Protein structure prediction via combinatorial assembly of sub-structural units

Yuval Inbar\textsuperscript{1,\ast}, Hadar Benyamini\textsuperscript{2}, Ruth Nussinov\textsuperscript{2,\,3} and Haim J. Wolfson\textsuperscript{1}

\textsuperscript{1}School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel, \textsuperscript{2}Sackler Institute of Molecular Medicine, Sackler Faculty of Medicine, Tel Aviv University, Tel Aviv, Israel and \textsuperscript{3}Basic Research Program, SAIC-Frederick, Inc, Laboratory of Experimental and Computational Biology, NCI - FCRDC, Bldg 469, Rm 151, Frederick, MD 21702, USA

Received on January 6, 2003; accepted on February 20, 2003

ABSTRACT
Following the hierarchical nature of protein folding, we propose a three-stage scheme for the prediction of a protein structure from its sequence. First, the sequence is cut to fragments that are each assigned a structure. Second, the assigned structures are combinatorially assembled to form the overall 3D organization. Third, highly ranked predicted arrangements are completed and refined. This work focuses on the second stage of this scheme: the combinatorial assembly. We present CombDock, a combinatorial docking algorithm. CombDock gets an ordered set of protein sub-structures and predicts the inter-contacts that define their overall organization. We reduce the combinatorial assembly to a graph-theory problem, and give a heuristic polynomial solution to this computationally hard problem. We applied CombDock to various examples of structural units of two types: protein domains and building blocks, which are relatively stable sub-structures of domains. Moreover, we tested CombDock using increasingly distorted input, where the native structural units were replaced by similarly folded units extracted from homologous proteins and, in the more difficult cases, from globally unrelated proteins. The algorithm is robust, showing low sensitivity to input distortion. This suggests that CombDock is a useful tool in protein structure prediction that may be applied to large target proteins.

Supplementary information: More tables and figures are available at www.cs.tau.ac.il/~inbaryuv/combdock/

Contact: inbaryuv@tau.ac.il

Keywords: structure prediction, multiple docking, hierarchical model, combinatorial assembly.

INTRODUCTION
The assignment of a 3D conformation to a sequence of a polypeptide chain is known as the protein folding problem. Although it is one of the most studied problems in Bioinformatics, there is still no tool to reliably predict the fold of a new protein given its sequence. Currently, there are three major strategies for protein structure prediction: homology modeling, fold recognition and \textit{ab initio} prediction. Homology modeling (Sali and Blundell, 1993) is based on the observation that similar sequences fold into similar structures. For proteins with a similar sequence to a protein with a known structure, this is considered to be the most reliable prediction method (Lattman and Hopkins, 2001). Fold recognition (Jones, 1999) is based on the observation that there is a limited number of possible folds and therefore dissimilar sequences may fold into similar structures (Orengo \textit{et al.}, 1994). This approach involves threading of a query sequence onto known protein folds and evaluating its energy. Though these methods do not assume an obvious sequence homology to a protein with a known structure, they are still limited to structure prediction of proteins that belong to an already existing fold, and share some sequence similarity. The \textit{ab initio} methods (Covell, 1992; Ortiz \textit{et al.}, 1998; Sugita and Okamoto, 1999; Hardin \textit{et al.}, 2002) do not make any similarity assumptions. Rather they perform a conformational search on the target protein, seeking the lowest energy conformation which is referred to as the native structure. Semi \textit{ab initio} techniques use the protein structural database in order to make a knowledge-based conformational search (Simons \textit{et al.}, 1999; Kolodny \textit{et al.}, 2002). This approach is based on the observation that structures can be reconstructed using a relatively small library of short segments template structures (Unger \textit{et al.}, 1989).

According to the hierarchical model of protein folding suggested by Rose (1979), and by Lesk and Rose (1981), in the first stage of the folding process, short range interactions occur and form sub-structures of local fragments. Then, relatively stable local structural units with high population time join to larger structural units via non-local
interactions. These interactions follow the protein folding pathway that leads to the native structure. A line of experimental evidence suggests the existence of autonomous folding units within protein domains playing an important role in the protein folding pathway. Those include experiments with hydrogen exchange (Bai et al., 1995; Milne et al., 1998; Chamberlain et al., 1999; Rumbley et al., 2001), fragment complementarity (Dyson et al., 1992; Prat-Gay, 1996; Gegg et al., 1997; Zitzewitz et al., 1999) and folding studies using NMR (Park et al., 1997; Yang et al., 1998).

We have proposed a three-stage scheme for protein structure prediction that is based on this model (Fig. 1), (Haspel et al., 2003). In the first stage, a target sequence is cut into disjoint segments. Each segment is assigned a structural model. The second stage, the combinatorial assembly, gets as an input the structural models that were output by the first stage. It assembles them in an attempt to predict their relative 3D complexes. In the third stage, a completion and refinement of highly ranked models is performed.

