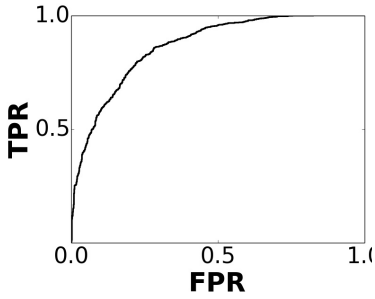
After this detailed comparison in terms of performance, timing, and model complexity, we have chosen bagging with $I = 10$ and J48 as the base classifiers for integration into idDock. The AUC of this model is 0.86 which is very much comparable with the other more complex models shown in Figure 6.0. An analysis of the variance of the area under the ROC for 10 fold is provided in the Appendix.



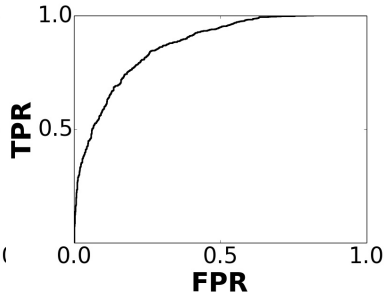
Decision Tree(J48)
AUC = 0.79

SVM (kernel=polynomial)
AUC = 0.60

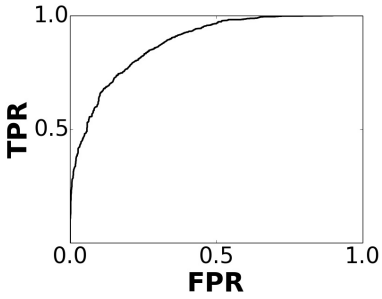
Bagging(J48, I = 10)
AUC = 0.86



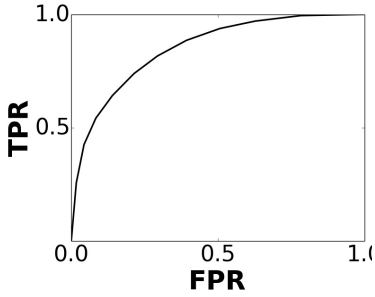
Bagging(J48, I = 20)
AUC = 0.86



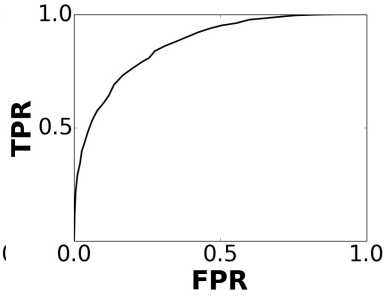
Bagging(J48, I = 50)
AUC = 0.87



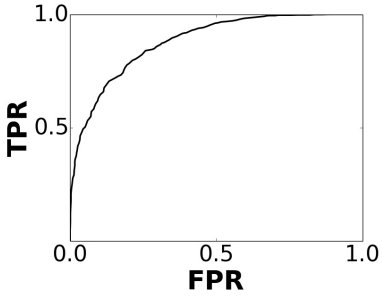
Bagging(Random Forest, T = 30)
AUC = 0.88



Random Forest(T = 10)
AUC = 0.84



Random Forest(T = 30)
AUC = 0.87



Random Forest(T = 100)
AUC = 0.88

Figure 6.0: ROC curves are shown for the different machine learning models. FPR stands for false positive rate and TRP for true positive rate. I and T stands for number of iterations and number of trees respectively. An area under the curve of 1.0 represents a perfect prediction while an area under the curve of 0.5 represents random guess.

6.2 Applications of idDock

idDock is implemented in C/C++ and run on ARGO. Testing is conducted on 15 known protein dimers as in table 2.1. On each testing system, idDock is run until either $10,000$ dimeric configurations are obtained or 7-days of CPU time have passed. This is repeated independently 5 times to account for probabilistic variance in the obtained results. Two different sets of analyses are conducted. First, the proximity of the configuration closest to the known native structure is reported for each system and compared to similar values reported by other state-of-the-art methods; other methods are also run 5 times in order to estimate variance of the results. Second, a more detailed analysis is conducted that looks at the relationship between energy and lRMSD to the known native structure to determine the likelihood of selecting a native-like configuration from energy-based selection mechanisms.

6.2.1 Comparative Analysis

Here we summarize the performance of idDock on each testing dimer in terms of the lowest lRMSD over all configurations to the known native structure. The same value is obtained for other state-of-the-art methods from published data or from data we have obtained by running methods available in software or web server form. Methods from other labs to which we compare idDock are FTDock-pyDock [77], ClusPro [27] and ZDock [59]. ZDock, ClusPro and pyDock are leading energy-driven methods. For completeness, we also compare idDock to one of our previous works, HopDock [5] for comparison. In HopDock, no machine learning model is integrated in the BH-based search, and a simple in-house energy function composed of van der Waals, electrostatic, and hydrogen bond terms is used. Each method is run 5 times on each protein system in order to estimate the variance between runs. It is worth noting that ZDock and ClusPro are deterministic methods.

Figure 6.1 summarizes the analysis comparing the lowest lRMSDs to the known native structure obtained from idDock and other state-of-the-art methods. Each method, idDock included, is run 5 times. The lowest lRMSD from each run of each method is recorded, and the range of lowest lRMSD values from all runs of all methods is plotted in gray for each of

the test dimers. The range of lowest lRMSD values from the 5 runs of idDock is highlighted in red line. Figure 6.1 clearly shows that even when considering variance between runs of all methods, idDock is one of the top performers in terms of the proximity to the known native structure for each of the test dimers. While idDock may not achieve the lowest lRMSD in each dimer (idDock achieves the lowest lRMSD on 5 dimers), the range of lowest lRMSDs obtained from different runs of it is in the bottom half of the range obtained from representative state-of-the-art methods.

The results summarized in Figure 6.1 effectively make the case for the contribution of the machine learning model in a probabilistic search algorithm such as idDock. On some dimers, energy-driven methods are far off from the native structure (e.g., 1WWW, 2HQS, 1G4Y and 1G4U), whereas idDock is able to report near-native structures on most of systems (except 1ZHI and 1G4Y, where the lowest lRMSD obtained from idDock is slightly higher than 5Å). Taken together, this analysis supports the claim that energy alone is not sufficient to drive a probabilistic search to the native structure.

6.2.2 Selection-based Analysis

In the following we take a closer look at the performance of idDock compared to other methods, particularly to understand what proximity to the native structure would be obtained by energy-based selection techniques. The baseline energy-based selection technique sorts configurations by energy, from lowest to largest, and then reports where in the sorted ordering the configuration with the lowest lRMSD to the native structure is found. This value is known as rank.

Figure 6.2 summarizes this comparative analysis in terms of rank. As before, each method, idDock included, is run 5 times, and the rank of the configuration with the lowest lRMSD to the native structure is recorded. For idDock, this configuration is found by sorting sampled configurations by their FoldX energies. For other methods, we rely on reported ranks. Methods such as pyDock and ZDock report rank. Since ClusPro provides top clusters with a total of 100 top structures, the rank we report for ClusPro is that within

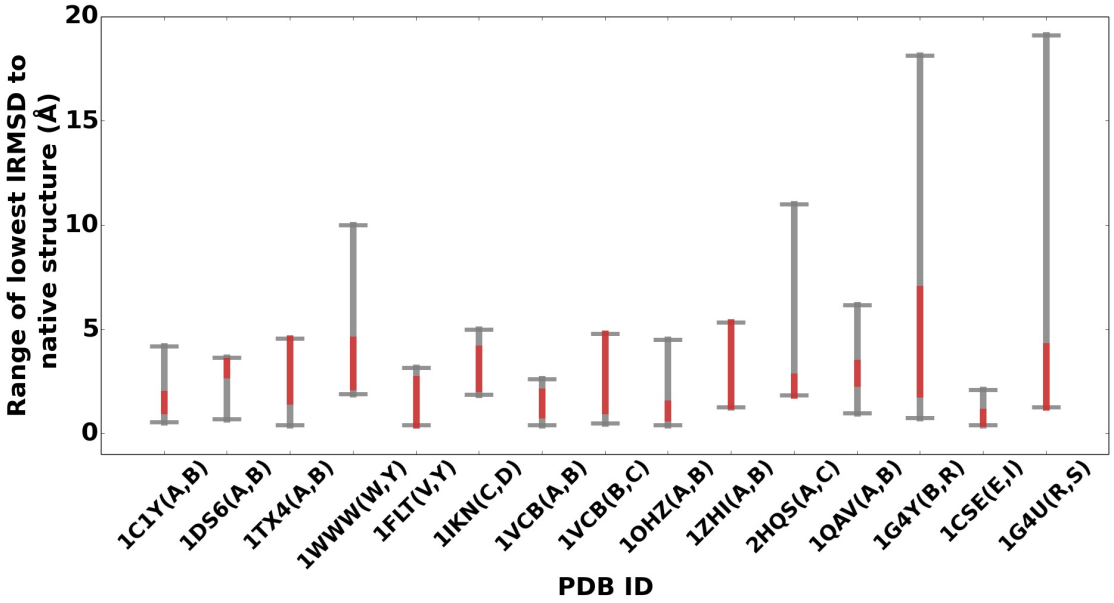


Figure 6.1: Comparison of idDock to other state-of-the-art methods in terms of lowest lRMSD to the native structure. Each method is run 5 times and lowest lRMSDs obtained from all runs of all methods are combined to obtain a range, drawn in gray. The range of lowest lRMSDs from all runs of idDock is highlighted in red.

the reported structures.

Rather than reporting the absolute position in this ordering of the lowest-lRMSD-to-native configuration, we report a normalized location, dividing by the total number of configurations. This allows a fair comparison across the various dimers and various methods, which report a different number of their top solutions. In the case of idDock, runs on some of the larger systems were terminated after 7 days. Cases where a rank of 100% is reported indicate those where the lowest-lRMSD to the native structure is beyond a stringent threshold of 5Å. The normalized ranks of all 5 runs of all methods are combined for each system, and their range is drawn in gray in Figure 6.2. The average rank obtained over 5 independent runs of idDock is indicated through a red circle.

Figure 6.2 shows that on 9/15 dimers, idDock shows comparable or better rank than other methods. These numbers include 4 dimers where three other methods are unable to find any solutions with lRMSDs less than 5Å. While not shown in detail, HopDock performs worst in terms of rank; this is due to a very simple energy function designed to guide the search in this method. For 7/15 of the dimers, the lowest-lRSMD idDock-obtained configuration is among the top 20% of energy-sorted configurations. For some dimers, such as those with PDB ids 1OHZ, 1VCB(A, B) and 1CSE, a normalized rank of close to 0 is reported, which means that the lowest-energy configuration obtained by idDock is indeed the one with the lowest lRMSD to the native structure. A detailed analysis showing the mean, average and variances lRMSD of all 5 runs and the actual average rank are provided in appendix section. Taken together, these results indicate that the machine learning model is effective at guiding the search in idDock towards the native structure, often working in concert with the energy function (a case-based analysis of this is provided below and in section 6.2.4).

We provide some more analysis as to why on some systems energy is a good selection criterion and on others it is not. Figure 6.3 plots FoldX energy versus lRMSD to the native structure for 6 selected dimers from a single run. Two dimers have been selected among those with very low rank; on these systems, energy is a good selection criterion.

