Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

PATRICK MAY AND THOMAS STEINKE

THESEUS– Protein Structure Prediction at ZIB

ZIB-Report 06-24 (November 2006)

THESEUS– Protein Structure Prediction at ZIB

Patrick May and Thomas Steinke

Computer Science Research, Dept. Computer Science, Zuse Institute Berlin (ZIB),

Junior Research Group "Alignment and Threading on Massively-Parallel Computers"

Berlin Center for Genom Based Bioinformatics (BCB)

November 6, 2006

Abstract

THESEUS, the ZIB threading environment, is a parallel implementation of a protein threading based on a multi-queued Branch-and-Bound optimal search algorithm to find the best sequence-to-structure alignment through a library of template structures. THESEUS uses a template core model based on secondary structure definition and a scoring function based on knowledge-based potentials reflecting pairwise interactions and the chemical environment, as well as pseudo-energies for homology detection, loop alignment, and secondary structure matching. The threading core is implemented in C++ as a SPMD parallelization architecture using MPI for communication. The environment is designed for generic testing of different scoring functions and search algorithms. A validation of the structure prediction results has been done on the basis of standard threading benchmark sets. THESEUS successfully participated in the 6th Critical Assessment of Techniques for Protein Structure Prediction (CASP).

Contents

1 Introduction

In post-genomics era protein structure prediction is still one of the major challenges in bioinformatic research because of the need to assign biological function to the thousands of uncharacterized transcribed genes discovered by the various genome sequencing projects. It is a fundamental axiom of molecular biology that the three-dimensional structure of a protein determines its function [1]. Although experimental methods like X-ray crystallography, NMR spectroscopy or cryo electron microscopy are providing high-resolution 3D structure information, they are still expensive due to wet-lab costs and expenditure in time. Existing computational methods for protein structure prediction can be split into the following categories:

- *Ab initio* methods (see, e.g., [2, 3]) adopt approximate physicochemical methods for the evaluation of the interaction of structural objects (atoms, residues etc.) and explore the phase space to find stationary states on the energy hyper-surface being of biological importance.
- Comparative modeling methods are based on identifying structural similarities with known 3D structures.

Comparative modeling is based on the observation that proteins with similar 3D structures exhibit, in general, similar functions. This is done by detecting close (homology modeling) or remote (fold recognition or threading) sequence homology between two sequences with inherent structural similarity. Fold recognition methods make a structure prediction for the target amino-acid sequence by recognizing a structural template structure representing a native fold. The template-based modeling approach will become increasingly useful for solving structures of proteins as more protein structures become available. For a recent review on fold recognition methods, see [4].

The basic idea behind threading is, that there are only a limited and rather small number (in the range of thousands) of different folds, also named architectures or *cores*, and that the various preferences of the different amino acids provide sufficient information to discriminate between different folds [5]. Threading methods usually consist of four components [6]:

1. a template structure library of known *cores*,
2. an empirical scoring function measuring the fitness of the target sequence for the *cores*,
3. a combinatorial search algorithm to optimally align the target sequence onto the *cores*, and
4. a statistical analysis to assess the significance of an optimal sequence-to-structure alignment found.

For each statistical significant sequence-to-structure alignment, the residues of the target sequence are predicted to have the backbone coordinates of the aligned residues in the template structure. Since fold recognition uses structural information in addition to sequence-based methods alone, it is often more effective than sequence-based methods like BLAST [7] or PSI-BLAST [8] for identifying native like folds.

In this work, we are introducing the THESEUS environment for protein structure prediction. THESEUS is a parallel implementation of a protein threading based on a multi-queued Branch-and-Bound optimal search algorithm to find the best sequence-to-structure alignment through a library of known structures. In section 2 we describe the template model based on secondary structure definition, the scoring function based on knowledge-based potentials and pseudo-energies, the search algorithm to find an optimal alignment between the target sequence and one template structure, and how we determine significant threading alignments from all alignments against the template library. Section 3 deals with the implementation details of the THESEUS threading environment. In section 4 we describe the results for several fold recognition benchmark sets, the participation at the 6th experiment on Critical Assessment of Techniques for Protein Structure Prediction (CASP) in 2004, and the performance tests in comparison with two other threading algorithms.

2 Protein Threading

2.1 Problem Definition

The optimal protein threading problem (OTP) is a variant of the general problem predicting the three-dimensional structure of a given amino acid sequence (the *target* sequence). *Threading* can be defined as a sequence-to-structure alignment between a target sequence S of unknown structure and a given *template* structure T_m. The alignment is given by a valid mapping $\vec{t}$ of a *template model* $\mathcal{M}_{T_m}$ (see subsections 2.2 and 2.4) onto the target sequence. The similarity between target sequence and template structure is scored by a given energy function f (see subsection 2.5). Since the scoring function uses pseudo-energies or knowledge-based potential, the OTP is defined as determining the sequence-to-structure alignment with minimal energy:

$$f(\vec{t}) = \min_{\vec{u} \in \tau} f(\vec{u}) \quad , \tag{1}$$

where τ represents the total search space of all valid threadings, with $\vec{u}$ being a member of it (see subsection 2.2).

2.2 The Template Model

To avoid the computational costs of an full atomic description of protein structures, the three-dimensional coordinates are replaced by an abstract, simplified template model. The definition of a template model $\mathcal{M}_{T_m}$ for a given protein structure T_m is as follows (Fig. 1): A template structure is a linear series of *core segments* that are connected via *loops*, representing the *core* of a protein structure. Each core segment represents a part of a secondary structure, namely an α-helix or a β-sheet. Each core segment $core_i$ $(head_i, tail_i)$ of a template is defined by the two parameters *head* and *tail*, where *head* is its start-position in the template and *tail* denotes its end-position in the template. Each $core_i$ consists of $(tail_i - head_i) + 1$ *core elements* or amino acids. The region between $tail_i$ and $head_{i+1}$ is called loop region $(loop_i)$ connecting core segments i and $i+1$. The M core segments of the template structure are enumerated from 1 to M in N- to C-terminal orientation and they must not overlap in the template sequence. This can be expressed mathematically as

$$1 \leq head_1 \leq tail_1 < head_2 \leq tail_2 < \ldots < head_M \leq tail_M \leq n \quad , \tag{2}$$

where n is the template sequence length.