Here we present the CombDock algorithm, which implements the combinatorial assembly stage. We have tested it on domains assembly. Many multi-domain proteins that have no overall sequence homology to any known structure contain domains that separately have good sequence homology to a solved structure. Existing techniques may predict a structural model for each domain. CombDock may then provide their spatial arrangement. We have further tested CombDock on assembly of smaller sub-structural units. Tsai and Nussinov (2001) have developed a dissecting algorithm which outputs a set of compact (sequence contiguous) building blocks for an input protein. The algorithm output was verified with limited proteolysis experiments (Tsai et al., 2002). We have used the cutting algorithm to generate the input. The building blocks assembly examples demonstrate the potential usage of CombDock in protein structure prediction.

To the best of our knowledge, CombDock is the first docking based tool for the combinatorial assembly of multiple protein structural fragments. Our algorithm gets as input a set of protein structures, and outputs a list of ranked predictions for their organization. The algorithm is graph-theory based, utilizing a reduction of the multiple docking problem to the problem of finding a spanning tree in a complete graph. While the current work demonstrates the role of the algorithm in structure prediction, CombDock is also likely to be useful for the prediction of co-functioning multi-protein assemblies.

**COMBDOCK CONCEPT AND ALGORITHM**

The goal is to predict the overall structure. The input consists of known or predicted protein structural units (SUs). The structural units are domains, or building blocks which are substructures of domains. The algorithm assumes that the SUs are relatively accurate, although we show that it performs quite well even if the input is slightly distorted. Our approach is based on combinatorial pairwise interactions between the SUs. First, we give a mathematical description of the problem, followed by the algorithm outline and the main principles of its modules. We further describe how the design of the algorithm addresses the characteristics of the problem.

**Problem statement**

The input consists of an ordered set of $N$ SUs of a protein. The SUs are 3D conformations of disjoint segments of a protein sequence. The order of the SUs is induced by the N-to-C termini direction. Our goal is to construct a list of ranked complexes. Each complex contains $N$ input SUs, and represents a potential spatial arrangement of the SUs. The SUs are considered rigid bodies and can only be rotated and translated with respect to each other. The algorithm attempts to construct a near-native complex (similar to the native fold). This near-native complex should be ranked highly. Each output complex has to be a valid complex. A valid complex has to fulfill two types of constraints. The first is the penetration constraint, which avoids a high incidence of steric clashes between the atoms of the different SUs. The penetration is measured by sampling the Connolly’s surface (Connolly, 1983) of
one SU and checking how deep each sampling point penetrates the other SUs. We set the threshold constant at 3.0 Å. The second is the backbone constraint. Each SU has an N-terminus and a C-terminus. In nature, there is a physical restriction on the maximum Euclidean distance between the termini of consecutive SUs. This maximum distance is induced by the covalent bonds of the backbone and is reached when the polypeptide chain between the SUs is fully extended. Hence, the value of the distance constraint is a function of the number of amino acids between consecutive SUs.

Algorithm outline
In principle the assembly of SUs into one 3D complex is similar to the problem of solving a 3D jigsaw puzzle where the protein SUs are the puzzle pieces. There are two main differences between the two problems. First, the backbone constraints do not appear in the 3D puzzle problem; second, in a 3D puzzle there is no penetration between two pieces whereas in the assembly problem small extents of penetration are allowed in order to neglect minor side chain motions. When trying to solve a 3D puzzle, we normally follow an iterative algorithm. In each iteration we assemble pairs of the puzzle pieces. We assume that a good match between two pieces is not incidental and thus it is the correct orientation between them. Such a connected pair is considered one piece in the next iteration. Since we join one pair of pieces at each iteration, after \( N - 1 \) iterations we are left with only one piece, which is a complex of all the \( N \) original pieces. We hope that this complex is the correct solution of the puzzle. Our algorithm imitates this behavior. The puzzle is the protein native fold, the puzzle pieces are the protein structural units. The method has three modules (Fig. 1): (i) The all pairs docking module uses a geometric docking algorithm to obtain possible orientations that match all the different pairs of the puzzle pieces. For each pair of SUs, it outputs a set of transformations and their scores. (ii) The combinatorial assembly module gets these sets of transformations and uses them to construct complexes. Each complex contains all the SUs. (iii) The rescoring module clusters and rescres these complexes, in order to minimize the number of solutions and to re-rank them. In this stage the scoring function employs also physicochemical features.