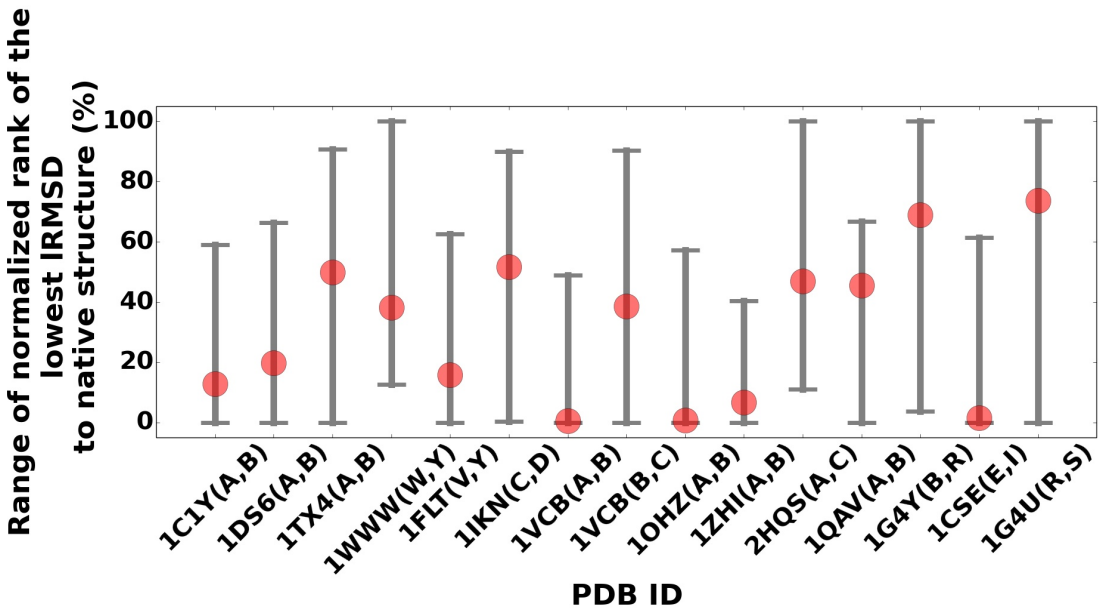


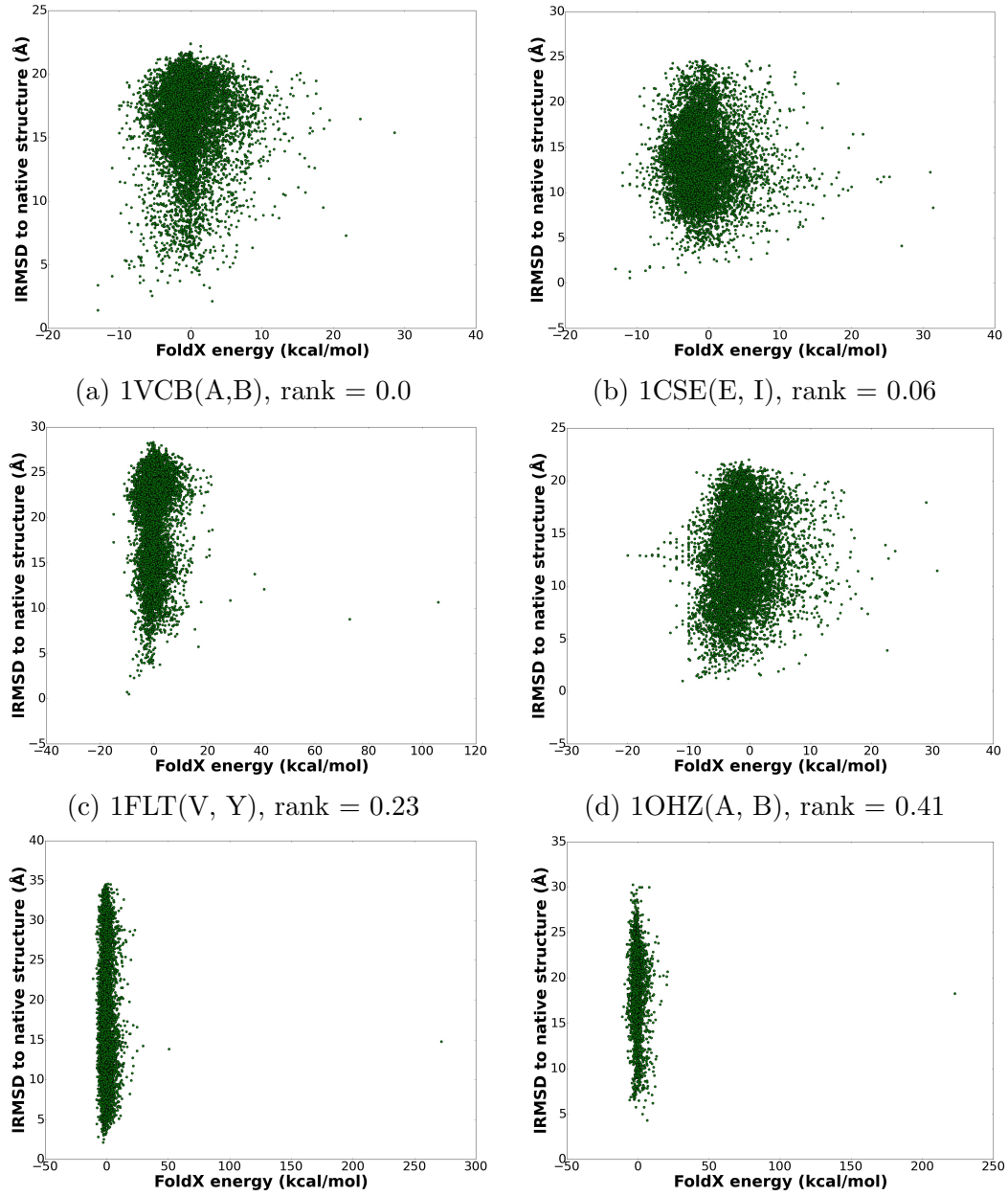
Figure 6.2: Normalized rank, measured as described above, is obtained from all runs of all methods and drawn as a range in gray. The average normalized rank over 5 independent runs of idDock is highlighted as a red circle.

As Figure 6.3(a)-(b) shows, this is because there is some correlation between interaction energy and lRMSD at the lower values. Figure 6.3(c)-(d) shows the relationship between energy and lRMSD for two systems with higher ranks. Again, some correlation is observed. No correlation is found for the two systems shown in Figure 6.3(e)-(f), which are selected among those with very high rank. On these systems, energy is not a discriminant.

In Figure 6.4 we draw some of the lowest-lRMSD or lowest-energy configurations for selected systems from a randomly selected run of idDock. Figure 6.4 superimposes these configurations over the corresponding native structure and draws them using Visual Molecular Dynamics (VMD) [10]. The chains are drawn in different colors, and the native structure is drawn in transparent. The top and middle panel in Figure 6.4 shows the lowest-lRSMD configurations for the 6 selected systems. The FoldX interaction energy and respective rank of the lowest-lRMSD configuration on each of these systems is also shown. The bottom panel in Figure 6.4 shows the lowest-energy configuration, instead, on 3 of the 6 selected systems. The lRMSD to the native structure of these configurations is also shown. On each of these 3 systems, the lowest-energy configuration is very far away from the native structure, including 1CSE (E, I), where a rank of 0.06 is found.

6.2.3 Timing Profile

This section analyzes the timing profile of idDock. We report the percentage of time each algorithmic component/operator in idDock needs using Gprof [81], a performance analysis tool for Unix-based applications. Reported values here are based on one single run and a randomly selected dimer (similar values are observed over different runs and different systems). Figure 6.5 shows that the local improvement operator, which makes use of the FoldX energy function, takes almost 80% of the running time. This time is spent on expensive energy evaluations. This is due to the fact that we employ a detailed and sophisticated energy function, FoldX, which on an average takes about 0.39 seconds per configuration. Moreover, to generate one local minimum, idDock can iterate over $k * m = 1,000$ energy evaluations. These evaluations are the main contributor to a running time of 7 CPU days



(a) 1VCB(A,B), rank = 0.0

(b) 1CSE(E, I), rank = 0.06

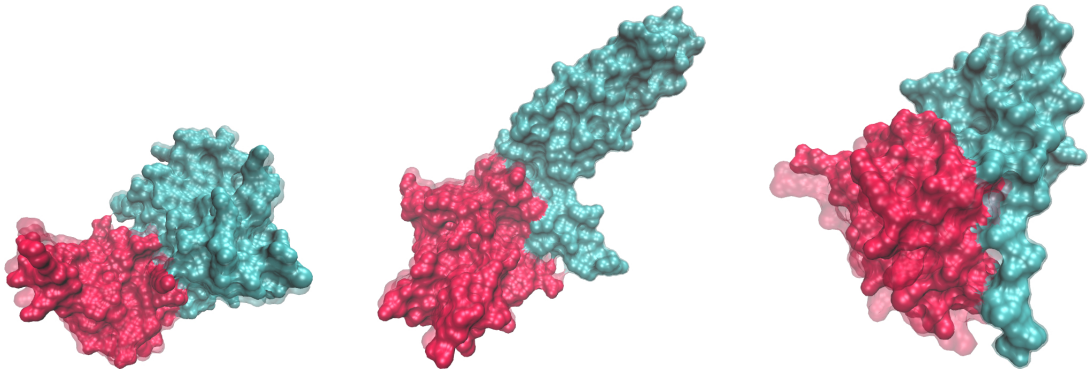
(c) 1FLT(V, Y), rank = 0.23

(d) 1OHZ(A, B), rank = 0.41

(e) 2HQS(A, C), rank = 16.61

(f) 1G4U(R, S), rank = 97.01

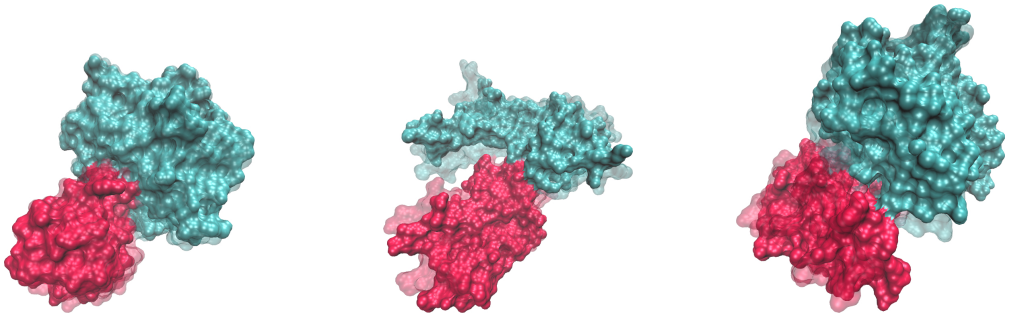
Figure 6.3: 6 systems have selected on which to plot the FoldX interaction energy vs. lRMSD to the native structure for idDock-generated configurations. The two systems in (a)-(b) have very low rank, followed by higher-rank systems in (c)-(d). The two systems in (e)-(f) have very high rank.



1C1Y(A,B)
1.4Å,−15 kcal/mol, 0

1FLT(V,Y)
0.4Å, −9.3 kcal/mol, 0.23

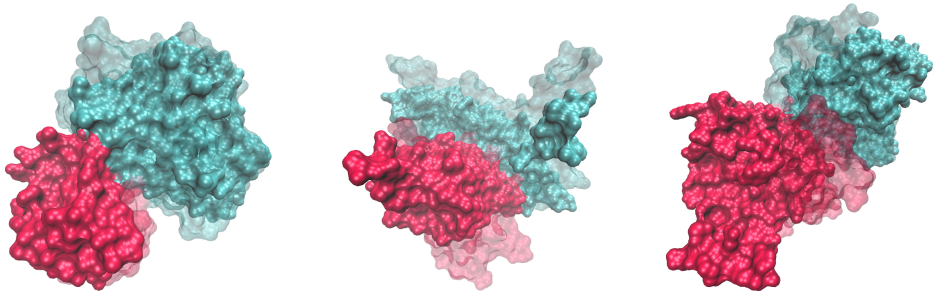
1VCB(A,B)
1.4Å, −13 kcal/mol, 0



1CSE(E,I)
0.5Å, −11 kcal/mol, 0.06

1WWW(W,Y)
4.5Å, −2.7 kcal/mol, 16.27

1TX4(A,B)
2.4Å, 4.5 kcal/mol, 90.62



1CSE(E,I)
10.74Å, −12 kcal/mol

1WWW(W,Y)
16.15Å,−13 kcal/mol

1TX4(A,B)
20.18Å, −12 kcal/mol

Figure 6.4: Top and middle panels draw the lowest-lRMSD to the native configuration for 6 selected systems and superimpose it over the corresponding native structure and lRMSD, FoldX interaction energy and normalized rank are shown under each sub figure. Bottom panel draws the lowest-energy configuration for 3 of the 6 selected systems and lRMSD and FoldX energy are shown. Chains are drawn in different color, and the native structure is drawn in transparent.

on some of the largest dimers. On the other hand, only 8.0% of the running time is spent on the perturbation operator, which includes generating a random configuration and testing the relevancy of it through the predictive machine learning model. The rest of the time is spent on miscellaneous tasks like I/O operations, vector manipulations, and so on.