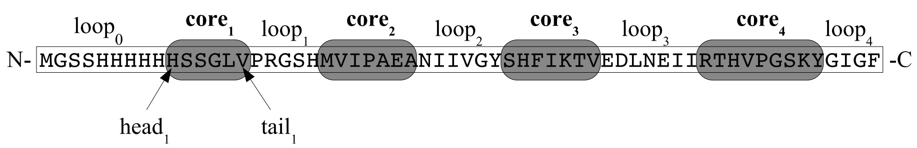


Figure 1: Template with four core segments and its five surrounded loops.

The template model $\mathcal{M}_{T_m}$ for a template structure T_m of sequence length m $(tm_1, \ldots, tm_m)$ with M core segments $(core_1, \ldots, core_M)$ and a target sequence $S =$

$(s_1,\ldots,s_n)$ of length n can be defined as a tuple $\mathcal{M}_{T_m} = (M, \vec{c}, \vec{\lambda}, \vec{l}_{min}, \vec{l}_{max}, I)$ with

$$\begin{aligned}
\vec{c} &= (c_1,\ldots,c_M) \ , &(3)\\
\vec{\lambda} &= (\lambda_0,\ldots,\lambda_M) \ , &(4)\\
\vec{l}_{min} &= (l_0^{min},\ldots,l_M^{min}) \ , &(5)\\
\vec{l}_{max} &= (l_0^{max}\ldots,l_M^{max}) \ , &(6)\\
I &= OCMG(T_m) \ , &(7)
\end{aligned}$$

and

$$m \in \mathbb{N}, \vec{c} \in \mathbb{N}^M, \vec{\lambda} \in \mathbb{N}^{M+1}, \vec{l}_{min} \in \mathbb{N}^{M+1}, \vec{l}_{max} \in (\mathbb{N} \cup \{\infty\})^{M+1} \ ,$$

to hold

$$\|T_m\| = m = \lambda_0 + \sum_{i=1}^{M} c_i + \lambda_i \ . \quad (8)$$

The length of core segment $core_i$ is given by c_i and the length of $loop_i$ by λ_i. I is given by the pairwise interaction graph $OCMG(T_m)$ representing pairwise structural contacts in T_m as defined in section 2.3. The two vectors $\tilde{l}_{min}$ and $\tilde{l}_{max}$ are denoting the minimal and maximal allowed loop lengths for a mapping of template model $\mathcal{M}_{T_m}$ onto the target sequence S, that can be calculated from the Euclidean distances between the cores of T_m and the length of the target sequence S:

$$\begin{aligned}
\forall i \in [1\ldots M-1] : l_i^{min} &= \left\lceil \frac{\delta^3(C_\alpha(tail(i)) - C_\alpha(head(i+1)))}{\triangle} \right\rceil \ ,\\
\forall i \in [1\ldots M] : l_i^{max} &= n - (l_0^{min} + \sum_{j=1}^{i} c_j + l_j^{min}) \ ,
\end{aligned}$$

where $C_\alpha(tail(i))$ and $C_\alpha(head(i+1))$ are the coordinates of the last C_α atom of $core_i$ and the first C_α atom of $core_{i+1}$, respectively. δ^3 is the Euclidean distance between the two three-dimensional vectors of the two C_α atoms, and $\triangle$ describes the averaged Euclidean distance between two neighbouring C_α atoms in amino acids. Since there is no information about the structure before the first and after the last core segment added to the template model, these values are set to $l_0^{min} = l_M^{min} = 0$.

A threading is then defined as a mapping of the template model onto the positions of the target sequence. Given a template model $\mathcal{M}_{T_m}$ and a target sequence S, than a *threading* of S through $\mathcal{M}_{T_m}$ is a vector

$$\vec{t} = (t_1,\ldots,t_M) \in [1\ldots n]^M \ ,$$

where t_i denotes the position of $tail(i)$ on the target sequence S.

For such a mapping (sequence-to-structure alignment) there exist some restrictions. The following constraints simply adopt the definitions of the template concerning ordering and overlapping to the mapped target sequence:

1. All core segments of M_{T_m} have to be placed onto S.

2. The core segments must be mapped to the target sequence with the same order as they have in the template sequence, i.e., $core_{i+1}$ must be placed after $core_i$ onto the target sequence.

3. The regions of the target sequence mapped to core segments must not overlap.

The mathematical definition of these restrictions is given in Eq, 13 in the subsection 2.4. Examples for restriction of valid threadings and invalid threadings are shown in Fig. 3 and 4.

During the sequence-to-structure alignment, gaps are only tolerated in loop regions between two core segments or in loop regions before the first ($loop_0$) or after the last core segment ($loop_M$, in case of M cores in template), while the alignment in core segments must be gap-free (gapped block). This rule represents the biological fact that core segments are highly conserved and therefore insertions or deletions are not supposed to take place inside the core segments. Gaps permitted anywhere in the structure would result in much larger search space sizes than gapped block.

2.3 Contact Graph

The score for each threading as expressed by the scoring function (see section 2.5) partly relies on contacts between residues in the template structure. Only interactions of residues inside core segments are taken into account. Interacting residues might be located in the same core segment or in two different, interacting core segments. Interactions with residues located in loop regions are not considered as it is believed that these interactions are relatively insignificant for fold recognition [9].