All pairs docking
In this stage we apply a geometrical pairwise docking algorithm on each of the \( N(N-1)/2 \) pairs of SUs. For each run, the docking algorithm returns a set of 3D transformations. By applying a transformation on the second SU of a pair, we change its orientation and position so it docks to the first SU. Each output transformation is scored by evaluating the geometric complementarity between the two SUs that we get by applying it. We keep the best \( K \) (user defined parameter) scored transformations for each pair of SUs. We have developed a docking algorithm that is a hybrid of two docking algorithms, Norel et al. (1995) and Polak (2002). The algorithm analyzes the surface of the two molecules (SUs) that it docks and by matching complementary local features, it calculates candidate transformations. The transformations are clustered, filtered and evaluated to find the \( K \) best ones with respect to geometrical complementarity. The clustering ensures that any two transformations of these \( K \) solutions yields significantly different complexes. The pairwise docking algorithm has a preprocessing stage which performs calculations involving only one molecule (in our case SU), and a stage which computes possible orientations between two SUs. \( N \) SUs define \( N(N-1) \) pairs. Since each SU is included in \( N - 1 \) different pairs, we save time by performing the preprocessing stage only once for each SU instead of \( N - 1 \) times.

Combinatorial assembly
This module constructs valid complexes by using only the input transformations generated by the all Pairs Docking module. We reduce the combinatorial assembly problem to the problem of finding spanning trees in a graph. The spanning trees must represent valid complexes. Each SU is represented by a vertex and each transformation between two SUs is represented by an edge between the two corresponding vertices. The input is therefore represented by a complete graph with \( K \) parallel edges between any pair of vertices. If we use one transformation to connect two SUs \( i \) and \( j \), and another transformation to connect SUs \( j \) and \( k \), we get a complex of three SUs where the transformation between SUs \( i \) and \( k \) is induced by the two transformations \((ij, jk)\). In the corresponding graph, this complex is represented by a subtree which contains the vertices \( i, j \) and \( k \) (which represents SUs \( i, j \) and \( k \) and the edges that represent the transformations that we have selected. Generalizing it to \( N \) SUs, a complex of \( N \) SUs is represented by a sub-graph with one connectivity component and no cycles, i.e. a spanning tree. Using this reduction, we can analyze the exponential characteristics of the problem. The number of different spanning trees of a complete graph with \( N \) vertices which has no parallel edges is \( N^{N-2} \). In our case we have \( K \) parallel edges between each pair of vertices. Each spanning tree of a graph with no parallel edges, represents \( K^{N-1} \) different trees (in the \( K \) parallel edges case) that have the same topology but differ in at least one edge. Therefore, there are \( N^{N-2}K^{N-1} \) different spanning trees for the input graph. Any spanning tree represents a complex, but the complex might be invalid. Since we seek the best spanning trees which represent only valid complexes, we cannot
use the polynomial running time algorithm that finds the best spanning tree because it searches the whole space of spanning trees, valid and invalid ones. In fact, the problem we are trying to solve is \textit{NP-Complete}. Even the 2D analogue is \textit{NP-Complete}. This can be shown using a simple reduction to the \textit{monkeys-puzzle} problem\(^1\). We have constructed a polynomial algorithm which provides a heuristic solution.

**The algorithm for the spanning trees selection**

There are three basic principles in the algorithm: (i) limiting the topology types of the spanning trees; (ii) an efficient search method, and (iii) a partially greedy selection of subtrees which bound the overall runtime and memory complexity of the algorithm.

The trees topology To explain the topology of trees which restricts our search, let us first introduce some terminology and definitions. A subtree of the input graph is a sub-graph that has only one connected component and no cycles. A subtree with \( i \) vertices represents a super structural unit (SSU). This SSU is a complex of \((1 \leq i \leq N)\) SUs, generated by applying the \( i - 1 \) transformations which the edges of the subtree represent. Two trees are adjacent if and only if there exists an index \( i \) so that vertex \( i \) is in one subtree and vertex \( i + 1 \) is in the other. We define recursively a sequential tree to be a tree of only one vertex or a tree that has an edge that connects two sequential trees that are adjacent. This inductive definition is motivated by the major role of local interactions within the protein. We restrict our search only to sequential trees. We define a simple folding step as a joining of two adjacent SSUs (represented by sequential subtrees) by applying an input transformation (an edge). In the generation of sequential trees (detailed in the following section) we use only simple folding steps.

The search method Exploiting the fact that different trees share common parts (subtrees), we have developed a hierarchical algorithm of \( N \) stages \((N \) is the number of structural units). At the \( i \)th stage, the algorithm generates sequential trees with \( i \) vertices. The first stage generates trivial trees, which are the vertices themselves. At the \( i \)th stage, the algorithm constructs trees with \( i \) vertices by connecting trees of smaller sizes generated earlier. At the \( N \)th stage, the valid spanning trees are generated. By using only simple folding steps we generate at each stage only sequential trees. A generated tree at each stage is discarded immediately, if it does not fulfill one of the constraints (backbone, penetration). Since we join two valid subtrees in order to get a new one, we need to check only the inter-subtrees constraints.