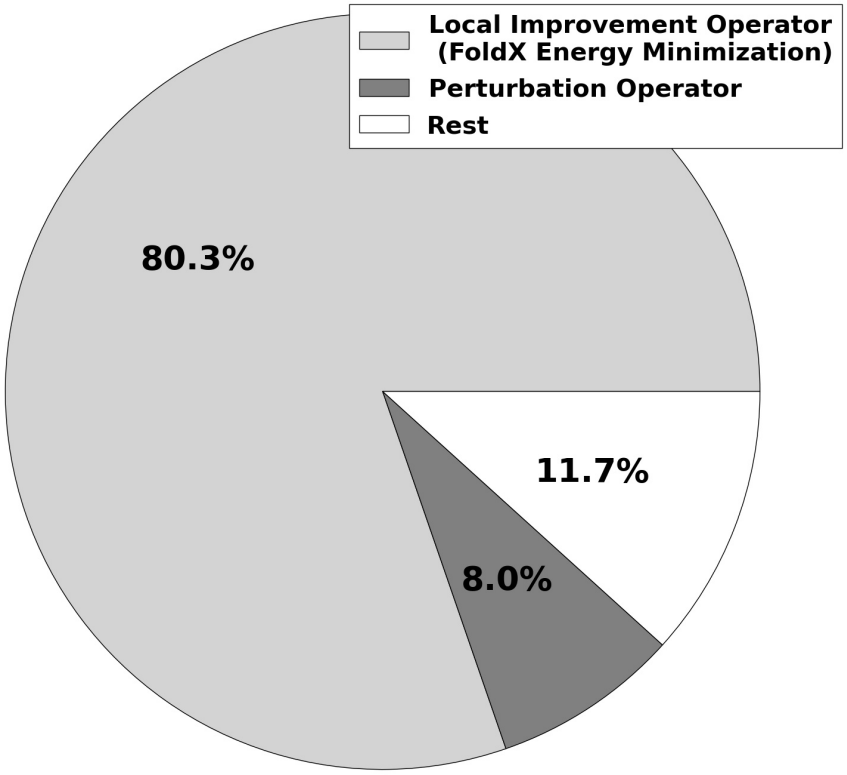


Figure 6.5: Timing profile of idDock is shown in a pie diagram. Different wedges show the percentage of time spent on different algorithmic components/operators in idDock.

6.2.4 Contribution of Machine Learning and Energetic Optimization: Protein System-Based Analysis

Here we provide a few more details on why idDock succeeds or fails on certain systems. We focus on the relationship between the ensemble learner and the minimization with FoldX, as these two can work concertedly to lower the lRMSD to the native structure or against eachother. We focus on four systems: one where the trained model performs well but nonetheless the energy function drives idDock away from the native structure; one where the trained model is inaccurate but the energy function nonetheless drives the search towards the native structure; one where both the trained model and the energy function work concertedly and lead idDock to the native structure; and one where both fail.

The first case is illustrated by the dimer with PDB id 1WWW, where the lowest lRMSD to the native structure is 4.5Å. On this system, the learned model performs well and evaluates as true configurations that have lRMSD to the native structure lower than 4.5Å. However, the energy function in the local improvement operator drives these configurations away from the native structure, resulting in an lRMSD higher than what the trained model would have reported in isolation. For instance, cases are found where the perturbed configuration that passes the stringent test of the learned model is less than 3.5Å in lRMSD to the native structure, but the minimization with FoldX modifies the configuration to one with higher lRMSD to the native structure.

The second case is illustrated by the dimer with PDB id 1VCB(A, B), where the lowest lRMSD to the native structure is 1.4Å. The particular configuration that achieves this lRMSD is obtained by minimizing a perturbed configuration that has passed the learned model but has lRMSD to the native structure of 6.40Å. This is an example where the energy function drives towards the native structure very effectively; the minimization lowers the interaction energy from 90.02 to -13kcal/mol. This represents an ideal case, where the best solution is obtained through energetic refinement.

The third case is illustrated by the dimer with PDB id 1FLT(V,Y), on which the learned

model and the energy function perform in concert with each-other. The configuration with the lowest lRMSD to the native structure is obtained from a perturbed configuration that has passed the learned model and has lRMSD of 3.61Å to the native structure. The minimization further lowers this lRMSD to 0.4Å, which is a significant improvement of 3Å.

The last case is illustrated by the system with PDB id 1G4Y(B,R) where neither the learned model nor the energy function are able to drive idDock towards low-lRMSD configurations. On this system, idDock obtains a lowest lRMSD to the native structure of 5.9Å.

Chapter 7: Class-specific Models of Interaction Interfaces

In idDock, a generalized predictive model is used to bypass energy evaluations. Here we investigate whether this predictive model can be further improved when restricted to specific functional classes of protein-protein assemblies. An interesting direction of research in machine learning for protein-protein docking is emerging that pitches baseline models [36, 82], such as the ones we developed and described in Chapter 6 against partner-specific models [83]. The latter are motivated by the fact that many proteins such as hubs that participate in several assemblies use different interaction interfaces with different protein partners. This direction of research is promising and has shown some improvement against baseline models, but one of its disadvantages is the current lack of structural, partner-specific data. In this chapter we pursue a new line of specialized models for which there are ample data for training. Essentially, datasets are grouped by their known functional classes, and models are built for each functional class. We investigate here two different ways, binary vs. real-valued, of constructing class-specific models and present some of our findings on integrating such models in protein-protein docking algorithms such as idDock.

7.1 Class-specific Predictive Models

We focus on 5 different functional classes. The datasets for each class are detailed in Section 7.1.1.

We then summarize two different strategies to build feature vectors. In Section 7.1.2 we present our results on models built for each class using the 7 real-valued features also employed by the baseline, general model used in idDock. In Section 7.1.3, we describe our second attempt on building models with features obtained from contrast set mining. In Section 7.1.4 we compare these two different strategies. In Section 7.1.5 we present some of

our findings on integrating class-specific models into idDock.

7.1.1 Obtaining Dataset for Function-specific Protein Class

We have extracted structures of protein dimers 5 different functional classes from the PDB. These classes are: Signaling, Transcription, Hydrolase Inhibitor, Growth Factor, and Immune. These classes have been defined according to PDB deposition. The classification is found in the PDB Format "HEADER" and under 'mmCIF' keyword search. The class is assigned during processing using a controlled vocabulary and is based on a comparison against the UniProt [84] sequence database. It is worth noting that wet-laboratory depositors of protein structures verify their class assignment using the PDB's available list of classes. A new functional class can also be added to the list, if it is annotated in other databases, such as SCOP [85].

We have performed the following PDB query to obtain dimeric proteins of each of these 5 classes and remove structures with 90% or higher sequence similarity:

1. From the top menus on the RCSB PDB home page, we select 'Search'>>'Drilldown Search'
2. Then we select 'Polymer Type'>>'Protein'
3. In the 'Query Refinements' section, we select 'Protein Stoichiometry'>>'Heteromer'
4. Again in the 'Query Refinements' section, we select 'Protein Stoichiometry'>>'Hetero 2-mer' to make sure all the proteins in the query result in dimer.
5. At the bottom of the 'Query Refinements' section, we select 'Refine Query with Advanced Search' to select function-specific proteins.
6. We click on 'ID(s) and keywords'>>'mmCIF keyword search (Classification). For example: to select 'Signaling' class protein we type Signaling in the corresponding box.
7. Then we select 'Submit Query'

8. Using the 'Reports' menu one can generate a quick report or a custom report of the fields that are of interest to the user.

Table 7.1 shows the total number of proteins obtained in this manner for each class.

Table 7.1: 5 functional classes and the number of proteins obtained from the PDB for each class are shown here.

Functional Class	Size of the dataset
Signaling	61
Transcription	45
Growth Factor	64
Immune	234
Hydrolase Inhibitor	171

For a given functional class, the positive dataset consists of proteins extracted from the PDB for that class. The negative dataset is composed of all the positive instances (proteins extracted from the PDB) for the other functional classes.

7.1.2 Predictive Machine Learning Model for Function-specific Protein Class: Real-valued Features

The interaction interface of an instance is first computed as in Chapter 6, and the interface is converted into a 7-entry real-valued vector, also described in Chapter 6. We have again employed the weka software package [80] to train various classifier models on the positive and negative datasets. Performance is recorded in a 10-fold cross validation setting for the purpose of comparison. The trained models are entropy-based Decision Tree (J48 implementation in weka), Random Forest(RF), Support Vector Machine(SVM), Bagging (with J48 and Random Forest), Boosting(AdaBoost implementation), Regression, Multilayer Perceptron.

7.1.3 Predictive Machine Learning Model with Mining Contrast Dataset for Function-specific Protein Class: Binary-valued Features

Here we pursue a second approach and employ Contrast Mining to identify contrasting features between the different functional classes. We employ the Search and Testing for Understandable Consistent Contrasts (STUCCO) strategy [86]. We briefly summarize STUCCO below, and then show its application by incorporating the rules it derives into predictive models.

Search and Testing for Understandable Consistent Contrasts (STUCCO)

We need two concepts to understand STUCCO, *Contrast-set* and *support* of a contrast-set. These are described in detail in [86]. Let $A_1, A_2 \ldots A_k$ be a set of k variables or attributes. Each A_i can take values from the set $V_{i1}, V_{i2} \ldots V_{im}$. A contrast set is an attribute-value pair defined on groups $G_1, G_2 \ldots G_n$. The support of a contrast set with respect to group G is the percentage of attributes in G where the contrast-set is true.

The goal of STUCCO is to find all the contrast-sets whose support differs significantly through all the groups. This is done as follows: the algorithm treats the learner as a tree search problem where the root node is an empty contrast set. Children of an internal node are generated by specializing the set through the addition of one more additional term or attribute. To avoid duplicacy, a canonical ordering of attributes is maintained. Children are formed, terms are added in a particular ordering. The search is conducted in a breadth-first manner as follows:

Given all nodes at a particular depth, the algorithm scans the database and counts the support for each group based on the following conditions:

- If each node is significant and large.
- If it should be pruned.
- If children should be generated.

After obtaining all significant contrast sets, results are reported based on statistical significance. The support count calculation is based on the null hypothesis, which states that the support is equal for all membership. The support count for each group is a frequency value, which can be analyzed through a contingency table where each row represents the truth value of the contrast set and each column shows the group frequency for that particular membership. If there is a difference between the frequency of a contrast set and the null hypothesis, the algorithm determines whether the difference is statistically significant or not. This is usually determined thorough the chi-square test.

Pruning of a node is based on following decision:

- Minimum deviation size: The maximum difference between the support of any two groups and this parameter is user-defined. We employ 10% as the minimum deviation size with 95% confidence interval.
- Expected cell frequency: The frequency of the cell in the contingency table needs to be above a certain threshold, otherwise the validity of the chi-square test is violated.
- χ^2 bounds: This is an upper bound employed in the distribution of a statistics calculated when the null hypothesis is true. Nodes are pruned when the boundary value is not met.