A *contact map* gives a detailed representation of the three-dimensional fold of a protein [10]. The *interaction graph* of a protein structure T_m of sequence length m (Fig. 2a) can be represented by the undirected *original contact map graph* $OCMG(T_m) = (V_O, E_O)$, where $E_O \subseteq V_O^2$ with V_O as the set of all residues of T_m, i.e.,

$$V_O = \{tm_i | 1 \leq i \leq m\} \quad , \tag{9}$$

and E_O as the set of edges with

$$\forall i, j \in V_O : e^o_{ij} = \begin{cases} 1 & \text{if } tm_i \text{ and } tm_j \text{ are in contact,} \\ 0 & \text{else} \quad . \end{cases} \tag{10}$$

$OCMG(T_m)$ represents the interaction graph I of the template structure T_m in Eq. 7. For reasons of simplicity we define for the template model $\mathcal{M}_{T_m}$ additionally the *simplified contact map graph* $SCMG(T_m) = (V_S, E_S)$ with $E_S \subseteq V_S^2$ for modeling the alignment. Here V_S is defined as the set of core segments

$$V_S = \{C_i | 1 \leq i \leq M\} \quad , \quad and \tag{11}$$

E_S is the set of edges with

$$\forall i, j \in V_C : e^s_{ij} = \begin{cases} 1 & \text{if } \exists\, tm_i \in core_i, tm_j \in core_j : e^o_{ij} = 1 \quad , \\ 1 & \text{if } j = i+1 \quad , \\ 0 & \text{else} \quad . \end{cases} \tag{12}$$

Each node in $SCMG(T_m)$ represents a core segment ($core_i$) of the protein template while each edge $e^s_{k,l} \in E_S$ represents an interaction between the two core segments k and l which denotes that there exists at least one pair of interacting residues located in $core_k$ and $core_l$. All interacting core segments and also all neighboring core segments are connected via edges.

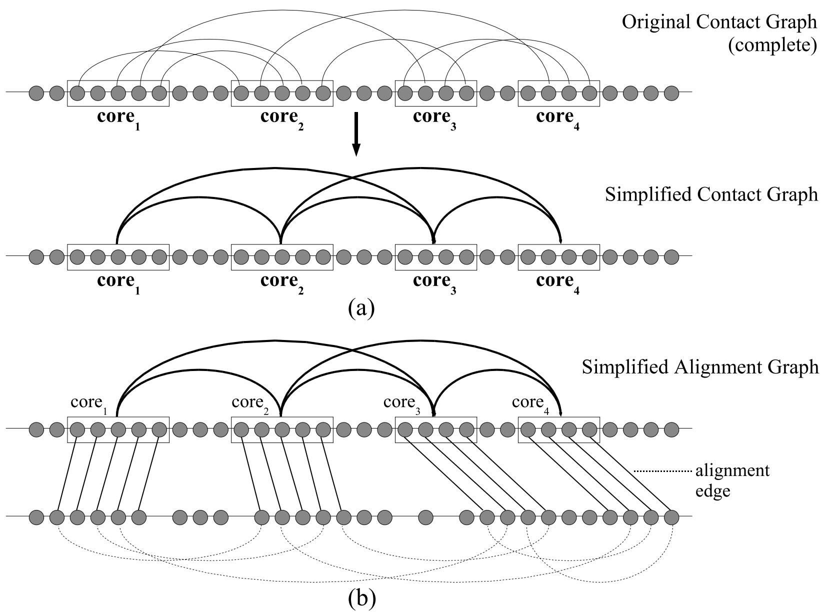


Figure 2: Source: [11]. (a): Template Contact Graph: Each residue of the template structure is represented by a circle, each core segment is marked as a box framing all residues located in that core segment. In the *OCMG* all interactions between residues are expressed by an edge between these two residues. In the *SCMG* these edges are replaced by only one edge connecting two core segments if there is at least one interaction edge in the original contact graph connecting residues located in these two core segments or the core segments are neighbors. (b): Alignment Graph: Each residue inside a core segment of the contact graph is aligned to exactly one residue of the target sequence symbolized by an alignment edge. Dashed arcs connect two residues of the sequence that are aligned to two interacting residues of the template graph and correspond to interactions in the original contact graph.

The interaction graph $SCMG(T_m)$ describes which core segments contain core elements that are interacting in some biological sense, i.e., that core elements are spatial neighbours in the three-dimensional protein structure. The interaction graph $OCMG(T_m)$ contains all interactions between core elements. $SCMG(T_m)$ can be derived directly from $OCMG(T_m)$ and the template model $\mathcal{M}_{T_m}$.

2.4 Sequence-to-Structure Alignment

A threading or sequence-to-structure alignment of a target sequence S and a template model $\mathcal{M}_{T_m}$ can also be expressed as a bipartite *alignment graph* as shown in Fig. 2b. In addition to the simplified contact graph from Fig. 2a the alignment graph contains another set of nodes $(s_1 \ldots s_n)$ representing residues 1 to n of the target sequence S. If residue i of the template structure is aligned to residue k of the target sequence there is an alignment edge $a_{i,k}$ connecting them. If residues i and j of the template structure are aligned to residues k and l in the target sequence this is equivalent to the fact that the two alignment edges $a_{i,k}$ and $a_{j,l}$ exist in the alignment graph. If the two edges $a_{i,k}$ and $a_{j,l}$ exist, and there exists an interaction edge $e_{i,j}$ in the complete contact map graph, then the two residues k and l of the target sequence are supposed to have an interaction too, symbolized by the dashed arcs in the alignment graph. Each of the alignment edges in one threading is weighted with a value representing the g_1 term of the energy function in Eq 15 introduced in the next section. The dashed arcs representing interactions are weighted with a value representing the g_2 term or pairwise interaction term of the energy function (see also next section). The score of one threading is then calculated by summing the weights of all alignment edges and interaction edges (dashed arcs) of the alignment graph. As mentioned already there are some restrictions for a placement to be valid. Non-overlapping core segments as well as the preservation of the core segment order correspond to non-conflicting alignment edges in the alignment graph. In the graphical representation of the alignment graph conflicting alignment edges can be defined as crossing edges (Fig. 3 and 4). A threading $\vec{t} \in \tau$ is called a valid threading from the set of all valid threadings τ if the following constraints are fulfilled:

$$\begin{aligned} 1+l_0^{min} &\leq t_1 \leq 1+l_0^{max} \\ t_i+c_i+l_i^{min} &\leq t_{i+1} \leq t_i+c_i+l_i^{max}, \qquad 1 \leq i < M \\ t_M+c_M+l_M^{min} &\leq n+1 \leq t_M+c_M+l_M^{max} \quad , \end{aligned} \tag{13}$$

where t_i denotes the position in the target sequence of length n that is aligned to $core_i$. The variables l_i^{min} and l_i^{max} are the minimal and maximal length assigned to $loop_i$ and c_i is the length of $core_i$:

$$c_i = tail_i - head_i + 1 \quad .$$

We say that $core_i$ is aligned to t_i if $core_i$ is aligned to target sequence positions $t_i \ldots t_{i+(c_i-1)}$.

These constraints directly guarantee that all pairs of neighboring alignment edges are not in conflict. It can be proven that this fact transitively guarantees for all pairs of alignment edges not to be in conflict (transitivity of non-conflict [11]). The constraints also guarantee that loop lengths are within their boundaries (l_i^{min}, l_i^{max}) and all core segments are aligned to sequence S.

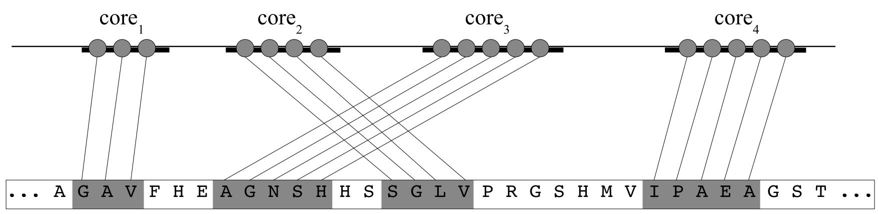
2.5 The Energy Function

In section 2.4 the alignment graph was introduced and the score of a valid threading was calculated as the sum of weights of all alignment and interaction edges (dashed arcs in Fig. 2b). These edge weights are assigned by the energy or scoring function f. The overall energy function represents the sum of energies over all alignment and interaction edges. The energy of one specific threading $t \in \tau$ is defined as:

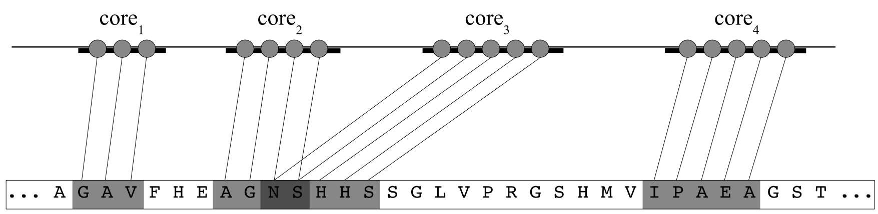
$$f(\vec{t}) = w_m E_m + w_{single} E_{single} + w_p (E_{p1} + E_{p2}) + w_g E_g + w_{ss} E_{ss} \tag{14}$$

The E_m term defines a mutation energy, E_{single} is the singleton energy , E_{p1} is the pairwise interaction energy term that considers interactions between residues located in the same core segment, E_{p2} is the energy term that considers interactions between different core segments. E_g represents the loop scoring and E_{ss} the secondary structure prediction term. Each w_i defines a weight factor for the corresponding energy term. In this section at first the formula for each term is presented followed by a brief introduction into the theory and construction of each energy term used in THESEUS.

Choosing an optimal set of weighting factors for the scoring function in Eq. 14 is crucial for the performance of THESEUS. The weight factors w_m, w_{single}, w_{pair}, w_g



(a)



(b)

Figure 3: Equivalence of conflicting (crossing) alignment edges in alignment graph model and invalid threading. Minimal and maximal loop length are not considered for reasons of clarity. (a) Alignment Graph showing invalid alignment (threading) where order of core segments in the template is not preserved in aligned target sequence. It is shown that this sort of invalid threading causes crossing alignment edges (conflicting edges). (b) Another invalid threading with overlapping (overlapping region in dark grey) core segments in target sequence also results in crossing alignment edges.

and w_{ss} were optimized by Taguchi orthogonal arrays [12] using the Fischer benchmark set [13] consisting of 68 target and 301 template structures (see section 4.1).