**The greedy selection of subtrees**

To address the memory problem, we have modified the data structure that holds and retrieves the sequential trees. Instead of saving all sequential trees of size \( s \) that begin at index \( i \), the data structure saves only the top \( D \) scoring trees, where \( D \) is a user defined parameter. We calculate the score of each tree комплекс by summing the scores of the transformations that were used to create it. This heuristic puts a polynomial upper bound on the runtime complexity. The memory complexity of each stage is \( O(DN) \), therefore the runtime complexity is \( O(D^2N^2K \log D) \). The memory complexity of the whole algorithm is \( O(N^2D^2) \), and its runtime complexity is \( O(D^2N^3K \log D) \). Instead of a brute force search, a selective restricted search is performed, which might cause the loss of good solutions. To evaluate this heuristic solution, it has to be tested on real structures, checking how well the native arrangement is predicted (see Results).

**Final scoring**

The input to this module is the set of complexes generated by the combinatorial assembly stage. It scores these final complexes and it performs an iterative clustering of the solutions in order to avoid redundant hypotheses. The scoring function combines both geometric and physico-chemical features. The geometrical component evaluates the geometrical complementarity between all the structural units in the given complex. Large interface area increases the score while penetration decreases it. Notice that in the complexes of the final stage, no structural unit penetrates another by more than the penetration threshold. The physico-chemical component measures the non-polar buried surface area. We mark the uncharged surface atoms of each structural unit. Then, given a complex we check if a marked atom becomes involved in an interface with another SU. If it is, we assume it became buried. The scoring function evaluates the surface area of these uncharged atoms that got buried in the input complex.

**RESULTS**

The goal of CombDock is to predict the protein structure from a given set of structural units (SUs) which are sequentially connected. However, we do not know how they are positioned and oriented in space with respect to each other. There are three types of cases that present different levels of difficulty:

**Type I:** Protein structural units from the same protein

An input of this type is generated by cutting a single protein to several SUs. Each unit is a 3D structure of

---

\(^1\) The monkeys-puzzle consists of \( N^2 \) square pieces. On each face of each piece a tail or a head of a monkey is drawn in a different color. The goal is to find an arrangement of the pieces, where we get a square of \( N \times N \) pieces, such that on the border of any two adjacent pieces, lies a complete monkey, i.e. head and tail of a monkey of the same color.
a single fragment of the protein sequence. The goal is to reconstruct the original overall structure. This case type resembles the so-called 'bound docking', where we separate a complex and re-dock it.

**Type II: Protein structural units from similar proteins:**
This input type consists of protein SUs with each one originating from a structure of a homologous protein. No two units are taken from the same protein molecule. The generation of this type of input involves: (i) A target protein (with a known structure) is chosen. (ii) The target structure is cut into SUs (domains or building blocks). (iii) For each original unit, an alternative unit with a high topological similarity is selected from a homologous protein. Here, the goal is to reconstruct a complex where the spatial arrangement of the units is similar to the arrangement in the target protein structure, i.e. the arrangement of the original units. Type II is computationally more difficult than type I, due to the structural and sequential variability in the interfaces of the units. Whereas in type I the geometrical complementarity between the units is practically perfect since they are derived from the same structure, this is not the case in type II where the units are derived from different sequences.

**Type III: Protein structural units from unrelated proteins:**
The input consists of SUs, each derived from a different protein with a different overall structure with low sequence homology (lower than in type II). This type of input presents a substantially more challenging problem since the alternative units are significantly less similar, sequentially and structurally, to the original ones in the native structure.

The output of CombDock is a set of scored complexes (solutions). To evaluate them, we seek the highest ranked solution whose Root Mean Square Deviation (RMSD) from the original protein (or units arrangement) is less than 5.0 Å. The higher is the rank of this solution and the lower is its RMSD from the native arrangement, the better is the result. Notice that since structural units of type II and

