The Dissertation Committee for Shruthi Viswanath
Certifies that this is the approved version of the following dissertation:

Scoring Functions for Protein Docking and Drug Design

Committee:

Ron Elber, Supervisor

Daniel Miranker

Pengyu Ren

Donald Fussell

Risto Miikkulainen

SCORING FUNCTIONS FOR PROTEIN DOCKING AND DRUG DESIGN

by

Shruthi Viswanath, B.Tech.Info.Tech.

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

The University of Texas at Austin

May 2014

Dedication

To my family.

To all dedicated and inspiring teachers.

Acknowledgements

Firstly, I am very grateful to have got the opportunity to work with my PhD advisor, Prof Ron Elber. He is a wonderful and inspiring example of a scientist and mentor. Working with him made everything seem easy. I am grateful for his faith in me, for being patient and giving me the space to grow at my own pace. He is incredibly helpful and generous with his time and is always available to answer any questions. His commitment, dedication and enthusiasm for science are contagious, and they ended up having a huge influence on me. He is a role model for life.

Thanks also to my other committee members, Prof Ren, Prof Fussell, Prof Miranker and Prof Miikkulainen for being very accommodating, especially during the time of my proposal exam. Special thanks to Prof Miranker for his extensive comments on my dissertation and for being supportive since my beginning days in grad school.

I wish to thank my former supervisor Dr Chengyong Yang in the Genetic Systems group of Thermo Fischer Scientific for teaching me several aspects of research. I am also grateful to Prof John Straub of Boston University and members of his group, Laura Dominguez and Leigh Foster, for collaborating with us on the membrane-docking project. Thanks are also due to the CAPRI community of researchers who are very encouraging of newcomers like me.

CLSB (Elber group) has been a fun place to be a graduate student. I would like to acknowledge the help I received as a beginner, from past members of the Elber group. Specifically, I am grateful to Ravikant for introducing me to the docking code and for laying the intellectual foundation for the work in my thesis. I would also like to thank Brinda Vallat, Thomas Blom and Baoqiang Cao for teaching me various aspects of

cluster computing and structure prediction. Other past and present members of CLSB who have been great at providing comments, technical help and general discussions: Szu-Hua Chen, Michele di Pierro, Mauro Mugnai, Juan Bello Rivas, Alfredo Cardenas, Serdal Kirmizialtin and Peter Ryumgaart.

I am very grateful to the ICES systems support staff for providing helpful and reliable technical support throughout the past few years. I also gratefully acknowledge the logistic help provided by Ruth Hengst of ICES and Lydia Griffith of CS. They bailed me out of trouble many times.

Szu-Hua and Minjung have been great friends in the 4th floor of ACES. I wish to thank my roommates and UT and non-UT friends for fun times and discussions: Akanksha, Subhashini, Sindhu, Aparna Roy and Srinath. My friends from undergrad have been especially supportive at all times in my grad school life: Pallavika, Priya and Swati. I wish to thank Archana and Mahesh for providing a home away from home in Austin.

I wish to thank my family, specially my parents and brother for their support and faith and my sweet in-laws for their encouragement. Finally I wish to thank my husband, Vishvas, for his enormous patience and for being my pillar of support throughout, and eagerly look forward to joining him after I graduate.

Scoring Functions for Protein Docking and Drug Design

Shruthi Viswanath, PhD

The University of Texas at Austin, 2014

Supervisor: Ron Elber

Predicting the structure of complexes formed by two interacting proteins is an important problem in computation structural biology. Proteins perform many of their functions by binding to other proteins. The structure of protein-protein complexes provides atomic details about protein function and biochemical pathways, and can help in designing drugs that inhibit binding. Docking computationally models the structure of protein-protein complexes, given three-dimensional structures of the individual chains.

Protein docking methods have two phases. In the first phase, a comprehensive, coarse search is performed for optimally docked models. In the second refinement and reranking phase, the models from the first phase are refined and reranked, with the expectation of extracting a small set of accurate models from the pool of thousands of models obtained from the first phase.

In this thesis, new algorithms are developed for the refinement and reranking phase of docking. New scoring functions, or potentials, that rank models are developed. These potentials are learnt using large-scale machine learning methods based on mathematical programming. The procedure for learning these potentials involves examining hundreds of thousands of correct and incorrect models. In this thesis, hierarchical constraints were introduced into the learning algorithm.

First, an atomic potential was developed using this learning procedure. A refinement procedure involving side-chain remodeling and conjugate gradient-based minimization was introduced. The refinement procedure combined with the atomic potential was shown to improve docking accuracy significantly.

Second, a hydrogen bond potential, was developed. Molecular dynamics-based sampling combined with the hydrogen bond potential improved docking predictions.

Third, mathematical programming compared favorably to SVMs and neural networks in terms of accuracy, training and test time for the task of designing potentials to rank docking models. The methods described in this thesis are implemented in the docking package DOCK/PIERR. DOCK/PIERR was shown to be among the best automated docking methods in community wide assessments.

Finally, DOCK/PIERR was extended to predict membrane protein complexes. A membrane-based score was added to the reranking phase, and shown to improve the accuracy of docking. This docking algorithm for membrane proteins was used to study the dimers of amyloid precursor protein, implicated in Alzheimer's disease.

Table of Contents

List of Tables

List of Figures

Chapter 1. Introduction

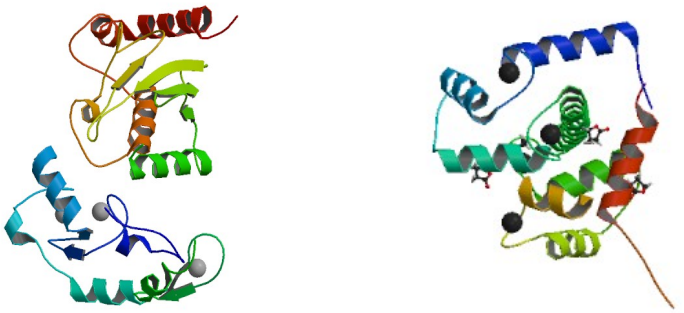
1.1 PROTEIN STRUCTURE PREDICTION

Proteins are important biomolecules and integral components of the cellular machinery. They are responsible for most cellular functions, including transport of molecules, immunity, movement, catalysis of reactions and signaling. Proteins are a linear chain of basic units: known as amino acids or *residues*. Amino acids differ from each other in their side chains, which give them each unique physical and chemical properties, and form the basis for the enormous diversity in protein structures.

Predicting the structure of proteins is an important problem in computational structural biology. Given the linear sequence of amino acids of a protein, also known as its primary sequence, one can predict its secondary structure (local patterns formed by the sequence) and tertiary structure (three-dimensional fold) computationally. Knowledge of the tertiary structure provides important clues about the function of a protein. The computational method of *homology modeling* [1] is the most widely used method for predicting the tertiary structure of a protein, given a) its sequence and b) another evolutionarily related protein, known as the template, whose structure is already known. Experimentally, protein structure is determined by the methods of X-ray crystallography or Nuclear Magnetic Resonance (NMR). All experimentally determined structures of proteins are added to an online database known as the Protein Data Bank.

Proteins do not exist in isolation in a cellular environment but accomplish their function by interacting with other proteins. Each protein is postulated to interact with at least ten other proteins during its lifetime [2-4]. Protein-protein interactions play a vital role in cellular processes and the function of a protein can be determined by its interactions [5]. The structure of protein chains interacting with each other is known as a *protein-protein complex*, or quaternary structure. Complexes can be formed in different

cellular environments: the majority are formed between proteins in water, or *soluble proteins*. These proteins exist in a *hydrophilic* (polar) environment. Protein complexes are also found in cell membranes, which is a *hydrophobic* (non-polar and water-repelling) environment. Figure 1.1 shows two complexes in the Protein Data Bank (PDB)[6, 7]. Figure 1.1 (a) is a structure of soluble complex (PDB ID 3hct) [6], which is a structure of Ubc13, a ubiquitin conjugating enzyme interacting with TRA6, a ubiquitin modulating protein. Figure 1.1 (b) is a complex (PDB ID 4ehq) [7] of membrane protein Orai1, a protein responsible for Calcium ion transport through the cell membrane, with Calmodulin, a protein that regulates calcium levels in the cell.



(a) (b)

Figure 1.1 (a) Complex 3hct [6], a soluble complex involving enzyme Ubc13. (b) Complex 4ehq [7], a membrane complex involving calcium ion transporting protein, Orai1.

The structure of protein-protein complexes is harder to predict through experimental techniques than tertiary structure. Complexes are generally too big to be solved by NMR techniques. There are about a thousand structures of protein-protein complexes in the Protein Data Bank, which is much smaller than the number of deposited tertiary folds, which is of the order of tens of thousands. One reason is that most protein complexes are formed only transiently. Another reason is that large structures of protein

complexes contain multiple domains and it is more difficult to crystallize multi-domain proteins, as they are less stable than single domain proteins[8]. Computational methods can be a quicker and cheaper alternative to get the structure of complexes than experimental studies. Note that the words *structure, model* and *conformation* mean the same thing.

1.2 PROTEIN-PROTEIN DOCKING

Protein-protein docking is a computational method that provides the atomically detailed structure of a complex formed by two proteins, given their individual tertiary structures. Usually the bigger of the two proteins is designated as the *receptor* and the smaller one is known as the *ligand*. The *interface* of a protein in a complex is the set of residues that are closest to, and interact with the other protein molecule.

Bound and unbound docking

The easy case of docking is known as *bound docking* in which we take apart the receptor and ligand from a known complex and find the binding pose of the ligand with respect to the receptor. This case is used to evaluate the performance of various docking algorithms. In the more realistic case, the tertiary structures of either the receptor or ligand or both, are not known and we need to model the 3D structures of the constituent proteins first. These approximate structures are then used to perform the docking, and this is known as *unbound docking*. Unbound docking is a harder problem since the individual structures of constituents are known only approximately.

Stages in protein docking algorithms

Protein docking algorithms generally consist of two phases: an initial rigid docking and coarse scoring phase, followed by a refinement and rescoring phase [9].

a. Rigid docking and coarse scoring

In this first phase, the individual protein structures are kept rigid. One of the molecules (usually the receptor) is kept fixed and a search is performed for various possible orientations of the ligand with respect to the receptor. This is a global search in 6-dimensional space (3 dimensions for rigid translation and 3 for rigid rotation of the ligand with respect to the receptor). There are many different search strategies that have been used by various groups.

One of the most widely used search algorithms for docking uses Fast Fourier Transforms (FFT) [10-13]. The interaction score between two molecules can be represented as a convolution and hence calculated using FFTs efficiently. Geometric Hashing[9, 14] is another search technique where instead of matching atoms or points in a grid between the two proteins, as in the case of FFT, a higher level matching is done: patches which denote the local shapes of a molecule are matched. Monte-Carlo searches in rigid body space have also been used a search strategy [15-18]. FFTs, unlike the other methods, enable an efficient exhaustive sampling of all rigid orientations (i.e. rotations and translations) of one molecule with respect to the other.

Scoring for models of the complex in this initial phase is coarse and not very detailed, and is usually based on the positions and types of residues (residue-based). More details on scoring functions are provided in the next section.

b. Refinement and Rescoring

The first search phase returns hundreds of thousands of models. In the second phase, these models are refined by local searches. Some amount of flexibility is introduced in the models [17, 19-23]. Models are reranked using more detailed scores and at the end of this phase, we are required to select the top few conformations of the

complex. But discriminating the best models from a given set has proved to be a harder problem than obtaining an initial large set of models containing a few good ones [24].

The changes to models are local, for example, side chain adjustments and limited adjustments of the protein backbone. The major search strategies are conjugate gradient [12],[25-27] and Monte-Carlo [15, 18, 28], and structures that are more chemically reasonable are obtained from the initial rigid docked models.

These altered structures are rescored using more detailed terms than the initial search stage, for example terms that are dependent on positions and types of atoms, as opposed to positions and types of residues[23, 29].

1.3 SCORING PROTEIN MODELS

Scoring functions assign scores to models that are expected to quantify how good a model is. Ranking a set of structures using a scoring function helps us find the best models in the set. Scoring functions can be classified as *coarse-grained* or *fine-grained* depending on whether the parameters for the potential are designed at the residue level or at the atomic level.

Energy landscape theory

Here, we introduce the term *energy* or *potential*, as an alternative to *score* and *scoring function*. All of these terms quantify models. The difference is that, for scores and scoring functions, higher is better, i.e. larger the score, better the model is expected to be. Whereas, for energy or potential, lower the energy, better the model is expected to be. Energy usually has a physical meaning and can be used in biochemical calculationsfor estimating stability of a protein, and other equilibrium properties. This follows from the thermodynamic hypothesis that the true structure (also known as the *native* structure) is

thermodynamically the most stable and hence lowest in free energy. This is depicted in Figure 1.2 that shows the energy landscape for protein folding or protein binding [30]. The native structure of the protein complex is at the bottom of the funnel shape and has the lowest energy. Structures that are close to the native, or *near-native* structures have lower energies than structures that are far away from the native, or *incorrect* structures, which are near the top of the funnel. For our purposes, the true structure is the experimental structure with which we compare our predicted models. We note that there can be multiple folds with the same lowest energy. However, we use the energy landscape as a working hypothesis as it has been shown to be an accurate model in a number of cases [30].

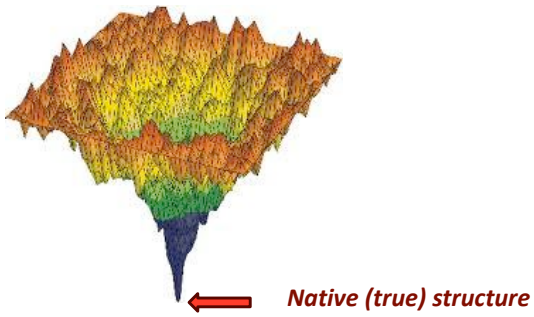


Figure 1.2 Cartoon diagram of the funnel-shaped landscape of protein binding[31].

Potentials are generally classified as *physics based* or *knowledge-based*, depending on how the parameters for the potential are derived.

Physics based scoring

Physics based potentials are also known as *molecular mechanics potentials*. They typically include bonded terms that measure deviations from ideal bonds (two-body), angles (three-body) and torsions (four-body), and non-bonded terms that include longer-

range interactions such as electrostatic interactions and van der Waals interactions [32]. The van der Waals interaction is modeled by the Lennard-Jones functional form, as shown in Eq. (1.1). These functions are continuous and differentiable in the atomic coordinate space. The parameters of molecular mechanics potentials such as $A, B, q, k_{bond}, k_{angle}, k_{torsion}, k_n$ are determined from physical properties of small molecules [32].

$$
\begin{aligned}
&E_{total} = E_{bonded} + E_{non-bonded} \\
&E_{bonded} = E_{bonds} + E_{angles} + E_{torsions} = \sum_{bonds} k_{bond}(l - l_{eq})^2 + \sum_{angles} k_{\theta}(\theta - \theta_{eq})^2 + \sum_{torsions,n} k_n(1 + \cos(n\phi + \gamma)) \\
&E_{non-bonded} = E_{vdw} + E_{electrostatics} = \sum_{i,j}\left(\frac{A_{ij}}{r_{ij}^{12}} - \frac{B_{ij}}{r_{ij}^{6}}\right) + \sum_{i,j}\left(\frac{q_i q_j}{\varepsilon r_{ij}}\right)
\end{aligned}
\tag{1.1}
$$

Knowledge based scoring

There is another class of potentials, which are termed as *knowledge-based* potentials, since their parameters are based not on experimental data from small molecules, but on statistical analyses of experimentally determined structures. Statistical potentials are an example of this kind. The potentials are derived based on Eq. (1.2) [33]. The energy of the current model *E(s)*, as a function of a geometric variable *s,* is given by the log odds ratio of the probabilities of the current state and reference state, as a function of the variable *s*. The numerator, $p^{struct}(s)$ depends on the geometry of the current structure and the denominator $p^{ref}(s)$ is based on the geometry of experimental structures in a reference database. The variable *s* is a geometric parameter such as a distance or torsion in the structure. *T* is the temperature and *k* is the Boltzmann constant and *kT* is a constant with a value of 0.593 kcal/mol at room temperature (300 K).

$$
E(s) = -kT \ln \frac{p^{struct}(s)}{p^{ref}(s)} \tag{1.2}
$$

Using the geometric parameter *s* as the distance between two particle types, where particle types can be residue-based or atom-based, a distance-dependent version of Eq. (1.2) is shown in Eq. (1.3). Here $e(i,j,d)$ is the contribution to the energy when the particle types *i* and *j* in the current model are at a distance range *d*. The particles can be atoms or residues. $n(i,j,d)$ is the number of times in the current model that particle types *i* and *j* are found at a distance range *d*. The probabilities of occurrence of particle types *i* and *j* in the reference state are assumed to be independent, and $n(d)$ is the probability that any pair of particle types occur at a distance *d*. The probabilities in the denominator of Eq. (1.3) are calculated from a set of existing experimental structures in a database, and the numerator is calculated from the geometry of the current model whose energy we want to compute.

$$e(i,j,d) = -kT \log\left[\frac{n(i,j,d)}{n(i)n(j)n(d)}\right] \tag{1.3}$$

The total statistical potential energy of a model is then the sum of all the pairwise particle interactions, as in Eq. (1.4).

$$E_{statistical} = \sum_{i=1}^{N} \sum_{j>i} e(i,j,d_{ij}) \tag{1.4}$$

1.4 MATHEMATICAL PROGRAMMING FOR DEVELOPING POTENTIALS

Mathematical programming is another method for deriving knowledge-based potentials. This approach was first proposed by Maiorov and Crippen[34] and later built upon by others[35-38]. This method involves solving a set of inequalities that specify that the energy of a correct structure $X_{correct}$ should be lower than the energy of an incorrect structure $X_{incorrect}$ as in Eq. (1.5).

$$E(X_{incorrect}) - E(X_{correct}) > 0 \tag{1.5}$$

The native fold (experimental fold) and the near-native models in the training set are used as correct structures, and all other models are used as incorrect structures, or decoys.

Using a similar distance-dependent formulation as in the case of statistical potentials, we have $u(i,j,d)$ representing the contribution to the energy when particle types i and j are at a distance range d. Representing the particle pair type (i,j) by a single parameter α, the energy of a structure can be written as the number of times, $n(\alpha,d)$, a contact between particle pair denoted by α is observed at a distance range d, times the weight for that contact type, $u(\alpha,d)$, as in Eq. (1.6).

$$E(X) = \sum_{\alpha,d} n(\alpha,d)u(\alpha,d) \tag{1.6}$$

Substituting Eq. (1.6) in Eq. (1.5) we get Eq. (1.7), which shows that the inequalities are linear in the parameters u, that determine the potential. This enables efficient calculation of the potential parameters through linear programming solvers like PF3 [39, 40].

$$\sum_{\alpha,d} u(\alpha,d)[n_{incorrect}(\alpha,d) - n_{correct}(\alpha,d)] > 0 \tag{1.7}$$

Using learning sets of hundreds of thousands of correct and incorrect structures, millions of inequalities of the type shown above are formulated and solved for the potential parameters u.

Note that to use these knowledge-based scoring functions for docking models, we do not consider all contacts in the model, but only contacts across the interface of the two proteins. That is, particles i and j belong to different (interacting) proteins in the complex, and d is the distance between them across the interface.

We note also that sampling and scoring are not independent: the potential parameters obtained from one set of models, from a given search algorithm may not be

applicable for scoring another set of models from a different docking algorithm. In this thesis, we develop potentials for the set of models sampled from our docking package DOCK/PIERR [13, 41].

Advantages over other knowledge-based potentials

Mathematical programming aims to produce potentials that explicitly model the energy landscape by stating that models near the bottom of the funnel should have lower energies than models near the top of the funnel.

In statistical potentials, statistics of negative examples i.e. incorrect structures are included in the reference distribution [2]. But mathematical programming is a discriminative learning technique: positive examples are separated from and explicitly compared against negative examples.

Exhaustive large-scale explicit examination of negative examples, as done in mathematical programming, provides potentials that have a positive distribution of $\Delta E = E(X_{incorrect}) - E(X_{correct})$ i.e. they always have higher energies for incorrect models, compared to correct models.

Moreover, quadratic programming based approaches have provably optimal convergence guarantees: it can be shown that for a given functional form, training set and error bound, ε, the set of parameters obtained from mathematical programming is optimal within the error bound [13, 42].

Another advantage over statistical potentials is that no assumption of reference state is required, as the reference state is modeled implicitly by sampling from the distribution of incorrect structures. Hence we do not need to assume independence of the distribution of particle types[2].

Disadvantages

The disadvantage of mathematical programming is that it requires special purpose solvers and large memory machines if the set of inequalities is very large.

Learning using mathematical programming is also restricted to simple functional forms, since the inequalities are linear in the parameters.

The large-scale examination of negative examples is prohibitively expensive for cases where we deviate from rigid docking and sample using extensive rearrangements of the structure, since the number of negative examples increases exponentially in such a search space.

1.5 METRICS FOR ASSESSING PROTEIN DOCKING MODELS

To measure if a computationally determined structure is accurate, we compare it to the native structure, which is the experimentally observed conformation deposited in the Protein Data Bank. There are different metrics that compare a given model to the reference structure or the native conformation. One popular measure is the *RMSD* or Root-Mean-Square-Deviation between the structures. It is a least-squares distance between the coordinates of the atoms in the model and reference structures after optimal superposition of the two structures. It is given by Eq. (1.8) where the *v*'s and *w*'s are coordinates of atoms in the reference and model structure respectively. To minimize the distance between the two coordinate sets, the coordinates of the model, *w*, are transformed by translating by *T*, a translation vector, and rotating by *U*, a unitary rotation matrix. *U* and *T* are computed analytically from the coordinates of atoms in the two structures[43].

$$D = \sqrt{\frac{\sum_{i=1}^{n} \left| v_i - w_i' \right|^2}{n}} \quad ; w' = U(w + T) \tag{1.7}$$

For comparing models of complexes, a couple of variants of RMSD are used. One is the *interface RMSD*, which is the RMSD calculated over the interface residues. Interface residues are defined as any residue of one of the proteins, whose atom is within 10 A distance from an atom of any residue in the other protein[44]. The interface RMSD between model and reference is calculated for the interface residues alone. Another type of RMSD is the *ligand RMSD,* where the RMSD is calculated over the ligand residues, keeping the receptor protein fixed.

TM-score, which has been used in parts of this thesis, is a variant of RMSD [45]. It is normalized based on the length of the proteins compared, and is a score with bounds [0,1] with a score of 1 meaning that the two structures are identical. A score of 0.5 or higher indicates structural similarity between two compared folds.

A different metric is the *fraction of native contacts*[44]. A contact in a protein-docking model is a pair of interface residues in the receptor and ligand. Interface residues are defined as in the interface RMSD case. The fraction of native contacts is the number of interface residue-residue contacts in the model that are also in the native or reference structure.

1.6 CONTRIBUTIONS OF THESIS

In this thesis, algorithms for the second phase of docking, i.e. reranking are developed, for the rigid docking code DOCK/PIE [13]. The new code is named DOCK/PIERR (DOCK/PIE + Refinement, Reranking). Specifically, side chain remodeling and energy minimization are introduced to the rigid docking structures, and an atomic potential is developed and used to rerank the refined structures. The atomic potential is developed using mathematical programming and while the learning algorithm

is similar to previous work [37], its novelty lies in introduction of hierarchical constraints to model the energy landscape. Though atomic potentials are short-range and noisy on unbound structures, they capture a signal different from residue potentials, hence their addition was shown to improve the accuracy of reranking. This work is discussed in Chapter 2.

Secondly, a new hydrogen bond potential is developed in Chapter 3 for the reranking phase, using a learning framework similar to that of the atomic potential. Developing hydrogen bond potentials on unbound structures was challenging, as the signal from hydrogen bonding is weak [46]. Molecular dynamics was used to amplify the hydrogen bond signal. Comparison with other learning algorithms on soluble and membrane protein data sets showed that mathematical programming was the best performing algorithm, closely followed by Neural Networks. Differences in the algorithms and their performance are discussed. Hydrogen bond potentials alone were found to be more accurate than residue and atomic potentials on membrane proteins, whereas their signal was weaker on soluble proteins. In soluble proteins, hydrogen bonds between proteins have to compete with water, while this competition is not present in the hydrophobic membrane environment. Hence hydrogen bonds performed better on ranking models of membrane protein complexes.

Third, the docking package DOCK/PIERR was applied to predict the structure of membrane proteins, as discussed in Chapter 4. The reranking algorithm was modified to include an environment-based score that characterized the suitability of a docked pose for the membrane environment. This modified prediction algorithm was shown to be comparable to the state-of-the-art membrane complex prediction algorithms. It was then applied to characterize the dimers of amyloid precursor protein. Docking results showed good agreement with results from another computational method: implicit solvent

simulations. Differences in structures characterized in different membrane environments and structures characterized by different computational methods were discussed.

Fourth, comparison of DOCK/PIERR to other leading algorithms by independent community wide assessments is shown in Chapter 5. It was ranked as the 4th best performing automated docking method for the period 2009-2013 [47]. This was encouraging since it is a new method, compared to most other methods in the field, which have been around for 10+ years. The advances made in DOCK/PIERR help establish automated docking methods as accurate methods for structure prediction and enables departure from previous methods that rely more on human intervention.

Finally, in Chapter 6, well-known graph algorithms from Computer Science were applied for the problem of finding reactant to product paths in computational analyses of networks from simulation data produced using the method of Milestoning [48-51]. An efficient path algorithm based on Dijkstra's shortest path algorithm [52, 53] was discussed and applied to two molecular systems. Different network representations of Molecular Dynamics simulation data processed with Milestoning[49] were discussed, and networks based on local information were shown to uncover incorrect reaction mechanisms.

The contributions to this thesis from a computer science perspective are firstly, the introduction of hierarchical constraints into the learning algorithm for developing potentials for ranking models. And secondly, a comparison of the learning approach based on linear programming is made to other well-known machine learning algorithms like SVMs and neural networks, for the purpose of ranking docking models. It is shown that the linear programming based approach compares favorably to the other algorithms, in terms of accuracy, training and test time.

Chapter 2. Atomic potential for ranking docking models

2.1 Background and related work

As mentioned in Chapter 1, docking algorithms typically consist of two phases: a rigid docking and coarse-scoring phase and a refinement and reranking phase[9]. In this chapter, we introduce methods for the refinement and reranking phase in the docking algorithm DOCK/PIE[13]. In the refinement and reranking phase, the structures obtained in the first phase are usually adjusted and reranked using fine-grained energy terms. For example, RosettaDock uses an iterative Monte-Carlo search starting from rigid docking structures, first rebuilding side-chains of existing structures, and then minimizing the rigid structure of the two proteins using an elaborate energy term, the Rosetta potential [17, 18, 54]. Monte-Carlo approaches have also been used by others to incorporate rigid-body and side chain movements in refining docked conformations [15].

Weng and co-workers developed RDOCK[23], a refinement algorithm, which uses energy minimization and re-ranks models with a combination of electrostatics and knowledge-based potentials representing desolvation. They later developed a faster algorithm for the second step, ZRANK[29], that is a linear combination of a knowledge-based atomic potential, ACE, with electrostatic and van der Waals terms. Wolfson and co-workers developed the refinement algorithms, FIREDOCK[21], which incorporates restricted side-chain flexibility and orientation adjustments and its improved version, FIBERDOCK[22], which incorporates backbone flexibility using normal modes in addition to side-chain flexibility.

GRAMM-X uses conjugate gradient minimization with a smoothed Lennard-Jones type potential and ranks the models with a scoring function that is a combination of residue-based and atom-based terms[12]. The Cluspro team developed a refinement method using Monte-Carlo runs with semi-definite programming with underestimation

(SDU)[11, 28]. Fernandez-Recio and co-workers use hydrogen-bond network optimization along with energy minimization of all-atom force-fields in order to refine docking poses[27]. Zhou and co-workers perform a short minimization and restricted re-sampling near existing models followed by re-ranking using DFIRE and EMPIRE energy functions[25].

Most of the methods so far address the case in which the constituents do not undergo drastic conformational changes in the complex compared to the unbound state. In an analysis of 178 unbound complexes in our learning set, we find the average TM scores between the unbound and bound chains to be 0.8953 and 0.8875 for the receptor and ligand respectively. Hence in this study too, we consider cases in which no large-scale movements take place in the individual constituents.

The rest of this chapter discusses the methods for the refinement and reranking phase in DOCK/PIE. This phase consists of a minimal refinement step and a fine-grained reranking step. The minimal refinement step alters nominally, the models created in the first phase of docking, by means of side-chain remodeling and energy minimization. The reranking step ranks the altered models using a combination of fine-grained atomic potential, PISA[41] and coarse-grained residue potential, PIE [13]. While PIE has been developed previously for the coarse scoring phase, the development of PISA using mathematical programming, is introduced in this chapter. The docking algorithm DOCK/PIE with the added refinement phase is renamed as DOCK/PIERR (DOCK/PIE + Refinement & Reranking), and compares favorably to other leading docking packages like ZDOCK, Cluspro and PATCHDOCK, on the ZLAB 3.0 Benchmark and an independent set of 30 novel complexes. We also discuss that coarse-grained potentials are more robust than atomic potentials for unbound docking, perhaps because atomic potentials are more sensitive to local errors. Still, it is found that atomic and residue

potentials capture different signals and hence combining their scores provides a significantly better prediction than either score alone.

2.2 METHODS

Learning and Test Sets

For optimizing the parameters of PISA, a learning set of 640 complexes developed in previous studies[13, 38] was used. It contains 460 bound and 180 unbound complexes. The new methods were tested on three datasets. The first dataset comprises of 124 complexes from the ZLAB Benchmark 3.0[55], a standard benchmark test set used by the protein-protein docking community. The second dataset comprises of 640 targets from our learning set. The third dataset is a set of 30 novel complexes that is independent from the learning set, and details of this dataset are available in the Results section.

Rigid Docking

Given the chains of the receptor and ligand molecules, we used our previously developed docking package DOCK/PIE[13], to generate a training set for refinement. We retain top scoring 2^{19}=524,288 FFT-based transformations for each complex. These transformations are then clustered using ligand RMSD and scored using the potential, PIE[13], which consists of a pairwise residue contact term along with van der Waals attraction and repulsion terms. Subsequently, the top scoring transformations are filtered for clashes, and clustered again using interface RMSD.

Side chain remodeling

The top 1000 models from DOCK/PIE rigid docking were chosen for refinement and reranking. The number of models must be large enough to include a near-native

model, and small enough to make the refinement process efficient. Our choice of the number of models was based on an examination of 22 targets in our CAPRI dataset [56]. 19 of 22 targets had near-native models (acceptable by CAPRI definition) in the top 1000, while the number dropped to 16 considering the top 500 models. Out of the remaining 3 targets that did not have near-native models in the top 1000, one target had a near-native model at position 3500, and the other two did not have any near-natives in the output after clustering. The choice of 1,000 candidates therefore seems reasonable.

In order to reduce the number of clashes, make the rigid docking poses chemically sound, and improve the energies of the models, we first performed side-chain refinement using SCWRL4[57]. SCWRL is a side-chain prediction program that uses graph-based decomposition to identify the set of optimal rotamers for the side chains of a given model. We used a cutoff distance of 6 Å between the two proteins to identify interface residues and modeled the side chains of only the interface residues using SCWRL.

Minimization

After side chain remodeling, clashes were removed by 100 steps of conjugate gradient energy minimization. The minimization was performed using the routine mini_pwl (conjugate gradient descent with Powell restart) of the molecular dynamics package MOIL[58] and the OPLS-AA force field. During the minimization, the receptor and ligand molecules are modeled as rigid bodies. Minimization both in vacuum and using implicit solvent models (GBSA) was performed: but no difference in the results between the two procedures was observed. Therefore we decided to use minimization in vacuum since it is more efficient. Overall the refined structures are not more similar to the X-ray structures compared to the unrefined complexes, and distance of the refined

structure in terms of RMSD from the initial structure is minute (~ 0.2 to 0.4 Å). The refinement is nevertheless useful since it allows for better ranking.

Atomic potential

We designed a distance-dependent pairwise atomic potential (PISA) to re-rank the top 1000 refined structures. Using the atomic potential on the refined structures, we expect to generate more hits in the top 10 (or top 1) than the rigid docking procedure alone. The parameters for the atomic potential were learnt using mathematical programming from the top 1000 refined models of each complex in our learning set.

The heavy (non-hydrogen) atoms were collected into 32 types, as reported earlier for threading potentials[37]. We employed three distance bins: 2-3.5 Å, 3.5-5 Å and 5-8 Å, the same bins as in [37] to recognize approximate structures for threading.

Knowledge-based pairwise atomic potentials are frequently modeled by a square-well potential, i.e. designate a single value, $u(i,j,d)$ for a distance range, r, and atom-type pair (i,j). For clarity we replace the pair of interaction (i,j) by a single index α of the interaction type. If an atom of type i is found within a distance d from an atom of type j, then the value $u(\alpha,d)$ is added to the energy of the structure. The energy or score, $E(X)$ of a complex X, with a receptor A and ligand B, is a sum of all pairwise interactions and is given by Eq. (2.1), where $n(\alpha,d)$ is the number of interactions of type α (i.e. we use a single index to describe the interaction of particles i and j) at distance d.

$$E(X) = \sum_{\alpha,d} n(\alpha,d)u(\alpha,d) \qquad (2.1)$$

Instead of rectangular bins, a better accuracy was obtained when a linear interpolation was used between the bins, as shown in Figure 2.1. The three distance bins, 2-3.5 Å, 3.5-5 Å and 5-8 Å have one single parameter value in the flat regions in the

middle of the distance bins. The outer one-third portion of the distance bins adjacent to neighboring bins is modeled by a straight-line interpolation between the bins.

The corresponding equations for the geometrical factor, $n(\alpha,d)$ are given in Eq. (2.2). Note that the values of $n(\alpha,i)$ are fixed by the geometry of the structure. For every distance bin, $(i=1,2,3)$ we identify a different multiplicative energy term $u(\alpha,i)$. The formulation above led to $p=1584 = \frac{32(32+1)3}{2}$ parameters for the potential.

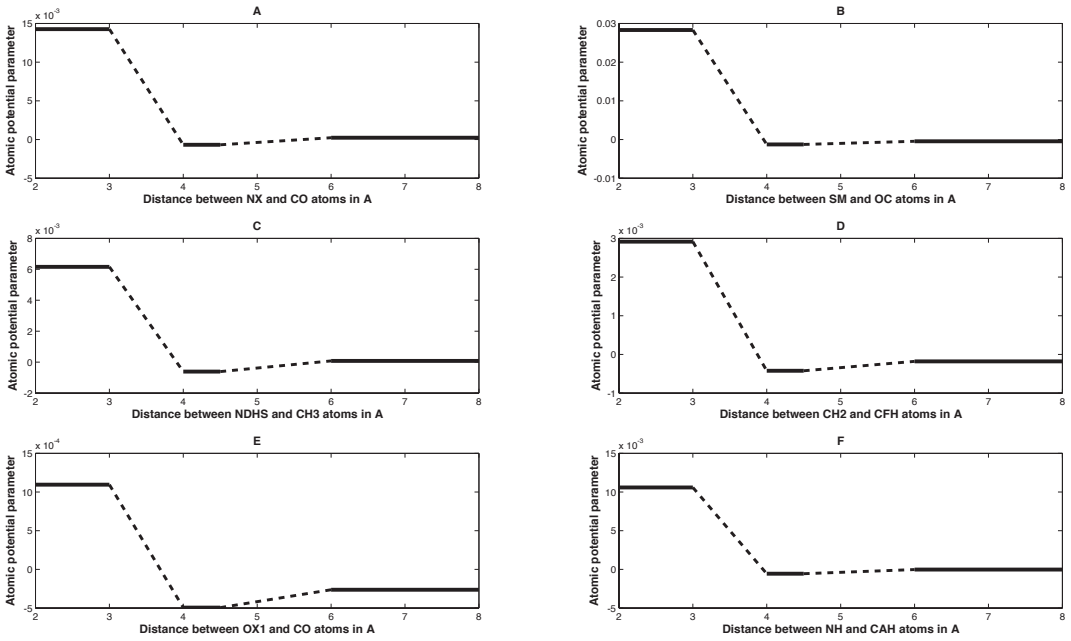


Figure 2.1 Value of $u(\alpha,d)$ for 6 different pairs of atom types: A) NX (LYS-NZ) and CO (carbon of backbone carbonyl). B) SM (MET-Sulfur) and OC (oxygen of carbonyl groups). C) NDHS (TRP-NE1) and CH3 (terminal aliphatic side chain carbon). D) CH2 (beta carbon) and CFH (aromatic side chain carbon). E) OX1 (ASP-OD1, OD2, GLU-OE1, OE2) and CO (carbonyl carbon). F) NH (amide nitrogen) and CAH (alpha carbon of amino acids).

$$
\begin{aligned}
& n(\alpha,1) = 1.0 && 2A \le r_{ab} < 3A \\
& \left\{ \begin{array}{l} n(\alpha,1) = 4.0 - r_{ab} \\ n(\alpha,2) = r_{ab} - 3.0 \end{array} \right\} && 3A \le r_{ab} < 4A \\
& n(\alpha,2) = 1.0 && 4A \le r_{ab} < 4.5A \\
& \left\{ \begin{array}{l} n(\alpha,2) = 4.0 - \frac{r_{ab}}{3.0} \\ n(\alpha,3) = \frac{r_{ab}}{3.0} - 3.0 \end{array} \right\} && 4.5A \le r_{ab} < 6A \\
& n(\alpha,3) = 1.0 && 6A \le r_{ab} < 8A
\end{aligned}
\qquad (2.2)
$$

Constraints used for learning

The parameters of the atomic potential were learnt taking into account known correctly and incorrectly docked structures. The energies of pairs of correct and incorrect structures are used to create inequalities of the type $E_{incorrect}$-$E_{correct}$>0 , as described in Chapter 1. The values of the parameters were obtained by solving the inequalities by linear programming using the LP solver, PF3[39].