The energy function given in Eq. 14 can easily be transformed into a very common form that Lathrop and Smith called the "general scoring function" [5]. This is obtained by including all non-pairwise scoring terms into one function $g_1 \in \mathbb{N}^2$ and all pairwise core segment interaction terms into a second function $g_2 \in \mathbb{N}^4$. The resulting general scoring function is defined as

$$E_{total} = \sum_{i=1}^{M} g_1(i,t_i) + \sum_{i=1}^{M} \sum_{i<j} g_2(i,j,t_i,t_j) \quad , \tag{15}$$

with

- $g_1(i,t_i)$ means, placing the $core_i$ onto position t_i, and
- $g_2(i,j,t_i,t_j)$ means, placing $core_i$ onto position t_i and $core_j$ onto position t_j.

g_2 has the following predicate:

$$g_2(i,j,t_i,t_j) \neq 0 \Leftrightarrow e^s_{ij} = 1, e^s_{ij} \in SCMG(T_m) \quad . \tag{16}$$

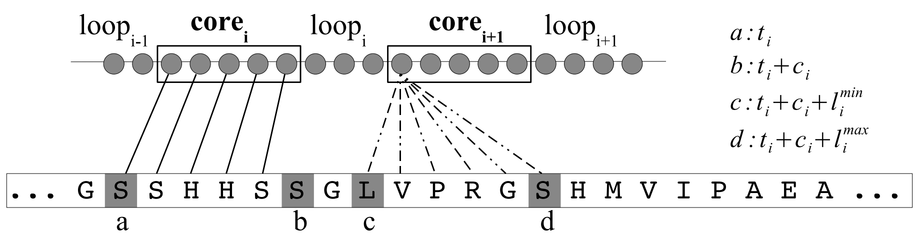


Figure 4: Visualization of the restrictions for valid threadings (Eq. 13). The dashed arcs show all valid alignment positions for $core_{i+1}$. In this example l_i^{min} is 2 and l_i^{max} is 7

Higher-order interactions can be included by extending the definition of the interaction graph to that of an interaction hypergraph. Furthermore, one has to introduce $2n$-ary functions g_n in order to implement n-ary interactions. If the template has M core segments, then $n \leq M$. Hence, the fully general scoring function is

$$\begin{aligned} E_{total} = &\sum_{i_1} g_1(i_1, t_1) + \sum_{i_1} \sum_{i_2 < i_1} g_2(i_1, i_2, t_1, t_2) + \ldots \\ &+ \sum_{i_1} \sum_{i_2 < i_1} \cdots \sum_{i_{M-1} < i_M} g_M(i_1, i_2, \ldots, i_M, t_1, t_2, \ldots, t_M). \end{aligned} \tag{17}$$

Triplet or higher interactions would arise in treatment of steric packing among multiple core segments, linked constraint equations on structural environments and detailed geometric or environment modeling [14]. Such interactions are computationally very expensive and yet there is no existing threading implementation considering triplet or higher interactions.

Mutation energy. The term E_m describes the compatibility of substituting one amino acid from the template structure by an amino acid from the target sequence. This energy is calculated using amino acid substitution matrices like PAM [15] or BLOSUM [16]. While in some implementations only the score in the substitution matrix is considered, the THESEUS implementation also makes use of a position specific frequency matrix (PSFM) calculated by PSI-BLAST [8]. PSI-BLAST constructs this matrix automatically by at first performing a BLAST search for the target sequence followed by a multiple alignment of the target sequence with the sequences of the best hits from which a position specific scoring matrix and a PSFM is generated. This procedure is iterated a user-specified number of times or until it converges. For a target sequence of length n the result is an $n \times 20$ matrix P where each column k describes the occurring frequency of all 20 amino acids at position k in the target sequence. The mutation function $mut(i,k)$ represents the mutation energy for template position i being aligned to position k in the target sequence:

$$mut(i,k) = \sum_{a=1}^{20} P(k,a) M(tm_i, a).$$

For each pair of aligned amino acids of target and template sequence we are summing up the products of the occurring frequency of amino acid a at target sequence position k and the substitution compatibility of amino acid a and the amino acid at position i in the template (tm_i) taken from substitution matrix M over all possible amino acids.

Currently the THESEUS pipeline is using the PAM250 substitution matrix which is supposed to be the best performing matrix for protein threading [13, 17]. If core segment i is aligned to sequence position t_i then $Mutation(i,t_i)$ is simply the sum of the mutation energies over all core segment positions c_i:

$$Mutation(i,t_i) = \sum_{r=0}^{c_i-1} mut(head_i + r, t_i + r).$$

The total mutation energy term E_m is then given by:

$$E_m = \sum_{i=1}^{M} Mutation(i,t_i) \quad ,$$

where M is the number of core segments.

The described sequence-to-profile alignment score is the mutation score averaged over homolog sequences. It has been shown that averaging energies over homologs can improve the alignment accuracy of fold recognition methods [18, 19]. The PSFM is applied in the same manner to all other scoring terms in Eq. 14.

Singleton and pairwise scoring terms. The terms E_{single} and E_p are knowledge based potentials. These potentials of mean force are estimated by applying the inverse Boltzmann relation to a given set of known 3D protein structures [20]. The advantage of knowledge based potentials is their small computational expense compared to more complex potential expressions used in today's force-fields. The disadvantage of the knowledge based potentials is the fact that they are derived from a specific dataset of proteins. Therefore they depend on the quality of the set and might vary considerably using different datasets.

The singleton term (singleton energy) characterizes the fitness of a single amino acid in its local chemical environment. The local environment of an amino acid is described by its affiliation to a secondary structure type and its solvent accessibility. Three secondary structure states according to the DSSP [21] classification

- helix (α-helix, 3_{10}-helix,π-helix)
- sheet (extended β strand, residue in isolated β-bridge)
- loop (all other residues)

and three states for the solvent accessibility

- buried (solvent accessibility $\leq 7\%$)
- intermediate ($7\% <$ solvent accessibility $\leq 37\%$)
- accessible (solvent accessibility $> 37\%$)

have been defined. The singleton energy of amino acid i, placed in environment with secondary structure type ss and solvent accessibility type sol is calculated using the formula

$$e_{single}(i,ss,sol) = -k_b T \ \ln \frac{N(i,ss,sol)}{N_E(i,ss,sol)} \quad ,$$

