idDock+:Integrating Machine Learning in Probabilistic Search for Protein-protein Docking

Irina Hashmi, Department of Computer Science, George Mason University, VA, USA
Amarda Shehu[1], Department of Computer Science, Department of Bioengineering, and School of Systems Biology, George Mason University, VA, USA

Predicting the three-dimensional functional structures of protein dimers, a problem known as protein-protein docking, is key to understanding molecular interactions in the cell. Docking is a computationally-challenging problem due to the diversity of protein-protein interactions and the high dimensionality of the associated configuration space. Many methods exist to address this problem, and the majority draw configurations systematically or at random from the configuration space and then rank them with energy scoring functions. In addition to the high dimensionality of the configuration space, the inaccuracy of current scoring functions presents major challenges. In particular, evidence is growing that optimization of a scoring function is an effective technique only once the drawn configuration is sufficiently similar to the functional dimeric structure. Therefore, in this paper we present a method that employs optimization of a sophisticated energy function, FoldX, only to locally improve a promising configuration. The main question of how promising configurations are identified is addressed through a machine learning method, an ensemble learner trained a priori on an extensive dataset of functionally-diverse protein dimers. To deal with a potentially vast configuration space, a probabilistic search algorithm operates on top of the learner, feeding to it configurations drawn at random. We refer to our algorithm as idDock+ for informative-driven Docking. idDock+ is tested on 15 protein dimers of different sizes and functional classes. Analysis shows that on all systems idDock+ finds a near-native structure and is comparable in accuracy to other state-of-the-art methods. idDock+ represents one of the first highly-efficient hybrid methods that combines fast machine learning models with demanding optimization of sophisticated energy scoring functions. Our results indicate that this is a promising direction to improve both efficiency and accuracy in protein-protein docking.

Categories and Subject Descriptors: G.3 [**Probability and Statistics**] Probabilistic algorithms; J.3 [**Life and Medical Sciences**] Biology and genetics

General Terms: Algorithms

Additional Key Words and Phrases: protein-protein docking; probabilistic search; informatics-driven; ensemble learning; stochastic optimization

[1]Corresponding author: amarda@gmu.edu

1. INTRODUCTION

Proteins are involved in the majority of biochemical processes in the cell. They assume specific three-dimensional structures to bind with other molecules and perform their cellular tasks. Many genetic diseases are caused by anomalous protein-protein interactions [Ritchie 2008]. In particular, protein dimers are ubiquitous in the cell. Predicting the three-dimensional functional structures of protein dimers, a problem commonly known as protein-protein docking, is fundamental to our understanding of the basis of biological function.

Experimental and computational methods are devoted to addressing the protein-protein docking problem. Several challenges regarding labor, time demands, desired resolution, and assembly size limit broad application of experimental methods [Dominguez et al. 2003; Kilambi et al. 2014] . Computational methods can in principle assist wet-laboratory investigations, but they also face challenges.

By now, protein-protein docking is recognized to be a computationally hard problem [Moitessier et al. 2009] for two major reasons: (i) There is great diversity among types of protein-protein interactions. Even proteins central to a known biological process may interact with other proteins not involved in that particular process. There seem to be no universal rules to predicting functional interactions. (ii) The configuration search space is vast. The configuration space of a dimer comprised of two protein chains A and B has $N_A + N_B + 6$ dimensions, where the N_A and N_B are the parameters to represent the internal structures of units A and B participating in the dimer, respectively, and 6 is the number of parameters needed to specify the rigid-body transformations that spatially arrange one mobile unit on top of the other reference unit.

To reduce the complexity of the configuration space, the parameters N_A and N_B can be removed from consideration and the problem becomes 6 dimensional; effectively, the problem is simplified to the rigid-body protein-protein docking, where the structures of the involved units are assumed to remain unchanged upon the formation of the dimer. Most of the current docking methods primarily address rigid-body docking and then in post-processing stages modify docked configurations by exploring limited internal fluctuations of each of the docked units. However, even the rigid-body docking problem remains non-enumerable due to the continuous space of rigid-body transformations.

Current computational methods employ two main schemes to guide and thus constrain the exploration of the configuration space even in rigid-body docking. Geometry-driven methods, such as SymmDock [Duhovny-Schneidman et al. 2005] and Combdock [Inbar et al. 2005], exploit analysis of surface curvature and use geometric complementary to filter out possibly irrelevant docked configurations. Short improvements of found configurations via optimization of a selected function measuring interaction energy are then carried out in a postprocessing stage. Geometry-driven methods are computationally efficient but largely recognized as less accurate than energy-driven methods [Ritchie 2008]. The latter forego this prior stage and instead conduct exploration of the configuration space in the context of optimization of a selected energy function. Current state-of-the-art methods that fall in this category include ClusPro [Comeau et al. 2004], RosettaDock [Lyskov and Gray 2008], SKE-DOCK [Terashi et al. 2007], GRAMM-X [Tovchigrechko and Vakser 2006], PIPER [], and others [Huang 2014]. State-of-the-art methods based on fast Fourier Transforms include ZDOCK [Chen et al. 2003], F2Dock [Bajaj et al. 2011], PIPER [Kozakov et al. 2006], and Hex [Macindoe et al. 2010].

Energy-driven methods are typically more computationally demanding than geometry-driven ones. Sufficient time needs to be allocated to explore the breadth of the dimeric configuration space in order not to miss configurations near the native structure. In addition, optimizations of molecular energy functions are expensive mainly due to the pairwise interaction terms in these functions. In addition to the computational demands, energy-driven methods may miss the native structure entirely. If the exploration does not draw a configuration sufficiently similar to the native structure, energy opti-

mization, which is essentially a local improvement technique, is not guaranteed to make sufficient structural improvements and reach the native structure. In many cases, the optimization itself may modify the configuration away from the native structure. Most energy functions linearly combine different weighted energy terms, where the weights are optimized for a particular dataset. For these reasons, even energy-driven methods can lead the search towards non-native structures [Halperin et al. 2002; Lensink and Wodak 2009]. A recent summary of the Critical Assessment of Predicted Interactions (CAPRI) results shows that, often, only $30 - 58\%$ of the target systems can be correctly predicted [Lensink and Wodak 2010].

One of the main lessons in computational protein-protein docking is that optimization of an energy function is at best a local technique to improve a configuration that is in the vicinity of the native structure in the configuration space. The question remains how to identify such configurations given unreliable energy functions. For this reason, a group of methods in protein-protein docking forego the traditional setting of predicting a native structure but instead focus on learning what makes true native interaction interfaces [Shoemaker and Panchenko 2007; Bordner and Gorin 2007; Zhu et al. 2006; Li et al. 2008; Li and Kihara 2012; Yan et al. 2004].

Given the recognized difficulties with both geometry-driven and energy-driven methods, and the growing knowledge on features that comprise native interaction interfaces, a third group of methods combine search and a priori knowledge, whether from experiment or from prior learning, to improve accuracy in protein-protein docking. These methods, which can be considered hybrid, include [Kanamori et al. 2007; Hashmi et al. 2011; 2012; Dominguez et al. 2003]. While work in [Dominguez et al. 2003] incorporates distance constraints obtained from the wet laboratory, work in [Kanamori et al. 2007; Hashmi et al. 2011; 2012] ranks configurations based on known features of native interaction interfaces, such as evolutionary conservation, prior to energy optimization.

In this paper, we advance work on hybrid methods for rigid-body protein-protein docking. Given the growing accuracy of machine learning methods in correctly classifying native from non-native interaction interfaces [Zhu et al. 2006], we present a novel algorithm that integrates machine learning models in its probabilistic search of the dimeric configuration space. From now on, we refer to this algorithm as idDock+ for informatics-driven Docking. A prior proof-of-concept demonstrating the promise of integrating a simple supervised learning model in probabilistic search for protein-protein docking was presented by us in [Hashmi and Shehu 2013b]. In this paper, we present a more mature and powerful algorithm, idDock+, which integrates an ensemble learner trained a priori on an extensive and carefully constructed dataset of functionally-diverse dimers in a probabilistic search algorithm.

idDock+ draws dimeric configurations at random, employing an evolutionary algorithm known as basin hopping (BH). Rather than immediately handing off drawn configurations for optimization to a sophisticated but computationally-demanding energy function, FoldX, idDock+ first evaluates configurations through the ensemble learner. Only configurations classified as native from the learner are then sent to the demanding optimization procedure. This hierarchy allows idDock+ to be more computationally-efficient than a typical energy-driven method, as the learner is a rapid filter of non-native configurations. It is also worth noting that idDock+ employs techniques from geometry-driven docking to rapidly draw docked configurations. Moreover, as the results presented here demonstrate, employment of the learner allows idDock+ to retain accuracy and make better use of optimization by applying it only on configurations deemed near the native structure. In many cases, our comparative analysis shows that idDock+ offers improvements over current state-of-the-art methods.

idDock+ represents one of the first hybrid methods that combines fast machine learning models with demanding optimization of sophisticated scoring functions for protein-protein docking. Results presented here indicate that this is a promising direction to improve both efficiency and accuracy in protein-protein docking.