In this study, we introduce hierarchical constraints for modeling the energy landscape of binding. More specifically the following types of inequalities are used:

a. Inequalities comparing near-native and misdocked models

'Near-native' models are those with an interface RMSD less than 2.5 Å to the native PDB structure. We call conformations 'misdocked' if they have an interface RMSD greater than 7 Å. We added the native structure to the set of near-natives for each target in our learning set. We then require that for each target, the atomic potential have a lower (better) energy for near-native models than for misdocked models, as shown in Eq. (2.3). Note that the set of models we consider here is restricted to the 1000 refined models of each complex.

$$E(X_{misdocked})-E(X_{near\text{-}native})>0 \qquad (2.3)$$

As illustrated in Chapter 1, the inequalities are linear in the parameters $u(\alpha,d)$ and hence, the inequalities can be solved efficiently by mathematical programming. In Eq. (2.3) the solution for the parameter set is up to a multiplicative positive constant λ. Hence, if $u(\alpha,d)$ is a solution so is $\lambda u(\alpha,d)$. It is therefore convenient to put the right hand side of the equation to 1 instead of zero. This choice sets a scale for the parameter values and makes the numerical calculation easier.

Further, we also allow for some errors in our solutions. It is not possible to satisfy all the inequalities because the functional form is not known exactly, and its current form is not flexible enough to solve all the constraints. On the other hand making the functional form more complex may lead to over-learning. New targets are obviously of more interest in practical applications and we aim for comparable performance on the training set and other targets. Therefore we remain with the simpler functional form while accepting some mis-classification.

The existence of mis-classifications is further amplified by the use of near-native structures as "correct" structures, instead of actual native complexes, and docking of unbound structures instead of bound structures of the individual chains in the complex. A near native structure as a target and the use of unbound chains mimics better the conditions of an actual prediction. However, it also increases the noise level and introduces uncertainties to the classification. Rather than the strict constraint in Eq. (2.3), we add to each inequality *i* a slack variable z_i. Eq. (2.3) then becomes:

$$E(X_{misdocked})-E(X_{near-native})>1-z_i \;;\; z_i > 0 \qquad (2.4)$$

b. Inequalities comparing high-quality hits and good hits

In the funnel shaped energy landscape described in Chapter 1, the distances between structures at the bottom of the funnel are smaller than distances between structures at the bottom and structures near the top. Hence, in order to provide a more

precise discrimination of the energies of near-native structures, we found it beneficial to add a new set of inequalities comparing near native structures with interface RMSD less than 1.5 Å compared to the native complex (call it high-quality hits) and hits with interface RMSD between 1.5 and 2.5 Å (known as good hits). The value of the cutoff 1.5 Å was obtained based on statistics of hits in the targets of the learning set. It was chosen such that there was an even distribution of models in the high-quality and good hit categories and the number of additional inequalities was maximized, as shown in Table 2.1. The new inequalities require that the high-quality hits will have lower energies than good hits.

$$E(X_{\text{good-hit}})-E(X_{\text{hq-hit}})>1-z_i \ ; z_i > 0 \qquad (2.5)$$

Table 2.1 Statistics of hits in the learning set

High-quality hit cutoff (hc) in Å	**Number of high quality hits (iRMSD < hc)**	**Number of good hits (hc<iRMS< 2.5 Å)**	**Number of resulting inequalities**
1.0	993	5416	40860
1.5	2341	4068	86930
2.0	4156	2253	79535

c. Inequalities comparing pairwise adjacent hits

In the third type of inequalities, we model the energy landscape by hierarchical inequalities. We sorted the hits (models with iRMSD less than 2.5 Å) by iRMSD, and formulated inequalities comparing energies of neighboring hits. For example, model *i* has lower iRMSD than model *i+1*. Therefore we expect the energy of the *i*th model to be lower than the energy of the model ranked *i+1*. Here n_{hits} is the total number of hits for a target in the learning set.

$$E(X_{\text{hit}}^{i+1})-E(X_{\text{hit}}^{i})>1-z_i \ ; i=1,2..n_{hits}-1; z_i > 0 \qquad (2.5)$$

We note that in principle, we could perform an all-vs-all comparison of docking models. However, the introduction of additional constraints not only increases the computational expense but also makes the inequality set more noisy [38]. This is because we use RMSD for ranking models, and models at higher values of RMSD are equally bad: a model with 10 Å RMSD is equally bad as a model with 12 Å RMSD.

Using these three types of inequalities, we had a total of 5,841,395 inequalities in our learning set. The complete set of constraints is now combined with an objective function that was minimized. The objective function is the sum of the parameters, $u(\alpha,d)$ and slack variables, z, where γ is an empirical constant that determines the weight of violations of the constraints relative to precise determination of the parameters.

$$\min\left|\sum_{\alpha,d} u(\alpha,d)\right| + \gamma\left\|\sum_{i} z_i\right\|_1 \qquad (2.6)$$

Using PF3[39], we solved 92.8% of the inequalities. We call the atomic potential PISA [Protein Interactions Scored Atomically] henceforth. We used the value of 1.0 for γ.

For each of the complexes in the learning set we mentioned previously, we used one thousand refined models along with the native structure for the complex to generate the three kinds of inequalities discussed above. For 67 of the complexes in the learning set (58 bound and 9 unbound), one or more backbone atoms in the PDB files were missing. The missing atoms prevent us from placing side chains or minimize continuous atomic energy using MOIL[58]. We attempted to add missing backbone atoms to the complexes using Modeller[59]. However, Modeller tends to move the modeled structure away from the template. We found that the results obtained by learning based on Modeller structures were worse than the results obtained by simply using the unrefined

(rigid-docking) models for complexes with missing atoms, as shown in Table 2.2. So for these complexes, we use the unrefined models for learning and testing.

Table 2.2 Performance with and without Modeller on 67 targets of the learning set. A hit is a model with an interface RMSD of 4 Å or less to the native.

Refinement method for targets with missing backbone atoms	**Number of hits in top 10/top1**	**Number of targets solved in top 10/top 1**
Modeller for modeling missing atoms, followed by SCWRL and minimization	81/15	38/15
Using unrefined models in case of missing main chain atoms	86/17	40/17

Combining atomic and residue scores for re-ranking

Though the atomic potential recognizes more hits in the top 100 than the previously developed residue based potential, PIE[38], it is not sensitive enough to recognize more hits in the top 10, or top 1. The reason for the lower performance of the atomic potential at the high end of prediction may be the more significant sensitivity of atomic interactions to structural details compared to interactions at the residue level. This sensitivity of the algorithm is amplified by the use of unbound (approximate) complexes rather than just bound complexes with atomically precise interactions.

Realizing that the atomic and residue potentials encapsulate different signals (atomic potentials captures shorter range interactions), we decided to combine the two, expecting the combined method to work better than the residue or atomic potentials alone. We used the following combination of potentials.

a. Product

For the atomic potential, PISA, the lower the energy, the better the model. Whereas for the residue score, PIE, higher the score, the better the model. Hence if we

take the product of the scores of PISA and PIE for each model, the lower the score of the product, the better the model should be.

$$C1 = PISA * PIE \tag{2.7}$$

b. Linear Combination

The second combination potential was a weighted linear combination of the atomic and residue potentials. The value of coefficient *a* was set to -0.2 by learning on the learning set.

$$C2 = PISA + a.PIE \tag{2.8}$$

c. Linear Combination with Product

Adding the individual values of the atomic and residue potentials to their product gave yet another potential.

$$C3 = c.PISA + d.PIE + PISA * PIE \tag{2.9}$$

Values of c and d, were found to be 0.1 and -0.8 respectively, as shown in Figure 2.2. The height represents the number of targets in the learning set with a hit (interface RMSD < 4 Å) in the top 10, for the combination of coefficients of the potential C3. The best values appear to be 0.1 for *c*, the atomic potential weight and -0.8 for *d*, the residue potential weight.

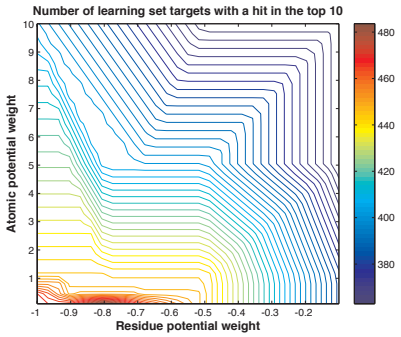


Figure 2.2 Contour plot showing parameter search for values of coefficients *c* and *d* in equation 11.

Tests on other docking packages

We compared our results to the ZDOCK[60], ZDOCK+ZRANK[29], CLUSPRO[11] and PATCHDOCK+FIBERDOCK[22] methods. For ZDOCK results on the ZLAB dataset, we used the latest version ZDOCK-3.0.2 with 6 degree Euler angle sampling and the results (RMSDs) as reported in the ZLAB website. For the other test sets, we used the downloaded packages for ZDOCK version 3.0.2 and ZRANK. For ZDOCK+ZRANK, we used the top 2000 conformations from ZDOCK as recommended[29]. We then added polar hydrogens to the models using SCWRL4[57] and reranked the models using ZRANK. For CLUSPRO, we used the results from the CLUSPRO server[11] which runs CLUSPRO version 2.0. We used the downloaded packages for PATCHDOCK and FIBERDOCK. FIBERDOCK is shown to be a refinement and re-ranking method over the same group's FIREDOCK. We used the top 500 models from PATCHDOCK as suggested[22] and refined the backbone and side chains of the models using FIBERDOCK, and re-ranked models with the FIBERDOCK energy term. The packages ZDOCK and CLUSPRO perform only rigid docking and no refinement or rescoring, and are meant to enrich the number of hits in the top 1000 or 2000 structures. It is interesting to note that ZDOCK in our hands scores better than ZDOCK+ZRANK.

2.3 Results and discussion

Creation of Test Sets

Performance of DOCK/PIERR (pronounced DOCK-by-PIER) was tested on three datasets. The first dataset comprises of 124 complexes from the ZLAB Benchmark

3.0[55]. The second dataset comprises of 640 targets from our learning set, described in the Methods section and used in previous work[38].

The third dataset is a set of 30 complexes that is independent from the learning set. These are a set of complexes that were deposited in the Protein Data Bank after September 22, 2010. To construct this test set, we queried the Protein Data Bank for soluble two-chain protein-protein complexes, with no DNA, RNA and free ligands in the structure. We discarded complexes with modified residues, and chains that were shorter than 50 residues in length. The query resulted in 126 complexes.

We then tested to see if these complexes were similar to any of the complexes in the learning set. We used TM-Align[45] on the individual chains of the bound complex and discarded all complexes where both chains had a TM score of 0.5 or higher with the chains of a target in the learning set. 45 of the complexes were not similar to any in the learning set. To perform unbound-unbound docking, we searched for homologs for the individual chains of these 45 complexes using PSI-BLAST[61]. We discarded the complexes with no homologs (BLAST expectation cutoff of 10e-3) for either chain.

Then we constructed a homology model for each chain, using the structure of the homolog and the sequence of the chain from the bound complex. We used MODELLER[62] to build the homology model and discarded complexes where the homology models had a TM score of less than 0.8 with the chain in the bound complex. We obtained a set of 30 complexes from this procedure, with 22 having both chains unbound and 8 with one chain unbound. Table 2.3 contains the list of homologs used for the unbound docking of novel complexes.

Table 2.3 List of unbound complexes in the novel test set of 30 targets along with the corresponding homologs used to model the receptor and ligand.

Target PDB ID	Receptor Homolog (PDB_chain) : Target Receptor Chain	Ligand Homolog (PDB_chain) : Target Ligand Chain
2xt4	2XT2_A:B	2XT2_A:A
2xty	2XTW_A:B	2XTW_A:A
3agx	3AGZ_A:A	3AGZ_A:B
3asy	1XRJ_A:A	1XRJ_A:B
3gt6	3GLA_A:A	3GLA_A:B
3lis	3LFP_A:A	3LFP_A:B
3m7f	3B7Y_A:B	1NRV_A:A
3mxj	3MXI_B:B	3MXI_B:A
3nfy	1T8P_A:B	1T8P_A:A
3oa9	3D6R_B:A	3D6R_B:B
3p2q	3KV7_A:A	3KV7_A:B
3pc6	3PC8_A:B	3PC8_A:A
3pge	3PGE_A:A	3L0W_A:B
3pra	3PRB_A:B	3PRB_A:A
3r8c	3R20_A:A	3R20_A:B
3rd6	3Q64_A:A	3Q64_A:B
3rkc	3HAG_A:B	3HAG_A:A
3t43	3LF6_A:A	3LF6_A:B
3te8	3LR5_A:B	3LR5_A:A
3u80	2UYG_A:A	2UYG_A:B
3umz	3UN0_B:A	3UN0_B:B
3vc8	3VCB_A:B	3VCB_A:A
2wfx	3HO4_B:B	2IBG_H:A
3d65	3D65_E:E	3BTM_I:I
3di3	3DI3_B:B	3DI2_C:A
3hct	1FXT_A:B	3HCT_A:A
3jrq	2IQ1_A:A	3JRQ_B:B
3l1z	3FSH_B:A	3L1Z_B:B
3m18	3M18_A:A	1I56_A:B
3nbp	1MU2_A:A	3NBP_B:B

Results on the ZLAB Benchmark

In the following tables we analyze the results by two metrics: interface RMSD and fraction of native contacts, as defined by the CAPRI assessment[44]. A hit defined in terms of interface RMSD, is a model with interface RMSD less than 4 Å, to the crystal structure of the complex, which is equivalent to an "acceptable" model in the CAPRI assessment. Similarly, a hit in terms of fraction of native contacts is a model with 10 percent or more native contacts, which is one of the criteria for an acceptable model in CAPRI.

We show in Table 2.4, the comparison of our docking software with ZDOCK, ZDOCK+ZRANK, CLUSPRO and PATCHDOCK+FIBERDOCK. We compare the performance of DOCK/PIE our rigid docking package, with the new DOCK/PIERR, which is DOCK/PIE with side chain remodeling, energy minimization and reranking. Reranking is done in various ways, using the atomic potential PISA alone, or the combination potentials, C1, C2 and C3, composed of the atomic and residue potentials.

In Table 2.4 and the succeeding results tables, the number of hits counts all acceptable predictions. Some of the targets can have multiple successful predictions, and all of these hits are counted in the entry "Number of hits". A target is considered solved when at least one prediction is in the top 1 or top 10 set. Only one hit per target is counted under "Number of targets solved".

DOCK/PIERR with C1 and C2 combination potentials performs better than the other DOCK/PIE versions. DOCK/PIERR picks a smaller number of hits than ZDOCK or ZDOCK+ZRANK in the top 10. However, DOCK/PIERR and its various versions are able to solve more targets than ZDOCK. ZDOCK is able to generate a lot of reasonable models for some targets. However, for some targets it does not generate hits at all. DOCK/PIERR is more uniform in the generation of hits. CLUSPRO is one of the best

methods, even though we include the results from the web server only, which does not include the more expensive refinement procedure.

Table 2.4 Comparison on the ZLAB 3.0 benchmark set of 124 targets.

Method	Interface RMSD		Fraction of Native Contacts	
	Number of hits in top 10/top1	Number of targets solved in top 10/top1	Number of hits in top 10/top1	Number of targets solved in top 10/top1
DOCK/PIE Rigid Docking	73/10	38/10	144/14	59/14
DOCK/PIERR Rerank with PISA	86/17	40/17	167/28	66/28
DOCK/PIERR Rerank with C1	107/19	50/19	190/32	72/32
DOCK/PIERR Rerank with C2	107/19	50/19	194/30	72/30
DOCK/PIERR Rerank with C3	102/15	46/15	175/23	63/23
CLUSPRO	63/19	50/19	172/31	69/31
ZDOCK	143/13	29/13	276/22	45/22
ZDOCK+ZRANK	96/12	23/12	208/26	50/26
PATCHDOCK + FIBERDOCK	21/2	15/2	56/4	33/4

Results on the novel set

Table 2.5 shows the comparison of DOCK/PIERR and other docking software on the novel set. For ZRANK, the authors recommend it to be used on the top 2000 models from ZDOCK. Besides applying ZRANK on the top 2000 models, we also applied ZRANK to the top 1000 models from ZDOCK, since we use the top 1000 models from our rigid docking procedure for reranking. We did similarly for FIBERDOCK, which is to be applied on the top 500 models from PATCHDOCK. CLUSPRO and ZDOCK have not been used so far for docking of homology models. Here, we are using homology

modeling to mimic a "real" docking experiment in which the bound structures are not known.

Table 2.5 Comparison of docking software on the novel set of 30 targets. Suffix of 1000 for example, means that the re-ranking was applied to top 1000 models from rigid docking.

Method	Interface RMSD		Fraction of Native Contacts	
	Number of hits in top 10/top 1	Number of targets solved in top 10/top 1	Number of hits in top 10/top 1	Number of targets solved in top 10/top 1
DOCK/PIE Rigid Docking	37/10	16/10	69/14	20/14
DOCK/PIERR Rerank with PISA	33/7	12/7	50/10	17/10
DOCK/PIERR Rerank with C1	41/7	15/7	70/11	21/11
DOCK/PIERR Rerank with C2	43/9	17/9	75/12	21/12
DOCK/PIERR Rerank with C3	44/10	17/10	72/12	22/12
ZDOCK	39/9	11/9	52/11	14/11
ZDOCK+ZRANK-2000	34/5	10/5	55/9	15/9
ZDOCK+ZRANK-1000	38/5	11/5	60/8	14/8
CLUSPRO	19/8	12/8	48/9	16/9
PATCHDOCK+ FIBERDOCK-500	18/4	5/4	32/4	11/4
PATCHDOCK + FIBERDOCK-1000	17/3	5/3	28/3	10/3

Figure 2.3 shows some of the models produced by different methods on the novel set. Since the chains are unbound-unbound there is a slight deviation between the receptor chains in the native and model. Table 2.6 shows the comparison of different docking methods on individual targets in the novel set. We see that targets that are hard for DOCK/PIE are generally also hard for the other docking packages. But there are some targets like 3asy, 3r8c and 3rd6, where the only software that was able to produce a hit in

the top 10 was DOCK/PIE (RR). For 3hct, only ZDOCK+ZRANK was able to produce a hit. For 3d65 and 3nbp only CLUSPRO was able to produce a hit in the top 1.

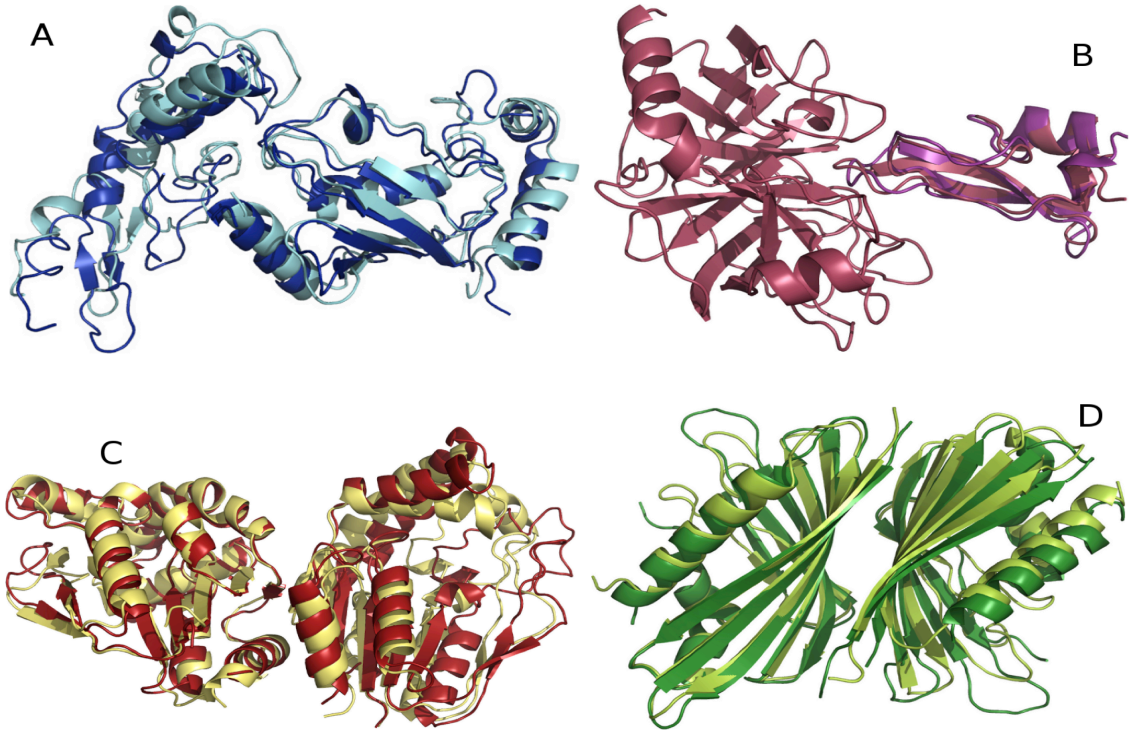


Figure 2.3 Models from three docking algorithms on complexes in the novel set. A) Native structure of 3hct (in blue) along with the best model produced for this complex, by ZDOCK+ZRANK (in cyan). B) Native structure of 3d65 (in purple) along with the best model by Cluspro (in raspberry). C) Native structure of 3asy (in brick red) superposed with the best model by DOCK/PIERR (in yellow). D) Native structure of 3rd6 (in dark green) superposed with the best model by DOCK/PIERR (in lemon yellow).

Table 2.6 Top 10 and top 1 hits per novel set target. D/P Rigid: DOCK/PIE. D/P PISA: DOCK/PIE with PISA. D/P CX: DOCK/PIE with combination potential CX. ZD: ZDOCK, CL: CLUSPRO. Suffix [N] implies reranking was applied to top N models. ZR [N]: ZDOCK+ZRANK & ZRANK applied to top N models from ZDOCK, PF [N]: PATHDOCK+FIBERODCK & FIBERDOCK applied on top N models from PATHDOCK.

Targets	**Number of irmsd hits in the top 10/top 1 for various docking software**										
	D/P Rigid	**D/P PISA**	**D/P C1**	**D/P C2**	**D/P C3**	**ZD**	**ZR 2000**	**ZR 1000**	**CL**	**PF 500**	**PF 1000**
2xt4	3/1	4/0	4/1	3/1	4/1	1/0	1/0	1/0	1/0	0/0	0/0
2xty	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0
3agx	3/1	3/1	3/1	3/1	3/1	7/1	6/1	7/1	1/1	4/1	4/1
3asy	2/0	3/1	4/0	3/0	3/0	0/0	0/0	0/0	0/0	0/0	0/0
3gt6	1/0	2/1	2/0	2/0	1/0	2/1	4/1	4/1	1/0	0/0	0/0
3lis	4/1	5/1	5/1	4/1	4/1	4/1	8/1	9/1	3/1	6/1	5/1
3m7f	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0
3mxj	1/0	0/0	1/1	1/0	1/1	0/0	1/1	1/1	2/1	0/0	0/0
3nfy	3/1	3/1	3/0	3/1	4/1	7/1	6/1	6/1	2/1	1/1	1/0
3oa9	1/1	4/0	3/1	2/1	2/1	7/1	2/0	3/0	1/1	0/0	0/0
3p2q	1/1	1/1	1/1	1/1	1/1	4/1	3/0	3/0	2/1	6/1	6/1
3pc6	2/1	1/0	1/0	2/1	2/1	1/1	0/0	0/0	0/0	0/0	0/0
3pge	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0
3pra	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0
3r8c	1/1	0/0	0/0	1/0	1/0	0/0	0/0	0/0	0/0	0/0	0/0
3rd6	3/1	3/1	3/1	3/1	3/1	0/0	0/0	0/0	0/0	0/0	0/0
3rkc	2/0	1/0	3/0	3/1	3/1	1/0	1/0	1/0	1/0	1/0	0/0
3t43	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0
3te8	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0
3u80	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0
3umz	6/0	3/0	5/0	6/0	6/0	1/1	2/0	2/0	1/0	0/0	0/0
3vc8	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0
2wfx	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0
3d65	2/0	0/0	1/0	2/0	2/0	0/0	0/0	0/0	3/1	0/0	0/0
3di3	2/1	0/0	2/0	3/0	3/0	4/1	0/0	0/0	0/0	0/0	1/0
3hct	0/0	0/0	0/0	0/0	0/0	0/0	0/0	1/0	0/0	0/0	0/0
3jrq	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0
3l1z	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0
3m18	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0
3nbp	0/0	0/0	0/0	1/0	1/0	0/0	0/0	0/0	1/1	0/0	0/0

On the novel set, DOCK/PIE with the residue potential seems to perform better than DOCK/PIERR with the atomic potential. DOCK/PIE rigid docking and DOCK/PIERR with potential C3 performs better than all other docking methods. Again, performance of ZDOCK is superior to ZDOCK+ZRANK. Also, ZRANK applied to top 1000 models seems to be better than the authors' recommendations of applying it on the top 2000 models. For FIBERDOCK, the author recommendation of applying it on the top 500 models seems to work better.

Results on the learning set

We report in Table 2.7, the performance of DOCK/PIE and various flavors of DOCK/PIERR on the learning set of 640 complexes.

Table 2.7 Comparison of DOCK/PIE and DOCK/PIERR on the learning set of 640 complexes.

Method	Interface RMSD		Fraction of Native Contacts	
	Number of hits in top 10/top1	Number of targets solved in top 10/top1	Number of hits in top 10/top1	Number of targets solved in top 10/top1
DOCK/PIE Rigid Docking	1646/376	466/376	2152/400	503/400
DOCK/PIERR Rerank with PISA	1764/334	459/334	2197/365	494/365
DOCK/PIERR Rerank with C1	2028/410	482/410	2486/433	508/433
DOCK/PIERR Rerank with C2	2024/411	477/411	2483/435	507/435
DOCK/PIERR Rerank with C3	2003/413	487/413	2487/430	520/430

The combination potentials generally perform better than the atomic potential, PISA alone, on all three datasets. Besides, Table 2.7 suggests that the atomic potential PISA seems to recover more hits in the top 10 than the residue potential in DOCK/PIE. But it is not as sensitive as the residue potential in DOCK/PIE when it comes to solving

more targets in the top ten, or top one. In other words, the atomic potential developed here is more useful for enriching the hit candidates than for sensitive prediction of hits from a small set of models.

Atomic potentials may be more sensitive to noise in the learning set. One source of noise in learning is the use of unbound complexes. In order to test this hypothesis that atomic potentials perform better on bound complexes than unbound, we compared the performance of DOCK/PIE rigid docking with the PIE potential, which includes residue based and van der Waals terms, with DOCK/PIERR reranking with the atomic potential. The results in Table 2.8 show that this hypothesis is not supported, since the atomic potential PISA has a higher percentage of solved targets for the unbound complexes than for the bound complexes. The atomic potential is also better than the residue potential on the unbound complexes. Hence we still do not know what makes atomic potentials less sensitive.

Table 2.8 Comparison of DOCK/PIE rigid docking and DOCK/PIERR on 460 bound and 180 unbound complexes in the learning set.

Type of complexes in dataset	**Docking Method**	**Interface RMSD**	
		Number of hits in top 10/top1	**Percentage of targets solved in top 10/top1**
Bound	DOCK/PIE Rigid Docking	957/278	74.1/60.4
Bound	DOCK/PIERR with PISA	915/228	69.5/49.5
Unbound	DOCK/PIE Rigid Docking	689/98	71/55.6
Unbound	DOCK/PIERR with PISA	849/106	78.9/60.2

Residue potentials are possibly more robust and are better able to capture enough of the overall structural features to recognize near-natives from a small set of models. Hence we use potentials that combine atomic and residue scores, hoping that they will be more robust, correlate well with RMSD, and enrich hits in the model set.

Run times

Approximate run times for DOCK/PIERR for different protein sizes are shown in Table 2.9. We estimate that other software packages we compared to in the present study are faster than DOCK/PIERR by a factor of about 10. So far we have focused our attention on getting higher accuracy and we did not focus on speed. ZDOCK is using essentially the same rigid docking algorithm as DOCK/PIERR (FFT) so we are hopeful that appropriate optimizations could be found. For example, DOCK/PIERR uses at present, double precision floating-point number in FFT calculations while ZDOCK uses only single precision numbers.

Table 2.9 Approximate run-times for DOCK/PIERR for different protein sizes. All runs were on 4 nodes of a Linux cluster with 8 cores each (32 cores total). Each core was an Intel Xeon X5460 processor with clock speed of 3.16 GHz. The memory size was 16GB for each node.

Receptor Length (number of residues)	**Ligand Length (number of residues)**	**Approximate run time in hours**
105	105	1.25
202	200	1.5
418	152	4.75
272	174	5.75
554	400	9

Analysis of the new atomic potential

On solving the inequalities generated from the learning set, we can calculate for each target in the learning set, the percentage of inequalities of that target that were not satisfied by the linear programming solution. Figure 2.4 shows the distribution of violations among targets in the learning set. We observe that there are a relatively small number of targets that contribute a large number of violated constraints.

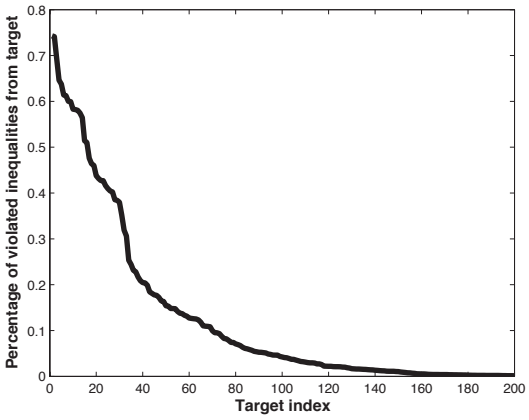


Figure 2.4 Percentage of violated inequalities for 200 targets of the learning set. The rest of targets are not shown as they have a negligible number of violated inequalities.

Some targets can be hard to dock if they have a very small number of native interface contacts. We show in Figure 2.5, the correlation between the number of native contacts in the target and the percentage of inequalities that were violated for that target. It is observed that the targets with low number of contacts have a high percentage of violated inequalities.

To assess the extent of redundancy among the inequalities in the linear program, we calculated the cosine of the angle between any two inequality vectors (the vectors are a function of α and d and of the form $[n_{misdocked}(\alpha,d) - n_{near-native}(\alpha,d)]$) and obtained the distribution of the cosine values. We did this for three different samples of inequality vector pairs sampled at random from the inequalities in our linear program: 1500, 2500 and 3500 pairs of inequalities. Figure 2.6 shows the distribution of cosine values, peaked around 0.0, which shows that a significant percentage of inequalities were orthogonal to each other. This suggests that most of the constraints offer new information and are independent of each other.

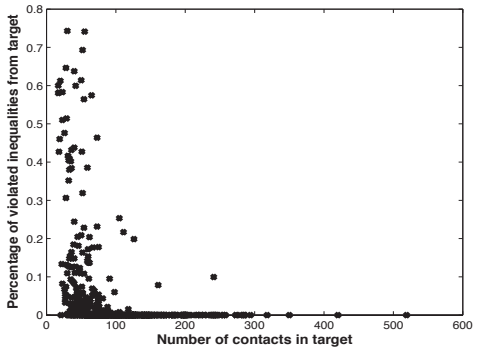


Figure 2.5 Percentage of violated inequalities per learning set target plotted against the number of contacts for the target.

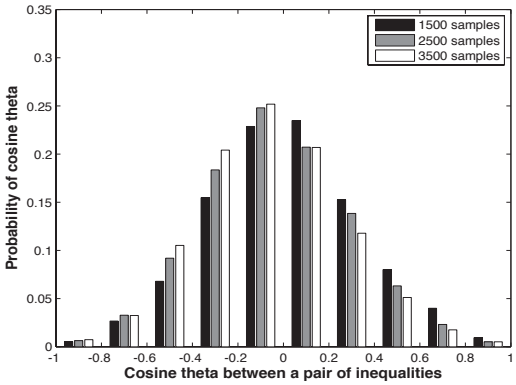


Figure 2.6 Percentage of violated inequalities per learning set target plotted against the number of contacts for the target.

Atomic and Residue Potentials on Refined and Unrefined Models

Here we explore the performance of atomic potentials on rigid docking models, as opposed to refined models. If we could obtain the same performance of atomic potentials on rigid docking models as on refined models, then the extra computational expense of side chain refinement and minimization can be avoided.

Figure 2.7 shows that the atomic potential works best when the parameters of the potential are learnt and applied to refined models. It has a better recognition capacity if trained and tested on refined models, compared to training and testing on the rigid

docking models. Interestingly, the potential trained and tested on unrefined structures is much worse than residue based rigid docking. Finally, the coarse-grained potential, PIE, performs worse when tested on refined structures, than when tested on unrefined structures. This is probably because it was trained on unrefined structures.

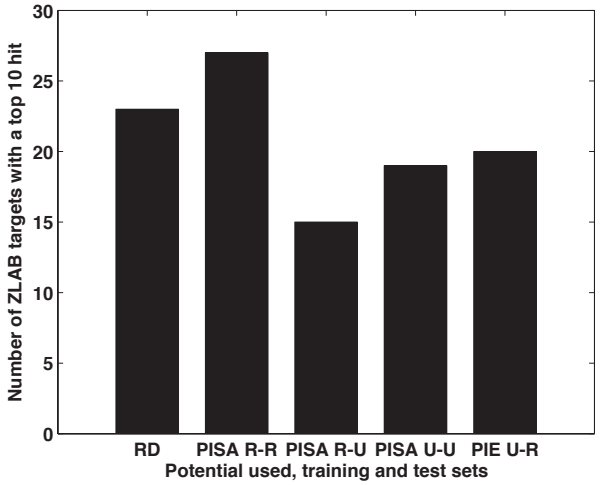


Figure 2.7 Performance of atomic potential PISA on refined and unrefined models. For each ranking method, the number of ZLAB targets with a hit with interface RMSD less than 2.5 A, in the top 10 models is shown. Abbreviations are: RD: Rigid Docking, U: unrefined, R: unrefined. PISA-R-U for example, means that the re-ranking potential was PISA, which was learnt on refined learning set models and tested on unrefined ZLAB structures.

Examining hit cutoffs and distance bins for the atomic potential

Tables 2.11-2.14 show numerical experiments on different definitions of near-native and misdocked conformations and different distance bins for the ZLAB benchmark and the training set. All these different variations of the atomic potential were learnt on the training set and tested for performance on the ZLAB set. The best performing definition on the ZLAB set was chosen for the actual potential, PISA.

Models were classified as near-native if they had interface RMSDs lower than the hit cutoff and were classified as misdocked models if they had an RMSD higher than the

misdocked model cutoff. Near-native cutoffs of 4 A and 2.5 A were tried and misdocked model cutoff of 5.0, 6.0 and 7.0 A were tried and we chose near-native cutoff of 2.5 A and misdocked model cutoff of 7 A.

We also experimented with different distance bins for the pairwise atomic potential, PISA. We show that the distance bin 2-3.5; 3.5-5; 5-8A works best. Incidentally these are also the distance bins used for detecting approximate structures for threading[37].

Table 2.10 Comparison of different cutoffs for near-native or misdocked complexes on the ZLAB benchmark of 124 complexes.