Table 1: The THESEUS singleton scoring matrix with singleton energy for each amino acid (first column) in all 9 environments as described in the text

	Helix			Sheet			Loop		
	Buried	Inter	Exposed	Buried	Inter	Exposed	Buried	Inter	Exposed
ALA	-0.578	-0.119	-0.160	0.010	0.583	0.921	0.023	0.218	0.368
ARG	0.997	-0.507	-0.488	1.267	-0.345	-0.580	0.930	-0.005	-0.032
ASN	0.819	0.090	-0.007	0.844	0.221	0.046	0.030	-0.322	-0.487
ASP	1.050	0.172	-0.426	1.145	0.322	0.061	0.308	-0.224	-0.541
CYS	-0.360	0.333	1.831	-0.671	0.003	1.216	-0.690	-0.225	1.216
GLN	1.047	-0.294	-0.939	1.452	0.139	-0.555	1.326	0.486	-0.244
GLU	0.670	-0.313	-0.721	0.999	0.031	-0.494	0.845	0.248	-0.144
GLY	0.414	0.932	0.969	0.177	0.565	0.989	-0.562	0.299	0.601
HIS	0.479	-0.223	0.136	0.306	-0.343	-0.014	0.019	-0.285	0.051
ILE	-0.551	0.087	1.248	-0.875	-0.182	0.500	-0.166	0.384	1.336
LEU	-0.744	-0.218	0.940	-0.411	0.179	0.900	-0.205	0.169	1.217
LYS	1.863	-0.045	-0.865	2.109	-0.017	-0.901	1.925	0.474	-0.498
MET	-0.641	-0.183	0.779	-0.269	0.197	0.658	-0.228	0.113	0.714
PHE	-0.491	0.057	1.364	-0.649	-0.200	0.776	-0.375	-0.001	1.251
PRO	1.090	0.705	0.236	1.246	0.695	0.145	-0.412	-0.491	-0.641
SER	0.350	0.260	-0.020	0.303	0.058	-0.075	-0.173	-0.210	-0.228
THR	0.291	0.215	0.304	0.156	-0.382	-0.584	-0.012	-0.103	-0.125
TRP	-0.379	-0.363	1.178	-0.270	-0.477	0.682	-0.220	-0.099	1.267
TYR	-0.111	-0.292	0.942	-0.267	-0.691	0.292	-0.015	-0.176	0.946
VAL	-0.374	0.236	1.144	-0.912	-0.334	0.089	-0.030	0.309	0.998

where k_b is the Boltzmann constant, T the absolute temperature and $N(i, ss, sol)$ is the number of amino acids of type i in the environment ss and sol.

$N_E(i, ss, sol)$ is defined as:

$$N_E(i, ss, sol) = \frac{N(i)N(ss)N(sol)}{N^2}$$

denoting the estimated number of amino acids of type i in the defined environment. All numbers were defined by counting each occurrence in a representative database. Currently THESEUS uses the sequence-independent, representative protein structure database derived by Fischer *et al.* [13, 22].

Combining each of the three states of secondary structure and solvent accessibility make nine different structural environments. The singleton energy for all amino acids can be stored in a 20×9 matrix as shown in Table 1.

The singleton energy averaged over homologs for $core_i$ aligned to position t_i is defined as

$$Singleton(i, t_i) = \sum_{k=0}^{c_i - 1} \sum_{a=1}^{20} P(head_i + k, a) e_{single}(a, ss(head_i + k), sol(head_i + k)) \quad ,$$

where $ss(y)$ is the secondary structure classification and $sol(y)$ the solvent accessibility of residue y in the template structure. The singleton energy term is given by:

$$E_{single} = \sum_{i=1}^{M} Singleton(i, t_i).$$

The interaction potential measures the preference of pairs of amino acids to be in a short distance in 3D structure. A protein's conformation and its fold respectively are determined by weak non-covalent interactions of residues that are close in 3D space. In the energy function an interaction of two amino acids is assumed, if

- the distance between their C_β-atoms is within a specified cutoff distance which is set to 7Å in THESEUS, and
- their distance on the sequence is greater than 2 residues.

The interaction potential of a pair of two amino acid types i and j is calculated as:

$$e_{pair}(i, j) = -k_b T \ \ln \frac{M(i, j)}{M_E(i, j)} \quad ,$$

with $M(i, j)$ denoting the occurrence frequency of pairs of amino acid types i and j within the cutoff distance found in the protein structure database. The reference state $M_E(i, j)$ is defined as

$$M_E(i, j) = \frac{M(i)M(j)}{M} \quad ,$$

denoting the estimated number of pairs assuming that amino acid types i and j are forming independently interactions within the cutoff distance. $M(x)$ is the number of pairs within cutoff where at least one amino acid is of type x, and M is the total number of pairs within cutoff distance. Table 2 shows the 20×20 interaction potential matrix from Xu et al. [9]. The pairwise energy $Pair(i, j, t_i, t_j)$ for two core segments averaged over homologs is defined as:

$$Pair(i, j, t_i, t_j) = \sum_{k=0}^{c_i-1} \sum_{l=0}^{c_j-1} \sum_{a_i, a_j=1}^{20} P_i(t_i + k, a_i) P_j(t_j + l, a_j) e_{pair}(a_i, a_j) co_{ij}(k, l) \quad ,$$

with the boolean function

$$co_{ij}(k, l) = \begin{cases} 1 & \text{if } (head_i + k, head_j + l) \in E_O(OCMG), \\ 0 & \text{else} \ , \end{cases}$$

where *OCMG* represents the original contact map graph defined in section 2.3.