---

**Table 1. Assembly of protein domains: Input analysis and results**

<table>
<thead>
<tr>
<th>Target</th>
<th>input type</th>
<th>Domain</th>
<th>Input domain source, PDB(position)</th>
<th>% seq. identity (similarity)</th>
<th>% structurally unmatched</th>
<th>best solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Glycosyl transferase (PDB: 1a47)</td>
<td>I</td>
<td>1 (1-495)</td>
<td><em>T.here, 1a47</em>(1-495)</td>
<td>100</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 (496-578)</td>
<td><em>T.here, 1a47</em>(496-578)</td>
<td>100</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 (579-683)</td>
<td><em>T.here, 1a47</em>(579-683)</td>
<td>100</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>II</td>
<td>1 (1-495)</td>
<td><em>B.sp, 1d7f</em>(1-496)</td>
<td>71 (84)</td>
<td>68 (82)</td>
<td>1.5(1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 (496-578)</td>
<td><em>B.stear, 1cyg</em>(492-574)</td>
<td>75 (72)</td>
<td>69 (82)</td>
<td>0.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 (579-683)</td>
<td><em>B.circulans, 1kcl</em>(582-686)</td>
<td>75 (89)</td>
<td>69 (82)</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td>III</td>
<td>1 (1-495)</td>
<td><em>A.niger, 2aaa</em></td>
<td>27 (44)</td>
<td>27 (44)</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 (496-578)</td>
<td><em>Epp, 1k3m</em>(538-639)</td>
<td>NS</td>
<td>NS</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 (579-683)</td>
<td><em>A.niger, 1kul</em></td>
<td>37 (60)</td>
<td>37 (60)</td>
<td>20</td>
</tr>
<tr>
<td>Elongation factor (PDB: 1b23)</td>
<td>I</td>
<td>1 (1-212)</td>
<td><em>E.coli, 1d2e</em>(A:297-393)</td>
<td>100</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 (213-312)</td>
<td><em>E.coli, 1d2e</em>(A:213-312)</td>
<td>100</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 (313-405)</td>
<td><em>E.coli, 1d2e</em>(313-405)</td>
<td>100</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>II</td>
<td>1 (1-212)</td>
<td><em>B.taurus, 1d2e</em>(A:55-250)</td>
<td>62 (78)</td>
<td>57 (74)</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 (213-312)</td>
<td><em>T.here, 1xem</em>(A:213-312)</td>
<td>97 (98)</td>
<td>95 (96)</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 (313-405)</td>
<td><em>E.coli, 1dg1</em>(H:297-393)</td>
<td>76 (86)</td>
<td>71 (82)</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>III</td>
<td>1 (1-212)</td>
<td><em>M.musculus, 1f6q</em></td>
<td>NS</td>
<td>NS</td>
<td>44</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 (213-312)</td>
<td><em>B.stear, 1d1n</em></td>
<td>NS</td>
<td>NS</td>
<td>82</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 (313-405)</td>
<td><em>A.sulfolobus, 1jay</em>(A:323-429)</td>
<td>NS</td>
<td>NS</td>
<td>52</td>
</tr>
</tbody>
</table>

The data set of three input types of domain assembly for both glycosyl transferase and elongation factor (EF-Tu). Type I is self-assembly. Type II involves substitution of domains from homologous proteins. In type III the domains were taken from proteins with globally different structures. The sequential and structural similarity of the input domains to the target is noted. The domain columns present data that refer to the input domain while the overall columns refer to the source protein from which the domain was taken. Sequence identity and similarity percentages were calculated using NCBI Blast (default parameters). NS = not significant. For structural comparisons, we superimposed the source structure from which the domains were taken on the target structure. The structure alignment finds correspondence between ‘matched’ residues, which are within a distance of 2.5 Å when superimposed. The number of residues without correspondence indicates the structural variance (distortion) or dissimilarity between the aligned structures. The average values is the weighted (by the domain size) average of corresponding values of the domains. The best solution is defined as the highest ranked solution with an RMSD of less than 5.0 Å. We superimposed the solution complex on the complex which represents the arrangement of the domains in the target protein, and calculated the RMSD between all sets of Ca atoms.

---

**Downloaded from https://academic.oup.com/bioinformatics/article-abstract/19/suppl_1/i158/227881 by guest on 27 March 2019**
Protein structure prediction via combinatorial assembly of sub-structural units

(a) The source structures of the input domains
(b) Superposition
(c) The native structure (left) and best solution (right).

Fig. 2. Domain assembly of Cyclodextrin glycosyltransferase, type III (a) Each input domain was extracted from a different source. The sources of the α-amylase domain (red on the left) and the starch-binding domain (green on the right) are single domain proteins, while the E-set domain (blue) source has 3 domains. (b) The structural alignment demonstrates the significant structural difference between the input domains and the target structure (PDB 1A47). The target structure backbone is in gray. (c) On the right is the best solution, which was ranked first with an RMSD of 4.2 Å from the native structure (on the left).

Fig. 2. Domain assembly of Cyclodextrin glycosyltransferase, type III (a) Each input domain was extracted from a different source. The sources of the α-amylase domain (red on the left) and the starch-binding domain (green on the right) are single domain proteins, while the E-set domain (blue) source has 3 domains. (b) The structural alignment demonstrates the significant structural difference between the input domains and the target structure (PDB 1A47). The target structure backbone is in gray. (c) On the right is the best solution, which was ranked first with an RMSD of 4.2 Å from the native structure (on the left).

type III are sequentially different from the target protein, we do not apply the physico-chemical component of the final scoring function, rather we use a geometrical scoring function.

Below we present our results. First, we detail the cases of protein domains assembly. The domain classification is derived from SCOP (Murzin et al., 1995). Next, we present cases of building blocks assembly.