The whole outcome of the algorithm is a subset of rules that essentially contrast one set from the another. We employ these rules to build our binary feature vector for each instance in the positive and negative datasets of each functional class.

Binary Feature Construction

For each instance, we determine whether a particular rule reported by STUCCO is met or not.

For instance, let us suppose that for class A STUCCO generates n different rules. Let us also suppose that $rule_i$ combines m different features. We can represent $rule_i$ as "$f_{i1} - low_{i1} - high_{i1}$" & "$f_{i2} - low_{i2} - high_{i2}$" ... "$f_{im} - low_{im} - high_{im}$", where each f_{im} represents

the m^{th} feature of rule i, low_{mi} is the lower range of the value of feature f_{im} value, and $high_{im}$ is the higher range of the value of feature f_{im} in that particular rule. Now we check each instance of class A on whether that particular instance contains all the feature values of $rule_i$. If the instance exhibits all, then we place a "1" or a "0" otherwise. In this way the dimension of the feature vector for each instance is equal to the total number of rules generated by STUCCO (n in this example). These binary features can be employed to represent any instance and train machine learning models.

7.1.4 Comparative analysis between real-valued and binary-valued feature models

We show here the results obtained from different classifiers, such as Support Vector Machine (SVM), Decision Tree, Bagging, Random Forest, AdaBoost, Multi-layer Perceptron (MLPerceptron), Linear and Logistic Regression. Where applicable, we have tuned the parameters to obtain the optimal performance of each classifier. We measure F-measure, precision, recall, but report here only area under the ROC (auROC).

Tables 7.2 - 7.6 show a detailed comparative analysis between the two settings, real-valued vs. binary feature representations, for each of the five functional classes. The first column in each table shows the classifier. The next column shows the parameter settings for that classifier, and the next two columns show auROC for the real-value and the binary-value settings, respectively.

We see that the performance of the classifiers in the real-valued setting is better than that in the binary-valued setting. This is due to the fact that for some of the protein classes, the number of generated features in STUCCO is very low (the growth class is an exception). We also observe that on some of the cases, non-linear classifiers, such as MLPerceptron and logistic regression perform reasonably well in the binary setting in certain functional classes (see growth and immune classes). This is due to the fact that binary features exhibit the well known "XOR" problem when the data are not linearly separable [87].

Taken all these together, we make the following observations: 1) in order to employ

models build from STUCCO, more features need to be extracted for the different functional classes. Adding more feature along with STUCCO generated features may provide better results. New features can be added by ranking the original 7 features and taking top n features to be added. 3) Due to possible non-linear separability, more complex models are needed to obtain optimal performance for binary-valued features. This is an interesting direction but beyond the scope of the work presented in this thesis.

Classifier	Parameter Settings	$auROC_{real}$	$auROC_{binary}$
SVM	C=0.01, kernel = poly	0.5	0.62
SVM	C=0.1, kernel = poly	0.55	0.69
SVM	C=1.0, kernel = poly	0.6	0.69
SVM	C=10.0, kernel = poly	0.63	0.69
SVM	C=0.01, kernel = RBF	0.5	0.5
SVM	C=0.1, kernel = RBF	0.5	0.5
SVM	C=1.0, kernel = RBF	0.49	0.62
SVM	C=10.0, kernel = RBF	0.57	0.69
Decision Tree (DT)	Weka default	0.84	0.68
Bagging (with J48)	I = 10	0.92	0.69
Bagging (with J48)	I = 15	0.92	0.68
Bagging (with J48)	I = 20	0.92	0.68
Random Forest	T = 10	0.93	0.68
Random Forest	T = 15	0.94	0.68
Random Forest	T = 20	0.94	0.68
Bagging (with RF)	I = 10	0.94	0.69
Bagging (with RF)	I = 15	0.94	0.68
Bagging (with RF)	I = 20	0.94	0.68
Adaboost	I = 10, Classifier = J48	0.93	0.68
Adaboost	I = 15, Classifier = J48	0.94	0.68
Adaboost	I = 20, Classifier = J48	0.94	0.68
MLPerceptron	Weka default	0.8	0.69
Linear Regression(Simple Logistic)	Weka default	0.67	0.67
Logistic Regression (Logistic)	Weka default	0.67	0.68

Table 7.2: Comparative analysis between the real-valued and binary-valued settings for the signaling class in terms of auROC. The parameter settings for each classifier are shown in second column, where an I = nr. of iterations, C = SVM parameter, T = nr. of trees, and weka default represents the default parameter setting used for that particular classifier.

Classifier	Parameter Settings	$auROC_{real}$	$auROC_{binary}$
SVM	C=0.01, kernel = poly	0.5	0.6
SVM	C=0.1, kernel = poly	0.56	0.59
SVM	C=1.0, kernel = poly	0.67	0.6
SVM	C=10.0, kernel = poly	0.68	0.6
SVM	C=0.01, kernel = RBF	0.5	0.53
SVM	C=0.1, kernel = RBF	0.5	0.53
SVM	C=1.0, kernel = RBF	0.49	0.6
SVM	C=10.0, kernel = RBF	0.64	0.56
Decision Tree (DT)	Weka default	0.83	0.59
Bagging (with J48)	I = 10	0.92	0.59
Bagging (with J48)	I = 15	0.93	0.59
Bagging (with J48)	I = 20	0.93	0.59
Random Forest	T = 10	0.92	0.59
Random Forest	T = 15	0.93	0.59
Random Forest	T = 20	0.94	0.59
Bagging (with RF)	I = 10	0.94	0.59
Bagging (with RF)	I = 15	0.94	0.59
Bagging (with RF)	I = 20	0.94	0.59
Adaboost	I = 10, Classifier = J48	0.93	0.59
Adaboost	I = 15, Classifier = J48	0.94	0.59
Adaboost	I = 20, Classifier = J48	0.95	0.59
MLPerceptron	Weka default	0.81	0.6
Linear Regression(Simple Logistic)	Weka default	0.69	0.59
Logistic Regression (Logistic)	Weka default	0.69	0.59

Table 7.3: Comparative analysis between the real-valued and binary-valued settings for the transcription class in terms of auROC.

Classifier	Parameter Settings	$auROC_{real}$	$auROC_{binary}$
SVM	C=0.01, kernel = poly	0.49	0.75
SVM	C=0.1, kernel = poly	0.64	0.77
SVM	C=1.0, kernel = poly	0.68	0.78
SVM	C=10.0, kernel = poly	0.69	0.78
SVM	C=0.01, kernel = RBF	0.51	0.51
SVM	C=0.1, kernel = RBF	0.51	0.73
SVM	C=1.0, kernel = RBF	0.51	0.75
SVM	C=10.0, kernel = RBF	0.66	0.77
Decision Tree (DT)	Weka default	0.66	0.79
Bagging (with J48)	I = 10	0.94	0.83
Bagging (with J48)	I = 15	0.95	0.83
Bagging (with J48)	I = 20	0.95	0.82
Random Forest	T = 10	0.94	0.82
Random Forest	T = 15	0.94	0.85
Random Forest	T = 20	0.95	0.85
Bagging (with RF)	I = 10	0.95	0.86
Bagging (with RF)	I = 15	0.95	0.85
Bagging (with RF)	I = 20	0.95	0.85
Adaboost	I = 10, Classifier = J48	0.94	0.84
Adaboost	I = 15, Classifier = J48	0.95	0.84
Adaboost	I = 20, Classifier = J48	0.95	0.84
MLPerceptron	Weka default	0.85 0.86	
Linear Regression(Simple Logistic)	Weka default	0.76	0.82
Logistic Regression (Logistic)	Weka default	0.77	0.84

Table 7.4: Comparative analysis between the real-valued and binary-valued settings for the Growth class in terms of auROC.

Classifier	Parameter Settings	$auROC_{real}$	$auROC_{binary}$
SVM	C=0.01, kernel = poly	0.5	0.62
SVM	C=0.1, kernel = poly	0.54	0.68
SVM	C=1.0, kernel = poly	0.62	0.68
SVM	C=10.0, kernel = poly	0.67	0.68
SVM	C=0.01, kernel = RBF	0.5	0.54
SVM	C=0.1, kernel = RBF	0.5	0.52
SVM	C=1.0, kernel = RBF	0.5	0.65
SVM	C=10.0, kernel = RBF	0.54	0.68
Decision Tree (DT)	Weka default	0.75	0.67
Bagging (with J48)	I = 10	0.84	0.7
Bagging (with J48)	I = 15	0.84	0.7
Bagging (with J48)	I = 20	0.85	0.7
Random Forest	T = 10	0.84	0.7
Random Forest	T = 15	0.84	0.69
Random Forest	T = 20	0.85	0.69
Bagging (with RF)	I = 10	0.86	0.7
Bagging (with RF)	I = 15	0.86	0.7
Bagging (with RF)	I = 20	0.86	0.69
Adaboost	I = 10, Classifier = J48	0.8	0.7
Adaboost	I = 15, Classifier = J48	0.8	0.71
Adaboost	I = 20, Classifier = J48	0.79	0.7
MLPerceptron	Weka default	0.71	0.7
Linear Regression(Simple Logistic)	Weka default	0.68	0.7
Logistic Regression (Logistic)	Weka default	0.69	0.7

Table 7.5: Comparative analysis between the real-valued and binary-valued settings for the Immune class in terms of auROC.

Classifier	Parameter Settings	$auROC_{real}$	$auROC_{binary}$
SVM	C=0.01, kernel = poly	0.5	0.6
SVM	C=0.1, kernel = poly	0.55	0.61
SVM	C=1.0, kernel = poly	0.62	0.61
SVM	C=10.0, kernel = poly	0.63	0.61
SVM	C=0.01, kernel = RBF	0.5	0.5
SVM	C=0.1, kernel = RBF	0.5	0.51
SVM	C=1.0, kernel = RBF	0.5	0.61
SVM	C=10.0, kernel = RBF	0.56	0.61
Decision Tree (DT)	Weka default	0.75	0.58
Bagging (with J48)	I = 10	0.87	0.6
Bagging (with J48)	I = 15	0.87	0.59
Bagging (with J48)	I = 20	0.87	0.59
Random Forest	T = 10	0.85	0.58
Random Forest	T = 15	0.86	0.58
Random Forest	T = 20	0.87	0.59
Bagging (with RF)	I = 10	0.88	0.59
Bagging (with RF)	I = 15	0.88	0.59
Bagging (with RF)	I = 20	0.88	0.59
Adaboost	I = 10, Classifier = J48	0.85	0.6
Adaboost	I = 15, Classifier = J48	0.85	0.6
Adaboost	I = 20, Classifier = J48	0.85	0.6
MLPerceptron	Weka default	0.77	0.62
Linear Regression(Simple Logistic)	Weka default	0.69	0.61
Logistic Regression (Logistic)	Weka default	0.69	0.58

Table 7.6: Comparative analysis between the real-valued and binary-valued settings for the Hydrolase Inhibitor class in terms of auROC.

7.1.5 Incorporating Function-specific Models in Docking

When investigating the incorporation of class specific models in stochastic optimization idDock, we discovered that the models provided very high rejection rates and made it very difficult for idDock to obtain configurations that were deemed native-like for further optimization. The result of more than 70% rejection rates were high computational demands to obtain even a few hundred configurations.

Therefore, we decided to pursue the following experiment. We select Bagging with J48 tree with iteration I to be 10, and we randomly select one native structure from one of our systems of study, PDB id 1DS6, which is a signaling protein. Next we generate 2000 configurations by perturbing the native structure. The perturbation is done as follows: randomly select one of the principle axes and perform a rotation by an angle sampled uniformly at random in the $[0°, 30°]$ range. The perturbation operator is applied onto native structure until $2,000$ configurations are obtained within < 10Åin lRMSD to the native structure. These 2000 configurations are then passed to the Bagging with J48 tree model trained on the functional class of the selected system (signaling in this case).