2. MATERIALS AND METHODS

idDock+ is a BH-based algorithm. In esence, idDock+ hops in configuration space between neighboring configurations that are local minima of some employed scoring/energy function. It makes use of repeated application of two operators, a perturbation operator and a local improvement operator.

In idDock+, docked configurations are generated in a trajectory, $C_1, C_2, \ldots, C_n$. The algorithm hops between two consecutive local minima configurations C_i and C_{i+1} through an intermediate perturbed configuration $C_{perturb,i}$. The perturbation operator takes a local minimum configuration C_i as input and applies a move to obtain a perturbed configuration $C_{\text{perturb,i}}$ and escape the current minimum C_i. In a baseline realization of the BH algorithm, the perturbed configuration $C_{\text{perturb,i}}$ would be sent to a local improvement operator that is in essence an energy minimization procedure. In idDock+, however, the perturbed configuration $C_{\text{perturbed,i}}$ is first sent for evaluation to the predictive, learned model to determine whether the configuration has a true/native interaction interface or not. The minimization procedure is only applied when the interaction interface in $C_{\text{perturb,i}}$ is labeled true; a probabilistic criterion is applied to determine whether the result of the minimization procedure should be accepted as the next local minimum configuration C_{i+1} in the growing trajectory. Otherwise, if the interaction interface in $C_{\text{perturb,i}}$ is not deemed true from the learned model or the result of the minimization procedure on $C_{\text{perturb,i}}$ fails the probabilistic test, a new perturbed configuration $C_{\text{perturb,i}}$ is attempted.

We now describe in detail the perturbation operator and local improvement operators. Both use a specific representation of docked configurations that builds on concepts and techniques from geometry-driven approaches in rigid protein-protein docking. Hence, we first describe this representation. We end this section with a description of the learned model.

2.1 Rigid-body Transformations to Obtain Docked Configurations in idDock+

One unit, chosen arbitrarily, say A remains static and is the reference unit. The other unit, B, is moving, and rigid-body transformations in SE(3) are applied onto B to obtain docked configurations. A transformation in SE(3) can be represented as $T < r_x, r_y, r_z, t_x, t_y, t_z >$ where r_x, r_y and r_z are the rotation and t_x, t_y and t_z are the translation components along the x, y and z axes, respectively.

Instead of drawing transformations from SE(3) at random, idDock+ borrows from geometry-driven methods and directly calculates transformations that align geometrically-complementary regions of the moving unit onto the reference unit. In order to find geometrically-complementary regions on the molecular surface of a protein unit, idDock+ makes use of two different molecular surface representations, first the Connolly representation [Connolly 1983] and then the Shuo representation [Lin et al. 1994]. Details on these representations are provided elsewhere [Fischer et al. 2005]. In essence, the result of employing them is that a list of points deemed Shuo critical points are obtained. These points cover key locations of a molecular surface. The Shuo critical points are categorized as *caps*, *pits*, and *belts* which correspond to convex, concave, and saddle regions on a molecular surface, respectively.

Once the molecular surface of each unit participating in a dimer is discretized and represented in terms of a list of Shuo critical points, transformations in SE(3) can be easily obtained. First, three critical points are sampled from a molecular surface to define a triangle. The first point for a triangle is chosen arbitrarily. The other two are sampled from the compiled list of critical points so that they lie within a neighborhood of the first. The idea is to construct triangles that cover neither too small nor too large regions on a molecular surface. The distance constraints used for such neighborhoods are now well-established and described in [Fischer et al. 2005]. In addition, the points to define a triangle are sampled so that the triangles represent concave or convex regions on a molecular surface. Once such triangles are easily sampled from a list of critical points on a unit, then a transformation in SE(3) can be computed directly by trying to align a triangle T_B sampled from the Shuo critical points on unit

B onto a geometrically-complementary triangle T_A sampled from the Shuo critical points on unit A. The transformation is unique, as it tries to align a local frame corresponding to T_A onto another local frame corresponding to T_B in three dimensions. Since many triangles can be sampled from each unit, different transformations in SE(3) can be obtained, thus resulting in different docked configurations.

2.2 Perturbation Operator in idDock+

idDock+ is initialized with a docked configuration C_1 that is a local minimum. In essence, a transformation is obtained as above, the resulting configuration is subjected to the learner, and then to the local improvement operator if deemed native-like by the learner. This process repeats until a C_1 is obtained that has passed the learner and is a local minimum. In general, once a configuration C_i in the current growing trajectory has been obtained that is a local minimum, C_i is subjected to the perturbation operator.

A local minimum configuration C_i is associated with two triangles T_A and T_B as described above (as it is the result of aligning two geometrically-complementary triangles, with minor modifications from the local improvement operator). A new configuration $C_{\text{perturb,i}}$ is then sought that is neither too similar to nor too different from C_i. To achieve this, a triangle $T_A^{'}$ is sampled from the Shuo critical points over unit A in the vicinity (a d-radius neighborhood) of T_A and another, $T_B^{'}$, is sampled from the Shuo critical points over unit B in the vicinity (a d-radius neighborhood) of T_B. This process is repeated until $T_A^{'}$ and $T_B^{'}$ are geometrically-complementary. Once these triangles are obtained, a new transformation can then be defined that aligns them and yields a new docked configuration, $C_{\text{pertrurb,i}}$.

In previous work, we have focused on analyzing BH-based algorithms in various structure prediction applications [Olson et al. 2012]. Our analysis shows that it is important that the new triangles be in the vicinity of those in C_i; that is, the selection of d is crucial to preserve the adjacency relationship between C_i and $C_{pertub,i}$. A random or a very large values of d will effectively make a BH algorithm behave like a random restart. On the other hand, too small values of d risk $C_{\text{perturb,i}}$ being in the same minimum as C_i; the subsequent local improvement operator would then effectively reproduce C_i and not yield a new minimum. Hence, here we set d to be 5Å, which we have demonstrated to be effective in our prior applications of BH for protein-protein docking [Hashmi and Shehu 2012; 2013a]. A detailed prior analysis on the impact of d can be found in [Hashmi and Shehu 2013a].

Unlike prior realizations of BH for protein-protein docking, idDock+ does not send $C_{\text{perturb,i}}$ to the local improvement operator for mapping to a nearby local minimum. Instead, the perturbed configuration is sent for evaluation to a machine learning model. If the model predicts the interaction interface in $C_{\text{perturb,i}}$ to be false, the perturbation operator is applied anew on C_i to obtain a new perturbed configuration $C_{\text{perturb,i}}$. Otherwise, $C_{\text{perturb,i}}$ is sent to the local improvement operator.

2.3 Local Improvement Operator in idDock+

The local improvement operator modifies $C_{pertub,i}$ to project it to a nearby local minimum. The operator is an energetic refinement/minimization procedure, which can be computationally costly when employing sophisticated energy functions, as we do here. However, one of the main advantages in idDock+ is that the operator is applied judiciously, only on perturbed configurations deemed promising from the learned model.

Given a perturbed configuration $C_{\text{perturb,i}}$ that has passed the learned model, the local improvement operator proceeds as follows. A maximum of $m = 50$ moves are attempted. In each move, an axis is selected uniformly at random from the three xy, z axes. In addition, an angle θ is sampled uniformly at random in $[-5°, 5°]$. Rotation by θ along the chosen axis is then applied onto the previous configuration (starting with $C_{\text{perturb,i}}$) to get a new configuration. The procedure terminates if $k = 10$ consecutive

moves fail to lower the interaction energy. These parameters have been carefully tuned and adapted from previous work by us and others [Hashmi and Shehu 2013a; Kanamori et al. 2007].

The local improvement operator seeks to lower the FoldX interaction energy [Schymkowitz et al. 2005]. FoldX is specially designed for protein assemblies, and its terms are weighted using data obtained from protein engineering experiments. The terms include solvation energy, van der Waals potential, hydrogen bond potential, electrostatic potential, entropic and clash penalty.