A. Assembly of protein domains We have chosen two multi-domain proteins cyclodextrin glycosyltransferase and elongation factor:
Target 1: Cyclodextrin Glycosyltransferase (PDB: 1a47)((Strokopytov et al., 1996)). According to the SCOP database classification, this protein contains four domains: (i) the N-terminal domain of α-amylase, (ii) the C terminal domain of α-amylase (iii) the E-set domain and (iv) the starch binding domain. The first two domains, the α-amylase N and C termini, do not appear separately in any solved structure. Therefore, we treated this protein as containing three domains: α-amylase, E-set and starch binding.
Target 2: Elongation factor (PDB:1b23)(Nissen et al., 1999). This protein contains three different domains of three different folds (SCOP (Murzin et al., 1995)): (i) Elongation factor Tu (EF-Tu), N-terminal (G) domain (ii) EF-Tu, domain 2 and (iii) EF-Tu, C-terminal domain.

For each target protein we generated the three input types. For input type 1, we just cut the target protein to its domain, using them as the input domains. The native domains were replaced in the input for type II by domains from homologous proteins. Note that the input type III domains has a different structural context than the original domains, in order to make the prediction more difficult (Fig. 2a). Table 1 details the sequential and structural similarity between the input domains to the target domains. We have run CombDock on these six cases (2 × 3). For both targets and for each input type a near-native solution was found. In five of the cases the near-native solution was also ranked in the top five solutions (Table 1). The running times of the combinatorial assembly stage were very short, less than 4 minutes for each example, since there are only three domains that are assembled in these examples. We discuss combinatorial assembly running time in the following building blocks section, where the examples are of higher variability.

Degradation of the predictions due to distorted input Running CombDock on the three input types for the same two targets (Table 1) enabled us to analyze the effect of distorted input. As expected the most accurate predictions were for input type I examples. The predictions were degraded when the input was distorted (type II and III). Comparing the results of type II to type III show that the more distorted is the input, the less accurate are the results. The worst prediction was for the elongation factor, input type III where the near-native prediction was ranked only 42nd. The fact that a near native solution was still found suggests that an enhanced scoring function may enable us to better predict the arrangement of such a distorted input.

Performance of the pairwise docking is more sensitive to input distortion Input distortion affects the geometrical complementarity between the domains. Since we apply a geometrical docking in the all pairs docking stage, it is not surprising that the predictions were less accurate. For the cyclodextrin glycosyltransferase the pairwise docking performance were very good for input type I: for two out
of the three pairs ($\alpha$-amylase:E-set and $\alpha$-amylase:starch binding) the top rank transformation had an RMSD of around 1.0 Å. For the same pairs, the predicted pairwise interactions of input type II were slightly less accurate: an RMSD of around 1.5 Å and lower rank. The pairwise docking performance for type III was the worst; no solution in the top 100 had an RMSD under 5.0 Å for any pair. However, the combinatorial assembly is less sensitive to input distortion. It used pairwise transformations that were ranked very lowly by the all pairs docking stage in order to generate a near-native complex. This complex was the top ranked solution since the multiple approach of the combinatorial assembly considers the whole complex rather than just the pairwise interactions. Moreover, in the best solution complex, we obtained a better orientation between the E-set and the starch binding domains (which was induced by the two transformations that were used to generate the complex) than any of the transformations that the pairwise docking suggested.

B. Assembly of protein building blocks

Building blocks (BBs) are compact and relatively stable domain substructures. Tsai and Nussinov (2001) have developed a dissecting algorithm which outputs a set of compact (sequence consecutive) building blocks. These building blocks usually consist of one or more secondary structure elements. We used the algorithm to dissect the target proteins. Both building blocks and domains are protein structural units consisting of a single segment of the polypeptide chain. Additionally, in both cases, the single polypeptide chain of the protein induces distance constraints between any two consecutive units. These constraints depend on the length of the linkers, which relates to the ‘unassigned’ regions between the two. ‘Unassigned’ regions are the segment of the sequence which have not been assigned into any building block by the building block dissecting algorithm.

Despite the common characteristics of the problems, in general the assembly of building blocks is computationally more difficult than the assembly of domains, due to two main reasons: (i) On average, the number of building blocks in a single domain is higher than the number of domains in a multi-domain protein. (ii) The linkers are longer with respect to the size of the structural units, in building blocks than in domains. This makes the linkers distance constraints significantly more restrictive in the domain assembly problem, reducing the size of the arrangement search space. Nevertheless, as we detail below, the CombDock algorithm was able to find near-native solutions for various cases of building blocks assembly.

Input Type I: assembly of the target protein building blocks

We show seven examples of input type I targets. Table 2 lists the targets, their properties and a summary of the results. These examples vary in the number of building blocks (from 4 up to 13 BBs), the maximum linker length (from 1 up to 13 residues), the fold class of the target protein and the average size of the BBs. The CombDock algorithm was able to find good solutions for all targets, and has further ranked each one in the top 10 solutions. Since the BBs were taken from the native structures and are not distorted, we could examine how other input properties influence the algorithm performance.