Then the total pairwise energy for interactions within a single core segment E_{p1} is given by:

$$E_{p1} = \sum_{i=1}^{M} Pair(i, i, t_i, t_i) \quad ,$$

and the total pairwise interaction energy for interactions between two distinct core segments by:

$$E_{p2} = \sum_{i=1}^{M-1} \sum_{j=i+1}^{M} Pair(i, j, t_i, t_j).$$

Table 2: The interaction potential matrix [9], amino-acid one-letter code in first column and bottom row.

A	-14																			
R	27	-2																		
N	10	-9	-43																	
D	22	-62	-42	2																
C	33	7	11	28	-192															
Q	3	-6	-20	7	19	-11														
E	12	-56	-14	14	12	1	7													
G	1	-8	-10	-27	9	-7	-3	-29												
H	6	-26	6	-45	19	27	-37	7	-45											
I	-11	11	35	32	15	24	29	18	29	-33										
L	-18	26	36	37	24	2	25	24	20	-16	-28									
K	12	31	-20	-56	25	-18	-67	10	5	19	18	12								
M	-7	30	31	21	5	3	14	1	-1	-1	-11	30	49							
F	-6	6	20	28	3	7	23	11	16	-10	-20	-2	-27	-21						
P	17	-3	-21	-3	11	-8	-10	-7	-6	37	22	-5	3	-2	-21					
S	17	-8	-22	-30	1	-16	-21	-19	-13	21	27	-6	19	11	-16	-18				
T	6	6	-23	-20	37	-15	-21	-7	-24	11	22	-2	16	28	-10	-22	-21			
W	5	-15	-2	10	5	-1	16	-7	-21	-2	8	3	-1	3	-43	13	9	-2		
Y	5	-13	5	27	6	-9	27	6	3	-16	-9	-31	-17	-1	-8	10	16	-10	-1	
V	-11	17	30	43	20	18	23	20	20	-23	-22	27	-5	-4	5	27	7	10	11	-32
	A	**R**	**N**	**D**	**C**	**Q**	**E**	**G**	**H**	**I**	**L**	**K**	**M**	**F**	**P**	**S**	**T**	**W**	**Y**	**V**

The secondary structure prediction term. The term E_{ss} measures the agreement between predicted secondary structure of the target sequence using PSIPRED [23] and the actual secondary structure of the template structure. PSIPRED assigns each position of the target sequence one of three secondary structure states (helix, strand, coil) together with a reliable score ranging from 0 to 9, where 0 means that the prediction is not reliable, and 9 very reliable. The score for a single position is then given by:

$$e_{sspred}(i,j) = \begin{cases} rel(i) & \text{if } pred(i) \text{ is equal to } ss(j), \\ -rel(i) & \text{else} \end{cases} ,$$

where $rel(i)$ is the reliability score of PSIPRED at position i, $pred(i)$ the predicted secondary structure state, and $ss(j)$ the secondary structure state of the aligned template structure position. Similar scoring functions for secondary structure are used in [24, 25]. The secondary structure prediction term for a single core segment is then defined as:

$$ssPred(i,t_i) = \sum_{k=0}^{c_i-1} e_{sspred}(t_i+k, head_i+k) \ ,$$

and the total secondary structure prediction score as:

$$E_{ss} = \sum_{i=1}^{M} ssPred(i,t_i).$$

The loop scoring term. The term E_g is built in two steps. Loops are per definition connecting two core segments or describing the sequence before the first or after the last core segment. First, a sequence alignment between the loop region in the target sequence and the corresponding loop region in the template is performed using a standard dynamic programming algorithm [26] and a PAM250 substitution matrix. Secondly, the chemical environment of the aligned residues is scored. The idea is that insertions and deletions will occur very frequently in loop regions. Often sequences with additional secondary structure are added so that a sequence alignment gap penalty is not able to capture the ability of a sequence region to be part of a loop region. Consequently, aligning a residue preferring a helical environment with a residue preferring an exposed loop environment should be penalized and aligning two residues preferring loops should be awarded.

The loop score for a single column of the alignment is defined as:

$$e_{loop}(i,j) = \begin{cases} e_{single}(i, ss(j), sol(j)) & \text{if } i \text{ and } j \text{ are aligned,} \\ 0 & \text{if } i \text{ or } j \text{ is a gap position} \end{cases} ,$$

where e_{single} describes the placement of residue i in the target sequence into the chemical environment of residue j in the template structure. The score for $loop_i$ is then given by:

$$Loop(i,b_i,e_i) = \sum_{i=b_i}^{e_i} \sum_{j=begin_i}^{end_i} e_{loop}(i,j) \quad ,$$

where b_i and e_i are the beginning and ending position in the target sequence and $begin_i$ and end_i the first and the last position of $loop_i$ in the template structure. The total loop score is defined as:

$$E_g = \sum_{i=0}^{M} Loop(i,b_i,e_i) \quad .$$

Alternatively, the user can use simple affine gap penalties introduced for sequence alignments, where the values of the gap opening g_o and the gap extension g_e penalty is user-specified:

$$Loop(i,b_i,e_i) = g_o + (e_i - b_i + 1)g_e \quad .$$

2.6 The Branch-and-Bound Search Algorithm

According to the formulated energy function the best (optimal) threading for a template and a target sequence is the one that minimizes the score f and is valid concerning formulated restrictions, as given in Eq. 1 in section 2.1. The task of finding the optimal threading is called the *optimal threading problem* (OTP) as formulated in section 2.1. It was proven by Lathrop that the optimal threading problem is NP-hard if pairwise interaction terms are considered [27]. There are two main approaches to solve this problem. One is to use heuristics that do find a good threading in polynomial time. But the solution found is not necessarily the optimal solution and therefore these algorithms sometimes do not solve the OTP exactly. The other strategy is to implement exact algorithms that find the optimal solution in acceptable time for most threadings but might have exponential runtime for some pairs of template structure and target sequence. The THESEUS threading environment implements the improved multi-queue variant of the Branch-and-Bound algorithm from Lathrop and Smith [5, 28]. The idea of Branch-and-Bound algorithms is to minimize the search space by splitting it subsequently into subsets (*branch step*). For each of the generated subsets of the original search space a lower bound within this subset is determined (*bound step*). The next branch step is then performed on the subset with the best lower bound.