Cutoff for near-natives and misdocked models		**Interface RMSD**	
Near-native cutoff in A	**Misdocked model cutoff in A**	**Number of hits in top 10/top1**	**Number of targets solved in top 10/top1**
2.5	7.0	86/17	40/17
4.0	7.0	81/15	39/15
2.5	6.0	86/16	39/16
2.5	5.0	82/15	38/15

Table 2.11 Comparison of different cutoffs for near-native or misdocked complexes on the learning set of 640 complexes.

Cutoff for near-natives and misdocked models		**Interface RMSD**	
Near-native cutoff in A	**Misdocked model cutoff in A**	**Number of hits in top 10/top1**	**Number of targets solved in top 10/top1**
2.5	**7.0**	**1764/334**	**459/334**
4.0	7.0	1424/255	305/255
2.5	6.0	1510/280	414/280
2.5	5.0	1627/292	393/292

Table 2.12 Comparison of different distance bins on the ZLAB benchmark of 124 complexes.

Distance bins used	Interface RMSD	
	Number of hits in top 10/top1	Number of targets solved in top 10/top1
2-3.5; 3.5-5; 5-8 A	**86/17**	**40/17**
3.5-5; 5-6.5; 6.5-8 A	86/13	39/13
2-4.5; 4.5-6; 6-8 A	85/13	38/13
2-3; 3-4; 4-5; 5-8 A	83/16	36/16
2-3.5; 3.5-5; 5-8; 8-10 A	84/18	38/18

Table 2.13 Comparison of different distance bins on the learning set of 640 complexes.

Distance bins used	Interface RMSD	
	Number of hits in top 10/top1	Number of targets solved in top 10/top1
2-3.5; 3.5-5; 5-8 A	**1764/334**	**459/334**
3.5-5; 5-6.5; 6.5-8 A	1310/278	367/278
2-4.5; 4.5-6; 6-8 A	1468/295	389/295
2-3; 3-4; 4-5; 5-8 A	1548/272	361/272
2-3.5; 3.5-5; 5-8; 8-10 A	1630/310	404/310

2.4 CONCLUSIONS

We have introduced an improvement to docking algorithms by introducing a new atomic potential and refinement and ranking algorithms. The refinement is small (~0.2 Å RMSD) and does not result in significant changes to the structure, however it makes the structures more chemically reasonable and improves the quality of the obtained potential. We show by extensive tests on three datasets of complexes that our methods outperform slightly, other state-of-the-art docking packages. We also observe that coarse-grained potentials are more robust to inaccurate structures produced by unbound docking. Nevertheless, we show that atomic and residue potentials capture different signals, and hence their combination works better than either of them individually. However, the

success rate of docking software even after refinement and improved reranking functions is still between 30 and 50%. One could envision designing multi-body potentials, orientation based potentials and potentials that are based on hydrogen bond interactions to capture more structural features that may lead to more accurate scoring functions and improve the success of computational docking procedures. We investigate one of these avenues, i.e. hydrogen bonding interactions, in the next chapter.

Chapter 3. Hydrogen bond potentials: comparison of learning algorithms and tests on soluble and membrane proteins

3.1 INTRODUCTION

In this chapter, we develop potentials describing hydrogen bond interactions in protein docking models [63]. The parameters of the potential are developed using different learning algorithms: pairwise learning using linear programming, linear and non-linear SVMs and neural networks. The same method referred to as linear programming in the previous chapter is called pairwise learning using linear programming, in this chapter, to distinguish it from linear SVMs. The distinction is explained in the section on comparison of learning algorithms.

We show that pairwise learning using mathematical programming has the best overall performance in terms of accuracy and training and test times, followed closely by neural networks. To see if the new hydrogen bond potentials improve the accuracy of reranking docking models, we additionally combine the hydrogen bond potentials from different learning methods with the residue and atomic contact potentials discussed in Chapter 2.

In chapter 2, the learning and test sets involved water-soluble protein complexes, or protein complexes formed in the aqueous solution in cells. In this chapter, we test also on protein complexes formed in a different environment. Transmembrane complexes are formed in the cell membrane, which is a hydrophobic environment.

For soluble complexes, hydrogen bond potentials alone possess a weaker signal for reranking, compared to interface atomic and residue potentials. However for transmembrane complexes, hydrogen bond potentials alone provide a better recognition capacity than residue and atomic potentials. We surmise that the weak nature of the hydrogen bond potentials for soluble proteins is possibly due to competition of interfacial

hydrogen bonds with water. Whereas in transmembrane proteins, the signal from hydrogen bonding is stronger possibly owing to the hydrophobic membrane core and lack of competition with water molecules.

Nevertheless, the addition of hydrogen bond potentials to atomic and residue potentials is shown to boost their reranking accuracy, for both soluble and membrane protein complexes.

3.2 BACKGROUND

A hydrogen bond is a non-covalent short-range electrostatic interaction between an electro-negative atom (called acceptor) and a polar hydrogen atom covalently bonded to another electro-negative atom (called donor). Here we examine hydrogen bonds formed across protein interfaces i.e. when the hydrogen and donor are from one protein and the acceptor is from the interacting protein. Hydrogen bonds formed at the interfaces of interacting proteins in biologically active complexes can play a role in stabilizing the interaction further, and also influence the choice of binding partner protein[46, 64]. A statistical examination of experimentally determined complexes revealed that the average protein-protein interface has around 10 hydrogen bonds [65, 66], [46]. However the role of hydrogen bonding in protein interactions is unclear.

The question we seek to answer is whether hydrogen bonding constitutes useful signal for model discrimination in protein docking. In [64], the authors find encouraging improvement in model discrimination for protein docking by using a hydrogen bond potential for ranking models. However their approach was tested on the easier case of bound docking and it is not known whether the results would be as good in the more realistic case of unbound docking. [66, 67] on the other hand report that the number of

intermolecular hydrogen bonds was not a very good predictor of a near-native model. We wish to know whether hydrogen bonding can enhance the signal of traditional atomic and residue potentials for reranking protein-protein docking models, and if so, by how much.

In protein-protein docking algorithms, hydrogen bonding is usually optimized as part of short-range electrostatic interactions. Examples are ZRANK[29], DOT[68], and FTDOCK [69]. However, a couple of docking algorithms use separate hydrogen bond potentials. Rosettadock [16, 17] uses a hydrogen bond potential [64, 70] that is based on propensities of hydrogen bonds observed in interfaces of experimental complexes. This potential contains both distance dependent and angle-based terms, and the coordinates of three atoms are considered: the polar hydrogen, donor atom that the polar hydrogen is covalently bonded to, and the electro-negative acceptor atom across the interface that the polar-hydrogen hydrogen bonds with [64, 70]. [71, 72] model the hydrogen bonding energy with spherical Gaussians centered at putative donor and acceptor positions. The optimization of hydrogen bond networks combined with all-atom force fields was also used to improve ranking of docking models [27].

In this work, we develop hydrogen bond potentials for unbound docking. Here, we formulate hydrogen bond potentials using several learning algorithms, and test their accuracy in improving reranking docking models of soluble and membrane protein complexes.

3.3 METHODS

Datasets of protein complexes

We use the same learning set used in Chapter 2, a set of 640 PDB complexes used in prior work [38], comprising of mostly soluble protein complexes and a small

percentage (5%) of membrane protein complexes. For soluble protein complexes, two test sets were used. One was the ZLAB 4.0 Benchmark set [41, 73] of 176 complexes: an extended version of the ZLAB 3.0 Benchmark used in Chapter 2. The other was an extended version of the novel test set described in Chapter 2. In this study, it was extended from 30 to 52 unbound complexes. The datasets are not large; however, they are representative of the non-redundant complexes available in the PDB today.

The additional 22 targets in the novel set were selected in a manner similar to the first 30. The PDB was queried for new protein-protein complexes (not membrane-based, not containing DNA/RNA) released between Feb 2012 and Aug 2013. A 70% sequence identity cutoff was used and 181 targets were identified. Of these, complexes containing peptides (monomer length less than 60 residues) were discarded and 126 complexes remained. These complexes were then tested for similarity to complexes in the learning set. The test for similarity was performed by comparing the receptor and ligand monomers of a new complex to the receptor and ligand monomers respectively, of every learning set complex, using TM-score [45]. For only 50 of the 126 complexes, both the receptor and ligand monomers were dissimilar (TM score less than 0.5) to the monomers of complexes in the learning set. PSIBLAST [61, 62] was then used to obtain homologs in the PDB for the individual monomers of those 50 complexes. 35 complexes had homologs (e-value greater than 0.001) for at least one monomer. Modeller [62] was then used to obtain homology models for the monomers, given the template from PSIBLAST. Homology models from Modeller that were dissimilar (TM score less than 0.8) to the bound structure of the monomer were discarded. For 22 targets, Modeller produced a model close to the bound structure, (TM score greater than 0.8), for both receptor and ligand. These 22 complexes were the set of additional targets added to the novel set, of which 10 are homodimers and 12 heterodimers. They are summarized in Table 3.1.

For membrane protein complexes, a test set of 30 unbound membrane protein complexes that was created in another study was used: this study on membrane proteins is described in Chapter 4. The hydrogen bonds for membrane complexes included those in the residues inside the hydrophobic core of the membrane as well.

Table 3.1 New set of 22 targets added to the independent novel test set of soluble protein complexes. Listed are the PDB chains used as receptor and ligand, along with the corresponding template used to obtain homology models (unbound structures) for docking.

Target PDB ID	**Receptor chain: Homolog**	**Ligand chain: Homolog**
2LYJ	A:1UTX_A	B:1UTX_A
2M0G	A:2M0G_A	B:1OPI_A
2Y9P	A:2Y9M_A	B:2Y9M_B
2YML	A:2NO4_A	B:2NO4_A
3TG1	A:1YW2_A	B:2OUC_A
3TZN	A:1AJ0_A	B:1AJ0_A
3VQL	B:3AAB_A	A:3AAB_A
3VVW	B:3ECI_A	A:3VVW_A
3VX7	A:3VH2_A	B:3VX7_B
4A5U	A:4A5U_A	B:2QOU_O
4B8A	B:1UOC_A	A:4B8A_A
4DHI	B:2ZFY_A	D:3HCT_B
4DUL	A:1UT4_A	B:1UT4_A
4EM8	A:3PH4_A	B:3HE8_A
4F4I	B:2CCJ_A	A:2CCJ_A
4GQX	A:3URR_A	B:3URR_A
4H6J	B:3F1N_B	A:3F1N_A
4H7A	A:2ZCA_A	B:2ZCA_A
4HYE	A:3B2N_A	B:3B2N_A
4ILH	B:3SBT_B	A:3SBT_A
4IP3	A:4IP3_A	B:3HCT_B
4JAK	A:3N4J_A	B:3N4J_A

Hydrogen bond potentials for reranking

As described in Chapter 2, the models from DOCK/PIERR rigid docking and minimal refinement are reranked using a combination of atomic and residue potentials [41]. In this study, we develop hydrogen bond potentials to be used in the last reranking phase along with the residue and atomic potentials.

Building polar hydrogens into docking models

Models of most complexes do not contain polar hydrogen atoms: polar hydrogens coordinates are not available from protein structures obtained by X-ray crystallography. Hence polar hydrogens needed to be added to each of the one thousand docking models for every complex in the learning and test sets. They were also required in the native (experimental) structures, for complexes in the learning set that did not contain hydrogen atoms. Only 68 of the 640 complexes in the learning set were NMR structures that already had hydrogens in the experimental structure. All polar hydrogens were built using the ready_pdb script of the MOIL MD package [58]. For 4 targets each in the learning and ZLAB sets, where MOIL hydrogen placement failed, due to missing residues in the PDB files, the HAAD program [74] was extended to multiple chains and used to add hydrogens. A small number of targets (8 targets in the learning set and 4 targets in the ZLAB set) for which both MOIL and HAAD failed to insert hydrogens were excluded from our analyses: the PDB files had several missing heavy atoms in these cases. Overall, we had 628 targets from the learning set, 165 targets from the ZLAB set and 52 from the novel set. These three formed the set of soluble protein complexes. Additionally, we used another test set of 30 unbound membrane protein complexes, described in the next chapter.

Enhancement of hydrogen bond signal using Molecular Dynamics

The number of hydrogen bonds in the interfaces of the protein-protein models is quite small; the order of magnitude is in the tens, even for an interface hydrogen to acceptor atom distance of 4 Å. We note that this distance is more permissive than the traditional hydrogen-acceptor distance in a typical hydrogen bond, which is around 2 Å. The longer distance range is to account for the additional error in unbound docking. In order to amplify this weak signal from hydrogen bonding, we ran a short simulated annealing molecular dynamics trajectory for each model (and also for the native structures in the learning set) and use the final structure at the end of the dynamics run, instead of the initial model, to calculate the hydrogen bonds. The inaccuracies of unbound docking models mean that some hydrogen bonds are not close enough in the original model to be captured within the cutoff. Hence the simulated annealing accumulates signal from additional nearby hydrogen bonds, increasing the number of hydrogen bonds within a distance of 4 Å by an order of magnitude, to hundreds. We show in the Results section, that this increase in the signal from hydrogen bonding leads to improved ranking using the resulting hydrogen bond potential.

The MD protocol was as follows: first, an initial short MD run: 50 steps of dynamics with 1 femtosecond time step at 300K, with the nbfi option in MOIL. The nbfi option replaces the hard Lennard-Jones repulsion potential with a softer Gaussian repulsion, and this reduces the number of hard collisions in the structures. Second, a short 10-step minimization using conjugate gradient descent implemented in the mini_pwl routine in MOIL: to make the structures at the start of the longer dynamics run more chemically reasonable. And third, a 10 ps simulated annealing dynamics run with 1 femtosecond time step and linear temperature cooling from 600K to 10K. The annealing (cooling) procedure resulted in a hydrogen bonding potential that was more accurate at

reranking docking models (improved the number of hits recovered), compared to MD with no annealing. All MD runs were performed in vacuum for simplicity.

Note that the three step short MD procedure is not intended as a refinement procedure as the changes to interface RMSD of the resulting models are small: within 1 Å. The structure from dynamics is used merely to calculate the hydrogen bond geometry, as it leads to a better set of hydrogen bonds. We also note that for a small number of models for each target for which dynamics failed to converge, the interface hydrogen bonds were simply calculated from the original model.

Functional form of the potential

We developed the functional form for the pairwise learning using linear programming approach first, and then extend the form to non-linear learning approaches like SVM and Neural Networks.

For the linear programming case, we formulated a simple distance dependent, double-binned hydrogen bond potential based on the coordinates of the polar hydrogen atom and acceptor atom at protein interfaces. The functional form of the potential is as in Eq. (3.1). This formulation is similar to that of the atomic potential in Chapter 2.

$$E_{hbond}(X) = \sum_{\alpha,d} n(\alpha,d)u(\alpha,d) \qquad (3.1)$$

The energy of a complex, X, $E(X)$ is dependent on the coarse-grained particle types of the polar hydrogens, h, and electronegative acceptor atoms, a, at the interface, and the distance d between them. Note that the polar hydrogen and acceptor atoms can come from either of the two interacting proteins in the complex. We denote the interacting particle pair type (h,a) by a single parameter, α henceforth. $n(\alpha,d)$ is the number of times a hydrogen acceptor particle pair of type α is at an interface distance range, d. The value of the vector n depends on the geometry of the docking model.

$u(\alpha,d)$ is the corresponding parameter in the potential associated with the particular pairwise interaction type of (α,d). The energy is linear in the parameters, u, that define the potential.

For the particle types of polar hydrogen and acceptor, the coarse-graining classification that resulted in the best accuracy with the minimum number of potential parameters, was based on the residue types of the polar hydrogen and acceptor. Residues were classified into four types: *Hydrophobic* (ALA, VAL, ILE, LEU, PRO, TRP, PHE, MET), *Polar* (SER, THR, CYS, ASN, GLN, TYR, GLY), *Positive charged* (ARG, LYS, HIS) and *Negative charged* (ASP, GLU). Polar hydrogens and electronegative acceptor atoms were accordingly classified into four different particle types based on their residue type. These are denoted as $\{hyd, pol, pos, neg\}$ in Eq. (3.1). Various other types of coarse-graining were attempted and are compared in the Results and Discussion section, under Development of the Potential. Note that the potential here is directional and not symmetric in the particle types, h and a, unlike traditional residue or atomic pairwise potentials. The total number of particle type pairs is hence $4 * 4 = 16$.

To model the distance between hydrogen and acceptor atoms, we used two distance bins [0-4, 4-8 Å]. While the usual interface hydrogen bond distance is around 2.5 Å [46], increasing the first distance bin to 4 Å from 2.5 Å led to an increase in accuracy of 15.8% in the ZLAB unbound docking test sets (accuracy was based on the number of targets with an acceptable model in the top 10 models). Using a smaller cutoff of 2.5 Å was found to be useful for bound docking but not for models from realistic unbound docking, where the monomer structures themselves are inexact by 1 Å or more. The addition of a second distance bin from 4-8 Å further increased the accuracy of the hydrogen bond potential in reranking. The data for this is shown in the Results section under Development of the Potential. The longer-range interactions might carry additional

signal possibly from water-mediated hydrogen bonding interactions. Others [75, 76] have previously shown that coarse-grained potentials for protein interaction are more accurate when a second well representing water-mediated interactions is added. It is also likely that unbound docking models are lower in accuracy by more than 2 Å, and hence long-range interactions are essential to describe the interactions in inexact models. For example, the ZLAB Benchmark 4.0 contains 21% of targets with unbound to bound interface RMSD greater than 2 Å.

The 2 distance bins and 16 atom type pairs result in a total of 32 parameters for the hydrogen bond potential. We note that the potential is purely distance-dependent and not angle dependent. Using a constraint on the hydrogen bond angle (angle between the donor-to-polar hydrogen and polar hydrogen-to-acceptor) to be between 120^0 and 180^0 decreased the accuracy of the hydrogen bond potential by 36.4% when reranking our models in the ZLAB unbound docking test set (accuracy again based on the number of targets with an acceptable model in the top 10 models). Hence we think that the angle dependence is more appropriate when using high-resolution bound docking models and is not suitable for the relatively imprecise unbound docking models.

Learning Algorithm 1: Pairwise Learning using Linear Programming (PLLP)

In pairwise learning for linear programming, a pair of energies, *E,* of correct ($X_{correct}$) and incorrect ($X_{incorrect}$) models is compared to obtain a set of inequalities of the type $E(X_{incorrect}) > E(X_{correct})$. These inequalities are solved to obtain the parameters of the potential, *u* in Eq. (3.1). The set of inequalities is linear in the parameters u , as noted in Chapter 2. Note that our definition of correct model (interface RMSD less than 2.5 Å) and incorrect model (interface RMSD greater than 7 Å) is the same as in Chapter 2.

The inequalities we solve are also the same set of inequalities as in Chapter 2. They are shown again in Eq. (3.2). The first inequality compares correct and incorrect models. The second and third inequalities help shape the binding funnel more precisely at the bottom of the funnel. The second inequality stipulates that the energy of high-quality hits or near-native models (with interface RMSD less than 1.5 Å) should be lower than that of good hits (with interface RMSD between 1.5 and 2.5 Å). The last inequality is based on a sorting based on iRMSD of hits within 2.5 Å interface RMSD for each learning set complex. The energy of a hit *i,* which is ranked just above the hit *i+1*, should be lower than the energy of hit *i+1*. These inequalities compare all n_{hits} pairwise adjacent hits of a target. z_i is the slack variable which is the error in satisfying each constraint.

$$\begin{aligned} &E(X_{incorrect} - X_{correct}) > 1 - z_i \quad ; z_i > 0 \\ &E(X_{good\_hit} - X_{high-quality\_hit}) > 1 - z_i \quad ; z_i > 0 \\ &E(X_{hit}^{i+1} - X_{hit}^{i}) > 1 - z_i ; i = 1,2,\ldots n_{hits} - 1 \quad ; z_i > 0 \end{aligned} \tag{3.2}$$

The objective function coupled with the constraints, during learning is shown in Eq. (3.3). The sum of errors in each constraint z_i is minimized along with the sum of parameter values, u. The constant $\gamma = 1$.

$$\min \sum_{\alpha,d} |u(\alpha,d)| + \gamma \left| \sum_i z_i \right|_1 \tag{3.3}$$

The three sets of inequalities in Eq. (3.2) coupled with the objective function in Eq. (3.3) form the linear program for the hydrogen bond potential, which was solved to obtain the parameters u. The top 1000 models of all the 628 targets in the learning set along with the native structure for these targets were used to formulate the inequalities. The simulated annealing MD procedure described earlier was performed on all the models and native structures, in order to enhance the number of interface hydrogen bonds. The total number of inequalities for the hydrogen bond potential was 5,820,745

and the number of parameters to be determined was 32. The linear program was solved using PF3 [39, 40], a parallelized linear programming solver designed for development of protein folding potentials. It is an extension of standard interior point solvers for the special case of protein folding problems where the number of inequalities (millions) is much higher than the number of parameters (~100's), which enables efficient parallelization of the constraint matrix.

Learning Algorithm 2: Support Vector Machines

Support vector machines are a class of learning algorithms which, when given a set of positive and negative examples, learn a model that maximizes the separation or margin between the positive and negative distributions [77]. Assume we are given a set of N_{train} training examples, $\left\{\left(x_i \equiv \left\langle x_{i,1}, x_{i,2} \ldots . x_{i,n} \right\rangle, y_i\right); i = 1,2 \ldots . N_{train}\right\}$, where each example is an input x_i an n-dimensional vector with a corresponding output $y_i \in \{-1,+1\}$ denoting the distribution (positive or negative) that the example belong to. Assuming the two distributions can be linearly separated with an *n*-dimensional hyperplane, SVMs seek to find the hyperplane that maximize the margin between the nearest examples of the two distributions. The optimization problem solved by the so-called soft margin SVM classifier [77, 78] is shown in Eq. (3.4).

$$\begin{aligned} &\min_{w,b} \|w\|_2 + C\sum_i z_i \qquad subject \quad to \\ &y_i\left(\vec{w}\cdot\vec{x}_i + b\right) > 1 - z_i \quad z_i > 0 \end{aligned} \tag{3.4}$$

It is called a soft-margin classifier, as the positive and negative examples need not be strictly well-separated. Misclassification of examples is allowed, and is denoted by the slack variable z_i, for each example, which represents the extent of deviation of the misclassified example from the hyperplane. w is the *n*-dimensional vector normal to the hyperplane we are looking for, and *b* is a bias constant. These two parameters determine

the SVM hyperplane, and their values are found by solving the quadratic program in Eq. (3.4), with one constraint per training example, and an optimization condition that requires minimizing the square of the vector *w* and the sum of non-negative errors, z_i. C is the cost parameter that controls the tradeoff between training error and margin. Larger the value of C, greater the penalty term so the margin for misclassification is smaller [77].

To model the hydrogen bond potential using SVMs, we used a variant of the binary classifier SVM, for regression, i.e. the output is not one of two classes as shown above, but a floating point value representing the hydrogen bond energy. Each model and native structure in the learning set was used as one training example: there were 625729 examples in all. The input features, or $\vec{x_i}$, in Eq. (3.4) was the set of 32 geometric contacts for a model, relevant to hydrogen bonding, depicted by $n(\alpha,d)$ in the section on linear programming. These feature values were scaled between -1 and 1 for numerical stability and to prevent the features with the largest fluctuation from dominating. The output, or y_i in Eq. (3.4), represents the hydrogen bond energy. The energy value used in training was the interface RMSD of the model to its native structure. A desirable property of a good potential is positive correlation with the interface RMSD, i.e. lower the RMSD, lower the energy and better the model. Hence interface RMSD was used as the training output.

Further, SVMs also allow for non-linear separation of distributions. This is done by mapping the input data to higher dimensions using kernel functions: the rationale is that it might be easier to linearly separate the data in higher dimensional space, compared to the original space. Kernel functions are defined on pairs of inputs and typical choices used are the polynomial kernel $K(x_i,x_j)=\left(s(x_i \cdot x_j)+c\right)^d$ where *s, c, d* are constants with *d* representing the degree of the polynomial; radial basis or Gaussian kernel

$K(x_i, x_j) = e^{-\gamma \|x_i - x_j\|^2}$ where γ is a constant to be tuned and sigmoid kernel $K(x_i, x_j) = \tanh(s(x_i \cdot x_j) + c)$ where s and c are constants [79]. The constant parameters in these kernels define the SVM potentials, and they need to be tuned while training.

SVMLight [79] was the package used for training and testing. The optimization problem in Eq. (3.4) is solved in SVMLight by a fast heuristic that involves stochastic sampling of the set of inequalities, and breaking them down into smaller quadratic problems that are solved at each step. Linear and non-linear SVMs were used to model the hydrogen bond potential. For linear SVMs, the cost parameter, *C,* was varied in powers of 2 $C \in \{2^{-5}, 2^{-3}, 2^{-1}, 2, 2^3, 2^5, 2^7, 2^9, 2^{11}\}$ to obtain different potentials. The potential with the best performance in ranking models on the learning set was chosen as the representative linear SVM potential. Similarly for the non-linear sigmoid kernel, the cost parameter was varied as above, while other parameters were kept fixed. The same was the case for the polynomial kernel, where in addition to the cost parameter, the degree of the polynomial was varied $d \in \{3,5,7,9\}$. Radial basis function kernel was omitted as the training time was too long (3+ days). One representative non-linear potential was chosen among the polynomial and sigmoid kernel potentials, according to the best ranking performance on the learning set.

We note that we also tried binary classification using SVMs to label model into one of two classes: correctly docked or misdocked structure, using an interface RMSD less than 4 Å to define correct structure. The idea was to see if this prediction could help fish out near-native structures from the set of models. However docking datasets are highly imbalanced: i.e. the number of negative examples is much higher than the number of positive examples. In these cases, binary SVM classifiers behave more like a majority classifier: classifying almost all examples as negative. SVMs allow for tuning of the cost

parameter for imbalanced datasets, such that different cost values *C* are used for positive and negative examples [80]. In our case, $C_+ >>> C_-$ i.e. the cost for misclassifying a positive example is much higher than the cost for misclassifying a negative example. However the models learnt using this fix in cost penalties did not turn out to be different and in general, classification was found to be not useful for reranking.

Learning Algorithm 3: Neural Networks

Artificial neural networks are another class of machine learning algorithms that seek to model the outputs of a problem as a non-linear function of the features or inputs. The non-linear model is that of a network with one input layer, one output layer and one or more hidden layers connecting the input and output layers. Each layer *i* contains a fixed number of neurons, which receive the input from all the neurons of the previous layer, *i-1,* and transform them by a some non-linear function (called activation function) on the inputs. Outputs from the neurons in layer *i* are propagated to neurons in layer *i+1*.

Eq. (3.5) is the Eq. for y_m^n, the output of the *m*th neuron in the *n*th layer[81, 82]. It is a non-linear function, *f,* of y_k^{n-1}, the outputs of all neurons indexed $k = 1,2,...N_{n-1}$ in the *n*-1th layer. The outputs are each scaled by a weight $r_{k,m}^{n-1,n}$, which is the weight of the connection between the *k*th neuron in the *n-1*th layer and the *m*th neuron in the *n*th layer. Also a bias constant *b* is added to the weighted linear combination of the previous layer outputs. The function *f* is usually the sigmoidal function, $f(x) = \frac{1}{1+e^{-x}}$, or the hyperbolic tangent $f(x) = \tanh(x)$ or Gaussian function $f(x) = e^{-ax^2}$. The resulting potential is continuous and differentiable in coordinate space.

$$y_m^n = f_m^n\left(b_m^n + \sum_{k=1}^{N_{n-1}} r_{k,m}^{n-1,n} y_k^{n-1}\right) \tag{3.5}$$

For the hydrogen bond potential, we used 1 and 2 hidden layers. The output of the single neuron of the output layer y_1^3 (for one hidden layer and 3 layers in all) and y_1^4 (for two hidden layers and 4 layers in all), represents the hydrogen bond energy as predicted by the Neural Network. The inputs to the neural network i.e. neurons of the first layer, $\left\{y_i^1; i = 1,2...32\right\}$ are the 32 geometrical features of hydrogen bonding for a model, represented by $n(\alpha, d)$, in the section on linear programming and SVMs.

The weights *r* and bias *b* completely determine the network. They are determined by iterative gradient descent on a training set. The objective function minimized during training is the Mean-Squared Error (MSE) between the current predicted outputs from the network and the correct outputs for the training set examples. The MSE at the *k*th iteration is shown in Eq. (3.6), where N_{train} is the number of training examples, $y_i^{NN,k}$ is the output of the neural network in the *k*th iteration for the *i*th training example and $y_i^{correct}$ is the correct output for that training example. We are assuming here that this neural network produces only a single output (hydrogen bond energy in our case).

$$MSE_k = \frac{1}{N_{train}} \sum_{i=1}^{N_{train}} \left(y_i^{NN,k} - y_i^{correct} \right)^2 \tag{3.6}$$

The algorithm used for updating the weights was Rprop [83], as it consistently produced networks with lower MSE than other weight update algorithms like Quickprop, Backpropagation and Batch Update [84]. In Backpropagation, the weights of the network are updated every time a new training example is seen, which means that they are updated multiple times per training iteration. Batch Update and Quickprop are advanced Backpropagation algorithms where weights are updated once all the training examples are seen i.e. only once per iteration. The above algorithms require one to tune additional parameters such as the step size or learning rate. In contrast, Rprop is one of the widely

preferred advanced update algorithms, and uses a dynamic step size for every step, updating the weights once per iteration. We used Rprop for training since it produced networks with lower MSE values than the other training algorithms for our case.

Different stopping conditions are used to terminate training in neural networks. They can be based on the number of iterations (train till a maximum number of iterations is reached), or a threshold on the training error (train till the MSE is less than ε, a small number) or a threshold on the training progress (train till improvement in MSE over the last T iterations is no more than ε) [85]. The disadvantage of the above approaches is that it is possible to overfit the network to the training set. Instead, we follow the common protocol of splitting the training set into an 80% set for training and 20% set for validation. The stopping conditions used were the following: Train until a maximum of $T = 500$ cycles (usually never reached). In each cycle, we train for 10 iterations. In each iteration, all the training samples in the 80% set are seen and weights of the network are updated. At the end of each cycle, we test the network on the 20% validation set and get the MSE on the validation set (validation error) as well as MSE on the training set (training error). If the percentage improvement in the validation error over the last 15 cycles was found to be less than 0.01%, then training was halted. Additionally if the improvement in the training error was found to be less than e-05 over the past 15 cycles, training was stopped.

The C-based neural network package, Fast Artificial Neural Networks (FANN) [84] was used for training and testing the networks. Fully connected networks were used, and inputs and outputs were scaled to [-1, 1]. The set of inputs, outputs and the training examples were exactly the same as that used for SVMs. Inputs were the 32 geometric features describing the hydrogen bond potential, output used to model the hydrogen bond

potential energy was the interface RMSD of the model to its native structure. In all, 625729 training samples were used.

Multiple architectures i.e. different number of hidden layers and neurons were experimented with, to obtain different models of the hydrogen bond potential. Also different hidden layer non-linear functions and output layer functions were explored. Linear and symmetric sigmoid functions were used for the output layer and networks with sigmoid outputs produced lower validation MSE. For the hidden layer, Gaussian and sigmoid functions were tried and networks with hidden layer sigmoid functions produced lower validation MSE. Fixing the training algorithm and hidden and output layer functions, different architectures of the network were tried and the final network selected was the one with the smallest validation MSE, as shown in the Results section.

We note that we also tried to perform classification instead of regression for neural networks, as in the case of SVMs. However due to the imbalance in the docking datasets: number of negative examples is much higher than the number of positive examples, the neural network behaved similar to the SVM and classified almost all models as negative examples, rendering the output to be uninformative for ranking.

3.4 RESULTS AND DISCUSSION

In this study, we first test the reranking performance of the newly developed hydrogen bonding potentials, by reranking the top 1000 models in the last step using the hydrogen bonding potentials alone, without C3. We then combine the best performing hydrogen bond potentials with C3, to improve the quality of final reranking.

The hydrogen bond potential is first developed in the linear programming framework, and various types of coarse-graining for atom types and distance bins are

systematically considered. Once the best performing coarse-graining is known from the linear programming results, the same functional form for the potential is used for the two other learning algorithms. This enables us to compare the performance of different learning algorithms. The results of the ranking performance of hydrogen bond potentials from different learning methods, alone and in combination with C3, are compared, on datasets of soluble and transmembrane complexes.

Development of the Hydrogen Bond Potential and Results from Pairwise Learning using Linear Programming

In this section we explore various types of coarse-graining models for the hydrogen bond potential derived using pairwise learning from linear programming (PLLP).

a. Using Molecular Dynamics improves the hydrogen bond signal

In Table 3.2, two different sets of models are used for calculating the hydrogen bond energy. These sets of models are compared based on their ability to produce accurate hydrogen bond potentials for reranking. In the first case, the hydrogen bond potential is derived from, and applied to, DOCK/PIERR models before the simulated annealing MD procedure described in this paper. These are models from rigid FFT docking which have previously undergone side chain remodeling and energy minimization [41]. In the second case, the learning and testing of the hydrogen bond potential is done on models that have undergone the simulated annealing MD procedure, in addition to the previous side chain remodeling and energy minimization. The results shown are for a simple hydrogen bond potential with 4 particle types each, for hydrogen and acceptor atoms, based on the residues they belong to (hydrophobic, polar, positive charged and negative charged) and a single distance bin from 0-4 Å. This resulted in 16 parameters in all, the values of which were obtained from linear programming using the

learning set models, as described previously in the **Methods** section. The number of near-native models or hits i.e. models within 4 Å interface RMSD of the experimental structure, within the top 10 and top 1 structures, are reported for the ZLAB and novel test sets. These final models are obtained by ranking the top 1000 models from DOCK/PIERR rigid docking using the hydrogen bond potential. Table 3.2 shows that the hydrogen bond potential has a better accuracy when simulated annealing is used to enhance the number of interface hydrogen bonds. So here onwards, we report results from potentials applied to models derived from the simulated annealing MD procedure.

Table 3.2 The performance of hydrogen bond potential on two different model sets: one without MD and one after MD is compared, on the ZLAB and novel test sets. The numbers of hits in the top 10 and top 1 and number of targets with at least one hit in the top 10 are reported. A hit is a model rated acceptable according to CAPRI i.e. with an interface RMSD of 4 Å or less. Note that the potential has 4 particle types of hydrogen and acceptor *(hyd, pol, pos, neg)* and 1 distance bin [0-4 Å], and is a simpler form of the final potential we derive.

Models used to derive and calculate hydrogen bond potential	**ZLAB test set (165 targets)**		**Novel test set (52 targets)**	
	Number of hits in the top 10/Number of targets with a hit in the top 10	**Number of targets with a hit in the top 1**	**Number of hits in the top 10/Number of targets with a hit in the top 10**	**Number of targets with a hit in the top 1**
Before simulated annealing MD	27/21	2	9/6	1
After simulated annealing MD	41/20	5	18/11	3

b. Exploration of particle types

Various types of coarse-graining were attempted for the particle types of hydrogen and acceptor. These ranged from the simple element level classification based

on the element types of the atoms; to the more complicated classifications that depend on the residue type, placement in the side-chain or backbone and hybridization of the atoms. Table 3.3 shows the number of parameters for each type of coarse-graining and the accuracy of the resulting hydrogen bond potential derived from linear programming.

Table 3.3 (a) Different definitions of hydrogen and acceptor particle types, and the corresponding number of parameters. The abbreviations are as follows: i. *residue types*: hyd: hydrophobic, pol: polar, pos: positive charged, neg: negative charged, ii. *element types*: N: Nitrogen, O: Oxygen, S: Sulphur and iii. *atom placement* : Bkbn: backbone, Sc: side-chain. Other abbreviations are standard 3-letter amino acid names.

Basis of classification	**Listing of hydrogen atom types**	**Listing of acceptor atom types**	**Number of parameters in the potential**
Element types of donor atom bonded covalently to hydrogen and acceptor atom	1. N 2. O 3. S	1. N 2. O 3. S	9
Residue type of hydrogen and acceptor atom	1. hyd 2. pol 3. pos 4. neg	1. hyd 2. pol 3. pos 4. neg	16
Element type as well as residue type of donor bonded to hydrogen, and acceptor atom	1. N: hyd 2. N: pol 3. N: pos 4. N: neg 5. O: pol i.e. SER/THR/TYR 6. S: pol i.e. CYS	1. O: hyd 2. O: pol 3. O: pos 4. O: neg 5. N: pos i.e. HIS 6. S: pol i.e. CYS	36

Table 3.3 (a) continues on the next page.