Linker lengths: The shorter the linker, the tighter is the backbone distance constraint between the two corresponding BBs. Indeed, as seen in Table 2, for the input instances with longer linkers, plastocyanin and lectin, the algorithm has generated more complexes than the examples with an equal number of BBs but shorter linkers: 650 complexes for the plastocyanin (4 BBs), only 365 for the interleukin (4 BBs), 3909 complexes for the lectin (6 BBs) and only 1476 for the nucleoside diphosphate kinase (6 BBs). For the same reason the running time of the algorithm also increases as the linker length increases.

Number of building blocks: As the number of building blocks increases, the running time of the algorithm becomes longer and the number of complexes produced by the combinatorial assembly stage increases. However, it is encouraging that the rank and RMSD of the near-native solutions only slightly decreased as the number of BBs increased. For the lectin example with 6 BBs, the top solution has an RMSD of 1.3 Å from the native. This is
almost as accurate as the solution for interleukin (0.8 Å) which has 4 BBs. Even for the citrate synthase which has 13 BBs, the algorithm found a 4.5 Å RMSD solution and ranked it in the top 10 results.

**BBs size:** The size appears to be the most important input property. The smaller the BBs, the worse is the performance of the CombDock algorithm. The lysin and the Nucleoside diphosphate kinase (NDK) examples which had the smallest sizes of BBs (23 and 21 respectively) also had the poorer results.

**Fold type:** This appears to have no impact on the performance of the algorithm. Both all α (interleukin, citrate synthase) and all β (plastocyanin, picornavirus, lectin) target proteins were well predicted. We attribute the less accurate predictions of the NDK (α + β) and Lysin (all α) examples to their small BBs.

**Input type II: building blocks extracted from similar proteins**

We generated three examples for input type II. As detailed in Table 2, for each example, one BB was extracted from the original target structure and the remaining BBs were extracted from other structures. In all cases a near-native solution was ranked in the top 5 solutions. The lectin had a better result (top ranked solution has an RMSD of 3.0 Å), despite the fact that the input BBs have a lower structural similarity to the target BBs than in the case of the NDK.

We attribute this to the difference in the sizes of the BBs. The average size of the lectin BBs is the highest of the three. As expected, for lectin and NDK, the performance of the algorithm was better in the type I examples than in type II. The type II distorted input BBs cause the top solution of the lectin to be within 3.0 Å RMSD from the native, while in type I (using the original BBs) the RMSD was only 1.1 Å. The near native solution for NDK in type I was ranked 1st while in type II only the 4th ranked solution was a near-native one.

**Input type III: building blocks from globally unrelated proteins**

Here we strictly selected the input BBs from proteins that belong to different super-families in SCOP. This restriction enforces that (i) only one of the BBs of the source proteins may be related to the target protein; (ii) two BBs which are in contact in the native structure must be derived from proteins with different overall structures. Details of two examples of this type are presented in Table 2. The first, leukocyte adhesion glycoprotein (LAG) is an α/β protein with 4 BBs. For this target we kept one original BB (BB no. 3) and the rest were extracted from proteins with an average overall sequence identity of 17%. These 3 input BBs, when superimposed on the target structure have 13% of mismatch residues (13 out of the 96 Cα are not structurally aligned within the 3.0 Å range for the 3 BB). The linker lengths in this input are 5, 7 and 1 which further complicates the assembly problem. Indeed, the number of complexes that were generated is very high for a structure with 4 BBs (Table 2). Nevertheless, after the final scoring stage, the 3rd ranked solution had an RMSD of 3.4 Å. The second example, the myoglobin, is an all α protein with 6 BBs. The average identity percentage is 25% and the structurally mismatched residues average is 5%. The linkers are all of length 1 except the first linker (length 2). CombDock has generated 710 complexes, 100 were left after clustering, and the top ranked solution has an RMSD of 3.0 Å (Fig. 3). The myoglobin example gave better results than LAG. It appears that the larger number of BBs affected mostly the running time of the algorithm, while the linker lengths decreased the quality of the prediction.

**DISCUSSION**

**Algorithm performance**

we have shown the advantages of CombDock over pairwise docking algorithms. The results indicate that not only was the algorithm able to select the right pairs of SUs with a significant interface area, but it also chose the correct transformations between them. While these transformations were sometimes ranked lowly by the pairwise docking, they were re-ranked higher when we applied the multiple approach, which considers the whole complex rather than just the pairwise interactions. For instance, in the assembly of glycosyl transferase (PDB: 1a47) input type III, the predicted complex with an RMSD of 4.25 was generated using the top ranked transformation output by the pairwise docking between the α-amylase and the E-set domains. It further used the 301st ranked transformation between the α-amylase and the starch binding. For the E-set and the α-amylase the pairwise docking had no reasonable solution within the top 2000 ranked transformations, nevertheless the top ranked solution of the combinatorial assembly was able to find their correct orientation. By testing a broad range of examples, we have learned that increasing the number of SUs increases the running time of the algorithm but only slightly decreases the prediction quality. On the other hand, the linker lengths and the size of the SUs have a strong impact on the performance of the algorithm. Predictions improve when the linker lengths are shorter and the SUs size is larger. Additionally, large SUs may compensate for long linkers, since this lowers the degrees of freedom of the conformational search. This is demonstrated in the assembly of legume lectin BBs using input type III. We have also shown that a distorted input caused CombDock to produce a less accurate prediction (Type I solutions are better than type II and III). However,
the results of the type II and III examples demonstrate that even when the input is distorted, CombDock predicts a near-native solution. This robustness is extremely important since in the structure prediction scheme (Fig. 1), the input SUs constitute predicted models for segments of the target sequence. In the best scenario these models are near-native models; never the exact ones.