The result of the local improvement operator is a low-energy configuration that is a representative of a local minimum; this is a working definition of a local minimum, as no exact minimization procedure can be afforded to truly obtain the bottom of a local minimum. However, such a working definition is often sufficient when employing non-exact optimization techniques. It is also worth noting that the resulting configuration is not immediately added as C_{i+1} to the growing trajectory of sampled local minima. Instead, a probabilistic criterion is employed, per the Metropolis criterion. The criterion is based on the energy difference between two consecutive local minima, $\delta E = E(C_{i+1} - E(C_i))$. A probability $e^{-\delta E \cdot \beta}$ is calculated, where β is a scaling parameter. If C_{i+1} passes this criterion, it is added to the trajectory. Otherwise, the process begins anew, with another perturbation operator attempted on C_i. We have selected β to be 0.3kcal/mol^{-1} here, which indicates that an energy increase of 2kcal/mol will be accepted with a probability of 0.55. This effectively means that an immediate energetic increase of 2kcal/mol will be accepted slightly more than half the time.

The idea behind employing a probabilistic criterion is to allow idDock+ to accept some higher local minima to escape deep basins and enhance its exploration capability. In essence, the trajectory is a Metropolis Monte Carlo trajectory of consecutive local minima.

2.4 Machine Learning Model to Evaluate Perturbed Configurations in idDock+

In a proof-of-concept demonstration of the ability to integrate a simple machine learning model in a BH-based algorithm, we constructed a decision tree model on a small dataset of protein dimers in earlier work [Hashmi and Shehu 2013b]. In this work, we take a more systematic approach to first carefully construct a large and diverse training dataset and then identify an optimal learning model among many state-of-the-art ones in machine learning.

We first describe in detail the training dataset, the representation we employ to train machine learning models, and then the various models we pitch against one another to determine an optimal one for integration in idDock+.

2.4.1 *Training Dataset.* The training dataset consists of 2062 protein-protein data. The positive dataset consists of 1071 true/native interaction interfaces found on experimentally-obtained assemblies extracted from a PDBbind protein-protein dataset [Wang et al. 2005]. The PDBbind database is obtained from the Protein Data Bank (PDB) [Berman et al. 2000] by scanning for protein-protein complexes. According to PDBBind's definition, a protein unit is said to be in contact with another unit if there are at least 10 interaction residues on each of the chains. The chain length for each protein is longer than 20 residues.

The negative datasets of 991 instances consists of three smaller sets. The first is constructed by randomizing the positive dataset. Units selected randomly from two different complexes from the positive dataset are docked with a random rigid-body transformation. A total of 456 dimers are obtained by repeated randomization. The second set consists of 76 crystal packing structures obtained from [Zhu et al. 2006]. Crystal packing structures are generated due to the crystallization process in X-ray structure determination techniques. Hence, these structures are not biologically relevant. The third set consists of 459 dimeric structures which are $5{-}12\mathring{A}$ away in lRMSD from the native. This third set is generated by pyDOCK [Cheng et al. 2007].

2.4.2 *Feature vector.* Each training instance is converted into a vector of 7 entries, referred to as a feature vector, so that machine learning models can discriminate between positive and negative instances in a 7-dimensional spce.

An interaction interface is first defined over an instance. As in [Zhu et al. 2006], an amino acid is said to be on the interface if the solvent accessible surface area (SASA) decreases by $> 1\text{Å}^2$ upon the formation of a complex. The interaction interface consists of the amino acids so determined to be on the interface from each of the units and is converted to a feature vector. The features chosen here to represent an interaction interface are interface area, interface area ratio, 4 composition-based features, and conservation score.

The first entry of a feature vector is the interface area which is defined as in [Zhu et al. 2006]:

$$\text{InterfaceArea}_{\text{A+B}} = 0.5 \cdot (\text{SASA}_{\text{A}} + \text{SASA}_{\text{B}} - \text{SASA}_{\text{A+B}}), \tag{1}$$

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

The second entry is the interface area ratio obtained defined as in [Zhu et al. 2006]:

$$\frac{\text{InterfaceArea}_{\text{A+B}}}{min(\text{SASA}_{\text{A}}, \text{SASA}_{\text{B}})}, \tag{2}$$

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

$$\frac{NAA_k^{interface}}{NAA_k^{surface}}, \tag{3}$$

where, $NAA_k^{interface}$ is the number of amino acids of type k on an interaction interface, and $NAA_k^{surface}$ is the number of amino acids of the same type k the surface. As before, an amino acid is determined to be on the surface if it loses about $< 1\text{Å}^2$ in its SASA upon the formation of the complex [Zhu et al. 2006]. Type *k* includes hydrophobic, hydrophilic, basic, and acidic.

The last entry in the feature vector is the average conservation score of the interaction interface, measured over conservation scores of amino acids on the interface. The conservation score of an amino acid is measured using the iterative Joint Evolutionary Trees (iJET) algorithm [Engelen et al. 2009]. iJet is based on multiple sequence alignment and associates a conservation score with an amino acid from 0 to 1, where 0 is least conserved, and 1 is most conserved.

2.4.3 *Analysis of Interaction Properties.* We have performed a statistical analysis on these 7 features over positive and negative instances. Figure 1 shows the distributions over the negative and positive datasets. The x axis shows the value range for a particular feature, while the y axis represents the percentage of the structures that fall within that particular range.

Figure 1 shows that the interface area of positive/native instances is overall higher than that of most negative instances, while no conclusive observations can be made regarding the interface area ratio. Native instances tend to have more hydrophobic atoms than non-native ones, and a similar observation can be drawn regarding ratio of hydrophobic amino acids. On the other hand, acidic and basic compositions do not seem to discriminate between native and non-native instances. One does observe more native than non-native instances have average conservation scores > 0.6.

2.4.4 *Identification of Optimal Classification Model.* To summarize, the interaction interface in a training instance is first computed, and the interface is converted into a 7-entry vector as described

Fig. 1: Figure shows the distribution of the first $3/7$ features/interaction properties computed over positive/native and negative/non-native instances in bar diagrams. The x axis represents the value range for the properties, and the y axis shows the percentage of instances that fall within a particular value range. Positive and negative instances are shown in white and diagonal lined bars, respectively .

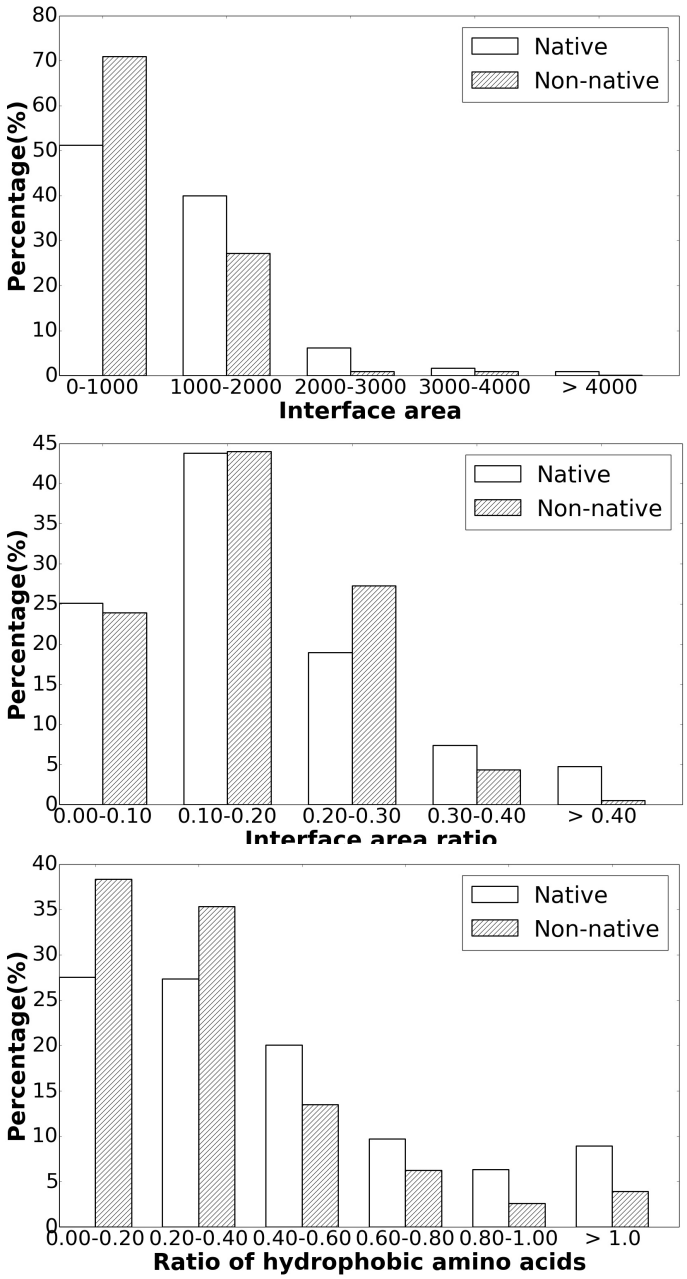


Fig. 1: (Continued) Figure shows the distribution of the other $3/7$ features.