Basis of classification	Listing of hydrogen atom types	Listing of acceptor atom types	Number of parameters in the potential
Residue type, element type, side-chain/backbone placement of donor bonded to hydrogen, and acceptor atom	1. Bkbn N: hyd 2. Bkbn N: pol 3. Bkbn N: pos 4. Bkbn N: neg 5. Sc N: hyd i.e. TRP 6. Sc N: pol i.e ASN/GLN 7. Sc N: pos i.e. ARG/LYS/HIS 8. Sc O: pol i.e. SER/THR/TYR 9. Sc S: pol i.e. CYS	1. Bkbn O: hyd 2. Bkbn O: pol 3. Bkbn O: pos 4. Bkbn O: neg 5. Sc O: pol i.e. SER/THR/TYR/ASN/GLN 6. Sc O: neg i.e. ASP/GLU 7. Sc N: pos i.e. HIS 8. Sc S: pol i.e. CYS	72
Residue type, element type, side-chain/backbone placement and chemical similarity (e.g. hybridization) of donor bonded to hydrogen, and acceptor atom	1. Bkbn N: hyd 2. Bkbn N: pol 3. Bkbn N: pos 4. Bkbn N: neg 5. Sc N: TRP 6. Sc N: ASN/GLN 7. Sc N: ARG-NE 8. Sc N: ARG-NH1/NH2 9. Sc N: LYS 10. Sc N: HIS 11. Sc O: SER/THR/TYR 12. Sc S: CYS	1. Bkbn O: hyd 2. Bkbn O: pol 3. Bkbn O: pos 4. Bkbn O: neg 5. Sc O: SER/THR/TYR 6. Sc O: ASN/GLN 7. Sc O: ASP/GLU 8. Sc N: HIS 9. Sc S: CYS	108

In Table 3.3 (a), various types of coarse graining of hydrogen and acceptor particle types are shown along with the corresponding number of parameters in the resulting potential. Table 3.3 (b) shows the effect of different types of coarse graining of atom types, on the accuracy of the hydrogen bond potential for reranking.

Table 3.3 (b) The performance of hydrogen bond potentials with one distance bin [0-4 Å] and various coarse-graining types for hydrogen and acceptor atoms is shown. The hydrogen bond potential is applied for reranking the top 1000 models from DOCK/PIERR rigid docking of each target, followed by side chain remodeling, minimization and simulated annealing MD. The number of hits in the top 10 and top 1 and number of targets with at least one hit in the top 10 are reported for the ZLAB and novel test sets. A hit is a model rated acceptable according to CAPRI i.e. with an interface RMSD of 4 Å or less.

Number of potential parameters based on atom type coarse-graining in Table 3 (a)	**ZLAB test set (165 targets)**		**Novel test set (52 targets)**	
	Number of hits in the top 10/Number of targets with a hit in the top 10	**Number of targets with a hit in the top 1**	**Number of hits in the top 10/Number of targets with a hit in the top 10**	**Number of targets with a hit in the top 1**
9	18/14	2	8/6	1
16	41/20	5	18/11	3
36	28/19	4	12/8	1
72	33/20	4	14/9	1
108	35/24	5	17/10	2

The number of distance bins is fixed to one [0-4 Å]. The hydrogen bond potential is applied for reranking the top 1000 models from DOCK/PIERR rigid docking followed by side chain remodeling, minimization and simulated annealing MD. We first start with the smallest number of parameters (9 parameters), based on just the element types of the acceptor and donor atom, which is covalently bonded to the polar hydrogen atom. Next we explore the classification based on the residue type of the hydrogen and acceptor (16 parameters). We then incrementally add complexity to the coarse-graining by including the residue type along with element type (36 parameters), adding backbone/side-chain distinction (72 parameters) and coarse-graining finally based on chemical similarity (hybridization for instance, 108 parameters).

In Table 3.3 (b), we note that coarse-graining according to residue types (potential with 16 parameters) is better than the coarse-graining according to element type, indicating that protein-protein interactions seem to be residue-specific. Additional complexity, with potentials with more than 16 parameters, does not lead to significantly improved performance on both the test sets. Hence we chose to retain the simple potential with 16 parameters for further calculations. In order not to over-fit the potential by including too many parameters, we further did not consider potentials with 200 and more parameters for atom types. We note again that the above calculation corresponds to a single distance bin, and in the succeeding section we explore the effect of adding distance bins, and consequently additional parameters related to that.

Exploration of distance bins

In Table 3.4, we show the effect of additional distance bins longer than the initial [0-4 Å] bin. We add a second distance bin for interactions in the range [4-8 Å] and even a third one in the range [8-12 Å]. As explained in the section on Functional form, the longer-range interactions represent signal possibly from water-mediated hydrogen bonding interactions, or from interactions in unbound docking models that are imprecise. The number of atom type pairs is fixed to 16, as per the results of the previous section. It is seen that the second distance bin [4-8 Å] provides additional signal over the first one. However, the [8-12 Å] distance bin representing long-range electrostatic interactions does not carry additional signal over the previous 2 bins. Hence we use the version with 2 distance bins.

Table 3.4 The performance of hydrogen bond potentials with different distance bins is shown. The number of hydrogen and acceptor atom type pairs is fixed to 16. The hydrogen bond potential is applied for reranking the top 1000 models from DOCK/PIERR rigid docking of each target, followed by side chain remodeling, minimization and simulated annealing MD. The number of hits in the top 10 and top 1 and number of targets with at least one hit in the top 10 are reported for the ZLAB and novel test sets. A hit is a model rated acceptable according to CAPRI i.e. with an interface RMSD of 4 Å or less.

Distance bins used in Å	**ZLAB test set (165 targets)**		**Novel test set (52 targets)**	
	Number of hits in the top 10/Number of targets with a hit in the top 10	**Number of targets with a hit in the top 1**	**Number of hits in the top 10/Number of targets with a hit in the top 10**	**Number of targets with a hit in the top 1**
[0-4]	41/20	5	18/11	3
[0-4,4-8]	50/25	5	28/15	3
[0-4,4-8,8-12]	49/23	7	28/15	5

This leads to a hydrogen bond potential with 16 atom type pairs and 2 distance bins, a total of 32 parameters. Solving for the parameters of the potential using linear programming, we obtain a solution with 71.4% of the inequalities satisfied on the learning set. We note that this percentage is significantly less than that of previously developed atomic and residue potentials [41]. One reason for could be the number of parameters; the number of parameters in the previously developed atomic potential was 2 orders of magnitude larger (1584 parameters) and the residue potential was an order of magnitude larger (252 parameters). The other reason could be that traditional residue/atomic interactions are more specific than hydrogen bonding interactions in soluble protein interfaces; possibly because of the competition of protein-protein hydrogen bonds with water.

For final reranking of the top 1000 models from DOCK/PIERR, we combine the hydrogen bond potential along with the potential C3 [41], a previously developed combination of interface residue and atomic potentials. We first calculate the C3 term from the rigid docking models that are refined i.e. subject to side chain remodeling and energy minimization [41]. Then the hydrogen bond term is calculated on the models after the additional simulated annealing MD procedure described in this paper. A linear combination of C3 with the hydrogen bonding term is used for reranking the top 1000 models, as shown in Eq. (3.7). Weight of the hydrogen bonding term is derived from the performance on the learning set targets, to be 4.0. The performance of the hydrogen bond potential from linear programming is further discussed in Table 3.5 for soluble proteins and Table 3.7 for membrane proteins. The performance of the linear combination of the hydrogen bond term with C3 is shown in Table 3.6 for soluble protein datasets and Table 3.8 for membrane protein datasets.

$$E_{total} = C3 + w.E_{hbond} \tag{3.7}$$

Results from SVM potentials

SVM regression potentials were derived for three different kernel choices: linear, sigmoidal and polynomial kernels, and various choices of cost parameters and degree of polynomial, as mentioned in the Methods section. For the linear and sigmoidal kernels, each value of cost parameter produced a new SVM potential. For the polynomial kernel, each combination of cost parameter and degree of polynomial produced a new SVM potential. One linear and one non-linear SVM potential were chosen; the potentials were chosen based on the performance of the resulting potential in reranking the top 1000 models of targets in the learning set. In particular, the linear kernel with cost $C = 2^9 = 512$ was chosen, as it produced the highest number of learning set targets with a

hit in the top 10, as shown in Figure 3.1 (a). Similarly among non-linear kernels, the polynomial kernel with degree $d = 5$ and cost $C = 0.12$ was chosen for the same reason. In this case, multiple cost values for polynomial kernel $d = 5$ performed equally well, as shown in Figure 3.1 (b). Hence we chose the potential with the smallest C value, or widest (most general) margin between correctly and incorrectly docked models.

Table 3.5 shows the performance of the chosen non-linear and linear regression SVM hydrogen bond potentials alone, without C3, for reranking docking models on the soluble protein test sets. Performance of the non-linear SVM potential is not better than that of the linear SVM potential. This indicates that the set of models is linearly separable and use of non-linear functions should be avoided as this can lead to overfitting. Further, linear combinations of the SVM linear potential with C3 and SVM non-linear potential with C3 were obtained separately, as in Eq. (3.7). The weight, i.e. the parameter w in Eq. (3.7) was fixed to be 0.008 for the linear SVM and 0.005 for the non-linear SVM, using the linear combination with the best ranking performance on the learning set. The performance of the two linear combination potentials is discussed in Table 3.6 for soluble protein datasets and Table 3.8 for membrane protein datasets.

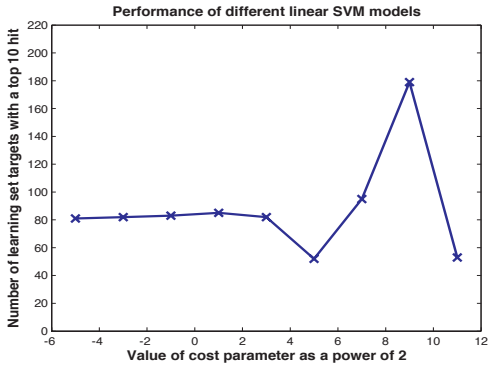


Figure 3.1 (a) Model selection for linear SVMs. The accuracy of each model (in terms of number of learning set targets with a top 10 hit) is plotted as a function of the cost parameter. The linear SVM with cost $C = 2^9 = 512$ produces maximum number of targets with a hit.

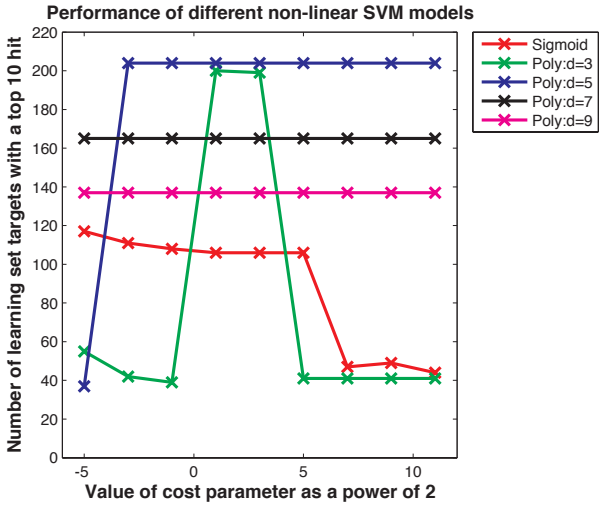


Figure 3.1 (b) Model selection for non-linear SVMs: sigmoid and polynomial kernels with degrees 3,5,7 and 9. The accuracy of each model (in terms of number of learning set targets with a top 10 hit) is plotted as a function of the cost parameter. The polynomial SVM with degree $d = 5$ produces maximum number of targets with a hit.

Results from Neural Network potentials

Different network architectures were attempted for modeling the hydrogen bond potential with neural networks: one hidden layer with 2, 5, 10, 15, 20, 32, 40 and 50 neurons and 2 layers with 2, 5 and 7 neurons. We did not increase the number of neurons or layers further, as the mean-squared error at the end of training was not higher for the larger networks compared to the networks we report here. Figure 3.2 shows the behavior of MSE as a function of the number of network layers and neurons. The network with one hidden layer and 10 neurons was chosen as it had the lowest MSE. The neural network hydrogen bond potential was combined with atomic and residue potentials in C3 for reranking, as in Eq. (3.7), fixing the weight w to be 3.3 based on ranking performance on the learning set. The performance of the neural network hydrogen bond potential alone is in Table 3.5 for soluble proteins and Table 3.7 for membrane proteins, while the performance of the linear combination with C3 is in Tables 3.6 and 3.8.

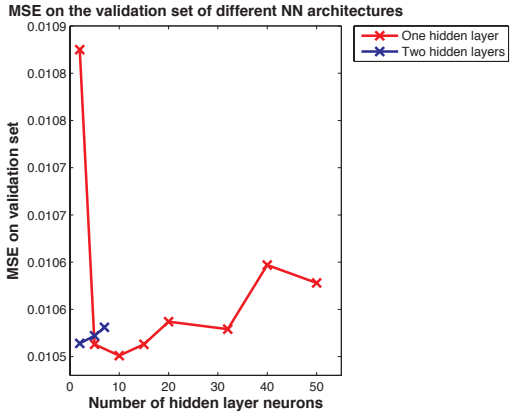


Figure 3.2 Model selection for neural networks. The number of hidden layer neurons is plotted against the Mean Squared Error on the validation set during training. The networks with one hidden layer are shown in red while the networks with two hidden layers are shown in blue. The network with one hidden layer and 10 neurons has least error.

Performance of Hydrogen Bond Potentials on Soluble Protein Complexes

The hydrogen bond potential developed by pairwise learning using linear programming performs the best while the neural network potential performs next best. Also, as discussed before, the non-linear SVM is not necessarily better than the linear SVM. Table 3.6 shows the performance of the hydrogen bond potentials in combination with C3.

In combination with C3, the neural network potential performs best overall, followed by the linear programming potential. Addition of the neural networks hydrogen bond potential results in a 16.94% increase in the number of targets with a top 10 hit in the ZLAB set increases, while the number of targets solved in the novel set is about the same. But the number of top 10 hits is enriched for both the ZLAB and novel sets, by 14.39% and 20.63 % respectively.

Table 3.5 The performance of hydrogen bond potentials from different learning algorithms is shown on ZLAB and novel test sets. A hit is a model rated acceptable according to CAPRI i.e. with an interface RMSD of 4 Å or less.

Learning method for generating hydrogen bond potential	**ZLAB test set (165 targets)**		**Novel test set (52 targets)**	
	Number of hits in the top 10/Number of targets with a hit in the top 10	**Number of targets with a hit in the top 1**	**Number of hits in the top 10/Number of targets with a hit in the top 10**	**Number of targets with a hit in the top 1**
Pairwise Learning using Linear Programming	50/25	5	28/15	3
Linear SVM, *c=512*	17/12	1	17/8	3
Non-linear SVM, Polynomial kernel, *c=0.12, d=5*	17/9	1	11/10	1
Neural Network 1 hidden layer with 10 neurons	40/24	3	23/10	4

We note that the hydrogen bond potential that works best in combination with C3 (neural networks) is not the one that works best alone. This discrepancy maybe because the atomic potential in C3 and the hydrogen bond potential used the same set of inequalities for learning and there is the possibility of overlapping in learning leading to some redundancy in the linear combination signal. This suggests that using different learning algorithms for different reranking potentials might be useful to capture heterogeneous signal. SVM potentials perform the worst and do not seem to add much signal to the atomic and residue potentials already present in C3. We also note that the

ZLAB and novel test sets behave slightly differently. This is because of the nature of difficulty of the datasets: in the novel set all targets chosen were such that monomer unbound to bound distance was within 1-2 Å, while in the ZLAB test set, 21% of the targets had monomer unbound to unbound distance greater than 2 Å.

Table 3.6 The performance of hydrogen bond potentials from different learning algorithms in combination with C3 is shown on ZLAB and novel test sets. A hit is a model rated acceptable according to CAPRI i.e. with an interface RMSD of 4 Å or less.

Learning method used to derive the hydrogen bond potential	**Potential used for reranking**	**ZLAB test set (165 targets)**		**Novel test set (52 targets)**	
		Number of hits in the top 10/Number of targets with a hit in the top 10	**Number of targets with a hit in the top 1**	**Number of hits in the top 10/Number of targets with a hit in the top 10**	**Number of targets with a hit in the top 1**
NA	C3	132/59	19	63/26	13
Pairwise Learning using Linear Programming	Linear combination with C3	142/60	25	71/29	13
Linear SVM, *c=512*	Linear combination with C3	135/61	18	63/27	12
Non-linear SVM, Polynomial kernel, *c=0.12, d=5*	Linear combination with C3	136/61	22	66/26	13
Neural Network 1 hidden layer with 10 neurons	Linear combination with C3	151/69	21	76/27	14

We also observe that the signal for reranking obtained from hydrogen bond potentials alone is weaker than the signal obtained from atomic and residue potentials. Indeed, while the average interface is stipulated to have around 10 hydrogen bonds [86], [46], an analysis of the set of native structures in our learning set suggested that a significant fraction 105/628 did not have any hydrogen bonds within a distance of 4 Å. It is possible that the hydrogen bonds between protein interfaces have to compete with those between water and protein, and this results in the hydrogen bond signal being weak. The competition with water is not present in hydrogen bonds in membrane protein interfaces, which we discuss next.

Hydrogen Bond Potentials for Transmembrane Complexes

Till now, the discussion has centered on protein complexes in aqueous solution. However, protein complexes integral to the cell membrane form another important class of complexes: they perform critical functions like cell signaling and transport, and their misfolding and aggregation results in diseases like Alzheimer's and Parkinson's [87]. Hence we examine the performance of docking potentials on membrane proteins too. Due to the abundance of experimental data for soluble proteins, the potentials used in docking algorithms are based on soluble protein complexes. However, recent studies have shown that these algorithms and potentials can be applied to predict membrane complexes with reasonable accuracy [87]. Here we explore the performance of the developed hydrogen bond potentials on test sets of membrane proteins.

For transmembrane complexes, hydrogen bond potentials alone seem to be more accurate than the atomic and residue potentials in C3, as Table 3.7 suggests. The increased signal in this case could be because of lack of competition with water for hydrogen bond formation. The SVM potentials, which did not perform well in the soluble

protein case, perform better here. The linear programming potential performs next best in reranking membrane protein models.

In a previous study of extension of DOCK/PIERR to membrane protein docking, it was found that adding an energy term (MTE) that mimics the membrane environment was beneficial in ranking (see Chapter 4). In Table 3.8 we show the linear combination of C3 with the hydrogen bonding term from different learning algorithms combined with MTE. It is shown that the use of the hydrogen bond potential can also contribute slightly to improved ranking of membrane protein models.

Table 3.7 The performance of hydrogen bond potentials from different learning algorithms is shown on a test set of 30 homology modeled membrane protein complexes. A hit is a model rated acceptable according to CAPRI i.e. with an interface RMSD of 4 Å or less.

Learning method for generating hydrogen bond potential	**Unbound membrane proteins set (30 targets)**	
	Number of hits in the top 10/Number of targets with a hit in the top 10	**Number of targets with a hit in the top 1**
C3 (No hydrogen bond potential)	2/2	0
Pairwise Learning using Linear Programming	12/7	3
Linear SVM, *c=512*	8/6	3
Non-linear SVM, Polynomial kernel, *c=0.12, d=5*	15/11	4
Neural Network 1 hidden layer with 10 neurons	5/5	1

Table 3.8 The performance of hydrogen bond potentials from different learning algorithms in combination with C3 is shown on a test set of 30 homology modeled membrane protein complexes. A hit is a model rated acceptable according to CAPRI i.e. with an interface RMSD of 4 Å or less.

Learning method used to derive the hydrogen bond potential	**Potential used for reranking**	**Unbound membrane proteins set (30 targets)**	
		Number of hits in the top 10/Number of targets with a hit in the top 10	**Number of targets with a hit in the top 1**
NA	C3*MTE	14/11	7
Pairwise Learning using Linear Programming	$(C3+w.E_{hb}).MTE$	17/11	8
Linear SVM, *c=512*	$(C3+w.E_{hb}).MTE$	16/11	7
Non-linear SVM, Polynomial kernel, *c=0.12, d=5*	$(C3+w.E_{hb}).MTE$	16/11	7
Neural Network 1 hidden layer with 10 neurons	$(C3+w.E_{hb}).MTE$	15/11	6

Analysis of hydrogen bond potential

In general, pairwise linear programming performs well as a learning algorithm overall on soluble and membrane protein datasets. It is also the learning method whose parameters are easier to interpret biochemically. In Tables 3.9 (a) and 3.9(b) we show the potential parameters from linear programming for the first (0-4 Å) and second (4-8 Å) distance bins respectively. We note that the most significant parameter values are in the

short-range distance bin. In particular, hydrogen bonding between backbone atoms seems to influence the statistics more than side-chain interactions. Particularly favorable interactions are found between backbone-backbone hydrogen bonds in hydrophobic-hydrophobic and hydrophobic-charged residue interactions. Unfavorable interactions are significant for negative-charged residues at the interface, which suggests that they may like to form hydrogen bonds with water.

Table 3.9 (a). Value of potential parameters for the short-range distance bin (0-4 Å). The rows represent hydrogen particle types and columns represent acceptor particle types. All potential values are multiplied by 1000.

Hydrogen residue type ⇓ Acceptor residue type ⇒	**Hydrophobic**	**Polar**	**Positive-charged**	**Negative - charged**
Hydrophobic	-2.712378	-1.988873	-3.209258	-3.227944
Polar	0.863366	-0.382273	-0.379366	-0.332003
Positive-charged	0.764617	0.091558	0.127956	-1.083882
Negative-charged	3.096665	2.101862	2.158442	-1.898721

Table 3.9 (b). Value of potential parameters for the long-range distance bin (4-8 Å). The rows represent hydrogen particle types and columns represent acceptor particle types. All potential values are multiplied by 1000.

Hydrogen residue type ⇓ Acceptor residue type ⇒	**Hydrophobic**	**Polar**	**Positive-charged**	**Negative - charged**
Hydrophobic	-0.346644	-0.305750	-0.178341	0.367602
Polar	-0.088703	-0.088758	-0.054776	0.115549
Positive-charged	-0.024214	0.180884	0.212524	-0.498265
Negative-charged	-0.299279	-0.565140	0.540539	2.069571

Comparison of different learning algorithms

a. Differences in theory and implementation of methods

The first distinction we make is between neural networks and the other two methods. Methods like SVMs and pairwise learning using linear programming (or linear programming, in short) are based on solving a set of inequalities (linear or quadratic programs) to obtain the parameters of the potential, while neural networks use gradient descent based minimization of error on the training set.

The next distinction is between the theory of SVMs and linear programming. The linear programming method finds a set of parameters, *u,* defining a hyperplane, such that for each pair of (correct and incorrect) structures, the resulting energy is higher for the incorrect structure. Whereas SVMs find a set of parameters *w,* defining a hyperplane, such that the margin between the correct and incorrect structures (defined by the hyperplane) is maximized. In linear programming, inequalities comparing *pairs of models* are solved in order to get the potential parameters. The inequalities solved in linear SVMs are similar: the difference is that the constraints are formulated *per model* and not per pair of models.

We also distinguish between the implementation of methods to solve the optimization problems in linear programming and linear SVMs. The set of inequalities arising in both linear programming [39] and SVMs [79] can be implemented in principle using the same optimization method, for example interior point methods. However the underlying implementations in typical SVM packages are different from those in linear programming solvers. Firstly, the linear programming solver PF3 that we use, solves the entire set of inequalities at once. The size of the matrix involved in the optimization problem is $N_{train} * n_{\dim}$ where N_{train} is the number of training examples and $n_{\dim}$ is the dimensionality i.e. number of features [39]. The highly asymmetric matrix sizes for

problems in protein structure prediction ($N_{train} >> n_{\dim}$) i.e. millions of constraints and hundreds of parameters, leads to efficient parallelization schemes for interior point algorithms [39, 40] in which the constraint matrix solved at each step is a square matrix of dimension $n_{\dim}$. As a result, the entire set of inequalities can be solved efficiently. In contrast, in SVMs the matrix size of the dual problem solved is $N_{train} * N_{train}$ [79]. This set of inequalities is solved by heuristics that use stochastic sampling, to solve a subset of inequalities at a time. The advantage of stochastic sampling is that it ensures that the problem is solved with reasonable memory resources [79]. However in practice, convergence can take longer for stochastic sampling based methods, and sampling subsets of inequalities can be less accurate than learning methods that solve all inequalities at once.

The second difference in implementation between the SVM and linear programming packages we used, is the underlying method to solve the set of inequalities. The linear programming solver PF3 [39, 40] uses Newton's method while the SVM software SVMlight [79] uses a quadratic programming solver based on Gauss- Seidel's method for solving the set of inequalities.

We note that we have used off-the-shelf software for comparisons on SVMs and neural networks. A better comparison would have been to implement non-linear SVMs and linear SVMs in the same package, so that the same underlying algorithms are used for solving the optimization problems. However, this comparison can still be useful in practice since off-the-shelf tools are blindly used, and the various packages have been optimized for performance over the years.

As an aside, we also note that the optimization problem in linear programming method is also similar to ranking SVMs [88], which is a much more computationally expensive learning method, where all-versus-all inequalities are formulated and solved.

For example, if we want to find a set of parameters such that the input examples $\{A1, A2, A3, A4\}$ are ranked in the order $y(A1) > y(A2) > y(A3) > y(A4)$, where y is the SVM output, inequalities that compare A1 to A2, A3 and A4, A2 to A3 and A4 and A3 to A4 are formulated. This is an all-versus-all set of inequalities, which results in a large mathematical program (number of inequalities is $\frac{N_{train} * (N_{train} - 1)}{2}$ where N_{train} is the size of the learning set). In our approach, we solve a subset of these all-versus-all inequalities. In particular, we use cutoffs to define a correct model (e.g. interface RMSD less than 2.5 Å) and incorrect model (e.g. interface RMSD greater than 7 Å) and the linear program only includes inequalities that compare correct and incorrect models. This procedure is not only less computationally expensive but also found to be less noisy for ranking docking models than an all-versus-all comparison[38]. One reason for this is that our ranking is based on RMSD[41], which is not meaningful at large values. For example models with RMSD of 10 and 11 Å are equally bad and ordering them is not helpful.

Linear SVMs are equivalent to neural networks with no hidden layers and multi-layer NNs can be expressed in terms of non-linear SVMs [89].

b. Accuracy

Pairwise learning using linear programming, seems to be one of the most accurate learning methods, with neural network regression performing second best. This is based on the ability of the hydrogen bond potential alone to rerank models on the soluble protein sets (Table 3.5). SVM regression potentials, in our experience perform much worse than these two.

As mentioned in the section on differences in algorithms, linear programming solvers like PF3 [39] solve the comprehensive set of inequalities all at once. While SVM packages like SVMLight [79] use heuristics for stochastic sampling of a subset of

inequalities, a few at a time. Using learning methods that solve the whole problem, without sampling subsets at a time seems to result in faster and better solutions.

c. Training time

Figure 3.3 shows that the linear programming method is fastest for training. For larger training set size and higher dimensionality of the problem (i.e. greater number of potential parameters), it will still be very efficient since the method is highly parallelized. One can get quick convergence on neural networks to a reasonable solution too, though convergence of neural networks is not very well-defined, and involves various stopping criteria to prevent overfitting [85].

The convergence of SVMs with stochastic sampling of inequalities is much slower. Note that in SVMs, the (dual) quadratic problem of dimension $N_{train} * N_{train}$ needs to be solved, while in our approach to learning which is based on linear programming, the dimension of the complete constraint matrix is much smaller and is $N_{train} * n_{\dim}$ where N_{train} is the number of training examples and $n_{\dim}$ is the dimensionality i.e. number of features [39]. PF3 further exploits this structure by splitting the matrices to dimension $n_{\dim} * n_{\dim}$ and solving each subset using Newton Raphson methods[39].

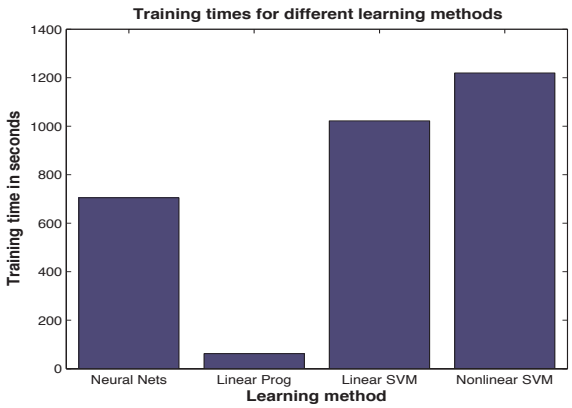


Figure 3.3 Average training time in seconds over all models obtained by different learning methods: Neural Networks, Pairwise Learning using Linear Programming, Linear and Non-linear SVMs. The times were calculated on single Intel® Xeon(® E5345 core of an 8-core machine with 8 GB memory and 2.33GHz clock speed.

d. Test time

Figure 3.4 shows the test time, i.e. time taken to obtain the hydrogen bond energy from various potentials, for 1000 models of a complex. Linear programming is the most efficient since it involves only the computation of a dot product, $n(\alpha,d)u(\alpha,d)$ for a given model.

The time taken by neural networks for predicting the energy is almost as small as the time taken by linear programming to predict the energy. This is interesting as neural network prediction is usually more expensive than SVM prediction, since it involves successive matrix multiplications while SVM prediction depends only on the number of support vectors. We note that the package FANN [84] used for neural network training and testing was optimized for good testing performance, to be used in real-time systems. In practice, neural network output calculation involves $n_l - 1$ successive matrix multiplications, where n_l is the total number of layers. The *i*th multiplication involves a matrix of dimension $N_i * N_{i-1}$ where N_i and N_{i-1} are the number of neurons in layer *i* and

i-1 respectively. In our case, the maximum size of the matrix is still quite small, and is *32*10* for the hidden layer neurons, with 32 input features and 10 neurons in the hidden layer.

On the other hand, SVM output prediction is linear in the number of support vectors used to describe the hyperplane. The support vectors are the training examples that lie on the margin. In our case, we had 6097 support vectors for the linear SVM and 1279 for the non-linear SVM. For linear SVMs, even though the energy (output) can be computed efficiently as a dot product like in the case of linear programming, practical implementations in SVM packages treat the linear case like the non-linear case, and use the set of support vectors to compute the output. Hence SVM predictions take longer.

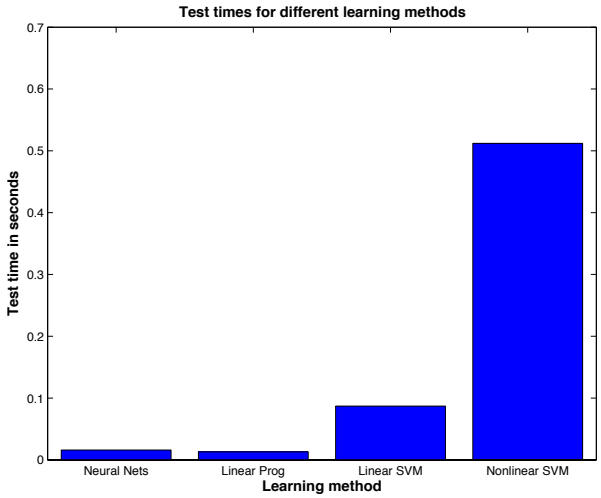


Figure 3.4 Total test time in seconds for calculating the energy of 1000 models of a complex containing 147 and 103 residues in receptor and ligand protein. Time obtained by different learning methods: Neural Networks, Pairwise Learning using Linear Programming, Linear and Non-linear SVMs, on a single Intel® Xeon(® E5345 core of an 8-core machine with 8 GB memory and 2.33GHz clock speed is shown.

e. Advantages and disadvantages

As mentioned before, the accuracy and training/test times of algorithms is in the order Pairwise Learning using Linear Programming > Neural Networks > SVMs. If a linear fit is good enough to obtain a potential, pairwise learning using linear programming seems to be the method of choice, while neural network regression can be used as the best option if a non-linear fit is desired

SVMs and Linear Programming both lends themselves to a geometric interpretation of the problem. Also, they both lead to sparse solutions in R^n where n is the number of features (dimension): this is ensured by the objective functions that minimize the sum (or sum of squares) of the parameters. Furthermore, the problems solved in these two cases are convex optimization problems, which when solved exactly, converge to a unique global minimum. On the other hand, neural networks rely on heuristics like gradient descent, that converge to a local minimum.

Linear programming and linear SVMs have the additional advantage that the parameters are readily amenable to physical interpretation. For example, the parameters $u(\alpha,d)$ in the hydrogen bond potential represent the weight of a contact between particle type pair α at a distance d. The non-linear SVM and neural network potentials are more complex and harder to interpret.

Further, SVMs and Neural Networks require the tuning of additional meta parameters during training, such as the cost parameter for SVM, degree of polynomial etc. for SVM, and the hidden and output activation functions for neural networks. Comparatively, the number of such parameters is very low in linear programming and their effect on the quality of solution is small.

Classification (not regression) using neural networks and SVMs is especially error prone for docking data, since the datasets have a much larger number of negative

examples than positive examples. Since classifiers are sensitive to imbalances in the training set, almost all models end up being classified as negative [80]. Hence it is better to use regression instead of classification for the purpose of ranking docking models.

Neural Networks are simpler to understand and implement, but more prone to overfitting [85]. However, by proper use of a validation set, this problem can be eliminated. Recently deep learning networks, or sophisticated networks with a large number of hidden layers and thousands of parameters, have been shown to outperform existing learning methods on a wide range of tasks [90]. The learning procedures are highly computationally intensive but can be parallelized using GPUs. Using deep learning for obtaining docking potential is expected to enhance the quality of the ranking further.

3.5 CONCLUSIONS

Using hydrogen bonding to distinguish between correct and incorrect binding for soluble proteins is hard as the net free energy gain upon binding is small [46]. This could be perhaps due to competition with water for interface hydrogen bonds. We see that hydrogen bond potentials carry much less signal than atomic and residue potentials for soluble protein complexes. In contrast, hydrogen bonding information is much more informative than traditional atomic and residue potentials in the context of membrane proteins, since their hydrophobic environment lacks competition from water. Nevertheless, the addition of hydrogen bonding potentials to atomic and residue potentials improves the accuracy of reranking in both soluble and membrane proteins.

An assessment of various learning algorithms for learning potential functions for protein docking is presented: this is the first such assessment of learning methods for reranking docking models, to the best of our knowledge. We show that pairwise learning

using linear programming performs best in terms of accuracy, training and testing time, followed by neural networks and SVMs. Future work could include obtaining potentials using recent machine learning methods such as deep networks.

Chapter 4. Docking membrane proteins

4.1 INTRODUCTION

In this chapter, we apply our docking package, DOCK/PIERR for predicting the structure of membrane protein complexes[91]. We introduce novel adjustments to the docking algorithm, to improve the accuracy of prediction for membrane proteins. We show that this membrane version of DOCK/PIERR, DOCK/PIERR-Membrane performs comparably to other leading docking packages. We further employ DOCK/PIERR-Membrane for predicting dimers of the amyloid precursor protein, an important membrane protein involved in the pathogenesis of Alzheimer's disease. Docking results are shown to agree well with results from implicit solvent MD simulation, another computational method that allows for significant protein movements. Finally, some interesting differences are uncovered between structures obtained by different computational methods (implicit and explicit solvent simulations) and structures from different membrane models (bilayer and micelle).

Membrane proteins are critical for transport of material across cell boundaries and for transmitting signals into and out of cells. Several diseases and aggregation phenomena have been associated with peptide interactions in membranes. Over 50% of current pharmaceutical drugs target G-Protein Coupled Receptors, a class of membrane proteins[87, 92]. Hence the study of membrane proteins and their aggregation is of general biomedical importance.

Rigid docking can be a useful computational tool for deducing membrane protein structure. Firstly, it can sample exhaustively, the set of all possible rigid conformations of the complex, on a lattice. This sampling is more comprehensive than the sampling obtained from equilibrium MD simulations. Second, docking, if established to be accurate, can be an efficient means of sampling higher order conformations of the peptide

(oligomers), hence providing atomic detail into the structure of aggregates, as a quicker computational alternative to MD simulations[92]. Finally, the potentials used in a docking algorithm such as DOCK/PIERR are based on contacts observed in protein interfaces and incorporate a different kind of information from the force fields used in simulation.

Docking algorithms like Cluspro[11] and Haddock[93] have been used previously to study the structures of several membrane complexes [94-97] . In a recent study [87], a comparison was made between different docking algorithms for predicting membrane protein complexes. Though docking algorithms have been designed primarily for aqueous solution, they are shown to be useful in predicting transmembrane complexes with only minor adjustments. However, docking methods have some drawbacks such as limited conformational flexibility, and not accounting for the membrane environment[92].