In Figure 7.1 we plot horizontal bars to show the rejection rate for a class-specific model. We divide the configurations into bins based on their lRMSD from the native structure. The x axis shows the rejection rate as a % for each bin. The bins are shown on the y axis.

This analysis allows making the following observations: 1) on an average, more than 75% configurations are rejected that are within $2\mathring{A}$ of the native structure. 2) More than 80% of configurations within 5Åof the native structure are rejected. This analysis highlights that class-specific models are very effective in the context of classification but are highly sensitive and thus not tolerant of approximations. However, in stochastic optimization, tolerance is key, as the goal of machine learning models is to identify promising configurations close enough to the native structure that then energetic minimization can further improve.

This simple yet effective analysis has alerted us to a very interesting challenge yet unrealized in the community for integration of machine learning models in stochastic optimization.

The observations made here lead to a new direction of research on how to balance prediction accuracy while allowing for enough error so machine learning methods are effective for guiding docking algorithms. Several directions present themselves, including the enrichment of training data with configurations sampled around native structures in ranges of tolerance needed of these models.

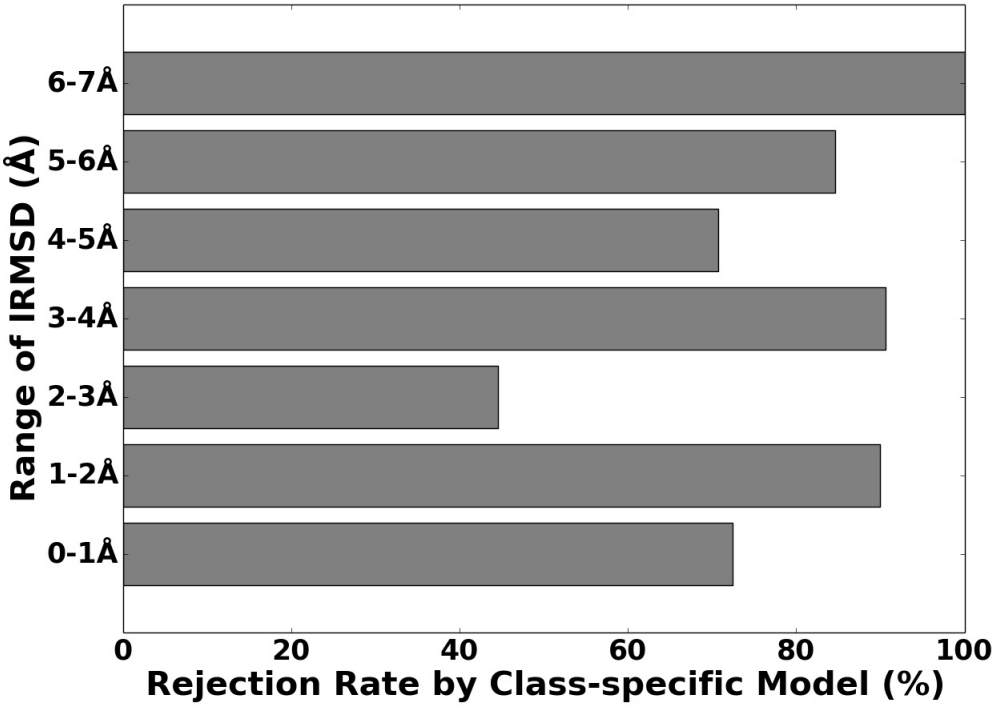


Figure 7.1: Horizontal bars show the rejection of a class-specific predictive model. The x-axis shows the rejection rate, and the y-axis shows the range of lRMSD to the native structure for randomly-generated configurations.

Chapter 8: Multimeric Docking: A Proposed EA-based Framework

This chapter investigates multimeric docking. The specific setting is as follows. 3d coordinates are provided for the $k > 2$ unbound protein units $U_1, U_2, \ldots U_k$. The units are assumed to interact with one another and form an assembly. The desired output is a set of near-native docked configurations As before, we focus on rigid-body docking; that is, no configurational changes occur in any of the units upon docking.

To the best of our knowledge, the only existing EA for multimeric docking is the Multi-LZerD method proposed in [54]. Multi-LZerD is very computationally demanding. The initial population it generates contains more than $50,000$ pairwise configurations. These are obtained via the geometry-driven method in [41]. The configurations are then clustered according to RMSD and a representative subset of 200 are then passed to the main multimeric component. The algorithm runs 3000 generations with a parent and offspring population size of 200 and 400, respectively. An expensive, custom-design energy function is used to evaluate configurations. Survival is based on fitness selection. The population of the last generation is passed through 2000 steps of MC-based energy minimization. Varied results are obtained on the ability to generate low-lRMSD configurations, depending on number of units and assembly size [54].

8.1 Components of Multimeric Docking Framework

Here we investigate a simpler EA for multimeric docking to gage the performance of a less computationally demanding algorithm.

8.1.1 Initial Population

The initial population contains configurations of all pairs of units. Our analysis shows that a random initial population does not help to converge the search. Therefore, our goal is to define a small but informative set of pairs for the initial population. This can be achieved in several ways. We have generated an initial population of size I in the following manner: we randomly choose one unit U_i as the base unit to be docked with another randomly-drawn unit U_j, where $j \neq i$ (U_j is the moving unit). Docking is achieved via evoDock, described in Chapter 3. The configuration is passed to the predictive machine learning model described in the context of idDOck in Chapter 6. Once a dimeric configuration passes the predictive model test, it is sent for energetic evaluation. Otherwise, the process begins anew.

8.1.2 Representation

Representation is an important component of any EA [88]. Here, each individual is represented through a set of triangles that represent the local coordinate systems of each of the moving units relative to an arbitrarily-designated base/reference unit.

8.1.3 Selection

Each newly-generated offspring configuration is added to the parent pool and competes for survival. The number of offspring generated equals that of parents. The offspring compete with parents to advance to the next level. Two selection mechanisms are considered, parental vs. survival selection. For parental selection, we have employed a stochastic selection technique to give equal chance to every parent to increase variation in the population. The survival selection mechanism is based on truncation selection.

8.1.4 Reproductive Operators

The reproductive operators and the selection mechanism should be chosen in a way so as to balance exploration and exploitation. Two types of perturbation operators are employed, mutation and expansion. The mutation randomizes the current transformation component.

This is achieved as follows: an angle θ is sampled uniformly at random in $[-15°, 15°]$, and an axis is chosen at random from the 3 principal axes x, y, and z. Rotation by θ along the chosen axis is applied once onto a randomly-selected moving unit in the selected configuration to get a new configuration. The mutation operator does not increase the number of docked units. Doing so is the objective of the expansion operator. This operator randomly selects a yet-unused moving unit and docks it onto an existing unit in the configuration.

Each operator is selected based on a dynamic probability scheme. At the beginning, we want to grow the configurations so as to explore more of the unexplored search space. Therefore, the expansion operator is selected with higher probability than the mutation operator. As the number of generations grows, more exploitation is expected of already explored regions. Hence, the mutation operator is selected with higher probability than the expansion operator. If a parent has already grown to a k-meric configuration, then only mutation is applied onto it.

8.1.5 Fitness Function

FoldX is used, as in idDock.

8.1.6 Population Size and Number of Generations

The main goal behind the designing and parameter selection for this framework is to apply a limited computational budget. Population size is 400, and the results reported here are on $100-400$ generations depending on the size of the protein assemblies considered here.

8.2 Applications of Multimeric Protein Docking

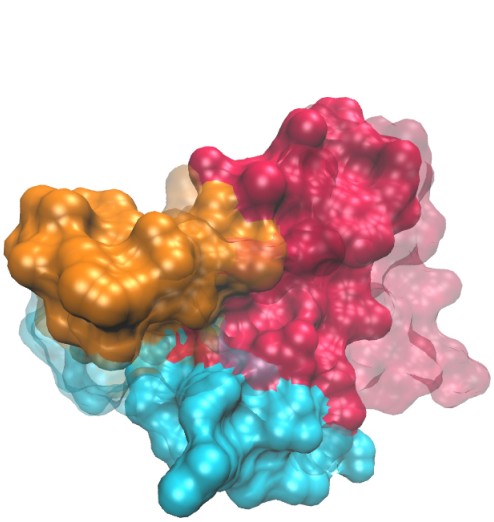
We report results on five multimers listed in Table 8.1. These systems are selected from systems already studied with the other two existing multimeric docking methods, CombDock [23] and Multi-LZerdD [54]. The last two columns show the lowest-lRMSD to the native structure obtained by CombDock and Multi-LZerD (reported from the last generation). The next column shows the lowest-lRMSD obtained by the simple EA described

above in the last generation, and the last column shows the lowest lRSMD obtained over all generations.

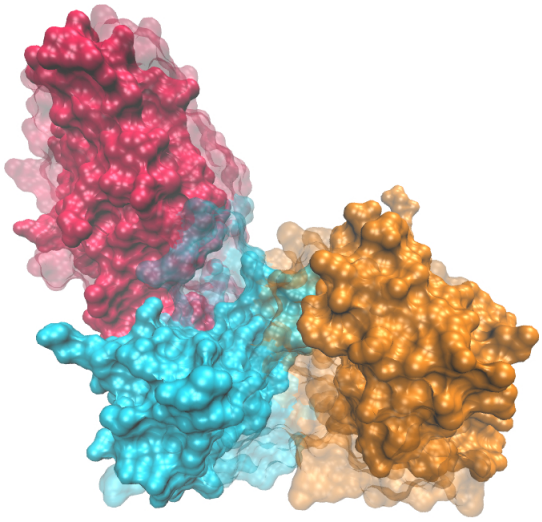
Table 8.1: Comparison on EAs for multimeric docking. NA implies data is not available.

PDB ID	Nrs of Units	Size (Nr. of atoms)	Functional Classification	CombDock (Å)	Multi-LZerD (lowest lRMSD in last generation) (Å)	Our (lowest lRMSD obtained in last generation) (Å)	Our (lowest lRMSD over all) (Å)
2AZE	3	2325	Transcription	0.79	0.73	19.39	18.30
1VCB	3	2601	Signaling Protein	0.55	1.09	10.34	7.28
6RLX	3	526	Hormone	9.52	NA	7.23	6.85
6RLX	4	697	Hormone	7.37	4.39	11.48	10.38
1LOG	4	3577	Isolectin	1.73	1.59	29.96	15.14

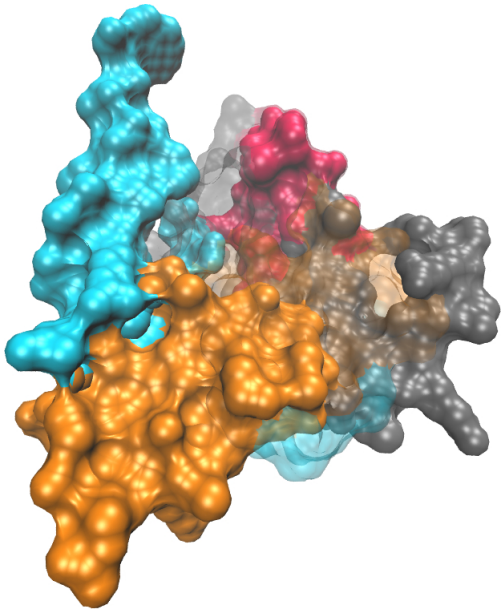
The results in Table 8.1 show that with a low computational budget the described EA generates configurations within $10 - 11$Å for more than half of the systems here. Performance deteriorates on assemblies with a higher number of units. Some of the best predicted configurations in the last generation are shown in 8.1. The chains are shown in opaque in different colors while the native is shown in transparent. A higher computational budget may improve lowest lRMSDs, but it is expected that further research into how to retain good structural features that emerge on incomplete configurations in early generations is more essential to improved performance.