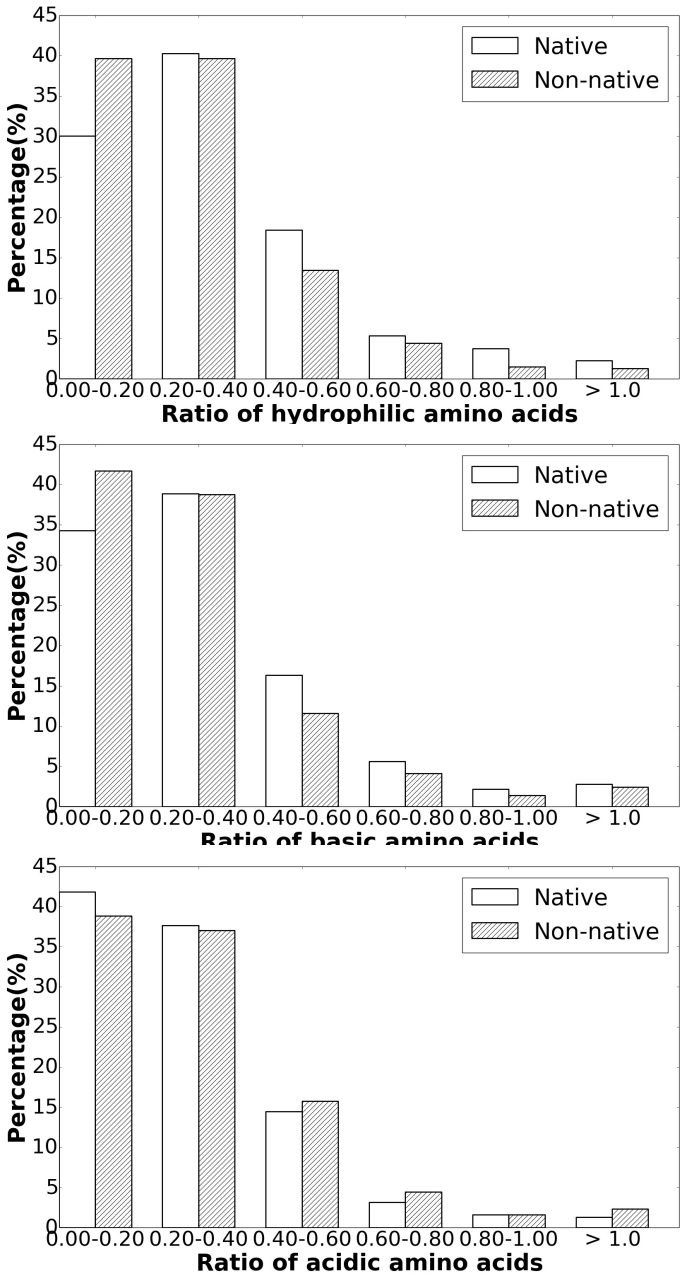
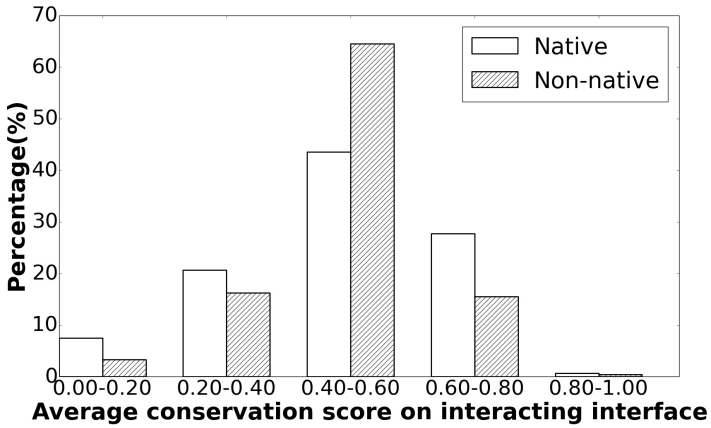


Fig. 1: (Continued) Figure shows the distribution of the last of the 7 features.

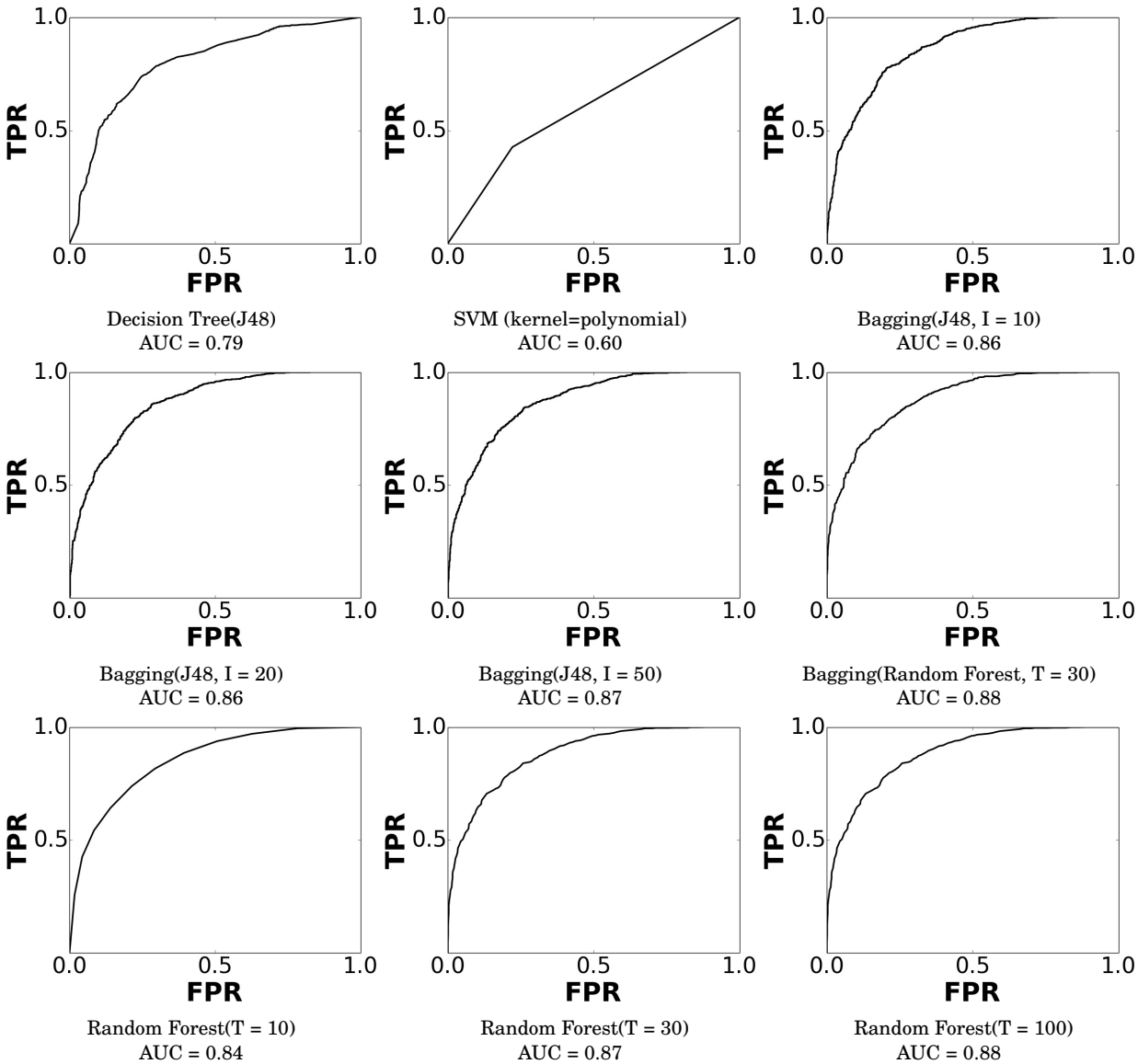


above. Various classification models available in the weka software package [Hall et al. 2009] are then trained, and their performance is recorded in a 10-fold validation setting for the purpose of comparison.