In this study we address the latter drawback by incorporating an additional energy term corresponding to the membrane environment. The membrane environment is known to influence the structure and function of proteins [98]. The energy term is a simple one-body term obtained by others to quantify the transfer energies of different molecules from aqueous solutions to membrane[99]. We show that by adding this simple energy term and retaining the rest of our docking algorithm, we are able to improve the accuracy of DOCK/PIERR in predicting transmembrane protein complexes. Earlier, others have used a membrane term in filtering rigid docking solutions from ZDOCK [97]. However, they only consider filtering based on orientation of monomers in the membrane. Here in addition to orienting the docking models in the membrane, we compute a novel membrane energy term using transfer free energies from simulation. Also, rather than patching algorithms of other groups, we refine our own method, which gives us easy access to the code and deeper understanding of the algorithm function.

DOCK/PIERR-membrane is applied to predict dimers formed by the 23-55 fragment of the amyloid precursor protein, APP-C99. The C99 amyloid precursor is a transmembrane fragment of 99 residues cleaved from the C-terminal end of the longer (600+ residues long) amyloid precursor protein. C99 is further cleaved by enzyme γ-secretase, to form the amyloid β peptide of length ranging between 38 and 43 residues, of which the peptides of length 40 and 42 residues are the most pathogenic. The aggregation of these peptides in the cell membrane results in formation of fibrils and ion channels, resulting in cell death[100].

Here we study the dimerization of the 23-55 fragment of the amyloid precursor C99 fragment. The 23-55 fragment includes the cleavage site for γ-secretase. By studying its dimer structure, we hope to elucidate factors affecting the stability of the dimer. The stability of the dimer affects the amount of amyloid β peptide released into the membrane, and hence affects the pathogenesis[101].

In this study, we dock the monomers of amyloid precursor (APP) obtained from simulation, and compare the results of simulation and rigid docking. Others have performed comparisons of rigid docking and simulation for dimers of Glycophorin-A and its mutants, and concluded that results from implicit solvent simulation match well with that from rigid docking. Here, we discover the same for the amyloid precursor protein, and additionally we discuss differences in structures obtained from different computational methods.

4.2 METHODS

In this section, we first describe the membrane score added to DOCK/PIERR to mimic the membrane environment. Second, we describe the dataset of unbound

membrane protein complexes, used for establishing performance of various docking algorithms, along with a brief note about the docking algorithms whose performance we compared. Finally we explain the approach used to dock the APP monomers obtained from simulation.

Docking Algorithm

The docking algorithm, DOCK/PIERR [13, 41] is used as in Chapter 2, to dock membrane proteins. In the first phase, an exhaustive set of structures is sampled using Fast Fourier Transforms and the residue-based potential, PIE with a van der Waals term. These structures are then clustered using ligand RMSD and interface RMSD to remove very similar structures, and additionally structures with too many clashes are eliminated. Refer [13] for details on this phase. In the second phase, the top 1000 models from the first phase are adjusted using side chain remodeling and minimization and reranked using the combination potential C3, a combination of interface residue potential PIE and interface atomic potential PISA.

Membrane potential for reranking docking models

The docking algorithm described above, only examines the interface contacts of the models and does not incorporate information about the environment surrounding the complex. The potentials PISA and PIE used for scoring interface contacts are derived empirically from datasets of experimental and model structures of globular protein complexes (their training set includes only 7 membrane proteins of a total of 640) [38].

Nevertheless, it is tempting to keep the designed potentials “as are” and look for an additional term to score the effects of the membrane. This will make the potential more modular, transferable and general. We add such a term that includes residue-specific information about membrane solvation, and show that it enhances prediction

accuracy in membrane complexes. This term is used along with C3 in ranking the top 1000 refined structures, in the last step of the docking procedure described above. We note it is also possible to add such a term in the coarse scoring step, but we added it in the last reranking step for convenience. We next describe how to compute this additional term.

a. Calculating membrane energy

Rather than design a membrane environment potential from scratch, we adopted a function that was developed by other investigators. Previous results from MD simulations by Tieleman and co-workers consider transfer free energy from aqueous solution to the center-of-membrane for each amino acid residue [102]. Their detailed and comprehensive simulations provided us with singe body adjustments that measure the costs (and rewards) of transferring each amino acid between the two environments. The underlying physical assumption is that the one-body term captures the environment effect and that the impact of the membrane on the two body interactions is significantly smaller and neglected. The drawback of our choice is that the atomically detailed simulations and our machine learning procedure are not necessarily compatible and some double counting of the same effect may occur. On the other hand, the combination of our potential with the Tieleman's energy does not include free parameters, making it relatively simple to verify the impact and the significance of the combination. We observe a large enhancement in prediction capacity, which suggests that the environment potential indeed captures a useful signal.

The membrane energy was calculated from these transfer energies using the following steps. First, each docking model was inserted into the membrane, by placing its center of mass at the center of the membrane, and by orienting the eigenvector corresponding to the smallest eigenvalue of the tensor of inertia of the model of the

protein complex along the membrane normal. This orientation is appropriate for elongate trans-membrane proteins such as helical proteins, which are our prime targets in the study of amyloid peptides. For wide proteins, a different orientation procedure will have to be used, since the eigenvector with the smallest eigenvalue is not necessarily in the direction normal to the membrane. Second, for each docking model, the relative solvent accessibility of every residue was calculated with the program DSSP [103]. Finally, the membrane energy was calculated as follows: each residue whose side chain center of mass was within a specified membrane width contributed to the membrane energy. The contribution from such a residue, *i,* was equal to the membrane transfer energy for that residue, t_i, weighted by its relative solvent (lipid) accessibility, a_i. As shown in Eq. (4.1), the membrane transfer energy, or *MTE,* for a model, is the sum of the transfer energy contributions from all residues *i,* within the membrane width.

$$MTE = \sum_i a_i t_i \tag{4.1}$$

We note that Tieleman and co-workers also provided water-to-hydrophilic membrane interface transfer energies, apart from water-to-center of membrane transfer energies. The addition of these extra parameters did not contribute to improved accuracy in ranking and hence they are not included in our docking algorithm for membrane complexes.

b. Membrane widths

The membrane half-width along the Z-axis is, important for our calculations since it determines the degree of exposure of different amino acid side chains to the membrane environment or to aqueous solution. However, membrane widths are not strictly fixed and can vary among different membrane proteins [104]. For experimentally determined structures the width is known; however, for model complexes and variable composition

of lipids it is not. Servers like TMDET [104] and databases like the PDBTM database [105] store pre-calculated widths for membrane proteins whose experimental structure has been determined. But these are difficult to use when ranking hundreds of thousands of models, with different effective membrane widths, and when studying complexes for which the experimental data is limited. To pick up a width which is consistent and optimal within our model, we use the following procedure: for each docking model, membrane transfer energies were calculated for a range of half-widths: 16 Å +/- 3 Å, in steps of 0.5 Å i.e. for 13.0, 13.5, 14.0, 14.5…16.0, 16.5, 17...19 Å respectively. For each width, only protein residues whose centers of mass are within the membrane boundaries are scored according to Eq. (4.1) and contribute to the membrane energy for that width. The lowest (best) membrane transfer energy over the range of widths was taken as the score for the docking model. Figure 4.1 shows an example of a model oriented in the membrane, and a particular residue, *i*, inside the membrane that contributes to the membrane energy.

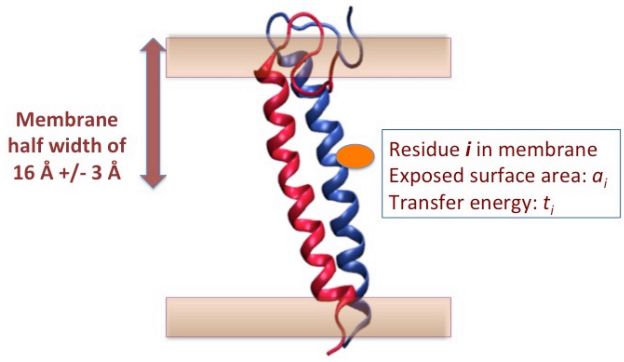


Figure 4.1 Example of a model oriented in the membrane, and a particular residue, *i*, inside the membrane that contributes $a_i t_i$ to the membrane energy, where a_i is the residue exposed surface area and t_i is the residue membrane transfer energy.

c. Bilayer versus micelle membrane environments

For docking membrane proteins characterized in a micelle environment, instead of the regular linear membrane model, a spherical membrane model with radius of 16 Å +/- 3 Å is used to calculate the membrane energy.

d. Combining membrane energy with docking scores

The membrane energy (henceforth known as MTE) was combined with C3 in a parameter-free fashion by using the product of C3 with MTE. The product energy in this study was formulated as $k*C3*MTE$ where $k=1.0$ if both C3 and MTE have positive values and $k=-1.0$ otherwise. This ensures that the product energy is negative when both energies are negative (favorable) and positive otherwise. We henceforth refer to the product energy as C3*MTE.

Other docking algorithms

The performance of DOCK/PIERR was compared to Cluspro [12, 28], GRAMM-X [12] and ZDOCK+ZRANK [29, 60]. We have compared our algorithm to these approaches in the past for the case of protein complexes in aqueous solution and it therefore makes sense to extend our comparison to membrane proteins. Previous comparative docking studies have shown that these algorithms were among the best performing algorithms for membrane protein docking [87, 106]. Results were obtained from the servers in case of Cluspro and GRAMM-X. For ZDOCK+ZRANK, the ZDOCK 3.0.2 package was downloaded and docking jobs were run locally. The top 2000 models from ZDOCK were rescored using the ZRANK scoring function.

Creation of unbound membrane protein complexes dataset

A data set of 30 transmembrane protein complexes was extracted from MPStruc [106], a database of membrane proteins from the White laboratory. Representative

structures were chosen from each of the classes to ensure functional and structural diversity. The membrane span of the selected proteins was checked using the PDBTM database [105], a database of transmembrane proteins in the PDB. Proteins selected from the MPStruc database, that had no entry in the PDBTM database, were discarded. Proteins classified as membrane proteins often do not span the entire length of the membrane and can interact with just one small region of it, e.g. peripheral membrane proteins. The PDBTM database was therefore used to determine the extent to which each protein was embedded in the membrane. Integral membrane proteins where the majority of the structure to be docked, lay in the transmembrane region were chosen.

We obtained 18 complexes for docking two separate protein chains. To increase the number of experimental models that we can study, we also considered single-chain multi-pass trans-membrane proteins (e.g. GPCRs) that we broke to two complementing fragments, at an extramembranous loop region, and re-assembled. We obtained 12 complexes this way. For each chosen GPCR, multiple independent splits were made, and each split produced two chains to be docked. Each independent split was taken as a separate target for unbound docking. Table 4.1 shows that we obtained 12 targets from the GPCRs 1C3W, 1H68, 1M0K and 2BRD, 3 per GPCR, in this manner. Finally, we also discarded transmembrane chains where the binding between the chains was intricate, i.e. one of the chains twisted around the other. For each protein complex chosen, Table 4.1 shows how we obtained the individual components to dock. The entry labels correspond to the PDB IDs.

Table 4.1 Targets and individual chains that formed the dataset of 30 transmembrane proteins.

Target	Original PDB	Receptor chain		Ligand chain	
		Chain: start residue	Chain: end residue	Chain: start residue	Chain: end residue
1A91	1A91	A:1	A:42	B:43	B:79
1BL8	1BL8	A:23	A:119	B:23	B:119
1C17	1C17	A:1	A:79	B:1	B:79
1C3W0	1C3W	A:75	A:231	A:5	A:74
1C3W1	1C3W	A:102	A:231	A:5	A:101
1C3W2	1C3W	A:5	A:129	A:130	A:231
1EHK	1EHK	B:3	B:168	C:2	C:34
1H2S	1H2S	A:1	A:225	B:23	B:82
1H680	1H68	A:94	A:219	A:2	A:93
1H681	1H68	A:2	A:119	A:120	A:219
1H682	1H68	A:2	A:150	A:151	A:219
1JVM	1JVM	B:24	B:123	C:24	C:120
1LGH	1LGH	A:1	A:56	D:1	D:56
1M0K0	1M0K	A:73	A:231	A:5	A:72
1M0K1	1M0K	A:106	A:231	A:5	A:105
1M0K2	1M0K	A:5	A:128	A:129	A:231
1M56	1M56	C:2	C:266	D:10	D:51
2BHW	2BHW	A:10	A:232	B:10	B:232
2BRD0	2BRD	A:66	A:228	A:7	A:65
2BRD1	2BRD	A:103	A:228	A:7	A:102
2BRD2	2BRD	A:7	A:129	A:130	A:228
2IRV	2IRV	B:93	B:271	A:92	A:273
2KSE	2KSE	A:1	A:40	A:150	A:186
2NRF	2NRF	A:91	A:272	B:91	B:272
2VT4	2VT4	A:40	A:358	B:39	B:359
2WIE	2WIE	A:2	A:82	B:2	B:82
3B45	3B45	A:169	A:270	A:91	A:168
3B4R	3B4R	A:3	A:220	B:3	B:218
3DWW	3DWW	A:11	A:152	C:11	C:152
3KCU	3KCU	A:29	A:280	B:29	B:280

Modeling unbound chains by homology and creating distorted structures by Molecular Dynamics

First, for each receptor and ligand sequence in the set of 30 transmembrane complexes, a search for homologs in the PDB was performed using PSI-BLAST [61]. For complexes for which homologs (E-value lower than 0.001 i.e. expectation that the two sequences are evolutionarily related by chance is less than 0.001) were found for receptor and/or ligand chains, Modeller [59, 62] was used to create a structure of the unbound receptor and ligand using the homolog as template. The TM score [45] of the bound to unbound structure was measured for each homology-modeled receptor and ligand chains. Unbound (modeled) conformations that were too different (i.e. TM score lower than 0.85) from the PDB (bound) conformation were discarded.

In all, we were able to successfully produce homologous unbound conformations for both chains in 19 of 30 complexes. Apart from these 19, 4 complexes had one unbound chain (receptor or ligand) with TM score lower than 0.85 to the bound structure, and the other chain with a TM score higher than 0.85 to the bound structure. For these 4 complexes, the unbound structures with TM scores lower than 0.85 were replaced with the bound (PDB) conformation and bound-unbound docking was performed. 4 other complexes had both receptor and ligand unbound conformations quite different (TM score lower than 0.85) from the bound conformations. And for 3 complexes, homologs were not found in the first step of PSI-BLAST. Hence the latter 7 complexes were treated separately and molecular dynamics was used to obtain the unbound conformations in these 7 cases, as is described next.

For the seven complexes for which homology modeling was unsuccessful, unbound conformations of the receptor and ligand were obtained from short Molecular Dynamics MD runs on the original PDB receptor and ligand structures. The receptor and

ligand were separately minimized in vacuum for 100 steps using mini_pwl, an energy minimization routine in the MD package MOIL [58] in order to remove high-energy contacts and clashes in the structures before the dynamics run. Then the minimized structures (receptor and ligand separately) were subject to a very short simulation of 0.1 ps at 300K (1000 steps with a time step of 0.0001 ps). The conformations obtained after the dynamics run were used as the unbound structures. These perturbed conformations had an average RMSD of 0.717 A to the original PDB structures, and a range of RMSDs between 0.618 A and 0.859 A. These RMSD values are smaller than typical homology models, however, MD under the above conditions distorts significantly, the structures of the proteins and therefore we did not push the simulations to produce higher RMSDs.

Approach for docking APP structures from simulation

A set of 50 dimers of the 23-55 segment monomer of APP-C99 corresponding to the lowest energy (based on the MD molecular mechanics energy) structures obtained from 100 ns equilibrium implicit solvent MD simulations at 300 K with the Martini force field in CHARMM [101], were used to test the performance of docking. Implicit solvent simulations represent the solvent e.g. water or membrane by a continuum model, while in explicit solvent simulations, the solvent is represented by discrete solvent molecules. Explicit solvent simulations are thus more computationally expensive, but also more accurate.

Both bound and unbound docking was performed on each set of simulation structures. In bound docking, the monomers i.e. individual helices of each simulated dimer were separated and docked, producing ten top scoring models from docking, for each simulation complex. For unbound docking, a simulation structure (say A) was chosen at random and its receptor (one of the helices in the simulation dimer) was docked

with a ligand (the other helix in the dimer) taken from another simulation structure (say B) in the same dataset. The models produced by docking A's receptor to B's ligand, were compared to the complex A. About 50 (or 30, depending on the number of complexes in the bound dataset) non-repeating A-B receptor-ligand pairs were docked. Since the monomer conformations themselves can be quite different (greater than 1 Å RMSD) from each other in simulations, the selection of complex B each time was constrained to those complexes where the ligand was within 1 Å RMSD from the ligand in complex A.

Additionally, as a final post-processing step for docking APP structures and comparing rigid docking procedure to simulations of peptide dimerization in membrane, anti-parallel dimer poses were filtered out from the final set of docking models, by making use of the additional information that the dimers found in the MD simulation are never anti-parallel. The last observation may reflect kinetic barrier and not necessarily thermodynamic preference, however, for comparison purposes the above filtering was found useful.

A cutoff of 1.5 Å interface RMSD was used as definition of "hit" or near-native structure, while evaluating docking methods on the APP dimers, since the monomer helices are short and only 33 residues long. This is in contrast to the usual cutoff, which is 4 Å for an acceptable model and 2.5 Å for a high-quality model in protein-protein docking assessments such as CAPRI [44, 56].

4.3 Results and Discussion

In this section, we first discuss results on prediction of membrane protein complexes. Second, we discuss the results from docking implicit solvent APP simulation dimers. Third, we discuss differences between dimers obtained from implicit and explicit

solvent simulations. Fourth, we touch upon differences in structures obtained from micelle and bilayer membrane environments.

Structure prediction of membrane protein interactions

a. Membrane protein interfaces can be predicted by solvated protein docking algorithms

We find that interfaces of membrane and water-soluble protein complexes are quite similar [87] and can be predicted with reasonable accuracy by current state-of-the-art protein-protein docking algorithms. This implies that protein-docking algorithms can be used as an additional and a reliable source of information for structural studies of membrane proteins. We note that protein docking algorithms use potentials that have been trained on datasets that are primarily composed of soluble proteins; for example, Cluspro and Gramm-X use the training set in [107] which consists of 621 protein complexes out of which only 6 are membrane proteins, DOCK/PIERR is trained on a dataset of 640 complexes with a similar percentage of membrane proteins, and ZDOCK's interface contact potentials are trained on a dataset [108] of 89 complexes with one membrane protein.

In spite of being trained on interfaces of soluble proteins, these docking algorithms succeed in predicting a near-native structure in the top ten models with reasonable accuracy. Table 4.2 shows the performance of 4 different docking algorithms on the dataset of 30 unbound transmembrane protein complexes. The measure of performance that we use here is the interface RMSD. Interface RMSD [44, 56] is a widely used measure of accuracy for docking predictions, and is the RMSD measured along the interface residues of the experimental complex. The second column in Table 4.2 shows the number of hits (near-native structures i.e. docking models that are within 4 Å interface RMSD from the bound structure) in the top ten models cumulative across all

30 complexes, followed by number of complexes for which at least one such hit was found in the top ten models. Depending on the algorithm, accuracy varies between 30-56.57% for unbound docking. Gramm-X performs the best in this study and is able to obtain a near-native structure in the top ten about 56.67% of the time in unbound docking. This is in agreement with an earlier study [87] that showed Gramm-X to have the best performance in docking membrane proteins.

Table 4.2 Docking performance of DOCK/PIERR with C3 and C3*MTE potentials, Gramm-X, Cluspro and ZDOCK+ZRANK on the dataset of 30 unbound membrane protein complexes.

Docking algorithm	**Top 10 Number of hits within 4 Å iRMSD/Number of targets with atleast one hit**
DOCK/PIERR Rerank with C3	2/2
DOCK/PIERR Rerank with C3*MTE	14/11
ZDOCK+ZRANK	10/9
Cluspro	17/14
Gramm-X	20/17

Table 4.3 shows the performance of docking algorithms in terms of number of top ten hits, split by target. DOCK/PIERR with the membrane score is able to dock complex 1H2S, which the other docking algorithms are not able to solve. Similarly, ZDOCK+ZRANK is able to solve uniquely 1JVM and 3DWW. Gramm-X is the only docking algorithm able to solve 3B4R.

b. Membrane energy contributes to improved recognition

As shown in Table 4.2, the inclusion of the membrane energy, significantly improves the recognition of the combination of atomic and residue potentials, C3.

DOCK/PIERR is able to obtain a near-native structure in the top ten in 36.67% of complexes.

Table 4.3 The numbers of models with interface RMSD less than 4.0 Å in the top 10 predictions of DOCK/PIERR with C3*MTE potential, Gramm-X, Cluspro and ZDOCK+ZRANK.

Target	**DOCK/PIERR with membrane score**	**ZDOCK+ ZRANK**	**CLUSPRO**	**GRAMM-X**
1A91	1	1	2	1
1BL8	0	0	0	0
1C17	0	0	1	1
1C3W0	1	0	1	1
1C3W1	1	0	1	1
1C3W2	0	1	1	1
1EHK	0	0	0	0
1H2S	3	0	0	0
1H680	0	0	1	1
1H681	0	0	0	0
1H682	0	0	2	2
1JVM	0	1	0	0
1LGH	0	0	0	0
1M0K0	1	0	1	1
1M0K1	1	1	1	1
1M0K2	0	1	1	2
1M56	0	0	0	0
2BHW	0	0	0	0
2BRD0	1	0	0	1
2BRD1	0	0	1	1
2BRD2	2	0	2	1
2IRV	1	1	0	0
2KSE	0	1	0	2
2NRF	0	0	0	0
2VT4	0	0	0	0
2WIE	1	2	1	1
3B45	1	0	1	1
3B4R	0	0	0	1
3DWW	0	1	0	0
3KCU	0	0	0	0

Docking and implicit solvent MD simulations agree on structures of APP dimers

In this section, we explore the structure of the dimer formed by the 23-55 segment of the APP-C99 protein using docking and implicit solvent MD simulations. Table 4.4 shows the performance of DOCK/PIERR for bound and unbound docking of 50 implicit solvent dimers from simulation. The docking performance was evaluated based on the number of models matching the corresponding MD structure within 1.5 Å interface RMSD. Table 4.4 reports the number of models in the top ten that matched the corresponding MD complex, across all 50 complexes. Also reported is the number of complexes out of 50, for which at least one model in the top ten matched the corresponding simulation structure. Docking and MD simulation show a good agreement with 42 out of 50 dimers from bound docking matching the corresponding MD structure, and 26 out of 50 dimers from unbound docking matching the MD structure. The accuracy of unbound docking is lower than that of bound docking, which is to be expected, as the interfaces of monomers from unbound docking do not match precisely.

Table 4.4 Bound and unbound docking results on 50 simulation structures from implicit solvent. The first number in the second column is the number of MD models recovered from docking across all 50 complexes: a hit is a model from docking that is within 1.5 Å interface RMSD to the corresponding simulation structure. The second number is the number of complexes for which at least one hit was found in the top ten models.

Docking type	**Top 10 Number of hits within 1.5 Å iRMSD to MD structure/Number of complexes with atleast one hit matching MD structure**
Bound	43/42
Unbound	26/26

Figure 4.2 shows the probability distribution of interface RMSDs for the top 10 docking models from bound and unbound docking of the 50 simulation dimers. In other words, this is a distribution across a set of 500 bound and 500 unbound docking models. Note that since we filter out anti-parallel orientations, the interface RMSD distribution stops at 10 Å (x-axis). There is a prominent tail near 1 Å, especially for bound docking indicating a significant number of near-native structures in the set of top 10 models. Another measure of confidence in docking predictions is the z-score. The average z-score of the C3*MTE energy across the 5 best docking models (best in terms of interface RMSD) was -4.2646 among the 500 bound docking models and -3.5062 among the 500 unbound docking models. More negative z-scores indicate that the potential can distinguish near-native structures more accurately.

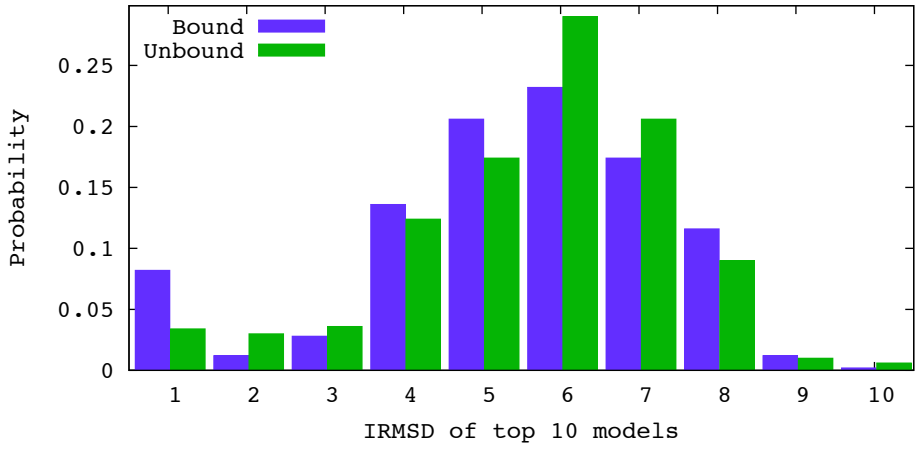


Figure 4.2 Probability density of the interface RMSD of top 10 docking models for 50 bound and unbound simulation dimers.

Further, APP dimers can be described by an order parameter based on the distance between the GLY-29 in the two helices[101]. If the distance is within 5 Å, the dimer is said to be in *Gly-in* conformation, if the distance is between 5 and 10 Å, the dimer is in *Gly-side* conformation and if the distance is above 10 Å, the dimer is in *Gly-out*

conformation. Based on this characterization, out of the 50 lowest energy simulation dimers from implicit solvent, 40 were of *Gly-side* type and 10 were of *Gly-in* type. There were no *Gly-out* structures in the 300 K MD ensemble. Table 5 shows the performance of docking in recovering the order parameters measured in the MD simulations for bound docking. The agreement between docking and simulation dimers is high (9/10) for *Gly-in* type structures and good (33/40) for *Gly-side* structures.

Table 4.5 Bound docking results on 40 *Gly-side* and 10 *Gly-in* simulation structures from implicit solvent. The first number in the second column is the number of docking models within 1.5 A interface RMSD from the corresponding simulation structure, across all complexes of the given dimer type. The second number is the number of complexes for which at least one hit was found in the top ten models for that dimer type.

Simulation dimer type [Number of simulation dimers]	**Top 10 Number of hits within 1.5 Å iRMSD to MD structure/Number of complexes with atleast one hit matching MD structure**
Gly-side [40]	34/33
Gly-in [10]	9/9

Figure 4.3 shows a couple of accurate docking predictions among the top ten models, superposed with the simulation structure they were assembled from. The *Gly-side* model was within an interface RMSD of 0.563Å from the simulation structure while the *Gly-in* model was within 0.632Å from the simulation structure. The figure shows that the backbones essentially overlap while the side-chains show minor differences.

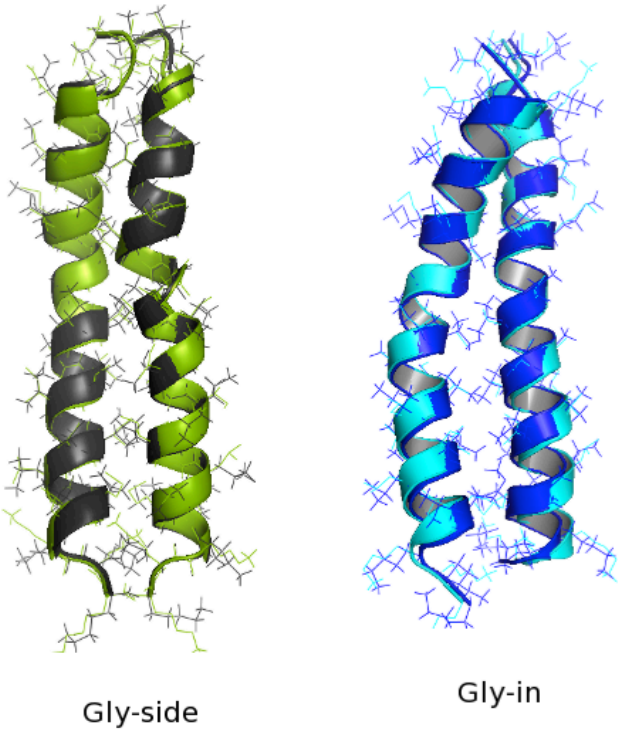


Figure 4.3 Left: A docking model (green) in the top 10 predictions, at an interface RMSD of 0.563 Å from the corresponding simulation structure (gray) of *Gly-side* type. Right: A docking prediction (cyan) in the top 10, at an interface RMSD of 0.632 Å from a *Gly-in* simulation structure (blue).

Structural differences between the results of explicit and implicit solvent methods to predict complexes of amyloid peptides

As Table 4.5 shows, DOCK/PIERR docking is reasonably accurate for *Gly-in* complexes generated by implicit solvent simulations in bilayer. However, when applied to dock 30 *Gly-in* complexes from explicit solvent POPC bilayer, it was observed that DOCK/PIERR fails to produce a single hit in the top ten models for any of the 30 complexes. These differences in docking performance hint at structural differences in the dimers from implicit and explicit solvation. The differences were investigated using the residue score PIE, which is represented as an energy here by inverting its sign (lower the energy, better the model).

Figure 4.4 is a distribution of the PIE energy for the implicit and explicit simulation dimers. The PIE energy is much lower for the implicit solvent dimers. This suggests that the number of inter-helical residue-residue contacts is higher for the implicit solvent dimers, leading to more favorable (lower) PIE energies for the latter. The contact based potentials in DOCK/PIERR favor the higher number of contacts in implicit solvent models, due to which docking models agree more with implicit solvent dimers than with explicit solvent dimers.

The compactness of helices in the dimers seems to be reason for different number of contacts in implicit and explicit solvent. This is seen in Figure 4.5, which is a distribution of the smallest eigen value of the tensor moment of inertia for each simulation structure. The smallest eigen value corresponds to the long axis and is hence a measure of how close the helices are to each other. The figure suggests that the implicit solvent dimer helices are closer than the explicit solvent dimers. In implicit solvent, the hydrophobic residues in the dimers form more contacts with each other, whereas in explicit solvent the residues form more contacts with the membrane. This leads to more compact dimers in implicit solvent. In explicit solvent models, perhaps protein-protein contacts are more easily replaced by protein-water contacts. In implicit solvent models, the protein contacts are not replaced.

Figure 4.6 illustrates that the implicit solvent models have helices closer to each other at the C-terminal (right hand side) end, whereas in explicit solvent models, the helices are further apart. This suggests that interactions formed by discrete water molecules are not fully captured by continuum models.

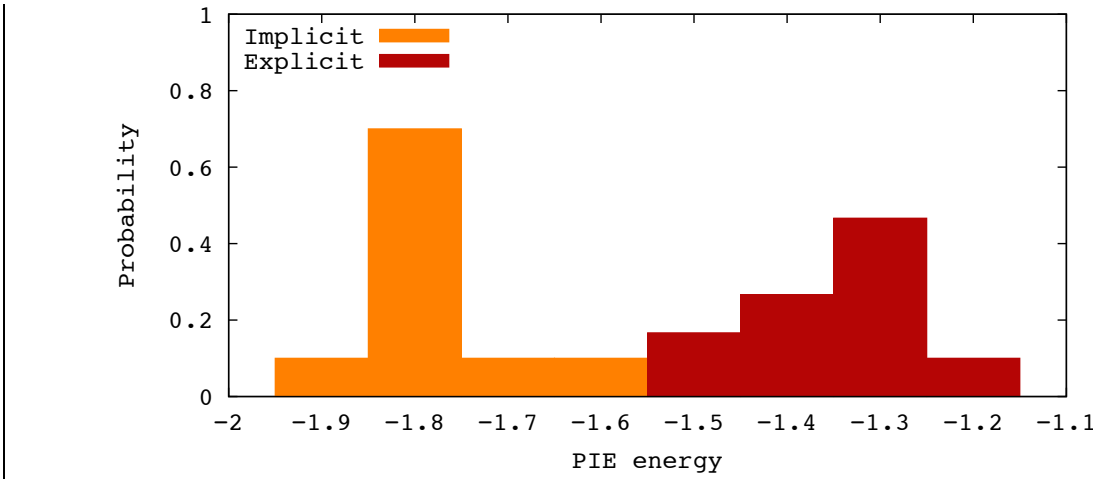


Figure 4.4 Probability distribution of PIE energy for 10 GLY-in implicit solvent dimers and 30 GLY-in explicit solvent dimers in POPC membrane that were bound docked.

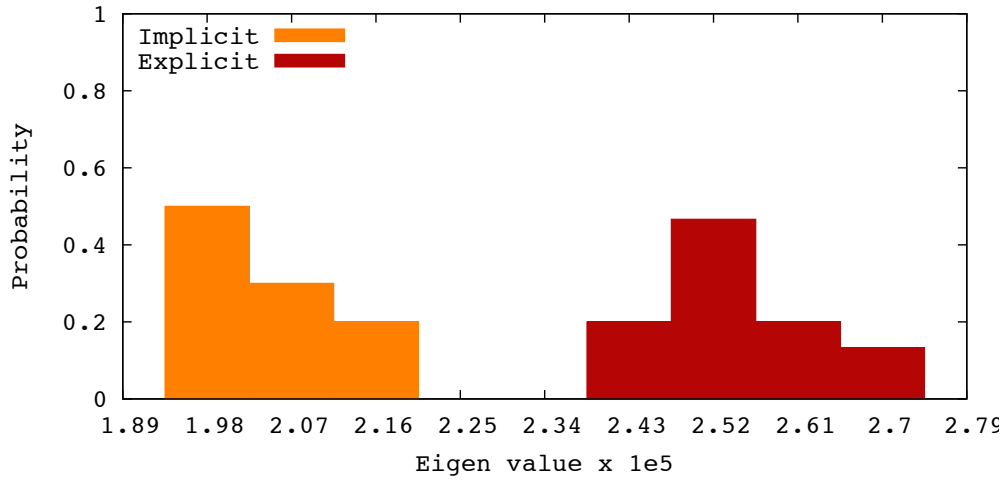


Figure 4.5 Distribution of the smallest eigen value of the tensor moment of inertia for 10 GLY-in implicit solvent dimers and 30 GLY-in explicit solvent dimers in POPC membrane that were bound docked.

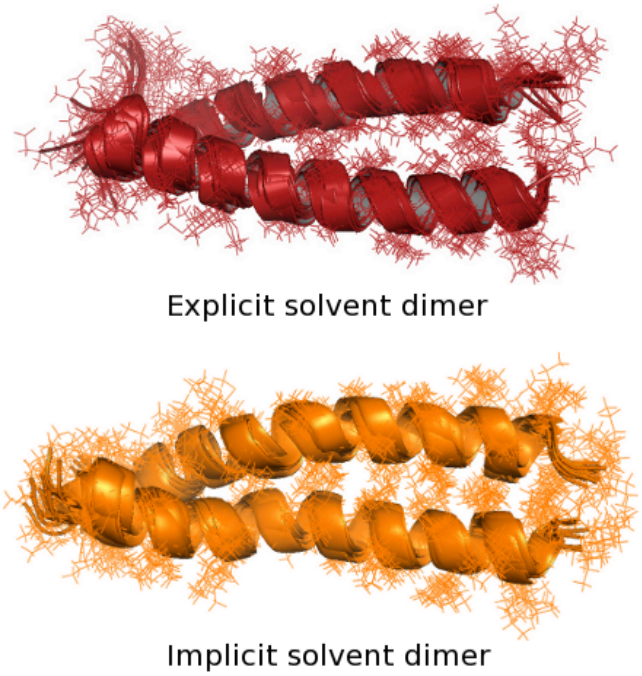


Figure 4.6 Top: 10 explicit solvent dimers superposed. Bottom: 10 explicit solvent dimers superposed. The dimers chosen were the top scoring simulation dimers, scored according to C3*MTE.

Differences between structures from micelle and bilayer environments

Further, we noticed that DOCK/PIERR is able to bound dock 17/30 simulation dimers from POPC bilayer membrane (i.e. a model within 1.5 A interface RMSD was found in the top ten models for 17 of 30 dimers) in Gly-out conformation. But the same experiment repeated on the Gly-out dimers in DPC micelle results in no hits in the top ten for any of the 30 dimers from micelle. Again the differences between the two docking accuracies hint at structural differences between dimers in different membrane environments. Differences between membrane protein structures characterized in micelle and bilayer environments have also been observed experimentally[98].

These differences were explored using PIE energy, and as Figure 4.7 shows, the PIE energy for bilayer and micelle simulation models is different. The PIE energy is more favorable for the bilayer models, due to higher number of inter-helical contacts in the dimers in bilayer.