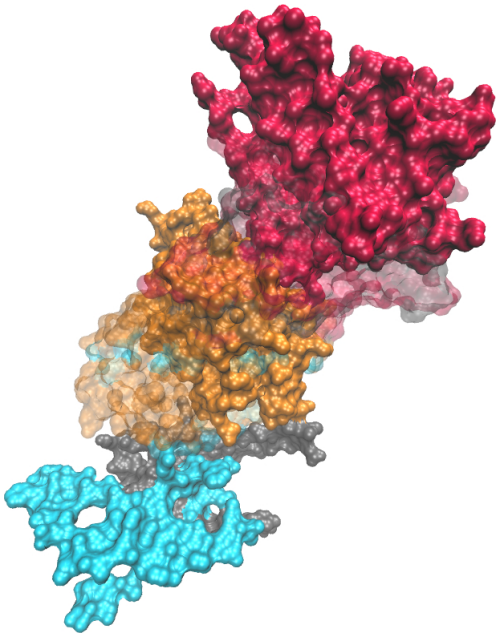
6RLX(A,B,C)
lRMSD = 7.23Å



1VCB(A,B,C)
lRMSD = 10.34Å



6RLX(A,B,C,D)
lRMSD = 11.48Å



1LOG(A,B,C,D)
lRMSD = 29.96Å

Figure 8.1: Some of the best predicted models in the last generation of EA-based multimeric docking. Chains are shown in different color and natives are shown in transparent. The lRMSD to native are shown under each subfigure. Figures are drawn in Visual Molecular Dynamics (VMD) [10]

Chapter 9: Conclusion and Future work

This thesis has made several contributions in proposing hybrid probabilistic approaches that integrate domain-specific insight into powerful stochastic optimization algorithms for the pairwise protein docking problem. Three different sources of domain-specific insight have been considered and integrated in docking methods. Evolutionary conservation stored in sequences of polypeptide chains is integrated in a geometry-driven method and shown to improve the performance of such methods for rigid-body pairwise docking. Qualitative information provided from wet-laboratory experts is integrated in a carefully thought-out, filter-based computational protocol to predict models of dimerization in GPCRs. Finally, machine learning models trained on known interaction interfaces are integrated for the first time in a stochastic optimization method. Additional contributions include integration of geometric complementarity in energy-driven methods, proposal of functional class-specific models, and evolutionary strategies for multimeric docking.

The work can be extended in few directions. The first direction is to incorporate more sophisticated and specific machine learning models. A possibly higher number of features characterizing interaction interfaces needs to be considered. However, as our investigation has shown, such models need to tolerate error so that they can allow stochastic optimization algorithms to identify configurations in the vicinity of the native structure for further energetic improvement. Some of our work on GPCR dimerization highlighted interesting Pareto-based analysis for decoy selection. The integration of such analysis in the current clustering-based techniques presents an interesting direction for future work in decoy selection.

The second direction of research concerns exploiting knowledge of prediction of interaction interface to design drug compounds. The machine learning model can be used to

identify which amino acids participate in a protein-protein interaction interface. This information can be valuable to guide the design of small molecule inhibitors that interact specifically with such amino acids. This direction has already begun to bear fruit [89], [90] by helping computational chemists design more effective, thermodynamically stable drugs to hinder signaling pathway generated upon binding between two proteins.

Finally, though not a major focus of this thesis due to the need to address outstanding challenges in pairwise docking, multimeric docking is expected to become more amenable in the near future. Near 30% of currently-deposited structures in the PDB are assemblies of more than two protein units. Since multimeric docking presents an exceptionally challenging combinatorial optimization setting, further research onto EAs for multimeric docking may prove more beneficial than extension of MC-based frameworks.

Appendix A: Appendix

A.1 Analysis of idDock Negative Dataset: Randomization of Positive Dataset

In this section we analyze the sequence similarity between two chains in randomized complexes that we have employed for negative dataset. For this experiment we have randomly selected 100 negative structures generated by randomizing the positive dataset and we measure similarity between two chains. The similarity is measured using protein-protein **B**asic **L**ocal **A**lignment **S**earch Tool [91] (BLASTp) server (`http://blast.ncbi.nlm.nih.gov/Blast.cgi`). A BLAST query enables a user to compare a query sequence with a library of sequences or a subject sequence to provide the similarity information. The basic working idea of blast is to detect local alignments between two protein sequences by computing all amino acid subsequences. The program then looks for the number of time and place in which these subsequences appear. It also looks for the closely matched subsequences between the query and subject. Each query is then scored by using a scoring matrix. For this work we have employed the default scoring matrix **BLO**cks **SU**bstitution **M**atrix (BLOSUM) [92] version 62 which employs statistical methods to obtain the similarity scores.

The x-axis of figure A.1 shows the sequence similarity between two chains and the y-axis denotes the frequency of the complexes for that similarity. A 0 sequence similarity means no significant similarity have been found between two chains and higher number reflects higher similarity. From the figure A.1 it can be seen that more than 55% complexes do not show any significant similarity and more than 40% chains shows $20 - 50\%$ sequence similarity. The rest higher similarity is cases where one chain is too small in terms of number of amino acids compared to the other chain.

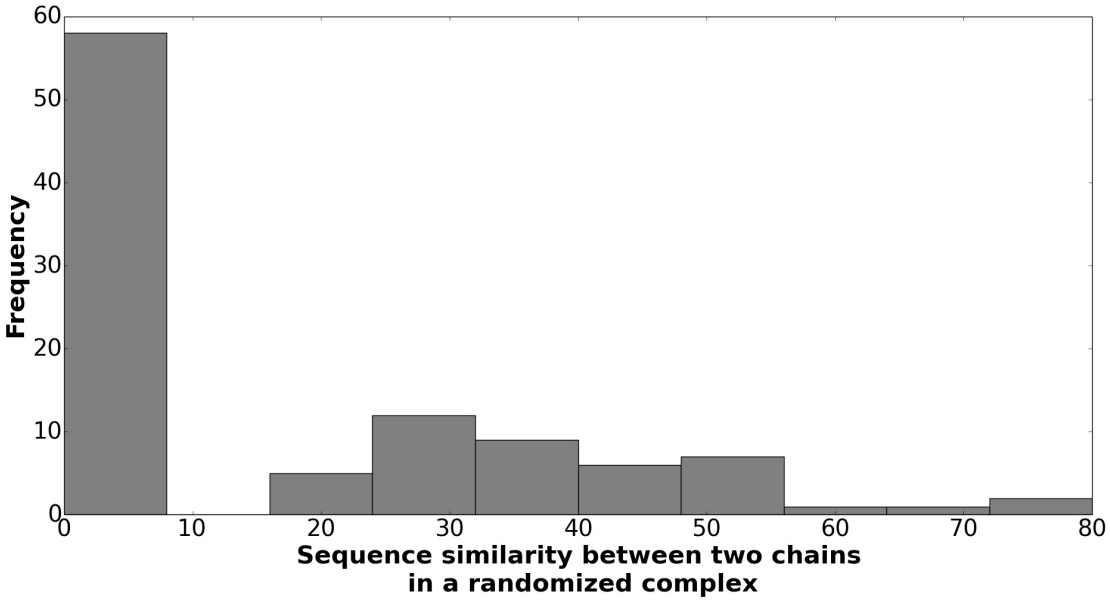


Figure A.1: Analysis of sequence similarity between two chains of a randomly generated negative complex. x-axis shows the actual similarity between two chains and the y-axis shows the frequency.

A.2 Variance Analysis of 10-fold Cross-Validation: Selection of Optimal Machine Learning Model in idDock

In this section we provide some detailed analysis on variance of area under the ROC (auROC) for each fold on some selected models of idDock. The reason for this analysis is to show the stability of each of the models. For this analysis we have selected 5 different models that we employed for choosing optimal model to be used in idDock framework. These are J48, SVM with polynomial kernel, bagging with J48 with 10 iterations, bagging with random forest with number of trees to be 10 and random forest with number of trees to be 10. Weka experimenter have been employed to obtain the auROC for each fold as follows:

- Open the Experimenter from the GUI Chooser
    1. Set the number of cross-validation folds (in this case 10)
    2. Add your dataset
    3. Set the Number of repetitions (in this case 1)
    4. Add the algorithm to be tested
- Go to the Run tab and Start the experiment and wait till it finishes
- Go to the Analyse tab and import the experiment results by clicking Experiment
    1. For Row select: Fold
    2. For Column select: Any desired measure (in this case area under the ROC)
    3. The specified results for each fold will be shown in the right for that particular model
    4. Calculate variance of all the folds using any standard variance calculator

The first column of table A.1 shows the model, the next column shows the parameter settings of the corresponding model. I represents the number of iterations and T represents the number of trees. The last column shows the variance of area under the ROC obtained over 10 fold cross validation from Weka.

Table A.1: Analysis of Variance of Area Under ROC (auROC) of each fold of cross-validation. I = number of iterations, T = number of trees.

Model	Parameter Settings	Variance of auROC
J48	Weka default	0.0008
SVM	Polynomial kernel	0.0012
Bagging with J48	$I = 10$	0.0005
Bagging with Random Forest	$T = 10$	0.0004
Random Forest	$T = 10$	0.0003

A.3 Detailed Comparative Analysis of idDock to Other State-of-the-art Methods

Table A.2 provide detailed analysis of comparing the results obtained by idDock to the other methods. The first column reports the PDB id for each of the protein system followed by their chains for each units in parenthesis. The second, third and fourth column presents the average lRMSD to native structure in Å obtained by pyDock, ClusPro and ZDock over 5 independent runs. Each of the measure is followed by the Average Normalized rank (ANR) in parenthesis again. The next column shows the same statistics for HopDock over 5 runs. The last four columns present various measures of idDock in terms of lowest-lRMSD to native in Å, average lRMSD to native in Å, the variance lRMSD to native in $\mathring{A}^2$ and the ANR over all runs.

Table A.2: Comparative analysis of idDock to other state-of-the-art methods on 15 different systems. The results of idDock is reported over 5 runs and shown in terms of lowest (η), average (μ), variance (σ^2) lRMSD and Average Normalized Rank (ANR). The average lRMSD to native structure and ANR (shown in bracket) of pyDock, ClusPro, ZDock and HopDock are also shown over 5 independent runs. All the lRMSDs are shown in terms of Å.

PDB ID (Chains)	pyDock [93](Å) (ANR)	ClusPro [27](Å) (ANR)	ZDock [43](Å) (ANR)	HopDock [5](Å) (ANR)	idDock η(Å)	idDock μ(Å)	idDock $\sigma^2(Å^2)$	idDock ANR
1C1Y (A,B)	3.2(4.3)	1.7(13)	0.7(0.3)	2.1(26.2)	1.1	1.5	0.1	**12.7**
1DS6 (A,B)	0.7(0)	1.8(0)	1.2(0.05)	2.6(31.4)	2.8	2.9	0.1	**19.7**
1TX4 (A,B)	0.4(0.015)	2.1(1)	0.7(0.1)	1.9(4.4)	1.5	2.6	2.2	49.9
1WWW (W,Y)	9.3(100)	10.0(100)	8.3(100)	2.8(47.8)	2.2	3.5	0.8	**38.1**
1FLT (V,Y)	0.5(0.75)	2.1(10)	1.1 (0.1)	2.4(35.4)	0.4	1.8	0.9	**15.9**
1IKN (C,D)	5.0(0.33)	4.2(27)	1.8(42.85)	2.9(45.7)	2.1	2.9	0.6	**51.7**
1VCB (A,B)	0.5(0.05)	1.5(49)	1.0(0.0)	2.1(11.4)	0.8	1.5	0.2	**0.4**
1VCB (B,C)	0.55(0)	1.9(0)	0.7(0.0)	1.8(10.1)	1.1	3.0	3.2	38.5
1OHZ (A,B)	1.2(0.08)	1.2(52)	0.7(1.95)	1.5(20.0)	0.7	0.9	0.1	**0.8**
1ZHI (A,B)	3.2(0.9)	1.4(7)	1.2(0.1)	3.1(30.8)	1.3	3.0	3.4	**6.6**
2HQS (A,C)	9.5(100)	9.7(100)	7.0(100)	3.2(53.6)	1.8	2.4	0.1	**47.0**
1QAV (A,B)	1.2(0.8)	1.7(0)	0.9(1.7)	2.7(32.0)	2.4	2.5	0.1	45.4
1G4Y (B,R)	18.3(100)	18.2(100)	0.7(3.7)	3.4(33.4)	1.8	3.8	3.8	**68.9**
1CSE (E,I)	0.8(0)	1.1(0)	0.5(2.4)	1.6(27.4)	0.5	0.9	0.2	**1.4**
1G4U (R,S)	15.2(100)	16.1(100)	1.3(0.05)	4.1(51.1)	1.2	2.7	1.5	**73.6**

In table A.2 we have highlighted the entire row if idDock is better than the other two methods in terms of average lRMSD to native. From the table we can see that on almost 10/15 of the systems idDock performs better than at least two other methods in terms of average lRMSD to native structure.