The models trained and compared here are entropy-based Decision Tree (J48 implementation in weka), Random Forest, Bagging, and Support Vector machines. The performance of each trained model is measured in terms of F-measure, precision, recall and area under Receiver Operating Curve (ROC) curves. We show here the ROC curves of each trained model in Figure 2. An ROC curve is generated by plotting the true positive rate against the false positive rate at different threshold settings. An area under the curve (AUC) of 1 represents a perfect prediction, while an AUC of 0.5 represents a random prediction.

As shown in Figure 2, the J48 tree and SVM with polynomial kernel perform worse among all models. It is worth noting parameters of each model have been tuned in order to get the best performance from each. The performance of bagging and random forest tree models grow as the number of iterations I and number of trees T increase. However, model complexity for each grows, as well, and raises the possibility of model overfitting. Therefore, after the detailed comparison in terms of performance, timing, and model complexity, we have chosen bagging with $I = 10$ and J48 as base classifiers for the learner integrated in idDock+. The AUC of this model is 0.86 which is very much comparable with the other more complex models shown in Figure 2.

Fig. 2: ROC curves are shown for the different machine learning models. FPR stands for false positive rate and TRP for true positive rate. I and T stands for number of iterations and number of trees respectively. An area under the curve of 1.0 represents a perfect prediction while an area under the curve of 0.5 represents random guess.



3. RESULTS

idDock+ is implemented in C/C++ and run on ARGO, a research computing cluster provided by the Office of Research Computing at George Mason University. Compute nodes used for testing are Intel Xeon E5-2670 CPU with 2.6GHz base processing speed and 3.5TB of RAM. Testing is conducted on 15 known protein dimers, detailed below. On each testing system, idDock+ is run until either $10,000$ dimeric configurations are obtained or 7-days of CPU time have passed. Two different sets of analyses are conducted. First, the proximity of the configuration closest to the known native structure is reported for each system and compared to similar values reported by other state-of-the-art methods. Second, a more detailed analysis is conducted that looks at the relationship between energy and lRMSD to native, and measures values, such as rank and hit-rate to determine the likelihood of selecting a native-like configuration upon different energy-based selection mechanisms.

3.1 Performance Measurements

One of the main measurements we employ here is Root-Mean-Square-Deviation (RMSD) to the known native dimeric structure to determine the quality of a generated configuration. RMSD is widely accepted now in docking methods and is reported in units of Å. RMSD measures the average atomic displacement between two configurations x and y under comparison:

$$\sqrt{\frac{1}{N}\sum_{i=1}^{N}\|x_i - y_i\|^2} \tag{4}$$

least RMSD (lRMSD) refers to the minimum RMSD over all possible rigid-body motions of one configuration relative to the other. A value between 2 and 5Å typically indicates a configuration that is highly similar to the known native structure. We use lRMSD here not only to determine the proximity of dimeric configurations generated by idDock+ to the known native structure but also to analyze the rank at which the lowest lRMSD configuration is obtained over an energy-ascending sorted ordering of configurations.

3.2 Protein Dimers Selected for Testing

15 known dimers have been selected for testing. These dimers are chosen because they vary in size, functional classification, have been used by other docking methods to measure performance, and some have also been CAPRI targets. It is worth noting that the dimers are a testing dataset with no overlap with the training dataset used to train the ensemble learner incorporated in idDock+. The dimers are listed in Table I, where column 1 shows the PDB identifier (ID) for each of the dimers followed by the chain identifiers in brackets. The next column shows the size of each of the chains in terms of total number of atoms. The last column shows the functional classification as obtained from the PDB [Berman et al. 2000].

3.3 Comparative Analysis

Here we summarize the performance of idDock+ on each testing dimer in terms of the lowest lRMSD over all configurations to the known native structure. The same value is obtained for other state-of-the-art methods from published data or from data we have obtained by running methods available in software or web server form. Methods from other labs to which we compare idDock+ are BUDDA [Polak 2003], FTDock-pyDock [Jimenez-Garcia et al. 2013] and ClusPro [Comeau et al. 2004]. The first, BUDDA, is a geometry-driven method which exhaustively samples geometrically-complementary triangles to generate new configurations. ClusPro and FTDock-pyDock are leading energy-driven meth-

Table I. : Details are listed here for each of the 15 dimers selected for testing. CAPRI targets are marked with an asterisk(∗). The size of each system is shown in terms of the total number of atoms.

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

Table II. : Comparison of idDock+ to other state-of-the-art methods

	lowest lRMSD to Native (Å)					
PDB ID	BUDDA	pyDock	ClusPro	HopDock	idDock	idDock+
1C1Y	1.2	10.4	7.2	1.8	2.7	**1.4**
1DS6	1.2	0.8	1.8	3.4	6.6	2.9
1TX4	1.4	18.5	4.7	1.0	4.7	**2.4**
1WWW	11.4	18.2	17.2	2.2	0.9	**4.5**
1FLT	1.5	2.8	4.7	1.5	0.6	**0.4**
1IKN	2.0	16.7	20.9	4.6	2.5	**2.4**
1VCB	0.7	1.4	1.9	3.6	0.9	1.4
1VCB	1.3	22.7	1.9	1.7	1.4	**1.1**
1OHZ	1.8	7.5	3.3	2.2	0.7	**0.9**
1ZHI	25.3	23.8	24.1	3.3	2.8	**1.5**
2HQS	29.1	15.2	16.6	2.6	4.5	**2.5**
1QAV	1.4	9.6	1.7	2.6	1.7	2.4
1G4Y	0.8	26.2	1.9	4.1	2.3	5.9
1CSE	0.7	13.2	1.1	2.7	1.2	**0.5**
1G4U	1.0	27.6	16.1	5.6	6.7	**4.2**

ods. For completeness, we also compare idDock+ to some of our previous work, HopDock [Hashmi and Shehu 2013a] and idDock [Hashmi and Shehu 2013b] for comparison. In HopDock, no machine learning model is integrated in the BH-based search, and a simple in-house energy function composed of van der Waals, electrostatic, and hydrogen bond terms is used. In idDock, a simple decision tree is trained over a small dataset and integrated into the BH search.

The comparative analysis summarized in Table II indicates that idDock+ not only performs comparably with the other methods but shows better performance than at least three other methods on 11 out of the 15 dimers (these cases are highlighted in bold). This effectively makes the case for the contribution of the machine learning model in the algorithm. We also see that on 9 dimers idDock+ performs better than idDock. This reflects that the larger dataset with ensemble learning results in improved performance.

It is worth noting that in some cases, BUDDA performs better than idDock+ due to it exhaustively considering all pairs of geometrically-complementary triangles. Additionally, the energy-driven approaches are far off from the native structure in some cases. This supports our observations that only energy is not sufficient to drive a probabilistic search to the native structure.

We point out that in some cases, idDock performs better than idDock+. This is due to the fact that the training dataset in idDock is much smaller (size 139), the machine learning model is a simple decision tree, and hence there is bias for certain features over others. For instance, idDock performs better than idDock+ on the system with PDB ID 1QAV, which is a membrane protein. This is due to the fact that the simple decision tree model in idDock emphasizes more hydrophobicity than other features related to the true interface. However, a larger training dataset with an ensemble learner as the model is less susceptible to this type of bias and is expected to perform well on a diverse set of testing systems.

3.4 Selection-based Analysis

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

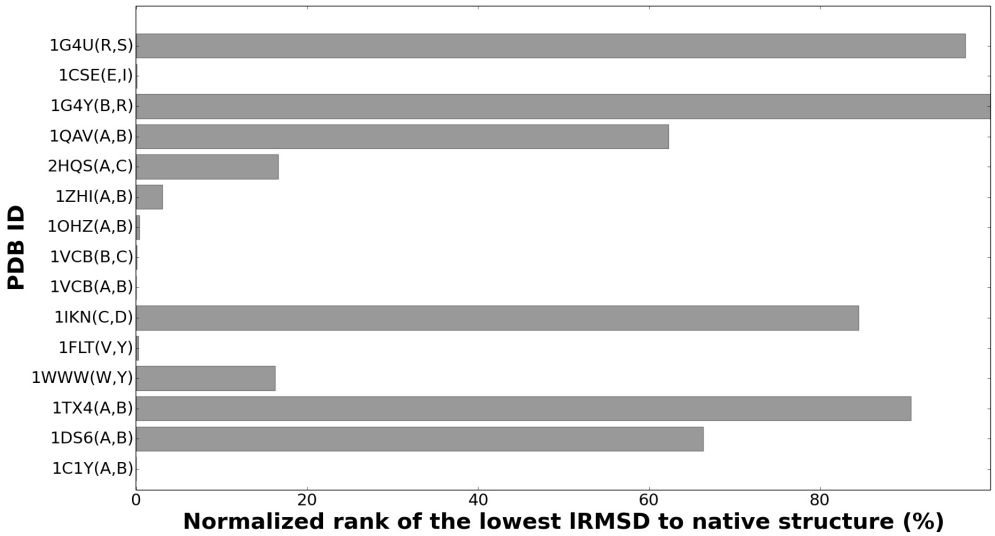
We report here the rank for each of the 15 test cases after sorting by FoldX energy. Rather than report the absolute position in this ordering of the lowest-lRMSD-to-native configuration, we report a normalized location, dividing by the total number of configurations. This allows a fair comparison among the various testing systems, since on some of the larger ones idDock+ was terminated after 7 days (see Implementation Details). Cases where a rank of 100% is reported include those where the lowest-lRMSD to the native structure is beyond a stringent threshold of 5Å.