**The role of CombDock in protein structure prediction:** Structure prediction techniques are usually classified into three types (i) comparative or homology modeling, which uses sequence similarity of the target sequence to proteins with known structures; (ii) fold recognition, which fits structural models to the target protein sequence; and (iii) *ab initio* techniques which perform conformational searches, and evaluate the resulting models by a physical or a knowledge-based scoring function.

The protein structure prediction scheme that we suggest is based on the hypothesis that protein folding is hierarchical. According to this scheme, the first stage in protein folding involves binding of short segments of the protein sequence to yield relatively stable local structures with high population time. Next, they interact with each other to form larger structural units. These interactions follow a pathway which eventually leads to (marginally) stable native structure. Our simplified scheme (Fig. 1) is divided into three separate components. The first attempts to predict local structure elements by cutting the target sequence into a set of stand-alone folding fragments covering (almost) the entire sequence. A conformation is assigned to each fragment to form structural units (SUs). The second component models the assembly of the local structural elements by a combinatorial assembly of these SUs. The final component models the local adjustments of the entire structure to reach the minimum energy conformation. CombDock is an implementation of the second stage. The first stage may be implemented either by (i) comparative modelling techniques applied to segments of the target sequence, (ii) a BBs database approach (Haspel *et al.*, 2003), or (iii) *ab initio* predictions (on segments of the target sequence).

While the prediction of the SUs may consist of known structures, the arrangement suggested by the CombDock algorithm may yield a fold different from those from which the SUs coordinations were extracted, and even may obtain structures that are not yet present. Indeed, the input type III examples for both domains and BBs show that even if the SUs are taken from proteins with different overall structures, CombDock was able to predict a near native structural model for the target protein. Notice that unlike the Rosetta method (Simons *et al.*, 1999) which uses a database of short segment structures for the conformational search, here we propose to predict structures for segments of the target sequence which fold into relatively stable structural units (BBs). We expect these segments to be at least 15 residues, but we do not expect them to be of the same length. Indeed
the BB lengths can be as long as a few hundreds of residues (Tsai and Nussinov, 2001; Fischer and Marqusee, 2000). Real structure predictions pose many challenges for the combinatorial assembly approach presented here. The reliability of structural units conformation prediction is limited and the linkers might be longer. We hope that future improvements of the different modules of CombDock will overcome these obstacles. The CombDock performance is currently being studied on a larger dataset.

CONCLUSIONS

Here we have presented an algorithm for the automated assembly of protein substructures. The algorithm gives a heuristic solution to a computationally hard problem (NPC). Our method, CombDock, addresses the second stage of a structure prediction scheme (Fig. 1) which is based on the hierarchical nature of protein folding.

The algorithm was able to predict near-native assemblies for various examples of both domains and building blocks with different levels of distortion. We have shown the power of the combinatorial docking approach and its advantage over the pairwise docking approach. This advantage arises from the fact that pairwise docking optimizes the interaction between two units, while the combinatorial docking optimizes the whole complex.

This CombDock prototype can clearly be improved. However, the running time, the quality of predictions and its low sensitivity to input distortion already suggest that CombDock may be used in protein structure prediction if the local structural units are given. The structural units may be predicted models, or experimentally solved substructures. In both cases, the method may help to obtain a structural model for much larger proteins than the current techniques can handle.

ACKNOWLEDGEMENTS

This research has been supported in part by the ‘Center of Excellence in Geometric Computing and its Applications’ funded by the Israel Science Foundation (administered by the Israel Academy of Sciences) and by grants of Tel Aviv University Basic Research Foundation. The research of H.J. Wolfson has been partially supported by the Hermann Minkowski-Minerva Center for Geometry at Tel Aviv University. The research of H. Benyamini has been supported by the Eshkol Fellowship funded by the Israeli Ministry of Science. The research of R. Nussinov has been funded in whole or in part with Federal funds from the National Cancer Institute, National Institutes of Health, under contract number NO1-CO-12400. The content of this publication does not necessarily reflect the view or policies of the Department of Health and Human Services, nor does mention of trade names, commercial products, or organization imply endorsement by the U.S. Government.

REFERENCES