Figure 4.8 shows that the angle between the helices in the simulation dimers is the reason for differences in number of contacts. It is a plot of the absolute value of the cosine of the angle between the helical long axes in the simulation structures. The dimers in bilayer have cosine values closer to 1, indicating that the helices are more parallel in bilayer. In contrast, the helices in micelle have a wider range of angles and favor non-parallel orientations, which are more "X"-like, with one helix making an angle with respect to the other.

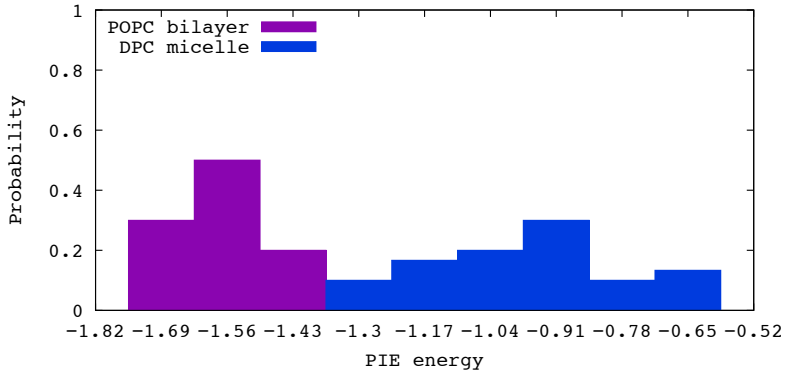


Figure 4.7 Probability distribution of PIE energy for 30 Gly-out explicit solvent dimers in POPC bilayer and and 30 Gly-out explicit solvent dimers in POPC membrane that were bound docked.

This is also illustrated in Figure 4.9, which shows the 30 bilayer models with parallel helices and 30 micelle models with "X"-shaped helical angles. The reason why helices in micelle environment adopt an "X"-shaped orientation maybe related to the entropic effect. A titled configuration allows for more entropy in the micelle than the parallel configuration.

Structures elucidated in micelle environments may differ from those elucidated in membrane environment. Hence this difference raises questions about the applicability of using micelle environments to substitute for membrane bilayers in membrane protein structure determination.

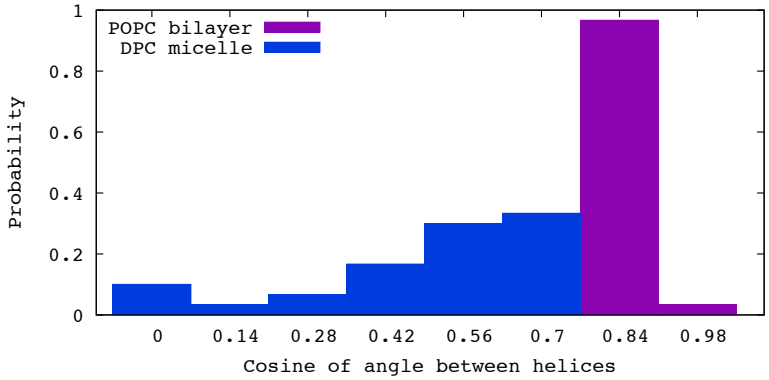


Figure 4.8 Distribution of cosine of angle between helices for 30 Gly-out explicit solvent dimers in POPC bilayer and 30 Gly-out explicit solvent dimers in POPC membrane that were bound docked.

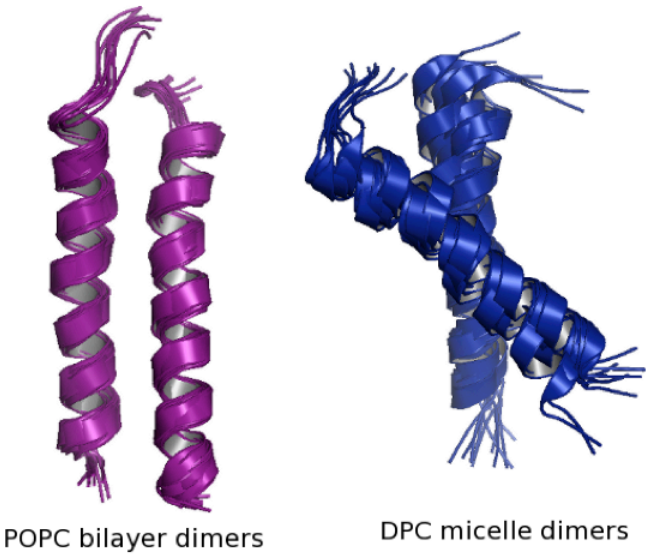


Figure 4.9 Left: Ten explicit solvent dimers from simulations in POPC membrane. Right: Ten explicit solvent dimers from simulations in DPC micelle. The ten models in each case were the top scoring dimers, as scored by C3*MTE.

4.4 CONCLUSIONS

In this chapter, we present the first comparative study of protein docking algorithms for docking *unbound* membrane proteins. It is also the largest comparison, including all comparative studies on bound and unbound membrane protein complexes. We show that including information about the membrane environment as an additional one-body residue-based energy term improves the prediction capacity of our docking algorithm, DOCK/PIERR, significantly. We use this method to study the dimerization of amyloid precursor protein. The results from docking match well with results from implicit solvent simulation. However, explicit solvent structures behave differently: explicit solvent structures have more protein-membrane contacts and implicit solvent structures have more protein-protein contacts. This difference shows that implicit solvent models and our docking procedure are not able to reproduce the contacts formed by discrete solvent molecules. Further, structures characterized in different membrane environments such as bilayer and micelle show significant differences. The dimers in bilayer have parallel helices while the dimers in micelle are more "X"-shaped, with helices oriented at an angle. This preference for "X"-shape can be explained on the basis of entropy i.e. rotational freedom of the dimers in micelle.

Predicting the structure of higher order amyloid aggregates and developing additional potentials trained on membrane protein interfaces represent some of the promising avenues for future work in the area of membrane complex prediction.

Chapter 5. Performance in CAPRI

5.1 INTRODUCTION

In this chapter, we discuss the performance of DOCK/PIERR in community-wide assessment of methods for protein docking. CAPRI (Critical Assessment of Predicted Interactions) [56] is an independent evaluation of current protein docking methods. Before the experimental structure of a newly discovered protein-protein complex is released online, information about its monomers is made available to the protein docking groups, and they submit their predictions of the structure of the complex.

These predictions are then evaluated by an independent assessment team, which decides the quality of the top ten predictions by each team based on criteria like *interface RMSD* (backbone RMSD of the residues in the interface of the reference complex; residues are in the interface if any atom of one residue is within 10 Å of an atom in the other protein), *ligand RMSD* (RMSD between the ligand molecule of predicted model and reference) and *fraction of native contacts* (percentage of residue-residue contacts in the reference structure that are also in the interface of the predicted model). Models are classified as *high-quality*, *medium, acceptable* and *incorrect* based on certain cutoffs of the evaluation metrics[44]. For example, a high-quality model needs to have an interface RMSD less than 1 Å or ligand RMSD less than 1 Å and fraction of native contacts greater than 50%.

There are different categories of participation: i) *server prediction* category, which is an assessment of automated docking methods and has a short prediction deadline of 24 hours, ii) *human prediction category*, which is a prediction competition with a longer deadline of 1-2 weeks allowing for manual correction of automated docking results using available literature information and iii) *scoring category*, where instead of

prediction, teams need to simply score the available models. There are generally about 1-3 rounds per year.

5.2 OVERALL PERFORMANCE OF DOCK/PIERR IN CAPRI

Table 5.1 shows the performance of DOCK/PIERR listed by target [41]. Only targets we participated in are shown. Hits refer to models that are of acceptable quality or better. "–" denotes no participation for that target, while 0 indicates no acceptable models were found for that target. Overall, we predicted a hit in the top 10 successfully for 4 out of 8 targets in the server category and 6 out of 9 targets in the scoring category. This is consistent with our results on the training set and benchmarks.

Table 5.1 Overall performance of DOCK/PIERR in CAPRI assessments.

Target	Total number of predictor groups	Number of predictor groups with hits	Total number of scorer groups	Number of scorer groups with hits	Server: number of top 10 hits	Scorer: number of top 10 hits
40	38	30	15	10	-	8
41	33	26	13	12	-	1
46	40	2	16	8	0	1
48	32	15	-	-	1	-
49	33	15	13	8	1	0
50	40	18	17	12	3	2
51	46	3	13	5	0	0
53	42	20	13	11	0	5
54	41	4	13	0	0	0
59	40	12	24	8	2	1

Table 5.2 shows the rank of DOCK/PIERR server and its earlier version, DOCK/PIE, in comparison with other automated servers. Rank of a server was determined based on both model quality and number of models. A server that submits high-quality/medium models is ranked higher than a server that submits acceptable

models. For servers that submit the same quality of models (e.g. acceptable), the number of acceptable models is chosen to determine the rank. In cases where we submitted all 10 incorrect predictions, the rank is not shown.

Table 5.2 Rank of DOCK/PIERR server per target

Target we participated in, as server	Number of acceptable or better models submitted by the server	Rank in server category	Number of servers participating for this target	Number of servers that submitted acceptable or better models for this target
T46	0	-	8	1
T48	1	2	6	2
T49	1	3	6	3
T50	3	1	6	2
T51	0	-	3	0
T53	0	-	8	4
T54	0	-	8	1
T59	2	2	8	3

Table 5.2 shows that for the four out of eight targets for which we submitted a correct model, the rank of the server was within the top three servers. Based on the above performance, DOCK/PIERR was ranked as the fourth most successful docking method in the automated server category of the CAPRI assessment of 2013 [47] .

5.3 PERFORMANCE BY TARGET

A target-wise discussion is presented in this section. The targets discussed are the ones that the author participated in. For a discussion of the performance of early versions of DOCK/PIERR on previous targets, refer [2].

T50

This complex, PDB 3R2X, is a protein interaction designed by the Baker lab, and is the structure formed by a hypothetical (designed) protein bound to the HA1 and HA2 domains of hemagglutinin in the influenza A virus [109]. The structure of hemagglutinin was provided. The sequence of the designed protein was provided along with a template structure. The structure of the designed protein was obtained by homology modeling using [59] . DOCK/PIERR obtained 3 acceptable or better hits in the server prediction round, including two medium hits and 2 hits in the scoring round. Figure 5.1 shows a medium quality model with interface RMSD of 1.487 Å superposed with the crystal structure of T59.

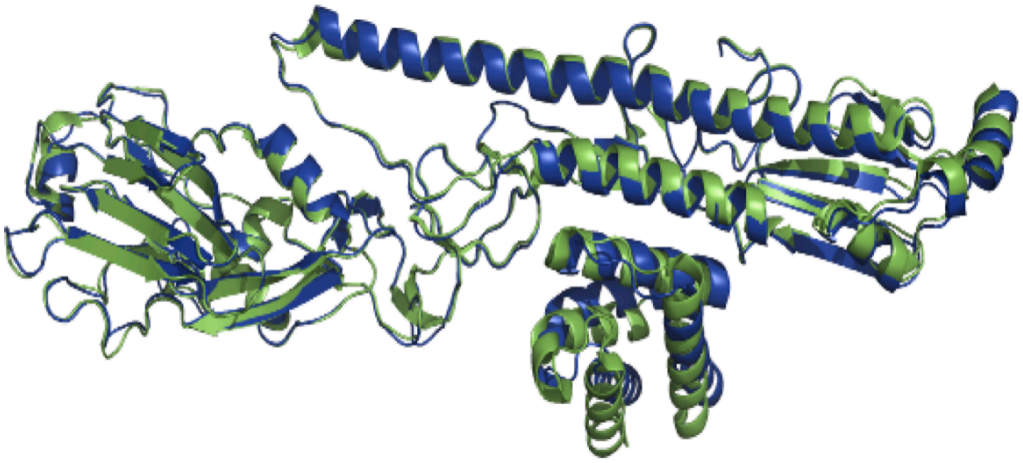


Figure 5.1 DOCK/PIERR medium-quality prediction (in blue) superposed with the crystal structure of T50 (in green).

T51

Target T51, PDB 4BXG, was a multi-domain target that involved assembly of the penta-modular cellulosomal arabinoxylanase structure [110]. The five domains that needed to be assembled were: GH5-CBM6-CBM13-Fn3-CBM62. An unpublished crystal structure was provided for GH5-CBM6, CMB13 was to be homology modeled, Fn3 had a

separate crystal structure deposited and CBM62 was free in the complex and could be ignored. This was solved by the assembly algorithm outlined in [2]. Three separate interface assessments were performed: between CBM13 and Fn3, between GH5-CBM6 and CBM13 and between and GH5-CBM6 and CBM13-Fn3. We did not obtain any hit in the scoring or prediction rounds. This was a hard target and only 3 of 35 prediction groups and 5 of 13 scorer groups got an acceptable model for this target.

T53

T53, PDB 4JW2, was a protein-protein complex between artificial alpha repeat proteins REP4 and REP2[111]. The structure of REP4 was available while that of REP2 needed to be modeled. DOCK/PIERR scoring produced 5 hits for this target while no correct predictions were made in the prediction round. Figure 5.2 shows one of the successful scoring predictions with interface RMSD of 1.21 Å from the crystal structure.

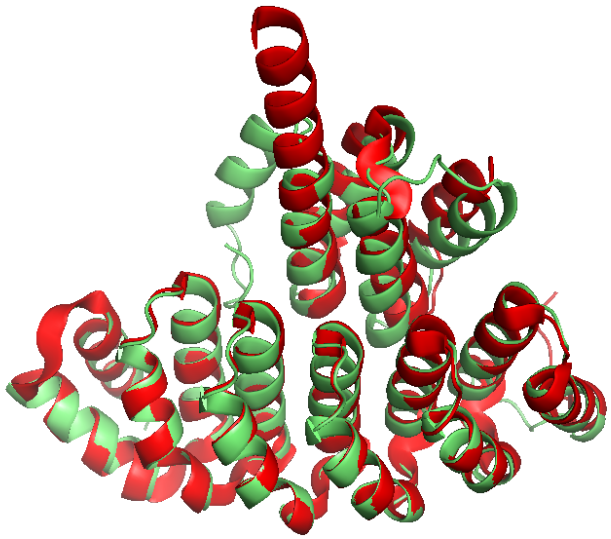


Figure 5.2 DOCK/PIERR medium-quality prediction (in green) superposed with the crystal structure of T53 (in red).

T54

T54, PDB 4JW3, was a complex between engineered neocarzinostatin and another alpha repeat protein, REP16 [111]. The structure of neocarzinostatin was available while the structure of REP16 was modeled from the sequence. We did not produce any hits in the prediction or scoring round here. This target was also found to be a hard target by other groups, since no group was able to get an acceptable or better model in the scoring round for T54, and only 4 out of 41 groups got an acceptable model in the prediction round.

T59

T59 was a complex between the EDC3 antibody domain (PDB 4A53) and RPS28B, an RNA decapping protein, whose sequence was provided. DOCK/PIERR successfully predicted 2 models in the server round and one model in the scoring round. A figure of the successful models is not provided, as the crystal structure coordinates are as yet unpublished.

5.4 DOCK/PIERR SERVER AND EXECUTABLES

Source code and Linux executables of the scoring functions developed in this thesis are found at http://clsb.ices.utexas.edu/web/dock_details.html. These scoring functions were recognized as some of the best scoring functions by researchers in the community [112]. All the methods described in this thesis are implemented in the DOCK/PIERR server [113] at http://clsb.ices.utexas.edu/web/dock.html. As of the time of this writing (March 2014) the server has 50+ users and 200+ submitted docking jobs. Apart from the application studies in this thesis, it has been used by a few others in their studies. DOCK/PIERR was used to suggest oligomeric conformations of a four-domain

orange-fluorescent protein (Ember) [114]. It was also used as one of the docking methods for predicting the complex between cytoplasmic dynein and pilin [115], to explore pathogenesis caused by bacterium pseudomonas aeruginosa.

The advances made in DOCK/PIERR help establish automated docking methods as accurate methods for structure prediction and enables departure from previous methods that rely more on human intuition. With more and more protein sequences and monomer structures being made available, automated docking methods such as DOCK/PIERR are slated to play an important role in large-scale prediction of complexes in the proteome.

Chapter 6. Algorithms for Network Analysis of Milestoning Data

6.1 BACKGROUND

Networks in Molecular Biology

Network analysis is becoming increasingly popular in computational molecular biology. For example, proteins interact with tens of other proteins during their lifetime to carry out their function. This web of interactions is represented by a protein-protein interaction network [3, 4]. Gene expression networks, analogous to protein interaction networks, provide insights into co-expression of genes. Other types of networks include gene regulatory networks, signaling networks and metabolic networks. The networks in molecular biology are massive and can be composed of millions of nodes and edges. They clearly require sophisticated computational tools to analyze them.

Networks from Molecular Dynamics Simulations

In this chapter, we discuss algorithms [116] for analyzing networks of molecular data gathered from molecular dynamics (MD) simulations. Molecular dynamics is a sampling technique where the time evolution of phase space points (space of coordinates and velocities) of the system is explored by solving Newton's laws of motion at each step. This sampling produces trajectories from an initial state (e.g. unfolded state of a protein) to a final state (e.g. folded state). Network analysis helps in mapping the continuous phase space trajectories from MD simulations, into a relatively small number of discrete states; this is useful in visualization of the data and in dissecting complex dynamics to concrete mechanisms. However, molecular networks from MD are getting increasingly complex, due to the growth in computer power that allows us to generate longer trajectories for larger systems. This increased complexity of the resultant networks makes simple interpretation and qualitative insight of the molecular systems more

difficult to achieve, necessitating the use of efficient and scalable algorithms for network analysis.

Milestoning

The algorithms discussed in this chapter are applied to data from the advanced MD sampling technique of Milestoning. Advanced MD techniques like Directional Milestoning[51], Markov State Models (MSM) [117-119], and Transition Path Theory (TPT) [120] are used to study the kinetics (mechanism and rates) of a long time scale cellular event in atomic detail, like the process of unfolding of a protein, or the binding of a small molecule to a protein. Long-time scale biological events (which take hours in real time) cannot be computed using straightforward MD simulations. Milestoning[49-51] is a theory and algorithm in which the overall trajectory of long time-scale events can be studied in a computationally efficient manner by breaking them down into shorter trajectories that can be run independently, in parallel, and then combined to get the overall chemistry (kinetics and thermodynamics). The parallel nature of the algorithm allows for efficient computation of long time-scale events even for large systems[121-123].

In Milestoning, the phase space (set of positions and momenta of the system) is divided into a set of anchors, or phase space points $\{X_\alpha\}_{\alpha=1}^{N}$, which provide coarse coverage of the phase space[121]. Milestones are then defined as interfaces, $\left\{I_j\right\}_{j=1}^{J}$, separating phase space volumes that are associated with the anchors, as in Figure 6.1. Henceforth, we use the indices $\{\alpha,\beta,\gamma..\}$ to denote the anchors and indices $\{i,j,k...\}$ to denote milestones.

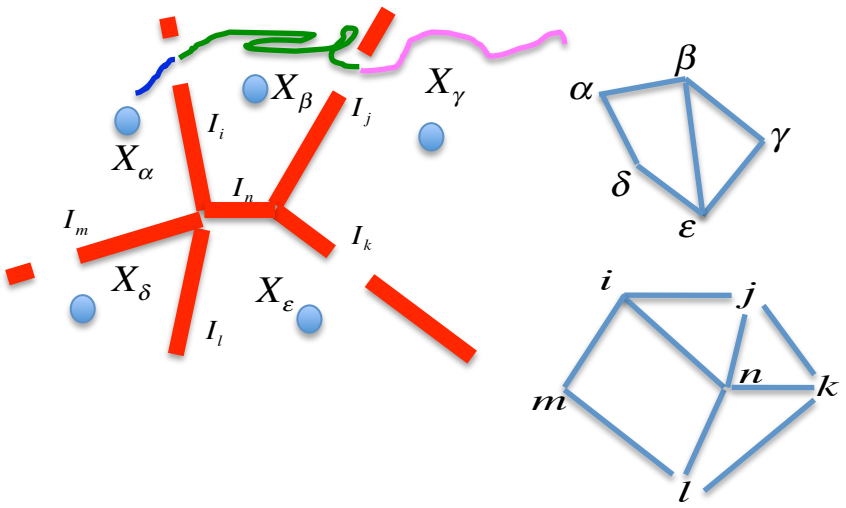


Figure 6.1 A schematic representation showing the mapping of continuous space and MD trajectories to a network.

The milestone, I_i between anchors X_α and X_β is a hyperplane, Y, in a coarse-grained space given by Eq. (6.1).

$$I_i(\alpha \to \beta) = \left\{ Y \mid d(Y,Y_\beta)^2 = d(Y,Y_\alpha)^2 + \Delta^2 \;\&\; \forall k d(Y,Y_\beta) \le d(Y,Y_k) \right\} \tag{6.1}$$

The flux at milestone I_i (the number of molecules that pass per unit time the i-th milestone) is denoted by q_i. The basic Milestoning equation[49] is of conservation of flux,

$$q_i(t) = 2 \cdot \eta_i \delta(t) + \sum_j \int_0^t K_{ij}(t-t') q_j(t') dt' \quad \forall i \tag{6.2}$$

where $q_i(t)$ is the flux through milestone I_i at time t, η_i is initial condition (the probability that the last milestone that passed before or at time zero is I_i), $K_{ij}(t)$ is the transition probability that a trajectory that starts at milestone I_i will pass through milestone I_j exactly after time t. Hence Eq. (6.2) keeps track of the number of trajectories and ensures that the flux is conserved.

For network calculations it is convenient to consider a stationary flux or steady state condition in which the flux, q_i, is time independent. The stationary matrix is K. It is

the time integrated transition matrix $\left(K_{ij} \equiv \int_0^\infty K_{ij}(t)dt \right)$ which gives the probability that a trajectory initiated at milestone I_i will hit (and terminate at) another milestone I_j before any other milestone. We obtain a stationary flux by setting cyclic boundary conditions. The final milestone *f* is set to return all the flux that arrives to it, to the first milestone. Hence the matrix element K_{fi} is set to one if milestone I_i is the first milestone and is set to zero otherwise. The above adjustment of *K* and the requirement of stationary flux / steady state results in a remarkably simple equation for the stationary flux [48].

$$q(Id - K) = 0 \qquad K_{fi} = \left\{ \begin{array}{ll} 1 & i = 1 \\ 0 & i \neq 1 \end{array} \right\} \tag{6.3}$$

where q is the vector of all stationary fluxes. Id is the identity matrix. As discussed extensively in earlier papers about Milestoning[49, 124] K is computed from atomically detailed trajectories as $K \approx n_{ij}/n_i$ where n_i is the number of trajectories initiated at milestone I_i and n_{ij} is the number of trajectories that started at I_i and the first milestone they reach (which is different from I_i) is milestone I_j. The length of the vector q is J and the dimensionality of K is JxJ, where J is the number of milestones.

In short, the only quantities needed to be estimated from short trajectories are the elements of the transition kernel, *K*. The flux can be derived from it by using Eq. (6.3). Trajectories between milestones are assumed independent, which allows us to calculate the trajectories in parallel. Moreover, because milestones are close to each other, the time scale of trajectories between two milestones is much smaller than the overall time scale of the biological process [121].

Contributions of this chapter

The questions we seek to answer are the following: “In what ways can we represent Milestoning data in terms of networks?”, “What are the important edges and

paths in these networks and how do we find them?" and "Are there bottlenecks in the networks?". Bottlenecks are edges of low flux in the network, that require significant efforts to pass through, and they must be crossed on the way from the initial (reactant) to the final (product) state. The information about pathways and bottlenecks is useful for qualitative analysis of the process and to gain more insight into the behavior of the system.

We propose Global Maximum Weight Pathways as a useful tool for analyzing molecular mechanism in Milestoning networks. A closely related definition was made in the context of Transition Path Theory [120]. We consider three algorithms to find these pathways: Recursive Dijkstra's, Edge-Elimination, and Edge-List Bisection. The asymptotic efficiency of the algorithms is analyzed and numerical tests show that Edge-List Bisection and Recursive Dijkstra's algorithms are most efficient for sparse and dense networks respectively. Pathways are illustrated for two examples: helix unfolding and membrane permeation. Finally, we illustrate that networks based on local kinetic information can lead to incorrect interpretation of molecular mechanisms.

6.2 NETWORK REPRESENTATION

The molecular process is represented as a weighted, directed graph $G=(V,E)$, where V is the set of vertices and E is the set of edges in G. An edge from vertex u to vertex v is represented as (u,v) and has a weight, $w(u,v)$. Note that the edges are directed, i.e. edge (u,v) is not the same as (v,u). The edge weights may have different physical realizations. For example, edge weights and states may be defined by the physical distance between two vertices (as is done by geometric clustering), the phase space flux between nodes, or the rate constant of transitions between the nodes.

The representation of directional milestoning data using networks has been previously described in[48, 51]. As Figure 6.1 shows, two types of network representation are possible for Milestoning data. One type of network is where metastable (stable intermediate) states are identified and are mapped to vertices (nodes). Transitions between the states are modeled by edges between the vertices. We have five cells in phase space denoted by $X_\alpha, X_\beta, X_\gamma, X_\delta, X_\varepsilon$. Each cell can be mapped to a network vertex and the edges would be between vertices, e.g. (α,β) and (β,γ). Sometimes the cells are represented by specific conformations (anchors) that are illustrated in the figure by the blue ellipses.

In an alternate network representation, the vertices can be interfaces or milestones denoted on the figure by dashed red lines, which indicate the boundary between domains. There are six milestones in the above figure, $I_i - I_n$. Continuous trajectories are mapped to the network either by their location in phase space, or by the last milestone that they have passed (color coded curves in the figure). On the right side of the figure we show network representations. Top figure is an anchor-based network and the lower figure is based on the milestones.

The dual representation, by anchors and milestones, makes it possible to visualize more than one network for the same process. Depending on the choice of nodes, as (i) anchors or (ii) milestones, we have two types of networks: (i) state-space network where the nodes are anchors or phase space volumes, and (ii) a flux-space network, where the nodes are milestones. There are more milestones than anchors and hence the picture obtained by the flux-space graph is more detailed and potentially includes more information than the state-space graph. But the state-space graph is simpler, and for interpretation purposes it can be beneficial to look at the system at the anchor level. We therefore convert the flux-space paths to state-space for visualization purposes.

For choice of edge weights between the nodes, we choose the flux, as flux between two nodes is the most informative quantity in Milestoning, that we can attach to an edge, as is done in Transition Path Theory[125] and the max flux formulations of optimal pathways[48, 126-128]. Below, we discuss how edge weights are obtained from fluxes for both state-space and flux-space graphs. We also discuss another graph representation based on rate coefficients instead of fluxes. For the flux-space graphs, we additionally explain how to convert the paths to state-space.

State-space (anchor-based) graphs with flux-based edge weights

We create a graph with one vertex per anchor. Consider two anchors α and β, which are associated in directional milestoning, with two fluxes, $q_{\alpha\beta}$ and $q_{\beta\alpha}$ corresponding to the interfaces (milestones) $\alpha \to \beta$ and $\beta \to \alpha$. The weight of the edge is the net flux $w(\alpha,\beta) = \left| q_{\alpha\beta} - q_{\beta\alpha} \right|$. The direction of the edge is decided according to the larger flux. Hence, if $q_{\alpha\beta} > q_{\beta\alpha}$ the direction of the edge is from α to β and vice versa. The main advantage of using graphs in anchor space, apart from the ease of interpretation, is that the size of graphs is smaller and hence calculating pathways is less expensive than the flux-space graphs. In directional milestoning for instance, the number of nodes i.e. milestones of a flux-space graph, *J*, is much larger than the number of anchors *N*, and *J* can be as large as $N(N-1)$. However anchor space graphs are more likely to be dense graphs.

Flux-space (milestone-based) graphs with flux-based edge weights

We have one vertex per milestone in this graph. The probability matrix K_{ij} sampled in Milestoning, determines the presence of edges between milestones [116]. An edge from milestone *i* to milestone *j* exists if the corresponding matrix entry is positive, ($K_{ij} > 0$). But determining edge weights is not obvious from first sight since the flux

information is for individual milestones, while the edge weights represent information between two connecting milestones. The following simple transformation converts vertex-based (milestone-based) weights to edge weights between pairs of milestones. For milestone pair i and j, the edge weight for edge (i,j) is $w(i,j)=|q_j|$ i.e. the flux associated with the second milestone.

On the path from start state s to end state t, the only milestone on the path whose flux we do not encounter as an edge weight on the path, is the starting milestone, since we consider only the flux of the latter milestone, j, for every edge (i,j). This is fixed by adding an extra (dummy) milestone, s' before the first vertex, with an edge from s' to s whose weight is $w(s',s)=q_s$ i.e. weight of the edge is equal to flux of the starting milestone. The pathway calculations are then performed from s' to t instead of s to t.

Note that, for a fixed milestone j, all the edges leading to milestone j in such a graph will have the same weight. In other words, $w(i,j)=|q_j| \quad \forall i,s.t.K_{ij}>0$. Hence many edges have the same weight (same flux in this case) and this can result in degenerate paths.

For visualization, we convert the resulting paths from milestone space to anchor space. For every milestone i in the milestone-based path, associated with anchors α and β, we add to the anchor-based path, an edge (α,β) between anchors α and β, with edge weight $w(\alpha,\beta)=q_i$ i.e. edge weight is the flux associated with the corresponding milestone. Note that adjacent milestones always share an anchor. For example, path $\langle i,j,k\rangle$ in milestoning space corresponds to path $\langle\alpha,\beta,\gamma,\delta\rangle$ in anchor space, assuming milestone i corresponds to anchor pair (α,β), milestone j corresponds to anchor pair (β,γ) and milestone k corresponds to (γ,δ).

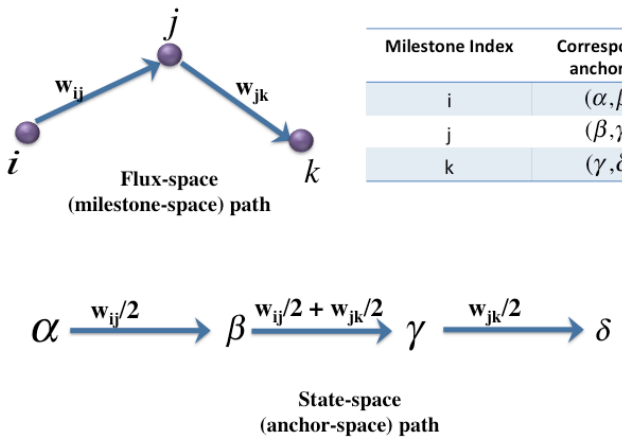
Flux-space (milestone-based) graph based on rate coefficients

An easier alternative to the flux based approach for getting edge weights is to weigh the edges of the graph with rate coefficients or energy barriers[129, 130]. This weighting is local and does not take into account global topology and local information can be misleading and point to less relevant portions of the graph. For example, using rate coefficients, it is possible to weigh some edges highly if they have a fast local transition. But at the same time, these edges may be off the main pathway receiving little reactive flux.

The rate coefficients of a Master equation between milestones can be computed directly using the Milestoning transition matrix *K* and the vector τ, the average lifetimes of the milestones [50, 131]. The rate coefficient for a transition between a milestone pair (i, j) (and the edge weight) is given by

$$w(i,j) = \frac{K_{ij}}{\tau_i} \tag{6.4}$$

Converting paths based on rate coefficients in flux-space (milestone-space) to state-space (anchor-space) is performed as follows. For every milestone *i* (a milestone between anchors α and β) on the path in flux-space, we add an edge in state-space between the anchors α and β. Each pair of milestones (i, j) is associated with three anchors, (α,β,γ) with milestone *i* associated with anchor pair (α,β) and milestone *j* associated with anchor pair (β,γ). Hence edge weight between a pair of milestones (i, j) in the path in milestone space is shared equally between anchor-based edges (α,β) and (β,γ). Edge weights from flux-space to anchor-space are converted as shown in Figure 6.2. Note that each milestone edge contributes to weights on two anchor edges and each anchor edge can get a contribution to its weight from two milestone edges (except the edges at the ends of the path).



Milestone Index	Corresponding anchor pair
i	(α,β)
j	(β,γ)
k	(γ,δ)

Figure 6.2 Conversion of a flux-space path with milestones as vertices, to a state-space path with the corresponding anchors as vertices. The table in the figure shows the mapping from milestone index to anchor index.

6.3 Definition of Pathways

Maximum Weight Path (MWP)

Given a start vertex, *s*, and end vertex, *t*, we seek a path between *s* and *t* in which has the maximum possible weight (of all paths between s and *t*) for the minimum weight edge on the path. Each of the paths from *s* to *t* paths has a bottleneck, that is the edge with the minimum weight (EMW) along the path. The *s*-*t* path with an EMW, which is larger than the EMW of all other *s*-*t* paths, is the maximum weight path between *s* and *t*. This has also been referred to as a dominant reaction pathway in Transition Path Theory[120]. The edge weights in the graph can also be referred to as capacities, and the maximum weight path is known as the maximum capacity path [132-135].

An example graph is illustrated in Figure 6.3 (a), which displays start and end vertices A and D respectively, and capacities (edge weights) marked along the edges. A path from vertex A to vertex D, passing through vertices B and C is written as $\langle A,B,C,D\rangle$. There are multiple paths between A and D, $\langle A,B,D\rangle$, $\langle A,C,D\rangle$ and

$\langle A,B,D\rangle$. Of these three paths, the maximum weight paths are $\langle A,B,D\rangle$ and $\langle A,B,D\rangle$, shown in green in Figure 6.3 (b), since the edge with minimum weight (EMW) on both these paths is the highest possible for an A to D path, and equal to 8, which is greater than 5, the minimum weight edge on path $\langle A,C,D\rangle$.

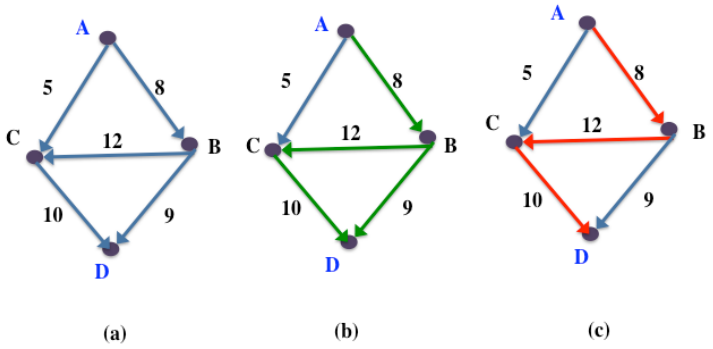


Figure 6.3 (a) An example graph with multiple paths between vertices of interest, A and D. (b) Maximum weight paths (MWP) between A and D shown in green. (c) Global maximum weight path (GMWP) between A and D shown in red.

Global Maximum Weight Path (GMWP)

The definition of maximum weight path stated above relies on just one edge in the path, i.e. the EMW. More than one path can share the same EMW as shown in the above example. In order to have a unique solution to the path determination problem, we define the global maximum weight path (GMWP), which is an optimal maximum weight path that is as close as possible to being unique. The global maximum weight path is referred to as the representative dominant reaction pathway in Transition Path Theory[120].

Let a path, *m*, be a maximum weight path between *s* and *t*. If for every pair of vertices on *m*, the subpath on *m* between those vertices is a maximum weight path, then *m* is a global maximum weight path (GMWP). The GMWP for a given pair of vertices is unique up to the degeneracy of paths branching from the same vertex in the graph. GMWP is analogous to a minimum energy path in continuous space, and the EMW is

analogous to a transition state. In previous studies, we defined a discrete version of the max flux path for a network as a GMWP[48, 136, 137].

In the example shown in Figure 6.3, both $\langle A,B,C,D\rangle$ and $\langle A,B,D\rangle$ are maximum weight paths, with the same EMW, (A,B). But the maximum weight path between B and D is $\langle B,C,D\rangle$ with minimum edge weight 10, and not $\langle B,D\rangle$, which has a minimum edge weight 9. Hence the global maximum weight path between A and D, shown in red in Figure 6.3 (c), is $\langle A,B,C,D\rangle$ since all its subpaths are also maximum weight paths.