Ranks are shown as horizontal bars in Figure 3. For about $9/15$ of the testing systems, a near-native model is found in the top 20% of energy-sorted configurations. For some systems, such as those with PDB ids 1C1Y and 1VCB, a normalized rank of 0 is reported, which means that the lowest-energy configuration is the one with the lowest lRMSD to the native structure.

We provide some more analysis as to why on some systems energy is a good selection criterion and on others it is not. Figure 4 plots FoldX energy versus lRMSD to the native structure for 6 selected dimers. Two dimers have been selected among those with very low rank; on these systems, energy is a good selection criterion. As Figure 4(a)-(b) shows, this is because there is some correlation between interaction energy and lRMSD at the lower values. Figure 4(c)-(d) shows the relationship between energy and lRMSD for two systems with higher ranks. Again, some correlation is observed. No correlation is found for the two systems shown in Figure 4(e)-(f), which are selected among those with very high rank. On these systems, energy is not a discriminant.

In Figure 5 we draw some of the lowest-lRMSD or lowest-energy configurations for selected systems. Figure 5 superimposes these configurations over the corresponding native structure and draws them using Visual Molecular Dynamics (VMD) [Humphrey et al. 1996]. The chains are drawn in different colors, and the native structure is drawn in transparent. The top and middle panel in Figure 5 shows the lowest-lRSMD configurations for the 6 selected systems. The FoldX interaction energy and respec-

Fig. 3: Normalized rank is drawn as horizontal bars on x axis for each of the 15 testing systems. The PDB ids of these systems are shown on the y axis.



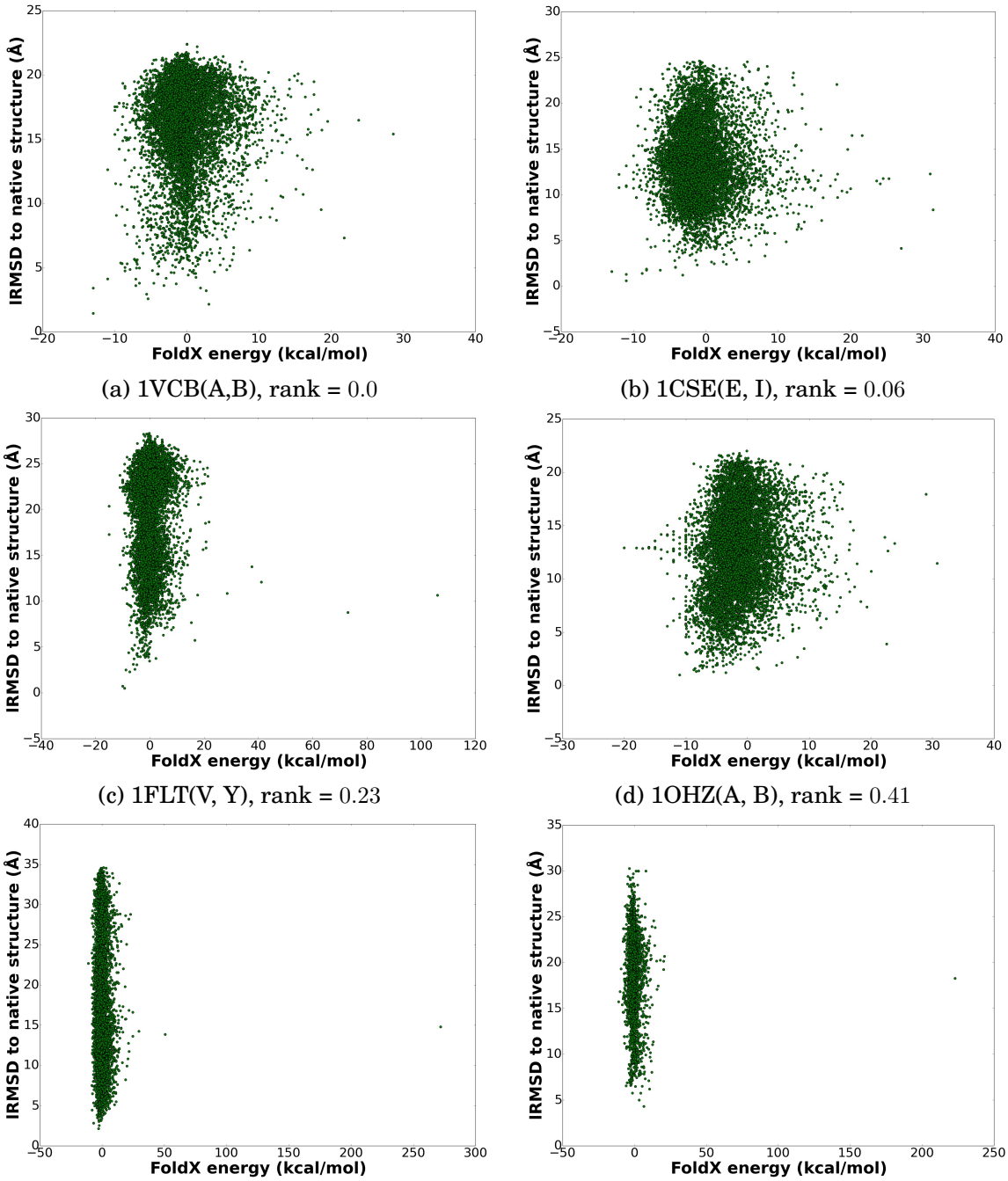
tive rank of the lowest-lRMSD configuration on each of these systems is also shown. The bottom panel in Figure 5 shows the lowest-energy configuration, instead, on 3 of the 6 selected systems. The lRMSD to the native structure of these configurations is also shown. On each of these 3 systems, the lowest-energy configuration is very far away from the native structure, including 1CSE (E, I), where a rank of 0.06 is found.

We introduce and analyze another measurement, hit rate, which is measured as follows: the idDock+-generated configurations are again sorted by FoldX energy in ascending order. We report the lowest lRMSD to the native structure on p% of the ordered configurations and vary p from $10-100$%. A p value of 10%, for instance, means that we consider only the top 10% of the energy-sorted configurations and report the lowest lRMSD to native among configurations in this subset. A p value of 100% means that all configurations are considered.

Figure 6 shows the hit rate at these 10% increments of p for each of the 15 systems. The results suggest that idDock+ is able to find the lowest lRMSD at $p = 40 - 50\%$ in the energy-sorted ordering for most of the systems.

The analysis on rank and hit rate suggests that even if one were to sort idDock+-generated configurations by FoldX interaction energy, the lowest-lRMSD configuration is expected to be found among no more than the top $2,000$ configurations in the sorted order. When proceeding with a post-processing stage, flexibility can be modeled only on these configurations and selection techniques can then focus on analyzing these configurations to draw from them the predicted native structure.

Fig. 4: 6 systems have selected on which to plot the FoldX interaction energy vs. lRMSD to the native structure for idDock+-generated configurations. The two systems in (a)-(b) have very low rank, followed by higher-rank systems in (c)-(d). The two systems in (e)-(f) have very high rank.



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

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

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

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

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

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

Fig. 5: Top and middle panels draw the lowest-lRMSD to the native configuration for 6 selected systems and superimpose it over the corresponding native structure. Bottom panel draws the lowest-energy configuration for 3 of the 6 selected systems. Chains are drawn in different colors, and the native structure is drawn in transparent.