We have also make the average normalized rank bold if it is comparable (within 20%) or better than at least two other methods. An ANR of 100% is reported if no configuration was found within $< 5\mathring{A}$ to native structure. From the table it is seen that on 12/15 of the idDock performs comparable or better than at least two other methods. We also notice that on systems: 1WWW, 2HQS, 1G4Y and 1G4U the energy-driven approaches failed completely, whereas idDock is able to report near-native structures on most of systems (except 1ZHI and 1G4Y, where the lowest lRMSD obtained is slightly higher than 5Å). We also notice that almost 7/15 systems the ANR is within 20% which essentially makes the case that idDock can be used successfully as first stage of docking where only first 20% of the generated configurations can be sent for further refinement and minimization. We further points out that on some of the systems like 1VCB(A,B), 1CSE and 1OHZ idDock report very low average normalized rank which is less than $< 1\%$. This essentially indicates that one of the lowest energy structure is indeed the lowest lRMSD to native structure.

Bibliography

Bibliography

[1] V. Polak, "Budda: backbone unbound docking application (master's thesis)," Master's thesis, Computer Science, Tel-Aviv University, Tel-Aviv,Israel, 2003.

[2] E. Kanamori, Y. Murakami, Y. Tsuchiya, D. Standley, H. Nakamura, and K. Kinoshita, "Docking of protein molecular surfaces with evolutionary trace analysis," *Proteins: Struct Funct Bioinf*, vol. 69, pp. 832–838, 2007.

[3] M. L. Connolly, "Analytical molecular surface calculation," *J Appl Cryst*, vol. 16, no. 5, pp. 548–558, 1983.

[4] S. L. Lin, R. Nussinov, D. Fischer, and H. J. Wolfson, "Molecular surface representations by sparse critical points." *Proteins*, vol. 18, no. 1, pp. 94–101, Jan. 1994.

[5] I. Hashmi and A. Shehu, "Hopdock: A probabilistic search algorithm for decoy sampling in protein-protein docking," *Proteome Sci*, vol. 11, no. Suppl 1, p. S6, 2013.

[6] ——, "Informatics-driven protein-protein docking," in *ACM Conf on Bioinf and Comp Biol Workshops (BCBW)*, Washington, D. C., September 2013, pp. 772–779.

[7] S. R. George and B. F. O'Dowd, "A novel dopamine receptor signaling unit in brain: heterooligomers of D1 and D2 dopamine receptors," *Scientific World J*, vol. 7, pp. 58–63, 2011.

[8] W. Guo, L. Shi, and J. A. Javitch, "The fourth transmembrane segment forms the interface of the dopamine D2 receptor homodimer," *J Biol Chem*, vol. 278, pp. 4385–4388, 2003.

[9] W. L. DeLano, "Pymol," *DeLano Scientific, San Carlos, CA*, vol. 700, 2002.

[10] W. Humphrey, A. Dalke, and K. Schulten, "VMD – Visual Molecular Dynamics," *J Mol Graphics*, vol. 14, pp. 33–38, 1996.

[11] D. S. Goodsell and A. J. Olson, "Structural symmetry and protein function," *Annu. Rev. Biophys. and Biomolec. Struct.*, vol. 29, pp. 105–153, 2000.

[12] C. B. Anfinsen, "Principles that govern the folding of protein chains," *Science*, vol. 181, no. 4096, pp. 223–230, 1973.

[13] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne, "The Protein Data Bank," *Nucl Acids Res*, vol. 28, no. 1, pp. 235–242, Jan. 2000. [Online]. Available: www.pdb.org

[14] M. Zacharias, "Accounting for conformational changes during protein–protein docking," *Curr Opinion Struct Biol*, vol. 20, no. 2, pp. 180–186, 2010.

[15] P. Uetz, "Two-hybrid arrays," *Curr Opinion Chem Biol*, vol. 6, no. 1, pp. 57–62, 2002.

[16] E. M. Phizicky and S. Fields, "Protein-protein interactions: methods for detection and analysis," *Microbiol Rev*, vol. 59, no. 1, pp. 94–123, 1995.

[17] R. M. Ewing *et al.*, "Large-scale mapping of human proteinprotein interactions by mass spectrometry. molecular systems biology," *Mol Syst Biol*, vol. 3, no. 1, p. 89, 2007.

[18] K. P. Kilambi, K. Reddy, and J. J. Gray, "Protein-protein docking with dynamic residue protonation states," *PLoS Comput Biol*, vol. 10, no. 12, p. e1004018, 2014.

[19] D. W. Ritchie, "Recent progress and future directions in protein-protein docking," *Curr Prot & Peptide Sci*, vol. 9, no. 1, p. 1, 2008.

[20] N. Moitessier, P. Englebienne, D. Lee, J. Lawandi, and C. R. Corbeil, "Towards the development of universal, fast and highly accurate docking/scoring methods: a long way to go," *Brit J Pharma*, vol. 153, no. S1, pp. S7–S27, 2009.

[21] T. M. Cheng, T. L. Blundell, and J. Fernandez-Recio, "pyDock: electrostatics and desolvation for effective scoring of rigid-body protein-protein docking," *Proteins*, vol. 68, no. 2, pp. 503–515, 2007.

[22] J. J. Gray, S. Moughon, C. Wang, O. Schueler-Furman, B. Kuhlman, C. A. Rohl, and D. Baker, "Protein–protein docking with simultaneous optimization of rigid-body displacement and side-chain conformations," *J Mol Biol*, vol. 331, no. 1, pp. 281–299, 2003.

[23] Y. Inbar, H. Benyamini, R. Nussinov, and H. J. Wolfson, "Combinatorial docking approach for structure prediction of large proteins and multi-molecular assemblies," *J Phys Biol*, vol. 2, pp. S156–S165, 2005.

[24] M. F. Lensink and S. J. Wodak, "Docking and scoring protein interactions: CAPRI 2009," *Proteins: Struct Funct Bioinf*, vol. 78, no. 15, pp. 3073–3084, 2009.

[25] ——, "Blind predictions of protein interfaces by docking calculations in CAPRI," *Proteins: Struct Funct Bioinf*, vol. 78, no. 15, pp. 3085–3095, 2010.

[26] D. Duhovny-Schneidman, Y. Inbar, R. Nussinov, and H. J. Wolfson, "PatchDock and SymmDock: servers for rigid and symmetric docking," *Nucl Acids Res*, vol. 33, no. S2, pp. W363–W367, 2005.

[27] S. R. Comeau, D. W. Gatchell, S. Vajda, and C. J. Camacho, "ClusPro: a fully automated algorithm for protein-protein docking," *Nucl Acids Res*, vol. 32, no. S1, pp. W96–W99, 2004.

[28] S. Lyskov and J. J. Gray, "The RosettaDock server for local protein-protein docking," *Nucl Acids Res*, vol. 36, no. S2, pp. W233–W238, 2008.

[29] G. Terashi, M. Takeda-Shitaka, K. Kanou, M. Iwadate, D. Takaya, and H. Umeyama, "The SKE-DOCK server and human teams based on a combined method of shape complementarity and free energy estimation," *Proteins: Struct Funct Bioinf*, vol. 69, no. 4, pp. 866–887, 2007.

[30] A. Tovchigrechko and I. A. Vakser, "GRAMM-X public web server for protein-protein docking," *Nucl Acids Res*, vol. 34, no. Web Server issue, pp. W310–4, 2006.

[31] D. Kozakov, R. Brenke, S. Comeau, and S. Vajda, "PIPER: an FFT-based protein docking program with pairwise potentials," *Proteins: Struct Funct Bioinf*, vol. 65, no. 2, pp. 392–406, 2006.

[32] S. Huang, "Search strategies and evaluation in protein–protein docking: principles, advances and challenges," *Drug discovery Today*, vol. 19, no. 8, pp. 1081–1096, 2014.

[33] I. Halperin, B. Ma, H. Wolfson, and R. Nussinov, "Principles of docking: An overview of search algorithms and a guide to scoring functions," *Proteins*, vol. 47, no. 4, pp. 409–443, 2002.

[34] C., B. R. Dominguez, and A. Bonvin, "Haddock: A protein-protein docking approach based on biochemical orbiophysical information," *J Am Chem Soc*, vol. 125, pp. 1731–1737, 2003.

[35] B. A. Shoemaker and A. R. Panchenko, "Deciphering protein–protein interactions. part ii. computational methods to predict protein and domain interaction partners," *PLoS Comput Biol*, vol. 3, no. 4, p. e43, 2007.

[36] A. J. Bordner and A. A. Gorin, "Protein docking using surface matching and supervised machine learning," *Proteins: Struct Funct Bioinf*, vol. 68, no. 2, pp. 488–502, 2007.

[37] H. Zhu, F. S. Domingues, I. Sommer, and T. Lengauer, "NOXclass: prediction of protein-protein interaction types," *BMC Bioinf*, vol. 7, p. 27, 2006.

[38] N. Li, Z. Sun, and F. Jiang, "Prediction of protein-protein binding site by using core interface residue and support vector machine," *BMC Bioinf*, vol. 9, p. 553, 2008.

[39] B. Li and D. Kihara, "Protein docking prediction using predicted protein-protein interface," *BMC Bioinf*, vol. 13, p. 7, 2012.

[40] C. Yan, D. Dobbs, and V. Honavar, "A two-stage classifier for identification of protein–protein interface residues," *Bioinformatics*, vol. 20, no. suppl 1, pp. i371–i378, 2004.

[41] V. Venkatraman, Y. D. Yang, L. Sael, and D. Kihara, "Protein-protein docking using region-based 3d zernike descriptors," *BMC Bioinf*, vol. 10, no. 1, p. 407, 2009.

[42] H. L. Wolfson and I. Rigoutsos, "Geometric hashing: an overview," *IEEE Comp Sci and Engineering*, vol. 4, no. 4, pp. 10–21, 1997.

[43] R. Chen, L. Li, and Z. Weng, "ZDock: an initial-stage protein-docking algorithm," *Proteins: Struct Funct Bioinf*, vol. 52, no. 1, pp. 80–87, 2003.

[44] C. L. Bajaj, R. Chowdhury, and V. Siddahanavalli, "F2dock: Fast fourier protein-protein docking," *IEEE/ACM Trans on Comput Biol and Bioinf*, vol. 8, no. 1, pp. 45–58, 2011.

[45] G. Macindoe, L. Mavridis, V. Venkatraman, M. Devignes, and D. Ritchie, "Hexserver: an FFT-based protein docking server powered by graphics processors," *Nucl Acids Res*, vol. 38, no. Suppl 2, pp. W445–W449, 2010.