More formally we define $W(s,t,p)$, weight of a path, p, from vertex s to t, as

$$W(s,t,p) = \min_{(u,v)\in p} w(u,v) \tag{6.5}$$

In Eq. (6.5), (u,v) represents an edge from vertex u to vertex v, and $w(u,v)$ is the weight or capacity of the edge (u,v). Eq. (6.5) states that the weight of a path p is equal to the weight of the edge with minimum weight (EMW) on the path. We define a path μ to be a maximum weight path between vertices s and t if μ satisfies Eq. (6.6).

$$W(s,t,\mu) \geq W(s,t,p) \quad \forall p \tag{6.6}$$

That is, the weight of path μ, from vertex s to t, is greater than the weight of all other paths p from s to t. Or, the EMW on path μ has a higher weight than the EMW of all other paths p. We also represent the EMW of the maximum weight path, μ, between s and t as $M(s,t) \equiv W(s,t,\mu)$.

We then define m as a GMWP from s to t, $m = \langle s,v_1,v_2....v_i,v_j..t\rangle$, if it satisfies

$$W(v_i,v_j,m) \geq W(v_i,v_j,p) \quad \forall p \quad \forall v_i,v_j \in m,\ i<j \tag{6.7}$$

Eq. (6.7) states that, for any two vertices, v_i and v_j on the path m, with v_i appearing before v_j on the path, the path between v_i and v_j that has the maximum weight, among all paths p from v_i to v_j is exactly the path through m. We now develop the algorithms for obtaining the MWP and GMWP.

6.4 DETERMINATION OF MAXIMUM WEIGHT AND GLOBAL MAXIMUM WEIGHT PATHS

Recursive Dijkstra's Algorithm for Global Maximum Weight Path

a. Modification to Dijkstra's Shortest Path Algorithm for Calculating the Maximum Weight Path

Dijkstra's algorithm[138] provides the base for efficient calculation of the GMWP using the Recursive Dijkstra's algorithm. Dijkstra's single-source shortest path algorithm finds the shortest paths and shortest path lengths from a single vertex of interest, s, to all other vertices, in a graph G, where a non-negative weight of an edge representing distance, $d(u,v)$, is associated with each edge (u,v). The length of the path, L, is determined by the sum of the edge distances.

Dijkstra's shortest path algorithm can be easily modified to obtain an algorithm to find a maximum weight path from a given vertex s to all other vertices. The two key points in the modification are that first, the minimization problem (shortest path length) in the previous case is converted to a maximization problem (maximum EMW or maximum capacity). Second, instead of using the length metric as the sum of distances in a path, we use the metric of the weight of the EMW along the path.

The algorithm for maximum weight path calculation finds at each step, the vertex, u, with the maximum weight (or maximum EMW) from s and updates the maximum weights of the vertices neighboring u. In other words, suppose we know the maximum weight of u, and say u is connected to v through edge (u,v). We can then update the maximum weight from s to v, if the weight of the path to v passing through u is higher than the current estimate of the maximum weight from s to v.

We arrive at the equality in Eq. (6.8) for each vertex v adjacent to vertex u, where $M(u)$ and $M(v)$ represent the current known maximum weight from the source vertex to u and v respectively, and $w(u,v)$ is the weight of the u-v edge. This is a slight

modification of the equality in the shortest path algorithm where the sum in the inner bracket is changed to a minimum of two edges and the min condition in the outer bracket is changed to max condition.

$$M(v) = \max\left(M(v), \min\left(M(u), w(u,v)\right)\right) \tag{6.8}$$

The algorithm to calculate maximum weight paths from a given source vertex *s* to all other vertices in a directed graph *G* is outlined in Table 6.1. The variable *M* keeps track of the weight of the bottleneck edge or EMW, on the maximum weight path from *s* to a particular vertex. The array *Q* is the priority queue in the shortest path algorithm which enables efficient extraction of the vertex with maximum weight at each step. The data structure *Adj* is an adjacency list representation of the graph. The EXTRACT_MAX operation extracts the current (unprocessed) vertex with the maximum weight from *s*.

An extra array called bottleneck is used here to store the actual vertices corresponding to the EMW (bottleneck edge) in the maximum weight path for a given vertex. This data structure is not required for calculating maximum weight paths, but is required later on, when we use this maximum weight path algorithm to calculate the global maximum weight path.

When the algorithm terminates, the maximum weight among all paths from *s* to a particular vertex, *i*, is retained in array element M[i] and the EMW for a particular vertex is in array bottleneck[i] (line 17). The proof of this algorithm is exactly analogous to Dijkstra's shortest path algorithm.

Table 6.1 Algorithm 1 - Modified Dijkstra's algorithm for finding maximum weights and bottleneck (EMW) edges from *s* to all other vertices in a graph *G*.

```
procedure MaxWeightPath(G,s)
for each v in G                                                    1
        M(v) = -1                                                  2
                                                                   3
M(s) = ∞                                                           4
Q = V     // Add all the vertices in G                             5
                                                                   6
while Q ≠ NULL                                                     7
        u = EXTRACT_MAX(Q)                                         8
        for each v in Adj(u)                                       9
                if M(v) < min(M(u),w(u,v))        // M(v)=max      10
(M(v),min(M(u),w(u,v)))
                        M(v) = min(M(u),w(u,v))                    11
                                                                   12
                        if M(u) < w(u,v)                           13
                                bottleneck(v) = bottleneck(u)      14
                        Else                                       15
                                bottleneck(v) = (u,v)              16
return bottleneck,M                                                17
```

The efficiency of the algorithm is the same as that of the shortest path algorithm. For a graph with V vertices and E edges, the best-known theoretical complexity of this algorithm is $O(V \log V + E)$, using Fibonacci heaps for efficiently extracting the next vertex with the smallest distance from s in $O(\log V)$ time, and adjacency lists for efficiently finding the neighbors of a vertex in $O(E)$ time across all vertices. For sparse graphs (i.e. $V \approx E$) the time complexity becomes $O(V \log V)$. For dense graphs, (where $E \approx V^2$), the time complexity is $O(V^2)$. For a simpler implementation of graphs with priority queue implemented using arrays and graphs implemented as adjacency matrices, the complexity is again $O(V^2)$ for this algorithm.

The maximum weight path is a path from the start to end state containing the transition edge, EMW, which is similar to the transition state of chemical reactions. The EMW is a good descriptor for processes dominated by a single and large free energy

barrier, in which case, the location of the transition edge is much more critical than the rest of the GMWP, and the algorithm outlined above can be used to compute this path efficiently. However, when the EMW is not dramatically lower in weight compared to other weights along the path, the location of the entire pathway matters, which brings us next to the calculation of Global Minimum Weight Path (GMWP).

b. Recursive Dijkstra's Algorithm for GMWP Calculation

We note that the GMWP is a special maximum weight path between *s* and *t*. It is a path where all subpaths between pairs of vertices on the same path are maximum weight paths. We now introduce a new algorithm, the Recursive Dijkstra's algorithm, that uses the maximum weight path algorithm (Algorithm 1) repeatedly to calculate the global maximum weight path. Given a pair of vertices *s* and *t*, we first use Algorithm 1, the maximum weight path algorithm, to get the EMW (u,v) between *s* and *t*. Note that Algorithm 1 returns the EMW from *s* to all other vertices, but we only need that piece of information for vertex *t*. Since $w(u,v)$ is the maximum weight that can pass between *s* and *t*, (u,v) is an edge common to all maximum weight paths between *s* and *t* and hence it exists also in the GMWP between *s* and *t*. We then have two subpaths to be determined in the GMWP, $p_1 = \langle s..u \rangle$ and $p_2 = \langle v..t \rangle$. We use the above technique recursively to find the EMW (bottleneck edge) edge between *s* and *u*, and between *v* and *t*. We note that once an EMW (u,v) is known, between *s* and *t*, the remaining subpaths p_1 and p_2 of the GMWP can be computed independently, since the edges on the subpaths will always be of higher weight than the EMW.

Thus each call to Algorithm 1 provides us with one edge on the GMWP. Once all subpaths are uniquely determined, we have the complete GMWP between *s* and *t*. Given vertices *s* and *t*, Algorithm 2 in Table 6.2 finds the global maximum weight path between them in a directed graph G.

There can be multiple maximum weight paths, all of them having the same EMW, but the GMWP is defined to be unique up to the possible accidental degeneracy of edge weights of alternate paths. If there are degenerate edges in the graph, there can be more than one GMWP, and hence it is recommended to compute the first path, remove the bottleneck edge and recompute the path, repeating this procedure till no more unique paths are found. This process guarantees that we get a complete picture of the reaction pathways.

Table 6.2 Algorithm 2 – Recursive Dijkstra algorithm to find the global maximum weight path between vertices *s* and *t*, in a directed graph, based on the modified Dijkstra algorithm for maximum weight paths.

procedure GlobalMaxWeightPath(G,s,t)	
// base case, return empty path	1
if s = t	2
return <>	3
	4
// call algorithm 2 to find bottleneck edge	5
(bottleneck,M) = MaxWeightPath(G,s)	6
(u,v) = bottleneck(t)	7
	8
// find subpaths by recursion	9
p_1 = GlobalMaxWeightPath(G,s,u)	10
p_2 = GlobalMaxWeightPath(G,v,t)	11
	12
// concatenate the subpaths	13
return $\langle p_1,(u,v),p_2 \rangle$	14

Each call to Algorithm 2 fixes one edge on the GMWP. With *V* vertices and *E* edges in the graph, the maximum length of a GMWP is of the order of V, so Algorithm 1 is called a maximum of *V* times from Algorithm 2. Note that Algorithm 1 itself takes $O(V \log V)$ for sparse graphs (with priority queue implemented using Fibonacci heaps and graphs implemented as adjacency lists) and $O(V^2)$ for dense graphs and for simple

implementations of sparse graphs. Hence Algorithm 1 takes $O(V^2 \log V)$ time for sparse graphs and $O(V^3)$ for dense graphs and for simple implementations of sparse graphs.

We note that we are doing some extra computations in Algorithm 2 that can be avoided. For example, we first call Algorithm 1 on the source vertex, *s* to get the EMW (bottleneck) of the destination vertex *t* from *s*. Then while computing the subpath from *s* to *u*, in the recursive step, we again call Algorithm 1 on *s* to get the EMW of *u* from *s*. But, Algorithm 1 calculates the EMWs for all vertices from *s*, and not just for one particular destination vertex, *t*. Hence we can just run Algorithm 1 once on each vertex, and store the EMWs of all other vertices from this vertex. This can be done by making bottleneck a 2D array i.e, bottleneck(i,j) will give the EMW for the maximum weight path from vertex *i* to vertex *j*. Each time we need the EMW from Algorithm 1 between two vertices *i* and *j*, we check whether Algorithm 1 has been already computed on vertex I, and only run Algorithm 1 when it has not been run on *i*.

The optimized procedure does not improve our bounds on the asymptotic time complexities outlined in the Efficiency section. In the worst case, the EMW between two vertices is always the first edge on the path between the two, in which case Algorithm 2 needs to be run on every vertex in the GMWP and optimization cannot be performed. Nevertheless, the optimization improves the runtime in the average case, and is useful in practice.

Comparison to Edge-Elimination based MaxFlux Algorithm

Previously, an approximate algorithm has been described for finding maximum flux path in the context of Directional Milestoning in [48]. Here we call it the "Edge-Elimination" Maxflux algorithm. The steps in the algorithm are:

1. Sort all the edges in the graph G based on their weight, into a list, L_w.
2. Initialize path *p*, between vertices *s* and *t* to an empty path. *p* on exit will be the GMWP.
3. While the vertices *s* and *t* are not connected in *p*, repeat the following steps.
4. Proceed to the next edge, (u,v) in L_w with the smallest weight.
5. Check if removal of (u,v) from *G* disconnects *s* from *t*.
6. If it does, then this is an edge that is critical to the GMWP, and hence it is added to *p*.
7. If not, then simply remove this edge from *G*, and proceed to the next edge in L_w.

Given a graph with E edges and V vertices, the time for sorting the edges is $O(E\log E)$. Checking if two vertices are connected in a graph can be done efficiently using graph traversal algorithms like breadth first search or depth first search [7], which take $O(V+E)$ if adjacency lists are used, or $O(V^2)$ if adjacency matrices are used to represent the graph. The maximum number of iterations we need is E (one per edge), so the time complexity becomes $O(E\log E+EV^2)$ when using a matrix representation of the graph and $O(E\log E+E(V+E))$ when using the adjacency list representation.

For dense graphs, where $E\approx V^2$, both the matrix and list representations yield a complexity of $O(V^4)$, whereas for sparse graphs where $E\approx V$, the matrix representation takes $O(V^3)$ while the list representation is faster and takes $O(V^2)$. Hence, the scaling behavior of the Edge-Elimination algorithm is worse than the Recursive Dijkstra algorithm.

Comparison to the Edge-List Bisection Algorithm

The approach for determining MWP and GMWP paths that we discussed is closely related to that of Metzner et al[120]. In [120] the network was based on Transition Path Theory (TPT) while our approaches use the formulation of Milestoning.

Here, we call the path algorithm given in[120] as the Edge-List Bisection algorithm and describe it below.

The overall approach used to identify Global Maximum Weight Paths in this algorithm, is identical to the Recursive Dijkstra's algorithm for GMWP calculation (Algorithm 3 in this chapter). That means, a bottleneck edge (u,v) is computed between vertices *s* and *t* first, and then the path between $\langle s..u \rangle$ and $\langle v..t \rangle$ is recursively identified. But the underlying algorithm to calculate a bottleneck each time (which in the Recursive Dijkstra's algorithm is a modification of the Dijkstra's algorithm) is a variant of the Edge-Elimination algorithm. The following steps describe how the bottleneck edge between two vertices *s* and *t* is selected each time.

1. Sort all the edges in the graph G based on their weight, into a list, $L_w = [e_1, e_2 .... e_{|E|}]$. The edges are stored in ascending order as in the Edge-Elimination algorithm.
2. If the last edge in L_w, $e_{|E|}$ is an edge between *s* and *t*, return the last edge as the bottleneck edge.
3. Go to the edge in the middle of the current sorted list, e_m. Let the weight of this edge be w_m.
    i) If *s* and *t* are still connected by removing all edges with weight less than w_m, then the bottleneck edge has a weight higher than w_m. Hence it is located in the second half of the edge list between $e_{m+1} .... e_{|E|}$, which is the part of the edge list we need to explore next.
    ii) Else if removing edges with weight less than w_m results in *s* being disconnected from *t*, then the bottleneck has a weight lower than w_m and is located in the first half of the edge list between $e_1 .... e_m$.

Note that we obtain from step 4, a sublist to be explored, and this sublist is half the size of the original sorted list. We then repeat steps 3 and 4, exploring the middle edge of the new sublist and using it to halve the edge list each time. These steps are

repeated till the final edge list consists of just one edge. This edge is the bottleneck edge returned by the algorithm.

Unlike the Edge-Elimination algorithm, where we go through each edge in the edge list one by one, here we traverse the edge-list in a bisected search manner, bisecting the edge list till we are left with a single edge. The overall algorithm runs in an identical manner to the Recursive Dijkstra's algorithm in terms of identifying the bottlenecks and reconstructing the path.

In contrast to the previous two algorithms the Edge-List bisection algorithm makes the following assumptions: (i) the graph has no edge degeneracy, (ii) the set of all the MWPs includes all the edges of the graphs, and (iii) there are no cycles in the graph. Assumption (ii) requires, for example, that the graph does not include dead-end branches. Hence some pre-processing of the graphs may be required.

To find a single bottleneck edge, the bisected edge list search examines $O(\log E)$ edges. And for each edge, one connectivity test is performed using Breadth-first Search or Depth-First Search, which takes $O(V+E)$ or $O(V^2)$ depending on whether the graph representation is in terms of the adjacency list or adjacency matrix. Hence the search for a single bottleneck edge takes $O(V^2 \log E)$ for the matrix representation, and $O(E \log E)$ for the list representation. Since there are atmost $O(V)$ edges on the GMWP, the overall algorithm takes $O(V^3 \log E)$ for the matrix representation and $O(VE \log E)$ for the list representation. Hence the complexity is $O(V^3 \log V)$ for all networks in the matrix representation and for dense matrices in the list representation, and becomes $O(V^2 \log V)$ for sparse networks in the list representation.

Note that the paths returned by all three algorithms above are identical.

6.5 RESULTS AND DISCUSSION

We considered two systems to demonstrate the paths: unfolding of a helix under stress and membrane permeation of DOPC. Below is a description of the systems and the paths we obtained in both.

Helix Unfolding under Stress

Alpha helices are prime secondary structure elements that are found in proteins. Their stability and folding/unfolding pathways are therefore of considerable interest. A recent study[136, 137] simulated a single molecule experiment of a ~100 amino acid helix, in which both terminals were pulled by an external force and unfolding events were recorded. For each of 10 load levels from 0pN to 100pN, 500 transition kernel matrices, K and milestone lifetimes, τ were sampled, from which fluxes were calculated using the Milestoning equation [136, 137]. We calculated paths (GMWP) on the average kernel matrices, lifetimes and fluxes, averaged over the 500 samples for each load level.

In this system the number of anchors was 14. For the different load levels, the number of milestones found were 129, 125, and 109 for 0, 30 and 70pN respectively. For path calculations, the starting anchor corresponded to the state *alpha3*, the fully folded α-helix state, with three hydrogen bonds wrapping an amino acid. The ending anchor corresponded to the unfolded state of the helix, in which no hydrogen bonds are formed and the dihedral angle is in the extended chain configuration, with $psi > 90$.

In milestone space, these start and end anchors corresponded to one start milestone and four end milestones, since there were multiple ways to reach the last anchor (unfolded state). All paths were converted to anchor space for visualization. Figure 6.4 demonstrates the complexity of the state-space and flux-space networks for the intermediate load level of 30pN.

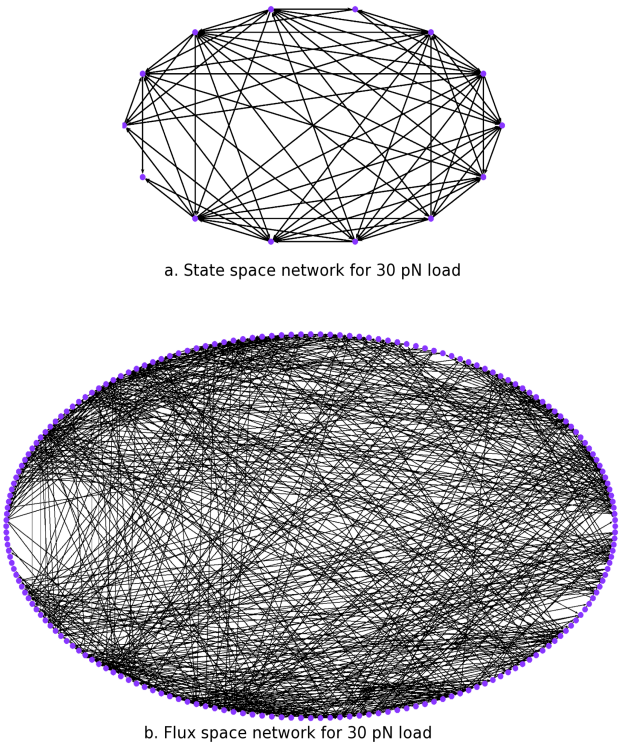
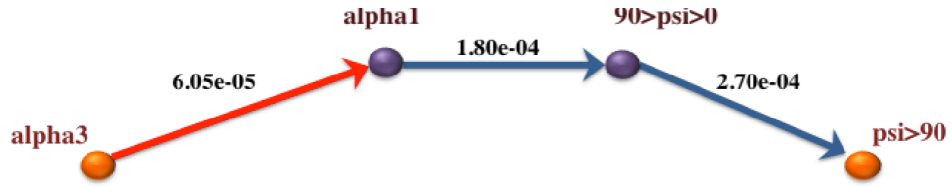
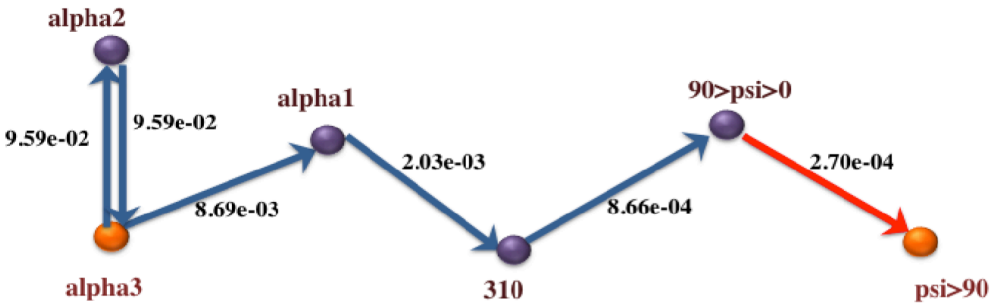


Figure 6.4 Visualization of average networks for helix unfolding under a load level 30pN in (a) state-space, with 14 anchors (vertices). (b) flux-space with 125 milestones (vertices). The graphs are to illustrate the complexity of analysis and were prepared with the Pajek program[139].

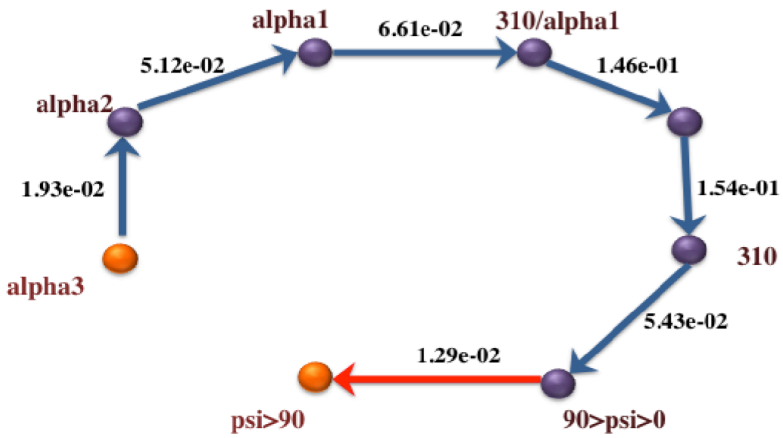
Figures 6.5-6.7 depict the global maximum weight pathways obtained from the three different graph representations: state-space graph, flux-space graph with flux-based weights and flux-space graph with rate coefficients, for three different load levels: 0pN, 30pN and 70pN. Intermediate vertices on the paths represent partially folded states like *alpha2* and *alpha1* with 2 and 1 hydrogen bonds remaining respectively, misfolded states like *310*, representing the 3_{10} helix, or nearly unfolded states, like $90 < psi < 0$.



a. 0pN, Path from state-space graph based on flux-based edge weights

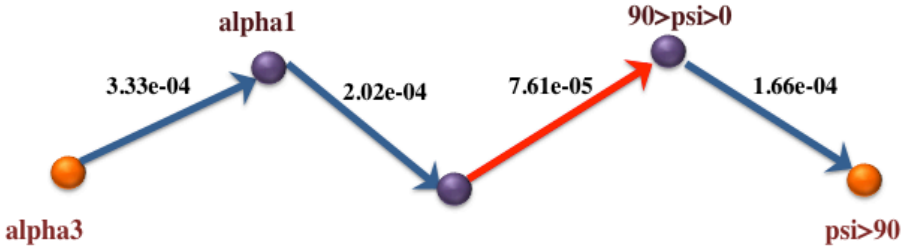


b. 0pN, Path from flux-space graph based on flux-based edge weights

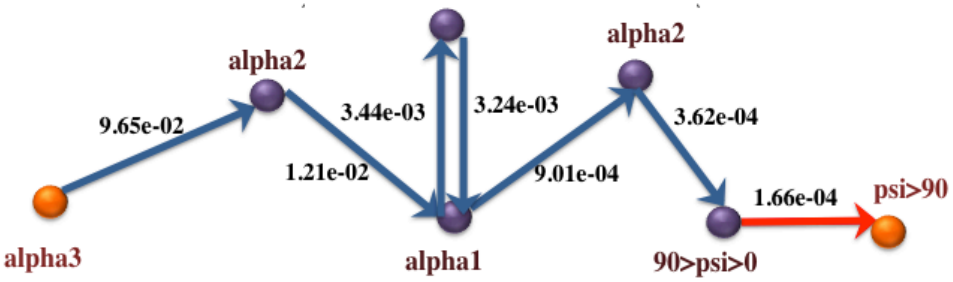


c. 0pN, Path from flux-space graph based on rate coefficients

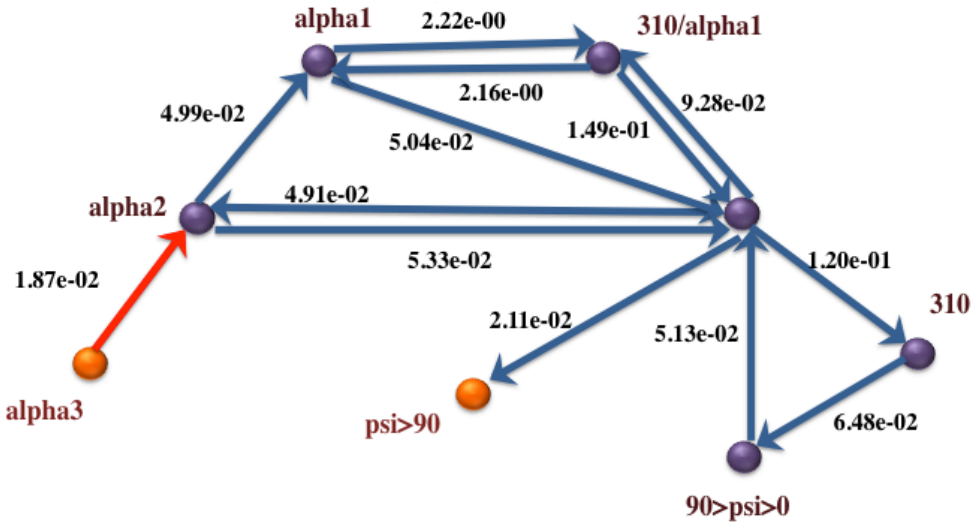
Figure 6.5 Global maximum weight paths using three different graph representations for helix unfolding under 0pN stress. Bottleneck edges (EMW) are in red.



a. 30pN, Path from state-space graph based on flux-based edge weights

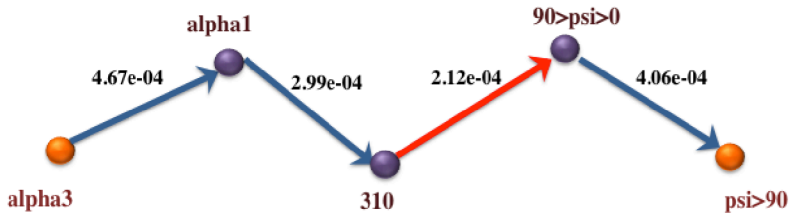


b. 30pN, Path from flux-space graph based on flux-based edge weights

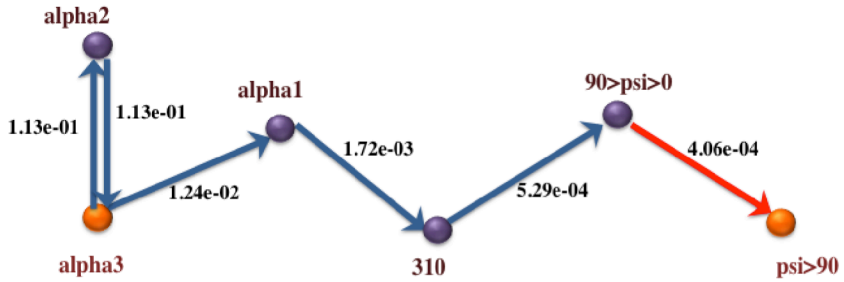


c. 30pN, Path from flux-space graph based on rate coefficients

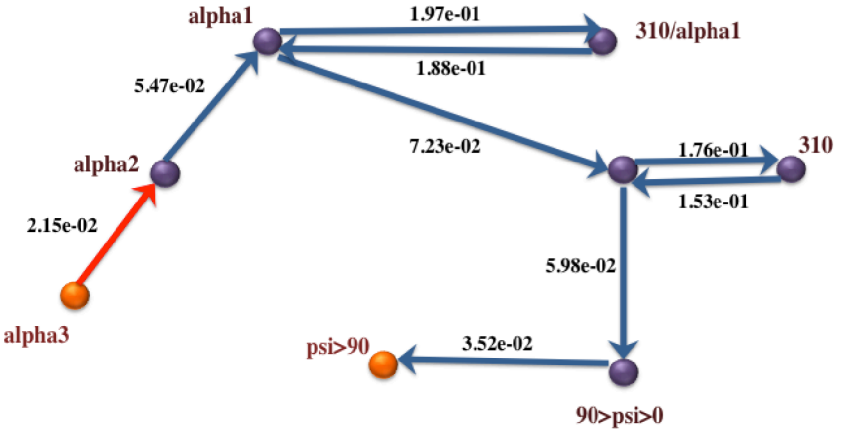
Figure 6.6 Global maximum weight paths using three different graph representations for helix unfolding under 30pN stress. Bottleneck edges (EMW) are in red.



a. 70pN, Path from state-space graph based on flux-based edge weights



b. 70pN, Path from flux-space graph based on flux-based edge weights



c. 70pN, Path from flux-space graph based on rate coefficients

Figure 6.7 Global maximum weight paths using three different graph representations for helix unfolding under 70pN stress. Bottleneck edges (EMW) are in red.

When examining the state-based pathways, we notice that the position of the EMW or the transition state is different for different loads. For example, the state-based path for 0pN path is direct and moves from the three-hydrogen-bond state to a state with one hydrogen bond and then to a state with positive backbone dihedral, *psi*. Finally the system transitions to the unfolded state, where no hydrogen bonds are present. The bottleneck is at the break of the first two hydrogen bonds. A similar path is followed at 30pN load, with the addition of one more (unlabeled) intermediate state with no hydrogen bonds. The dihedral angles of the unlabeled state are still in the folded region. Interestingly, the bottleneck at 30pN is different from the 0pN case, is shifted to a backbone conformational transition, and is not at the dissociation of a hydrogen bond. The 70pN path illustrates another twist in which a new intermediate hydrogen bond (3_{10}) is formed before the system unfolds. The bottleneck is shifted to the last state in which the psi dihedral completes the rotation to domains greater than 90 degrees. This is consistent with the application of additional load, since the 3_{10} helix is more extended than the α helix and it is preferred at the high load limit, compared to the random chain less-extended conformation (the unlabeled state) of the 30pN load.

The most complex paths are obtained at intermediate load level (30pN). One can understand this by considering the two limits of low and high loads. At low (zero) loads the system does not have sufficient energy to explore the energy landscape and is restricted to a few dominant and low energy reaction coordinates. At high load level, the large external force dominates the energy landscape. The external force washes out many of the molecular details and induces the system to fold in more direct and straightforward pathways. At intermediate load level, the external force is sufficient to reduce the free energy barrier to the extent that new states can be found and explored but it is not too strong to overwhelm the features of the energy landscape. This is also consistent with the

earlier observation[136, 137] that the mean first passage time through the system is longer for intermediate load levels.

We also explore different graph resolutions. The state-space graph is of the lowest resolution and the paths from this graph have the lowest level of details. A higher level of resolution is provided by the milestone-space graph. Milestones are the interfaces between states and obviously there are more interfaces than states. The last representation, which is based on rate coefficients between milestones, is not only more complex but also approximate. The significant differences from the kinetically exact MaxFlux path suggests that it is mechanistically incorrect.

Membrane Permeation of DOPC

Phospholipid membranes such as DOPC efficiently separate two aqueous solutions and support concentration gradients of different solutes that are necessary for life processes. However, the membrane barrier is not absolute and passive permeation is possible. It is an intriguing question whether basic ingredients of biological macromolecules (such as amino acids and sugar molecules) can permeate through membranes without the active assistance of transporters. Recently an investigation was initiated to accurately simulate the permeation of complex molecules through membranes[121, 140]. In particular, the translocation of a blocked tryptophan was simulated with Milestoning. A network was built that takes into account not only the center of mass of the permeant, but also the orientation of the molecule with respect to the membrane axis. The number of anchors here was 217 and the number of milestones was 1204. The start anchor (and milestone) corresponded to the permeant at the left of the membrane and the end anchor (and milestone) corresponded to the permeant in solution at the right side of the membrane.

The global maximum weight paths obtained using various graph representations for this system are shown in Figure 6.8. The paths based on fluxes, in both the flux-space and state-space graphs, are quite similar. But the path based on local rate coefficients is quite different and samples a different part of the conformation space. This example too suggests that the mechanisms obtained from local kinetic information can be different from those based on the exact kinetics. Nevertheless, the alternative path based on local kinetics is found at somewhat lower scoring GMWPs of flux-based graphs. Hence it is still a sensible choice with acceptable weight. For dense and degenerate graphs, multiple pathways of similar scores can be obtained, and this may be the case also here. The path based on rate coefficients is less "committed" to the low free energy minima shown in gray on the upper left side and lower right side of the plot in Figure 6.8.

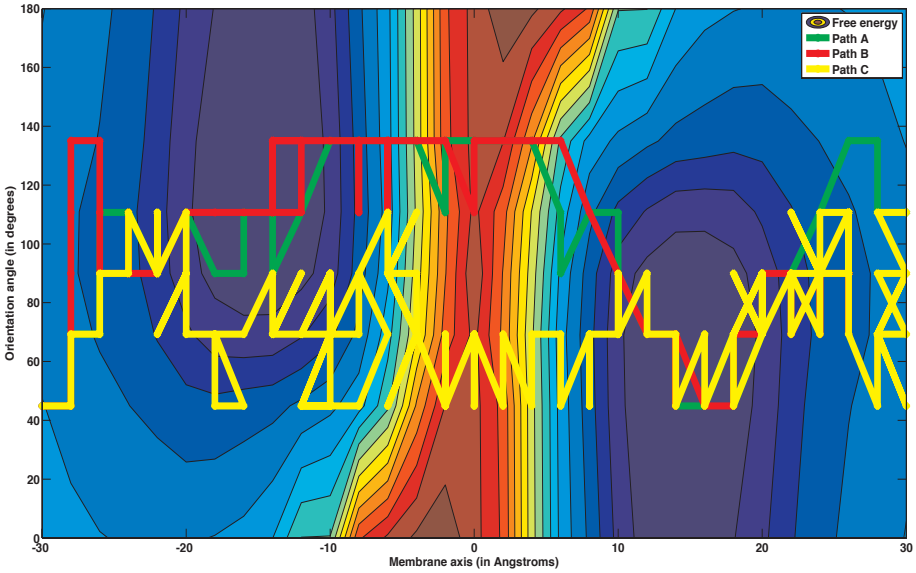


Figure 6.8 Global maximum weight paths for membrane permeation of DOPC. The graph representations are: Path A: state-space graph with flux-based weights. Path B: flux-space graph with flux-based weights. Path C: flux-space graph weighted by local rate coefficients.

6.6 Analysis of Run Times and Benchmarks

Table 6.3 summarizes the worst-case time complexities for dense and sparse graphs for various implementations of the path algorithms. *G: List* means the graph is implemented using adjacency lists, *G: Matrix* means the graph is implemented using adjacency matrices, *Q: Array* means the priority queue in Dijkstra's algorithm is implemented using arrays and *Q: Heap* means the priority queue is implemented using Fibonacci heaps. These scaling factors have been derived in the Efficiency section of each algorithm.

Note that the Edge-Elimination algorithm shows a marked difference in complexity between dense and sparse graphs. It is particularly inefficient for dense graphs and works best for sparse graphs when the number of edges is small. For dense graphs, the Recursive Dijkstra's algorithm shows the most favorable asymptotic time complexity. The Edge-List Bisection algorithm possesses complexities comparable to that of the Recursive Dijkstra's algorithm. Generally, state-space graphs maybe dense while flux-space graphs are usually sparse.

Table 6.3 Summary of asymptotic time complexities of various algorithms for dense $(E \approx V^2)$ and sparse $(E \approx V)$ graphs.

Dense graphs					
Recursive Dijkstra		Edge-List Bisection		Edge Elimination	
G: List Q: Heap	G: Matrix Q: Array	G: List	G: Matrix	G: List	G: Matrix
$O(V^3)$	$O(V^3)$	$O(V^3 \log V)$	$O(V^3 \log V)$	$O(V^4)$	$O(V^4)$
Sparse graphs					
Recursive Dijkstra		Edge-List Bisection		Edge Elimination	
G: List Q: Heap	G: Matrix Q: Array	G: List	G:Matrix	G: List	G:Matrix
$O(V^2 \log V)$	$O(V^3)$	$O(V^2 \log V)$	$O(V^3 \log V)$	$O(V^2)$	$O(V^3)$

Table 6.4 Average runtimes in milliseconds for random graphs with 100, 1000 and 10000 vertices, for the three algorithms.