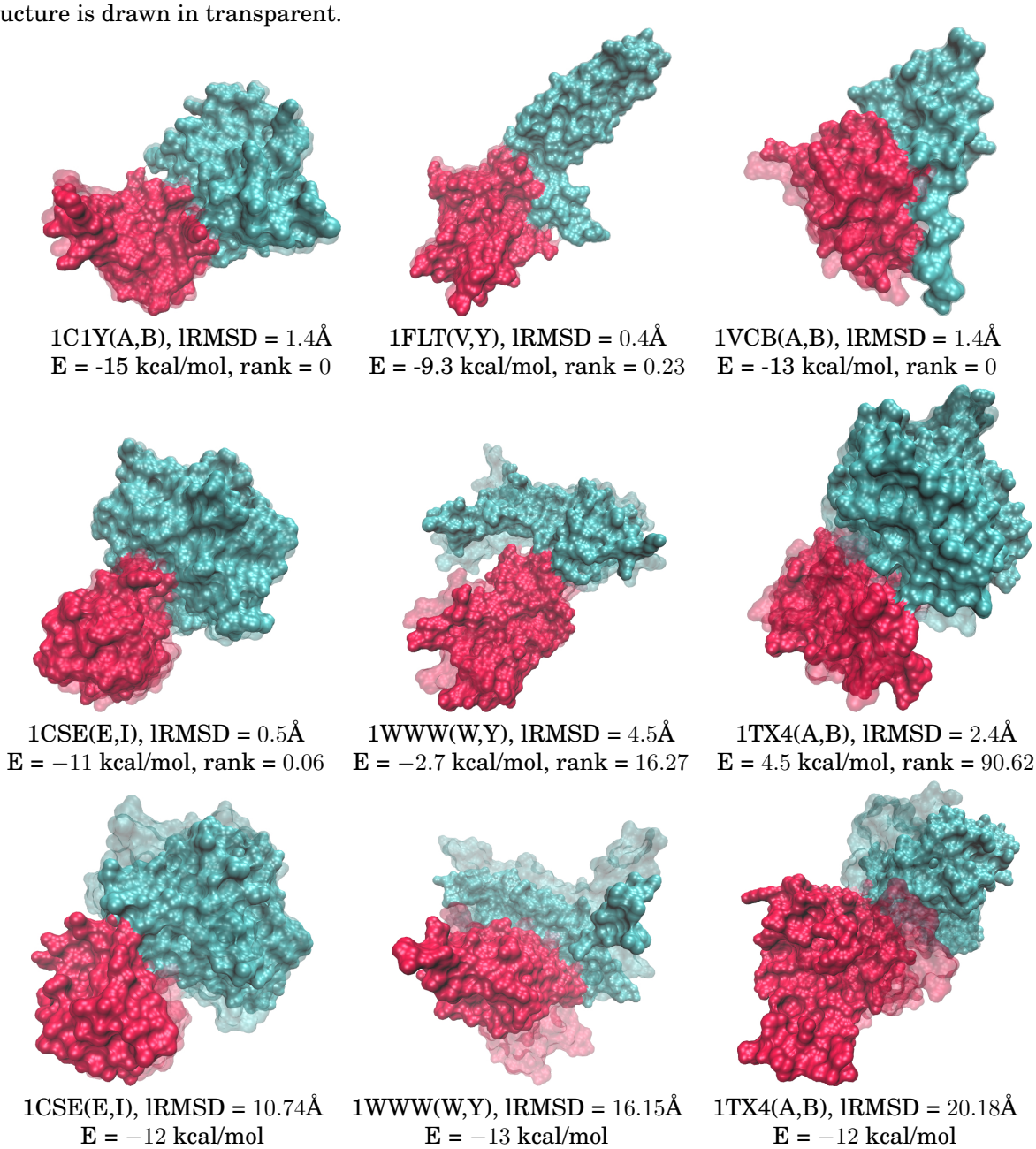
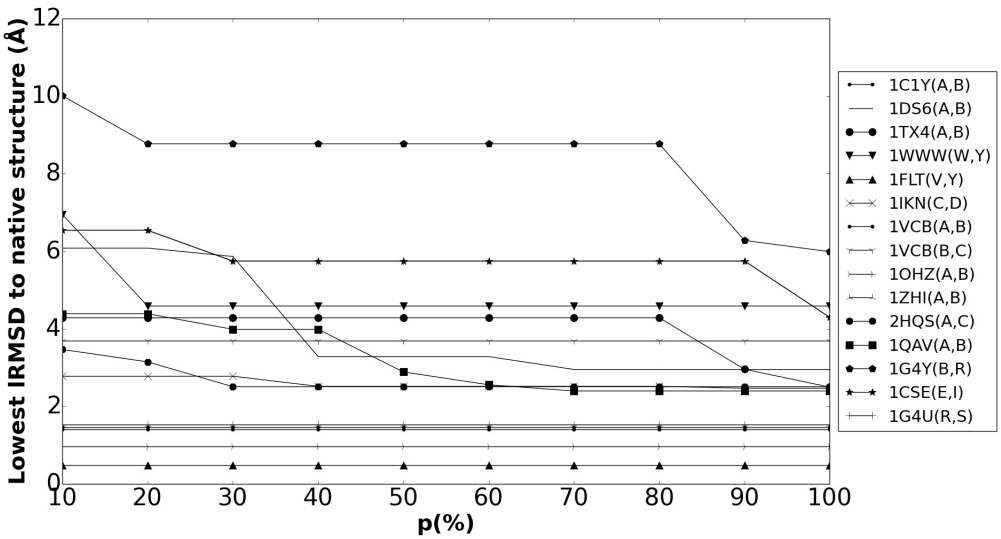


Fig. 6: Hit rate shown for all systems at 10% increments of p. p is shown on x-axis and hit rate is shown on y axis. Different markers are used for the different systems.



4. DISCUSSION

We have presented here idDock+, a novel algorithm for sampling low-energy configurations in rigid-body protein-protein docking. The algorithm employs concepts from both geometry- and energy-driven methods. Its probabilistic search is a realization of the Basin Hopping framework that allows generating a Metropolis Monte Carlo trajectory of consecutive local minima in the surface of a sophisticated energy function. However, one of the salient ingredients in idDock+ is its incorporation of a machine learning model to evaluate configurations prior to allocating to them a demanding budget for minimization. An ensemble learner is trained on a comprehensive training dataset and demonstrated effective in leading idDock+ towards near-native configurations.

In section 3 we focus on the overall performance of idDock+ and its comparison with other state-of-the-art methods that represent the various approaches to rigid protein-protein docking. Here we provide a few more details on why idDock+ succeeds or fails on certain systems. We focus on the relationship between the ensemble learner and the minimization with FoldX, as these two can work concertedly to lower the lRMSD to the native structure or against each-other. We focus on four systems: one where the trained model performs well but nonetheless the energy function drives idDock+ away from the native structure; one where the trained model is inaccurate but the energy function nonetheless drives the search towards the native structure; one where both the trained model and the energy function work concertedly and lead idDock+ to the native structure; and one where both fail.

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

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

The third case is illustrated by the dimer with PDB id 1FLT(V,Y), on which the learned model and the energy function perform in concert with each-other. The configuration with the lowest lRMSD to the native structure is obtained from a perturbed configuration that has passed the learned model and has lRMSD of 3.61Å to the native structure. The minimization further lowers this lRMSD to 0.4Å, which is a significant improvement of 3Å.

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

To our knowledge, idDock+ represents one of the first algorithms that integrate learned models in stochastic optimization for rigid protein-protein docking. Our results support the conclusion that this direction of research in protein-protein docking is worth investigating further, as the integration of an accurate learned model promises to address the issue with energy-driven optimization unable to solely drive the search towards near-native configurations. Consistently, studies show that energy-driven

optimization is effective on configurations in the vicinity of the native structure. It is the premise of learned models to identify such configurations efficiently.

Finally, it is worth noting that the algorithm presented here offers a roadmap to integrating other machine learning models in stochastic optimization approaches. As our detailed investigation of four selected systems indicates, there is room for improvement both in machine learning models and energy functions. Growing work in machine learning is expected to lead to more accurate models. These, coupled with increasingly accurate energy functions, will lead to further improvements in protein-protein docking.

Acknowledgement

Computations were run on ARGO, a research computing cluster provided by the Office of Research Computing at George Mason University, VA (URL: http://orc.gmu.edu). Funding for this work is provided in part by the National Science Foundation (CAREER Award No. 1144106). The authors are also grateful to the members of the Shehu lab for useful feedback during this work.

REFERENCES

Chandrajit L Bajaj, Rezaul Chowdhury, and Vinay Siddahanavalli. 2011. F2Dock: Fast Fourier Protein-Protein Docking. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)* 8, 1 (2011), 45–58.

H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne. 2000. The Protein Data Bank. *Nucleic Acids Research* 28, 1 (Jan. 2000), 235–242. www.pdb.org

A. J. Bordner and A. A. Gorin. 2007. Protein docking using surface matching and supervised machine learning. *Proteins: Structure, Function, and Bioinformatics* 68, 2 (2007), 488–502.