[46] E. Karaca, A. S. Melquiond, S. J. de Vries, P. L. Kastritis, and A. M. Bonvin, "Building macromolecular assemblies by information-driven docking introducing the haddock multibody docking server," *Mol and Cell Proteomics*, vol. 9, pp. 1784–1794, 2010.

[47] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *J Chem Phys*, vol. 21, no. 6, pp. 1087–1092, 1953.

[48] G. M. Morris, D. S. Goodsell, R. S. Halliday, R. Huey, W. E. Hart, R. K. Belew, A. J. Olson *et al.*, "Automated docking using a lamarckian genetic algorithm and an empirical binding free energy function," *J Comput Chem*, vol. 19, no. 14, pp. 1639–1662, 1998.

[49] E. J. Gardiner, P. Willett, and P. J. Artymiuk, "Protein docking using a genetic algorithm," *Proteins: Struct Funct Bioinf*, vol. 44, no. 1, pp. 44–56, 2001.

[50] E. Atilgan and J. Hu, "Improving protein docking using sustainable genetic algorithms," *Intl J. Comput Inf Syst Ind Man*, vol. 3, pp. 248–255, 2011.

[51] I. S. Moreira, P. A. Fernandes, and M. J. Ramos, "Hot spots-a review of the protein-protein interface determinant amino-acid residues," *Proteins*, vol. 68, no. 4, pp. 803–812, 2007.

[52] O. Lichtarge, H. R. Bourne, and F. E. Cohen, "An evolutionary trace method defines binding surfaces common to protein families," *J Mol Biol*, vol. 257, no. 2, pp. 342–58, 1996.

[53] S. Engelen, L. A. Trojan, S. Sacquin-Mora, R. Lavery, A. Carbone *et al.*, "A large-scale method to predict protein interfaces based on sequence sampling," *PLoS Comp Bio*, vol. 5, no. 1, p. e1000267, 2009.

[54] J. Esquivel-Rodriguez, Y. D. Yang, and D. Kihara, "Multi-lzerd: Multiple protein docking for asymmetric complexes," *Proteins: Struct Funct Bioinf*, vol. 80, no. 7, pp. 1818–1833, 2012.

[55] S. F. Altschul, T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman, "Gapped blast and psi-blast: a new generation of protein database search programs," *Nucl Acids Res*, vol. 25, no. 17, pp. 3389–3402, 1997.

[56] D. Fischer, S. L. Lin, H. L. Wolfson, and R. Nussinov, "A geometry-based suite of molecular docking processes," *J Mol Biol*, vol. 248, no. 2, pp. 459–477, 2005.

[57] E. Moreno, H. Hoffmann, M. Gonzalez-Sepúlveda, G. Navarro, C. V., A. Cortés, J. Mallol, M. Vignes, P. J. McCormick, E. I. Canela, C. Lluís, R. Moratalla, S. Ferré, J. Ortiz, , and R. Franco, "Dopamine D1-histamine H3 receptor heteromers provide a selective link to MAPK signaling in gabaergic neurons of the direct striatal pathway," *J Biol Chem*, vol. 286, pp. 5846–6854, 2011.

[58] B. Jiménez-García, C. Pons, and J. Fernández-Recio, "pydockweb: a web server for rigid-body protein–protein docking using electrostatics and desolvation scoring," *Bioinformatics*, vol. 29, no. 13, pp. 1698–1699, 2013.

[59] B. G. Pierce, K. Wiehe, H. Hwang, B.-H. Kim, T. Vreven, and Z. Weng, "Zdock server: interactive docking prediction of protein–protein complexes and symmetric multimers," *Bioinformatics*, vol. 30, no. 12, pp. 1771–1773, 2014.

[60] M. Torchala, I. H. Moal, R. A. Chaleil, J. Fernandez-Recio, and P. A. Bates, "Swarmdock: a server for flexible protein–protein docking," *Bioinformatics*, vol. 29, no. 6, pp. 807–809, 2013.

[61] S. Mondal, J. M. Johnston, H. Wang, G. Khelashvili, M. Filizola, and H. Weinstein, "Membrane driven spatial organization of gpcrs," *Scientific reports*, vol. 3, 2013.

[62] G. Y. Ng, B. F. O'Dowd, S. P. Lee, H. T. Chung, M. R. Brann, P. Seeman, and S. R. George, "Dopamine d2 receptor dimers and receptor-blocking peptides," *Biochem and Biophysic R Comm*, vol. 227, no. 1, pp. 200–204, 1996.

[63] W. Guo, L. Shi, M. Filizola, H. Weinstein, and J. A. Javitch, "Crosstalk in g protein-coupled receptors: changes at the transmembrane homodimer interface determine activation," *Proc Natl Acad Sci USA*, vol. 102, no. 48, pp. 17 495–17 500, 2005.

[64] W. Guo, E. Urizar, M. Kralikova, J. C. Mobarec, L. Shi, M. Filizola, and J. A. Javitch, "Dopamine d2 receptors form higher order oligomers at physiological expression levels," *The EMBO journal*, vol. 27, no. 17, pp. 2293–2304, 2008.

[65] Y. Liang, D. Fotiadis, S. Filipek, D. A. Saperstein, K. Palczewski, and A. Engel, "Organization of the g protein-coupled receptors rhodopsin and opsin in native membranes," *J Biol Chem*, vol. 278, no. 24, pp. 21 655–21 662, 2003.

[66] B. R. Brooks, R. E. Bruccoleri, B. D. Olafson, D. J. States, S. Swaminathan, and M. Karplus, "CHARMM: A program for macromolecular energy, minimization, and dynamics calculations," *J Comput Chem*, vol. 4, no. 2, pp. 187–217, 1983.

[67] N. Andrusier, R. Nussinov, and H. J. Wolfson, "Firedock: fast interaction refinement in molecular docking," *Proteins: Struct Funct Bioinf*, vol. 69, no. 1, pp. 139–159, 2007.

[68] J. Schymkowitz, J. Borg, F. Stricher, R. Nys, F. Rousseau, and L. Serrano, "The foldx web server: an online force field," *Nucl Acids Res*, vol. 33, no. Web server issue, pp. W382–W388, 2005.

[69] A. D. McLachlan, "A mathematical procedure for superimposing atomic coordinates of proteins," *Acta Crystallogr A*, vol. 26, no. 6, pp. 656–657, 1972.

[70] J. A. Vrugt, H. V. Gupta, L. A. Bastidas, W. Bouten, and S. Sorooshian, "Effective and efficient algorithm for multiobjective optimization of hydrologic models," *Water Resources Res*, vol. 39, no. 8, 2003.

[71] A. Roy, A. Kucukural, and Y. Zhang, "I-tasser: a unified platform for automated protein structure and function prediction," *Nature protocols*, vol. 5, no. 4, pp. 725–738, 2010.

[72] J. Moult, K. Fidelis, A. Kryshtafovych, and A. Tramontano, "Critical assessment of methods of protein structure prediction (casp) - round ix," *Proteins: Struct Funct Bioinf*, vol. 79, no. S10, pp. 1–5, 2011.

[73] O. Keskin, B. Ma, and R. Nussinov, "Hot regions in protein-protein interactions: the organization and contribution of structurally conserved hot spot residues," *J Mol Biol*, vol. 345, no. 5, pp. 1281–1294, 2005.

[74] B. Olson and A. Shehu, "Evolutionary-inspired probabilistic search for enhancing sampling of local minima in the protein energy surface," *Proteome Sci*, vol. 10, no. 10, p. S5, 2012.

[75] O. M. H.R. Lourenco and T. Stutzle, "Iterated local search," in *Handbook of Metaheuristics*. Kluwer Academic Publishers, 2002, vol. 57, no. 513, pp. 321–353.

[76] T. Kortemme and D. Baker, "A simple physical model for binding energy hot spots in protein–protein complexes," *Proc Natl Acad of Sci USA*, vol. 99, no. 22, pp. 14 116–14 121, 2002.

[77] B. Jimenez-Garcia, C. Pons, and J. Fernandez-Recio, "pyDockWEB: a web server for rigid-body protein-protein docking using electrostatics and desolvation scoring," *Bioinformatics*, vol. 29, no. 13, pp. 1698–1699, 2013.

[78] I. Hashmi and A. Shehu, "idDock+:integrating machine learning in probabilistic search for protein-protein docking." *J Comp Biol (JCB)*, vol. 22(9), pp. 1–18, 2015.

[79] R. Wang, X. Fang, Y. Lu, C.-Y. Yang, and S. Wang, "The pdbbind database: Methodologies and updates," *J Med Chem*, vol. 48, no. 12, pp. 411–419, 2005.

[80] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," in *SIGKDD*, vol. 11, no. 1. New York, NY, USA: ACM, Nov. 2009, pp. 10–18.

[81] S. L. Graham, P. B. Kessler, and M. K. Mckusick, "Gprof: A call graph execution profiler," in *ACM Sigplan Notices*, vol. 17, no. 6. ACM, 1982, pp. 120–126.

[82] N. London and O. Schueler-Furman, "Funnel hunting in a rough terrain: learning and discriminating native energy funnels," *Structure*, vol. 16, no. 2, pp. 269–279, 2008.

[83] L. C. Xue, R. A. Jordan, E.-M. Yasser, D. Dobbs, and V. Honavar, "Dockrank: Ranking docked conformations using partner-specific sequence homology-based protein interface prediction," *Proteins: Struct Funct Bioinf*, vol. 82, no. 2, pp. 250–267, 2014.

[84] U. Consortium *et al.*, "Reorganizing the protein space at the universal protein resource (uniprot)," *Nucl Acids Res*, p. gkr981, 2011.

[85] A. G. Murzin, S. E. Brenner, T. Hubbard, and C. Chothia, "Scop: a structural classification of proteins database for the investigation of sequences and structures," *J Mol Biol*, vol. 247, no. 4, pp. 536–540, 1995.

[86] S. D. Bay and M. J. Pazzani, "Detecting change in categorical data: Mining contrast sets," in *Proc of the Fifth ACM SIGKDD Intl Conf on Knowledge Discovery and Data Mining*. ACM, 1999, pp. 302–306.

[87] T. Pang-Ning, M. Steinbach, V. Kumar *et al.*, "Introduction to data mining," in *Library of Congress*, 2006, p. 74.

[88] K. A. De Jong, *Evolutionary computation: a unified approach*. MIT press, 2006.

[89] H. S. Park, Q. Lin, and A. D. Hamilton, "Modulation of protein–protein interactions by synthetic receptors: Design of molecules that disrupt serine protease–proteinaceous inhibitor interaction," *Proc Natl Acad Sci USA*, vol. 99, no. 8, pp. 5105–5109, 2002.

[90] A. Schön, S. Y. Lam, and E. Freire, "Thermodynamics-based drug design: strategies for inhibiting protein-protein interactions," *Future medicinal chemistry*, vol. 3, no. 9, pp. 1129–1137, 2011.

[91] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic local alignment search tool," *J Mol Biol*, vol. 215, no. 3, pp. 403–410, 1990.

[92] S. Henikoff and J. G. Henikoff, "Amino acid substitution matrices from protein blocks," *Proc Natl Acad Sci USA*, vol. 89, no. 22, pp. 10 915–10 919, 1992.

[93] T. M. Cheng, T. L. Blundell, and J. Fernandez-Recio, "pyDock: Electrostatics and desolvation for effective scoring of rigid-body protein–protein docking," *Proteins*, vol. 68, no. 2, pp. 503–515, Aug. 2007.

Curriculum Vitae

Irina Hashmi is pursuing her Ph.D. in Computer Science at George Mason University. She received her BS in Computer Science and Engineering from University of Dhaka, Bangladesh and MS from George Mason University. Her research interests include computational biology, evolutionary computation and data mining. Currently her work focuses on high-dimensional search optimization for the problems related to protein docking.