Dense graphs				
Graph size (nodes)	Number of edges	Recursive Dijkstra's	Edge-List Bisection	Edge Elimination
100	9,500	0.31	2.93	258.61
1000	600,000	98.70	404.06	1.56e+06
10000	60,000,000	49,045.3	117,248.62	-
Sparse graphs				
Graph size (nodes)	Number of edges	Recursive Dijkstra's	Edge-List Bisection	Edge Elimination
100	1,000	0.23	0.54	13.83
1000	10,000	47.81	36.18	7554.45
10000	100,000	11,188.2	7,228.17	-

To obtain a consistent and unbiased measure of the algorithm efficiency in practice, we recorded the runtimes of the two algorithms on random graphs. We generated several sparse and dense random graphs and runtimes were estimated by averaging the results over different random graphs and different start and end nodes. Simple implementations were used for both algorithms i.e. graphs were implemented using matrices and queues were implemented using arrays, since the asymptotic complexity is about the same for the simple versus the more sophisticated implementations, for either algorithm.

Table 6.4 shows the performance of the two algorithms on random graphs. Runtimes were calculated on a single core of an 8 core Linux Intel Xeon X5460 processor with clock speed of 3.16 GHz and 16GB memory shared among 8 cores. Runtimes were not calculated for the Edge-Elimination algorithm for 10000 vertices since the estimated runtime was too long. Also shown is the number of edges for each size of random graphs.

The Edge-Elimination algorithm is much slower than the other algorithms for all graph sizes, and its performance degrades significantly when transitioning from sparse to dense networks. The Recursive Dijkstra's algorithm, on the other hand, requires approximately the same order of magnitude of runtimes in both dense and sparse cases. The runtimes of the Edge-list Bisection algorithm are comparable to that of the Recursive Dijkstra's algorithm. We note that the Edge-List Bisection algorithm is most efficient for sparse graphs while the Recursive Dijkstra algorithm is most efficient for dense graphs.

We see that though the worst-case complexities of the algorithms are not very different, there is a wide difference in runtimes on the benchmark. Let us consider for example, the asymptotic complexities of the algorithms in Table 6.3 for sparse graphs using matrix representations of graph. For the Edge-Elimination algorithm, one needs to traverse through the list of sorted edges, checking for each edge, if its removal disconnects the two end vertices (an operation that takes $O(V^2)$ in this case), terminating only when the set of edges on the path is complete. In practice, this leads to a large number of edges being explored before we recover the complete path. So the average time complexity is closer to the worst-case time complexity for the Edge-Elimination algorithm.

In contrast, for the Recursive Dijkstra's algorithm and the Edge-list Bisection algorithm, we run the underlying bottleneck identification algorithms (which are the modified Dijkstra's algorithm which takes $O\left(V^2\right)$, or the bisection-based algorithm which takes $O(V^2 \log V)$ respectively in this case) only once per edge in the global maximum weight path. These means we only need to run these underlying bottleneck identification algorithms, E_p times, where E_p is the number of edges on the global maximum weight path. In practice, E_p can be far less than the number of vertices, which in turn is much less than the number of edges. Hence the average runtime for these

algorithms can be much smaller than the time predicted by the asymptotic analysis and is smaller than the time taken by Edge-Elimination.

6.7 CONCLUSIONS

Network representations are emerging from a number of enhanced sampling techniques for molecular kinetics using methods like Milestoning, TPT, MSM, and more. The push to longer time scales is obtained by calculation of local kinetic information by MD (e.g. local rate coefficients) and using the data in coarser equations such as in Milestoning. Networks offer a natural way for coarse-graining without losing too much in spatial resolution, while being able to push temporal scales to significantly longer domains (from nanoseconds to hours[121]). We expect the use of networks as well as the complexity of the networks (number of edges and vertices) to increase significantly in the future. This increase in network complexity is necessary to capture more details of chemical processes, allowing for the interactions of multiple coarse variables and going beyond one-dimensional reaction coordinates. However, the complexity of networks makes them harder to interpret and obtain qualitative insight, compared to lower dimensionality modeling. To obtain such qualitative interpretation, we identify in the network, dominant edges and paths that carry significant fluxes or trajectories and hence are more important than others. Maximum flux or global maximum weight pathways are discussed at length in the present paper as a natural choice for these analyses. Recursive Dijkstra's and Edge-List Bisection algorithms are proposed as efficient and scalable approaches to identify them. We also discussed the interpretation of molecular mechanisms using networks for analysis. We argue that using local kinetic information

(such as rate coefficients) instead of exact solution of the kinetic equations may lead to incorrect dominant pathways.

Code for calculating optimal pathways in networks is available as part of the analysis module of the molecular dynamics program MOIL[141]. It can be downloaded from http://clsb.ices.utexas.edu/web/moil.html

Chapter 7. Conclusions and Future Work

In this thesis, algorithms for improving protein-protein complex prediction were outlined. Large-scale machine learning methods like pairwise learning using linear programming were used to derive new potentials, or scoring functions. The training involved examination of hundreds of thousands of models, which included both correct and incorrect structures. This learning algorithm models the funnel energy landscape of protein complexes using constraints that stipulate that the energy of a correct model should be lower than the energy of an incorrect model. The contributions of this thesis from a computer science perspective are the introduction of hierarchical constraints into the learning algorithm, and a comparison to other well-known machine learning algorithms like SVMs and neural networks. Pairwise learning using linear programming compared favorably to the other algorithms, in terms of accuracy, training and test time.

From the perspective of structure prediction of protein complexes, new methods for reranking models were introduced. These methods were implemented in the docking package DOCK/PIERR [13, 41, 113]. Specifically, a new atomic potential [41] and a new hydrogen bond based potential [63] were developed. These potentials were combined with side chain remodeling, energy minimization and molecular dynamics-based sampling procedures to obtain more chemically reasonable structures starting from rigid docking models. The docking algorithm was shown to be comparable to the best docking algorithms in community wide assessments and benchmarks [47]. These advances help establish automated docking methods as accurate methods for structure prediction and enable departure from previous methods that rely more on human intervention.

The docking algorithm was extended to study membrane proteins [91], where a new membrane-based environment potential was introduced and shown to improve the

quality of predictions. Applications for docking of the amyloid precursor protein were studied. Finally, algorithms for calculating pathways in networks obtained from molecular dynamics simulations were discussed [116].

Protein-protein docking is an exciting field and there is lots of scope for improving structure predictions. Current methods can predict complexes correctly, in the top 10, about 40-60% of the time. We note that until this point, in DOCK/PIERR, we have only modeled rigid docking structures with minimal alterations to the structures, mostly in the side chains. A challenging next step is to model larger conformational change, by using algorithms that introduce backbone flexibility combined with side chain flexibility. Recent refinement algorithms based on unrestrained molecular dynamics with hybrid atomic and residue level modeling were found to improve the quality of docking solutions [19].

We note the deficiencies of our method identified in chapters 2 and 4: namely, that docking is inaccurate when the number of contacts in the native structure is low. This can be fixed by adding these low-contact native structures to the training, or training a separate potential for low-contact structures, which can be used when we have apriori knowledge that the number of contacts in the predicted model should be low.

Apart from sampling, scoring functions can also be improved in many ways. Recent developments that include entropic information about the model, have shown to improve scoring function accuracy. The size of clusters of models and stability of clusters was shown to aid in removing false positives during reranking [142]. Another successful method [143, 144] based on entropy involves kinetics-based approaches that used time-homogeneous Markov chain models, to determine transition probabilities between models, and selected models based on their equilibrium population. Graph-based approaches that represent the binding interfaces as networks, and use subgraph mining to

identify common motifs have also been recently popularized [145, 146]. Finally, another recently successful direction was the use of evolutionary information in developing multi-body potentials for ranking [144, 145].

The prediction of binding affinity from docking based potentials has been unsuccessful so far [145, 147, 148]. But new affinity benchmarks [149] have been recently, and machine-learning methods of the type described in this thesis can be used for developing potentials that can predict binding affinity, which quantifies the strength of the interaction, and can be used on the scale of proteomes to predict whether two proteins interact. This can help classify proteins whose functions are not yet known.

Membrane proteins are yet another relatively unexplored area for docking, where there are lots of important and open problems. The advantage of computational docking methods is that rigid docking is sufficient for a large number of membrane proteins [98]. Ensemble methods that combine the predictions from different sources like different docking algorithms, molecular dynamics simulations and experimental data, can provide more coverage of structural data about membrane proteins.

Further, scaling docking algorithms to predict higher order complexes, involving three, four and higher number of proteins, i.e. combinatorial assembly of proteins is a computationally challenging problem. It has been approached in the past by branch and bound techniques to eliminate a large number of intermediate solutions [2, 150] and graph-theoretical techniques [151].

On the machine learning front, developing new potentials based on recently popular methods like deep learning, and extending our linear programming based approach to non-linear kernels seem like promising future directions.

References

1. Baker D, Sali A. Protein structure prediction and structural genomics. *Science* 2001,**294**:93-96.
2. Ravikant D. Learning to Dock Proteins. Ithaca NY: Cornell University; 2011.
3. Uetz P, Giot L, Cagney G, Mansfield TA, Judson RS, Knight JR, *et al*. A comprehensive analysis of protein-protein interactions in Saccharomyces cerevisiae. *Nature* 2000,**403**:623-627.
4. Giot L, Bader JS, Brouwer C, Chaudhuri A, Kuang B, Li Y, *et al*. A protein interaction map of Drosophila melanogaster. *Science* 2003,**302**:1727-1736.
5. Janin J, Bahadur RP, Chakrabarti P. Protein-protein interaction and quaternary structure. *Quarterly Reviews of Biophysics* 2008,**41**:133-180.
6. PDB RCSB. Crystal structure of TRAF6 in complex with Ubc13 in the P1 space group. 2012.
7. PDB RCSB. Crystal structure of calmodulin binding domain of orai1 in complex with Ca2+ calmodulin displays a unique binding mode. 2012.
8. Dale GE, Oefner C, D'Arcy A. The protein as a variable in protein crystallization. *J Struct Biol* 2003,**142**:88-97.
9. Smith GR, Sternberg MJE. Prediction of protein-protein interactions by docking methods. *Current Opinion in Structural Biology* 2002,**12**:28-35.
10. Chen R, Weng ZP. Docking unbound proteins using shape complementarity, desolvation, and electrostatics. *Proteins-Structure Function and Genetics* 2002,**47**:281-294.
11. Comeau SR, Gatchell DW, Vajda S, Camacho CJ. ClusPro: a fully automated algorithm for protein-protein docking. *Nucleic Acids Research* 2004,**32**:W96-W99.
12. Tovchigrechko A, Vakser IA. Development and testing of an automated approach to protein docking. *Proteins-Structure Function and Bioinformatics* 2005,**60**:296-301.
13. Ravikant DVS, Elber R. Energy design for protein-protein interactions. *Journal of Chemical Physics* 2011,**135**.
14. Duhovny D, Nussinov R, Wolfson HJ. Efficient unbound docking of rigid molecules. *Algorithms in Bioinformatics, Proceedings* 2002,**2452**:185-200.

15. Lorenzen S, Zhang Y. Monte Carlo refinement of rigid-body protein docking structures with backbone displacement and side-chain optimization. *Protein Science* 2007,**16**:2716-2725.

16. Gray JJ, Moughon S, Wang C, Schueler-Furman O, Kuhlman B, Rohl CA, *et al*. Protein-protein docking with simultaneous optimization of rigid-body displacement and side-chain conformations. *Journal of Molecular Biology* 2003,**331**:281-299.

17. Wang C, Bradley P, Baker D. Protein-protein docking with backbone flexibility. *Journal of Molecular Biology* 2007,**373**:503-519.

18. Wang C, Schueler-Furman O, Baker D. Improved side-chain modeling for protein-protein docking. *Protein Science* 2005,**14**:1328-1339.

19. Zacharias M. Combining coarse-grained nonbonded and atomistic bonded interactions for protein modeling. *Proteins-Structure Function and Bioinformatics* 2013,**81**:81-92.

20. Bonvin AM. Flexible protein-protein docking. *Current Opinion in Structural Biology* 2006,**16**:194-200.

21. Andrusier N, Nussinov R, Wolfson HJ. FireDock: Fast interaction refinement in molecular docking. *Proteins-Structure Function and Bioinformatics* 2007,**69**:139-159.

22. Mashiach E, Nussinov R, Wolfson HJ. Fiber Dock: Flexible induced-fit backbone refinement in molecular docking. *Proteins-Structure Function and Bioinformatics* 2010,**78**:1503-1519.

23. Li L, Chen R, Weng ZP. RDOCK: Refinement of rigid-body protein docking predictions. *Proteins-Structure Function and Genetics* 2003,**53**:693-707.

24. Halperin I, Ma BY, Wolfson H, Nussinov R. Principles of docking: An overview of search algorithms and a guide to scoring functions. *Proteins-Structure Function and Genetics* 2002,**47**:409-443.

25. Liang SD, Wang GC, Zhou YQ. Refining near-native protein-protein docking decoys by local resampling and energy minimization. *Proteins-Structure Function and Bioinformatics* 2009,**76**:309-316.

26. Pierce B, Weng ZP. A combination of rescoring and refinement significantly improves protein docking performance. *Proteins-Structure Function and Bioinformatics* 2008,**72**:270-279.

27. Masone D, de Vaca IC, Pons C, Recio JF, Guallar V. H-bond network optimization in protein-protein complexes: Are all-atom force field scores enough? *Proteins-Structure Function and Bioinformatics* 2012,**80**:818-824.

28. Kozakov D, Hall DR, Beglov D, Brenke R, Comeau SR, Shen Y, *et al*. Achieving reliability and high accuracy in automated protein docking: ClusPro, PIPER, SOU, and stability analysis in CAPRI rounds 13-19. *Proteins-Structure Function and Bioinformatics* 2010,**78**:3124-3130.

29. Pierce B, Weng ZP. ZRANK: Reranking protein docking predictions with an optimized energy function. *Proteins-Structure Function and Bioinformatics* 2007,**67**:1078-1086.

30. Bryngelson JD, Onuchic JN, Socci ND, Wolynes PG. Funnels, Pathways, and the Energy Landscape of Protein-Folding - a Synthesis. *Proteins-Structure Function and Genetics* 1995,**21**:167-195.

31. Chaplin M. The folding energy landscape in the presence of low hydration. In; 2012.

32. Jorgensen WL, Tiradorives J. The Opls Potential Functions for Proteins - Energy Minimizations for Crystals of Cyclic-Peptides and Crambin. *Journal of the American Chemical Society* 1988,**110**:1657-1666.

33. Schwede T, Peitsch MC. *Computational structural biology : methods and applications*. N.J.: World Scientific; 2008.

34. Maiorov VN, Crippen GM. Contact Potential That Recognizes the Correct Folding of Globular-Proteins. *Journal of Molecular Biology* 1992,**227**:876-888.

35. Tobi D, Elber R. Distance-dependent, pair potential for protein folding: Results from linear optimization. *Proteins-Structure Function and Genetics* 2000,**41**:40-46.

36. Rajgaria R, McAllister SR, Floudas CA. A novel high resolution C-alpha-C-alpha distance dependent force field based on a high quality decoy set. *Proteins-Structure Function and Bioinformatics* 2006,**65**:726-741.

37. Qiu J, Elber R. Atomically detailed potentials to recognize native and approximate protein structures. *Proteins-Structure Function and Bioinformatics* 2005,**61**:44-55.

38. Ravikant DVS, Elber R. PIE-efficient filters and coarse grained potentials for unbound protein-protein docking. *Proteins-Structure Function and Bioinformatics* 2010,**78**:400-419.

39. Wagner M, Meller J, Elber R. Large-scale linear programming techniques for the design of protein folding potentials. *Mathematical Programming* 2004,**101**:301-318.

40. Meller J, Wagner M, Elber R. Maximum feasibility guideline in the design and analysis of protein folding potentials. *Journal of Computational Chemistry* 2002,**23**:111-118.

41. Viswanath S, Ravikant DVS, Elber R. Improving ranking of models for protein complexes with side chain modeling and atomic potentials. *Proteins-Structure Function and Bioinformatics* 2013,**81**:592-606.

42. Joachims T. Structured output prediction with Support Vector Machines. *Structural, Syntactic, and Statistical Pattern Recognition, Proceedings* 2006,**4109**:1-7.

43. Kabsch W. Solution for Best Rotation to Relate 2 Sets of Vectors. *Acta Crystallographica Section A* 1976,**32**:922-923.

44. Mendez R, Leplae R, De Maria L, Wodak SJ. Assessment of blind predictions of protein-protein interactions: Current status of docking methods. *Proteins-Structure Function and Bioinformatics* 2003,**52**:51-67.

45. Zhang Y, Skolnick J. TM-align: a protein structure alignment algorithm based on the TM-score. *Nucleic Acids Research* 2005,**33**:2302-2309.

46. Xu D, Tsai CJ, Nussinov R. Hydrogen bonds and salt bridges across protein-protein interfaces. *Protein Engineering* 1997,**10**:999-1012.

47. Lensink MF, Wodak SJ. Docking, scoring, and affinity prediction in CAPRI. *Proteins-Structure Function and Bioinformatics* 2013,**81**:2082-2095.

48. Kirmizialtin S, Elber R. Revisiting and Computing Reaction Coordinates with Directional Milestoning. *Journal of Physical Chemistry A* 2011,**115**:6137-6148.

49. Faradjian AK, Elber R. Computing time scales from reaction coordinates by milestoning. *Journal of Chemical Physics* 2004,**120**:10880-10889.

50. West AMA, Elber R, Shalloway D. Extending molecular dynamics time scales with milestoning: Example of complex kinetics in a solvated peptide. *Journal of Chemical Physics* 2007,**126**.

51. Majek P, Elber R. Milestoning without a Reaction Coordinate. *Journal of Chemical Theory and Computation* 2010,**6**:1805-1817.

52. Cormen T, Leiserson C, Rivest R. Introduction to Algorithms, MIT Press, 1992.

53. Dijkstra EW. A note on two problems in connexion with graphs. *Numerische mathematik* 1959,**1**:269-271.

54. Gray JJ. High-resolution protein-protein docking. *Current Opinion in Structural Biology* 2006,**16**:183-193.

55. Hwang H, Pierce B, Mintseris J, Janin J, Weng ZP. Protein-protein docking benchmark version 3.0. *Proteins-Structure Function and Bioinformatics* 2008,**73**:705-709.

56. Janin J. Assessing predictions of protein-protein interaction: the CAPRI experiment. *Protein Science* 2005,**14**:278-283.

57. Krivov GG, Shapovalov MV, Dunbrack RL. Improved prediction of protein side-chain conformations with SCWRL4. *Proteins-Structure Function and Bioinformatics* 2009,**77**:778-795.

58. Elber R, Roitberg A, Simmerling C, Goldstein R, Li HY, Verkhivker G, *et al*. Moil - a Program for Simulations of Macromolecules. *Computer Physics Communications* 1995,**91**:159-189.

59. Sali A, Blundell TL. Comparative Protein Modeling by Satisfaction of Spatial Restraints. *Journal of Molecular Biology* 1993,**234**:779-815.

60. Chen R, Li L, Weng ZP. ZDOCK: An initial-stage protein-docking algorithm. *Proteins-Structure Function and Genetics* 2003,**52**:80-87.

61. Altschul SF, Madden TL, Schaffer AA, Zhang JH, Zhang Z, Miller W, *et al*. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research* 1997,**25**:3389-3402.

62. Eswar N, John B, Mirkovic N, Fiser A, Ilyin VA, Pieper U, *et al*. Tools for comparative protein structure modeling and analysis. *Nucleic Acids Research* 2003,**31**:3375-3380.

63. Viswanath S, Elber R. A hydrogen bond potential for docking water-soluble and transmembrane protein complexes. *in preparation* 2014.

64. Kortemme T, Morozov AV, Baker D. An orientation-dependent hydrogen bonding potential improves prediction of specificity and structure for proteins and protein-protein complexes. *Journal of Molecular Biology* 2003,**326**:1239-1259.

65. Lo Conte L, Chothia C, Janin J. The atomic structure of protein-protein recognition sites. *Journal of Molecular Biology* 1999,**285**:2177-2198.

66. Norel R, Sheinerman F, Petrey D, Honig B. Efficient docking for antibody-antigen complexes. *Biophysical Journal* 2000,**78**:38A-38A.

67. Norel R, Sheinerman F, Petrey D, Honig B. Electrostatic contributions to protein-protein interactions: Fast energetic filters for docking and their physical basis. *Protein Science* 2001,**10**:2147-2161.

68. Mandell JG, Roberts VA, Pique ME, Kotlovyi V, Mitchell JC, Nelson E, *et al*. Protein docking using continuum electrostatics and geometric fit. *Protein Engineering* 2001,**14**:105-113.

69. Jackson RM, Sternberg MJE. A Continuum Model for Protein-Protein Interactions - Application to the Docking Problem. *Journal of Molecular Biology* 1995,**250**:258-275.

70. Morozov AV, Kortemme T. Potential functions for hydrogen bonds in protein structure prediction and design. *Advances in protein chemistry* 2005,**72**:1-38.

71. Fernandez-Recio J, Totrov M, Abagyan R. Soft protein-protein docking in internal coordinates. *Protein science : a publication of the Protein Society* 2002,**11**:280-291.

72. Fernandez-Recio J, Totrov M, Abagyan R. Screened charge electrostatic model in protein-protein docking simulations. *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing* 2002:552-563.

73. Hwang H, Vreven T, Janin J, Weng ZP. Protein-protein docking benchmark version 4.0. *Proteins-Structure Function and Bioinformatics* 2010,**78**:3111-3114.

74. Li YQ, Roy A, Zhang Y. HAAD: A Quick Algorithm for Accurate Prediction of Hydrogen Atoms in Protein Structures. *PLoS One* 2009,**4**.

75. Papoian GA, Ulander J, Wolynes PG. Role of water mediated interactions in protein-protein recognition landscapes. *Journal of the American Chemical Society* 2003,**125**:9170-9178.

76. Papoian GA, Wolynes PG. The physics and Bioinformatics of binding and folding - An energy landscape perspective. *Biopolymers* 2003,**68**:333-349.

77. Ben-Hur A, Weston J. A user's guide to support vector machines. *Methods in molecular biology* 2010,**609**:223-239.

78. Cortes C, Vapnik V. Support-Vector Networks. *Machine Learning* 1995,**20**:273-297.

79. Joachims T. Making large-Scale SVM Learning Practical. In: *Advances in Kernel Methods - Support Vector Learning*. Edited by Smola BSaCBaA: MIT Press; 1999.

80. Morik K, Brockhausen P, Joachims T. Combining statistical learning with a knowledge-based approach - A case study in intensive care monitoring. *Machine Learning, Proceedings* 1999:268-277.

81. Morawietz T, Sharma V, Behler J. A neural network potential-energy surface for the water dimer based on environment-dependent atomic energies and charges. *Journal of Chemical Physics* 2012,**136**.

82. Jose KVJ, Artrith N, Behler J. Construction of high-dimensional neural network potentials using environment-dependent atom pairs. *Journal of Chemical Physics* 2012,**136**.

83. Riedmiller M, Braun H. A Direct Adaptive Method for Faster Backpropagation Learning - the Rprop Algorithm. *1993 Ieee International Conference on Neural Networks, Vols 1-3* 1993:586-591.

84. Nissen S. Implementation of a Fast Artificial Neural Network Library (fann). In: Department of Computer Science University of Copenhagen (DIKU); 2003.

85. Prechelt L. Early Stopping - but when? In: Chapter 2, *Neural Networks: Tricks of the Trade*; 1997.

86. Sheinerman FB, Norel R, Honig B. Electrostatic aspects of protein-protein interactions. *Current Opinion in Structural Biology* 2000,**10**:153-159.

87. Kaczor AA, Selent J, Sanz F, Pastor M. Modeling Complexes of Transmembrane Proteins: Systematic Analysis of ProteinProtein Docking Tools. *Molecular Informatics* 2013,**32**:717-733.

88. Joachims T. Training linear SVMs in linear time. In: *ACM Conference on Knowledge Discovery and Data Mining (KDD)*; 2006. pp. 217-226.

89. Collobert R, Bengio S. Links between perceptrons, MLPs and SVMs. In: *Proceedings of the twenty-first international conference on Machine learning*: ACM; 2004. pp. 23.

90. Bengio Y, LeCun Y. Scaling learning algorithms towards AI. *Large-Scale Kernel Machines* 2007,**34**:1-41.

91. Viswanath S, Dominguez L, Foster LS, Straub JE, Elber R. An extension of a docking algorithm to membranes and applications to dimers of amyloid precursor protein. *in preparation* 2014.

92. Selent J, Kaczor AA. Oligomerization of G Protein-Coupled Receptors: Computational Methods. *Current Medicinal Chemistry* 2011,**18**:4588-4605.

93. Dominguez C, Boelens R, Bonvin AMJJ. HADDOCK: A protein-protein docking approach based on biochemical or biophysical information. *Journal of the American Chemical Society* 2003,**125**:1731-1737.

94. Comeau SR, Camacho CJ. Predicting oligomeric assemblies: N-mers a primer. *Journal of Structural Biology* 2005,**150**:233-244.

95. Cosconati S, Marinelli L, Lavecchia A, Novellino E. Characterizing the 1,4-dihydropyridines binding interactions in the L-type Ca2+ channel: Model construction and docking calculations. *Journal of Medicinal Chemistry* 2007,**50**:1504-1513.

96. Simon AC, Simpson PJ, Goldstone RM, Krysztofinska EM, Murray JW, High S, *et al*. Structure of the Sgt2/Get5 complex provides insights into GET-mediated targeting of tail-anchored membrane proteins. *Proceedings of the National Academy of Sciences of the United States of America* 2013,**110**:1327-1332.

97. Casciari D, Seeber M, Fanelli F. Quaternary structure predictions of transmembrane proteins starting from the monomer: a docking-based approach. *BMC bioinformatics* 2006,**7**:340.

98. Miao YM, Cross TA. Solid state NMR and protein-protein interactions in membranes. *Current Opinion in Structural Biology* 2013,**23**:919-928.

99. MacCallum JL, Bennett WFD, Tieleman DP. Partitioning of amino acid side chains into lipid bilayers: Results from computer simulations and comparison to experiment. *Journal of General Physiology* 2007,**129**:371-377.

100. Miyashita N, Straub JE, Thirumalai D, Sugita Y. Transmembrane structures of amyloid precursor protein dimer predicted by replica-exchange molecular dynamics simulations. *J Am Chem Soc* 2009,**131**:3438-3439.

101. Dominguez L, Meredith SC, Straub JE, Thirumalai D. Transmembrane Fragment Structures of Amyloid Precursor Protein Depend on Membrane Surface Curvature. *Journal of the American Chemical Society* 2014,**136**:854-857.

102. MacCallum JL, Bennett WF, Tieleman DP. Partitioning of amino acid side chains into lipid bilayers: results from computer simulations and comparison to experiment. *J Gen Physiol* 2007,**129**:371-377.

103. Kabsch W, Sander C. Dictionary of Protein Secondary Structure - Pattern-Recognition of Hydrogen-Bonded and Geometrical Features. *Biopolymers* 1983,**22**:2577-2637.

104. Tusnady GE, Dosztanyi Z, Simon I. TMDET: web server for detecting transmembrane regions of proteins by using their 3D coordinates. *Bioinformatics* 2005,**21**:1276-1277.

105. Tusnady GE, Dosztanyi Z, Simon I. PDB_TM: selection and membrane localization of transmembrane proteins in the protein data bank. *Nucleic Acids Research* 2005,**33**:D275-D278.

106. Jayasinghe S, Hristova K, White SH. MPtopo: A database of membrane protein topology. *Protein Science* 2001,**10**:455-458.

107. Glaser F, Steinberg DM, Vakser IA, Ben-Tal N. Residue frequencies and pairing preferences at protein-protein interfaces. *Proteins-Structure Function and Genetics* 2001,**43**:89-102.

108. Zhang C, Vasmatzis G, Cornette JL, DeLisi C. Determination of atomic desolvation energies from the structures of crystallized proteins. *Journal of Molecular Biology* 1997,**267**:707-726.

109. Fleishman SJ, Whitehead TA, Ekiert DC, Dreyfus C, Corn JE, Strauch EM, *et al*. Computational design of proteins targeting the conserved stem region of influenza hemagglutinin. *Science* 2011,**332**:816-821.

110. Bras JLA, Correia MAS, Romao MJ, Prates JAM, Fontes CMGA, Najmudin S. Purification, crystallization and preliminary X-ray characterization of the pentamodular arabinoxylanase CtXyl5A from Clostridium thermocellum. *Acta Crystallographica Section F-Structural Biology and Crystallization Communications* 2011,**67**:833-836.

111. Guellouz A, Valerio-Lepiniec M, Urvoas A, Chevrel A, Graille M, Fourati-Kammoun Z, *et al*. Selection of Specific Protein Binders for Pre-Defined Targets from an Optimized Library of Artificial Helicoidal Repeat Proteins (alphaRep). *PLoS One* 2013,**8**.

112. Rodrigues JP, Bonvin AM. Integrative computational modeling of protein interactions. *The FEBS journal* 2014.

113. Viswanath S, Ravikant DV, Elber R. DOCK/PIERR: Web Server for Structure Prediction of Protein-Protein Complexes. *Methods in molecular biology* 2014,**1137**:199-207.

114. Hunt ME, Modi CK, Aglyamova GV, Ravikant DV, Meyer E, Matz MV. Multi-domain GFP-like proteins from two species of marine hydrozoans. *Photochem Photobiol Sci* 2012,**11**:637-644.

115. Kausar S, Asif M, Bibi N, Rashid S. Comparative Molecular Docking Analysis of Cytoplasmic Dynein Light Chain DYNLL1 with Pilin to Explore the Molecular Mechanism of Pathogenesis Caused by Pseudomonas aeruginosa PAO. *PLoS One* 2013,**8**.

116. Viswanath S, Kreuzer SM, Cardenas AE, Elber R. Analyzing milestoning networks for molecular kinetics: Definitions, algorithms, and examples. *Journal of Chemical Physics* 2013,**139**.

117. Chodera JD, Singhal N, Pande VS, Dill KA, Swope WC. Automatic discovery of metastable states for the construction of Markov models of macromolecular conformational dynamics. *Journal of Chemical Physics* 2007,**126**.

118. Schutte C, Noe F, Lu JF, Sarich M, Vanden-Eijnden E. Markov state models based on milestoning. *Journal of Chemical Physics* 2011,**134**.

119. Berezhkovskii A, Hummer G, Szabo A. Reactive flux and folding pathways in network models of coarse-grained protein dynamics. *Journal of Chemical Physics* 2009,**130**.

120. Metzner P, Schutte C, Vanden-Eijnden E. Transition Path Theory for Markov Jump Processes. *Multiscale Modeling & Simulation* 2009,**7**:1192-1219.

121. Cardenas AE, Jas GS, DeLeon KY, Hegefeld WA, Kuczera K, Elber R. Unassisted Transport of N-Acetyl-L-tryptophanamide through Membrane: Experiment and Simulation of Kinetics. *Journal of Physical Chemistry B* 2012,**116**:2739-2750.

122. Elber R. A milestoning study of the kinetics of an allosteric transition: Atomically detailed simulations of deoxy Scapharca hemoglobin. *Biophysical Journal* 2007,**92**:L85-L87.

123. Elber R, West A. Atomically detailed simulation of the recovery stroke in myosin by Milestoning. *Proceedings of the National Academy of Sciences of the United States of America* 2010,**107**:5001-5005.

124. West AMA, Elber R, Shalloway D. Extending molecular dynamics time scales with milestoning: Example of complex kinetics in a solvated peptide. *Journal of Chemical Physics* 2007,**126**:145104.

125. E WN, Vanden-Eijnden E. Transition-Path Theory and Path-Finding Algorithms for the Study of Rare Events. In: *Annual Review of Physical Chemistry, Vol 61*; 2010. pp. 391-420.

126. Berkowitz M, Morgan JD, Mccammon JA, Northrup SH. Diffusion-Controlled Reactions - a Variational Formula for the Optimum Reaction Coordinate. *Journal of Chemical Physics* 1983,**79**:5563-5565.

127. Huo SH, Straub JE. The MaxFlux algorithm for calculating variationally optimized reaction paths for conformational transitions in many body systems at finite temperature. *Journal of Chemical Physics* 1997,**107**:5000-5006.

128. Zhao RJ, Shen JF, Skeel RD. Maximum Flux Transition Paths of Conformational Change. *Journal of Chemical Theory and Computation* 2010,**6**:2411-2423.

129. Czerminski R, Elber R. Reaction path study of confomrational transitions in flexible systems - applications to peptides. *Journal of Chemical Physics* 1990,**92**:5580-5601.

130. Czerminski R, Elber R. Reaction-path study of conformational transitions and helix formation in a tetrapeptide. *Proceedings of the National Academy of Sciences of the United States of America* 1989,**86**:6963-6967.

131. Shalloway D, Faradjian AK. Efficient computation of the first passage time distribution of the generalized master equation by steady-state relaxation. *Journal of Chemical Physics* 2006,**124**.

132. Hu TC. The Maximum Capacity Route Problem. *Operations Research* 1961,**9**:898-900.

133. Pollack M. The Maximum Capacity through a Network. *Operations Research* 1960,**8**:733-736.

134. Vassilevska V. Nondecreasing Paths in a Weighted Graph or: How to Optimally Read a Train Schedule. *Proceedings of the Nineteenth Annual Acm-Siam Symposium on Discrete Algorithms* 2008:465-472.

135. Vassilevska V, Williams R, Yuster R. All-Pairs Bottleneck Paths For General Graphs in Truly Sub-Cubic Time. *Stoc 07: Proceedings of the 39th Annual Acm Symposium on Theory of Computing* 2007:585-589.

136. Kreuzer S, Elber R. Catch bond-like kineics of helix cracking: Network analysis by molecular dynamics and milestoning. *Journal of Chemical Physics* 2013,**139**.

137. Kreuzer SM, Elber R, Moon TJ. Early Events in Helix Unfolding under External Forces: A Milestoning Analysis. *Journal of Physical Chemistry B* 2012,**116**:8662-8691.

138. Dijkstra EW. A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik* 1959,**1**:3.

139. Batagelj V, Mrvar A. Pajek: Program for Large Network Analysis. *Connections* 1998,**21**:47-57.

140. Cardenas AE, Elber R. Computational study of peptide permeation through membrane: searching for hidden slow variables. *Molecular Physics* 2013,**111**:3565-3578.

141. Elber R, Roitberg A, Simmerling C, Goldstein R, Li HY, Verkhivker G, *et al*. Moil a program for simulations of macrmolecules. *Computer Physics Communications* 1995,**91**:159-189.

142. Kozakov D, Schueler-Furman O, Vajda S. Discrimination of near-native structures in protein-protein docking by testing the stability of local minima. *Proteins-Structure Function and Bioinformatics* 2008,**72**:993-1004.

143. Torchala M, Moal IH, Chaleil RAG, Agius R, Bates PA. A Markov-chain model description of binding funnels to enhance the ranking of docked solutions. *Proteins-Structure Function and Bioinformatics* 2013,**81**:2143-2149.

144. Andreani J, Faure G, Guerois R. InterEvScore: a novel coarse-grained interface scoring function using a multi-body statistical potential coupled to evolution. *Bioinformatics* 2013,**29**:1742-1749.

145. Moal IH, Moretti R, Baker D, Fernandez-Recio J. Scoring functions for protein-protein interactions. *Current Opinion in Structural Biology* 2013,**23**:862-867.

146. Khashan R, Zheng WF, Tropsha A. Scoring protein interaction decoys using exposed residues (SPIDER): A novel multibody interaction scoring function based on frequent geometric patterns of interfacial residues. *Proteins-Structure Function and Bioinformatics* 2012,**80**:2207-2217.

147. Kastritis PL, Bonvin AMJJ. Molecular origins of binding affinity: seeking the Archimedean point. *Current Opinion in Structural Biology* 2013,**23**:868-877.

148. Kastritis PL, Bonvin AMJJ. Are Scoring Functions in Protein-Protein Docking Ready to Predict Interactomes? Clues from a Novel Binding Affinity Benchmark (vol 9, pg 2216, 2010). *Journal of Proteome Research* 2011,**10**:921-922.

149. Kastritis PL, Moal IH, Hwang H, Weng ZP, Bates PA, Bonvin AMJJ, *et al*. A structure-based benchmark for protein-protein binding affinity. *Protein Science* 2011,**20**:482-491.

150. Popov P, Ritchie DW, Grudinin S. DockTrina: Docking triangular protein trimers. *Proteins-Structure Function and Bioinformatics* 2014,**82**:34-44.

151. Inbar Y, Benyamini H, Nussinov R, Wolfson HJ. Protein structure prediction via combinatorial assembly of sub-structural units. *Bioinformatics* 2003,**19**:i158-i168.