R. Chen, L. Li, and Z. Weng. 2003. ZDock: an initial-stage protein-docking algorithm. *Proteins: Struct. Funct. Bioinf.* 52, 1 (2003), 80–87.

T. M. Cheng, T. L. Blundell, and J. Fernandez-Recio. 2007. pyDock: Electrostatics and desolvation for effective scoring of rigid-body protein–protein docking. *Proteins* 68, 2 (Aug. 2007), 503–515.

S. R. Comeau, D. W. Gatchell, S. Vajda, and C. J. Camacho. 2004. ClusPro: a fully automated algorithm for protein-protein docking. *Nucl. Acids Res.* 32, S1 (2004), W96–W99.

M. L. Connolly. 1983. Analytical Molecular Surface Calculation. *J. Appl. Cryst.* 16, 5 (1983), 548–558.

C. Dominguez, R. Boelens, and A. M. Bonvin. 2003. HADDOCK: A protein-protein docking approach based on biochemical orbiophysical information. *J. Am. Chem. Soc.* 125 (2003), 1731–1737.

D. Duhovny-Schneidman, Y. Inbar, R. Nussinov, and H. J. Wolfson. 2005. PatchDock and SymmDock: servers for rigid and symmetric docking. *Nucl. Acids Res.* 33, S2 (2005), W363–W367.

S. Engelen, A. T. Ladislas, S. Sacquin-More, R. Lavery, and A. Carbone. 2009. A Large-Scale Method to Predict Protein Interfaces Based on Sequence Sampling. *PLoS Comp Bio* 5, 1 (2009), e1000267.

D. Fischer, S. L. Lin, H. L. Wolfson, and R. Nussinov. 2005. A geometry-based suite of molecular docking processes. *J. Mol. Biol.* 248, 2 (2005), 459–477.

M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. 2009. The WEKA data mining software: an update. In *SIGKDD*, Vol. 11. ACM, New York, NY, USA, 10–18.

I. Halperin, B. Ma, H. Wolfson, and R. Nussinov. 2002. Principles of docking: An overview of search algorithms and a guide to scoring functions. *Proteins* 47, 4 (2002), 409–443.

I. Hashmi, B. Akbal-Delibas, N. Haspel, and A. Shehu. 2011. Protein Docking with Information on Evolutionary Conserved Interfaces. In *IEEE Intl Conf on Bioinf and Biomed Workshops (BIBMW)*. IEEE, Atlanta, GA, 358–365.

I. Hashmi, B. Akbal-Delibas, N. Haspel, and A. Shehu. 2012. Guiding Protein Docking with Geometric and Evolutionary Information. *J Bioinf and Comp Biol* 10, 3 (2012), 1242002.

I. Hashmi and A. Shehu. 2012. A Basin Hopping Algorithm for Protein-protein Docking. In *IEEE Intl Conf on Bioinf and Biomed (BIBM)*. IEEE, Philadelphia, PA, 466–469.

I. Hashmi and A. Shehu. 2013a. HopDock: A Probabilistic Search Algorithm for Decoy Sampling in Protein-protein Docking. *Proteome Sci* 11, Suppl 1 (2013), S6.

I. Hashmi and A. Shehu. 2013b. Informatics-driven Protein-protein Docking. In *ACM Conf on Bioinf and Comp Biol Workshops (BCBW)*. ACM, Washington, D. C., 772–779.

S. Huang. 2014. Search strategies and evaluation in protein–protein docking: principles, advances and challenges. *Drug discovery Today* 19, 8 (2014), 1081–1096.

W. Humphrey, A. Dalke, and K. Schulten. 1996. VMD – Visual Molecular Dynamics. *Journal of Molecular Graphics* 14 (1996), 33–38.

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

B. Jimenez-Garcia, C. Pons, and J. Fernandez-Recio. 2013. pyDockWEB: a web server for rigid-body protein-protein docking using electrostatics and desolvation scoring. *Bioinformatics* 29, 13 (2013), 1698–1699.

E. Kanamori, Y. Murakami, Y. Tsuchiya, D.M. Standley, H. Nakamura, and K. Kinoshita. 2007. Docking of protein molecular surfaces with evolutionary trace analysis. *proteinssfb* 69 (2007), 832–838.

K. P. Kilambi, K. Reddy, and J. J. Gray. 2014. Protein-Protein Docking with Dynamic Residue Protonation States. *PLoS computational biology* 10, 12 (2014), e1004018.

D. Kozakov, R. Brenke, S. Comeau, and S. Vajda. 2006. PIPER: an FFT-based protein docking program with pairwise potentials. *Proteins: Struct. Funct. Bioinf.* 65, 2 (2006), 392–406.

M. F. Lensink and S. J. Wodak. 2009. Docking and scoring protein interactions: CAPRI 2009. *Proteins: Struct. Funct. Bioinf.* 78, 15 (2009), 3073–3084.

M. F. Lensink and S. J. Wodak. 2010. Blind predictions of protein interfaces by docking calculations in CAPRI. *Proteins: Struct. Funct. Bioinf.* 78, 15 (2010), 3085–3095.

B. Li and D. Kihara. 2012. Protein docking prediction using predicted protein-protein interface. *BMC Bioinf* 13 (2012), 7.

N. Li, Z. Sun, and F. Jiang. 2008. Prediction of protein-protein binding site by using core interface residue and support vector machine. *BMC Bioinf* 9 (2008), 553.

S. L. Lin, R. Nussinov, D. Fischer, and H. J. Wolfson. 1994. Molecular surface representations by sparse critical points. *Proteins* 18, 1 (Jan. 1994), 94–101.

S. Lyskov and J. J. Gray. 2008. The RosettaDock server for local protein-protein docking. *Nucl. Acids Res.* 36, S2 (2008), W233–W238.

G. Macindoe, L. Mavridis, V. Venkatraman, M. Devignes, and D. Ritchie. 2010. HexServer: an FFT-based protein docking server powered by graphics processors. *Nucl. Acids Res.* 38, Suppl 2 (2010), W445–W449.

N. Moitessier, P. Englebienne, D. Lee, J. Lawandi, and C. R. Corbeil. 2009. Towards the development of universal, fast and highly accurate docking/scoring methods: a long way to go. *British J Pharmacology* 153, S1 (2009), S7–S27.

B. Olson, I. Hashmi, K. Molloy, and A. Shehu. 2012. Basin Hopping as a General and Versatile Optimization Framework for the Characterization of Biological Macromolecules. *Advances in AI J* 2012, Article 674832 (2012), 19 pages.

V. Polak. 2003. *Polak V 2003 Budda: backbone unbound docking application Master's Thesis School of Computer Science, Tel-Aviv University*. Master's thesis. Computer Science, Tel-Aviv University, Tel-Aviv,Israel.

D. W. Ritchie. 2008. Recent progress and future directions in protein-protein docking. *Current protein & peptide science* 9, 1 (2008), 1.

J. Schymkowitz, J. Borg, F. Stricher, R. Nys, F. Rousseau, and L. Serrano. 2005. The FoldX web server: an online force field. *Nucl. Acids Res.* 33, Web server issue (2005), W382–W388.

B. A. Shoemaker and A. R. Panchenko. 2007. Deciphering protein–protein interactions. Part II. Computational methods to predict protein and domain interaction partners. *PLoS computational biology* 3, 4 (2007), e43.

G. Terashi, M. Takeda-Shitaka, K. Kanou, M. Iwadate, D. Takaya, and H. Umeyama. 2007. The SKE-DOCK server and human teams based on a combined method of shape complementarity and free energy estimation. *Proteins: Struct. Funct. Bioinf.* 69, 4 (2007), 866–887.

A. Tovchigrechko and I. A. Vakser. 2006. GRAMM-X public web server for protein-protein docking. *Nucl. Acids Res.* 34, Web Server issue (2006), W310–4.

R. Wang, X. Fang, Y. Lu, C.-Y. Yang, and S. Wang. 2005. The PDBbind Database: Methodologies and updates. *J. Med. Chem.* 48, 12 (2005), 411–419.

C. Yan, D. Dobbs, and V. Honavar. 2004. A two-stage classifier for identification of protein–protein interface residues. *Bioinformatics* 20, suppl 1 (2004), i371–i378.

H. Zhu, F. S. Domingues, I. Sommer, and T. Lengauer. 2006. NOXclass: prediction of protein-protein interaction types. *BMC Bioinf* 7 (2006), 27.