In [5] Lathrop and Smith define the search space of the OTP as sets of valid threadings

$$T[\vec{b},\vec{d}] = \{\vec{t} \in \tau \mid b_i \leq t_i \leq d_i\}$$

representing a hyper-rectangle, whose corners are the vectors $\tilde{b}$ and $\tilde{d}$. The variables b_i and d_i with $1 \leq i \leq M$ define an interval $[b_i, d_i]$ containing all allowed positions in the target sequence for the alignment of $core_i$, where M is the number of core segments. All threadings in $T[\vec{b},\vec{d}]$ are valid with respect to the restrictions (13). The splitting of a set of threadings $T = T[\vec{b},\vec{d}]$ is performed by choosing a core segment $core_s$ and a split position t_s^{split} inside $core_s$. The set T is then split into three subsets $T_<$, $T_=$ and $T_>$.

$$\begin{aligned} T_< &= \{\vec{t} \in T \mid b_s \leq t_s < t_s^{split}\} \\ T_= &= \{\vec{t} \in T \mid t_s = t_s^{split}\} \\ T_> &= \{\vec{t} \in T \mid t_s^{split} < t_s \leq d_s\} \quad . \end{aligned}$$

The splitting starts with the core segment that has the most interactions with other core segments. Every splitting partitions a hyper-rectangle into smaller hyper-rectangles, at least one of which with lower dimension, i.e., with one or more additional fixed core segments. A priority queue holds a list of all currently instantiated hyper-rectangles sorted by lower bound. The queue is instantiated with a single threading set covering the entire search space of legal threadings. At each iteration, the threading set having the currently lowest lower bound is removed from the queue. If it contains a single threading, i.e., all core segments are fixed, the global optimum is found and the algorithm terminates. Otherwise the hyper-rectangle representing a threading set is split, and the resulting threading sets are merged into the queue according their lower bound. The evaluation of lower bounds is probably the most important part of the algorithm. The more precisely the lower bound the more rapidly the search converges towards the optimal threading. Besides the strength of the lower bound it is also important that it can be computed efficiently. Formula (18) defines a function $lb(T[\vec{b},\vec{d}])$ for evaluation of lower bounds which can be computed in polynomial time and is applicable for all energy functions that are compatible to the general scoring function (15):

$$lb(T[\vec{b},\vec{d}]) = \sum_i \left[\min_{b_i \leq x \leq d_i} g_1(i,x) + \sum_{j>i} \min_{\substack{b_i \leq y \leq d_i \\ b_j \leq z \leq d_j}} g_2(i,j,y,z) \right] \quad . \tag{18}$$

Lathrop and Smith had explored several alternative forms of lower bounds and denoted the following lower bound, which is implemented in THESEUS, as the most efficient variant:

$$\begin{aligned} lb(T[\vec{b},\vec{d}]) = \min_{t \in T[\vec{b},\vec{d}]} \sum_i \Big[\qquad & g_1(i,t_i) + g_2(i-1,i,t_{i-1},t_i) \\ & + \min\nolimits_{u \in T[\vec{b},\vec{d}]} \textstyle\sum_{\|j-i\|>1} \frac{1}{2} g_2(i,j,t_i,u_j) \Big]. \end{aligned} \tag{19}$$

In [28] Lathrop introduces a modified variant of the Branch-and-Bound algorithm, that uses $m+1$ priority queues, where queue Q_k holds the threading set of vector subspace dimension $m-k$, i.e., k of m core segments are fixed. The threading sets in each queue are sorted by their lower bound. Normally, at the beginning all queues are empty except queue Q_m, that is instantiated by a threading set covering the whole search space. If some prior knowledge about placing single core segments is available, this knowledge can be used to initialize a queue of lower dimensionality. At each step the algorithm sweeps across the different queues, always beginning with the queue of highest dimensionality, popping the threading set with the lowest bound, splitting it, and reinserting the new threading sets into the queues according their dimensionality. The idea is that threading sets enclosing good solutions will tend to sort to the front of the queues, and their children generated by splitting will migrate quickly to lower dimensions representing more fixed core segments. Sweeping across the different queues from high to low dimensionality will yield an optimal solution rapidly: one sweep can be done in $O(m)$ lower bound evaluations and guarantees to produce a legal threading in Q_0 at the end. The number of sweeps can be specified by the user. In THESEUS we are starting with 10 sweeps. Thereafter, if the global optimal solution is not found, the algorithm operates on the most productive regions of the search space explored by the sweep iterations.

2.7 Global Optimal Threading

The result from the Branch-and-Bound search algorithm is an optimal alignment of the target sequence with one template structure. Then, we have to determine which of the alignments against the template library is the most significant alignment. The alignments are characterized by the score of the energy function in Eq. 14 measuring the fitness of the alignment of target sequence and the template structure. The value of this score depends on the lengths of the target sequence as well as the template sequence and additional structural properties of the 3D template structure [29]. Some confidence measures like, e.g., the p-value can be computed only if the underlying score distribution is known. However, this is not the case for known fold recognition methods [30]. The THESEUS threading environment currently supports three measures to determine the significance of a given sequence-to-structure alignment. First, it ranks all alignments against the template library according to the *raw score* denoting the score of an optimal threading. Second, we are using $z-scores$, by using the score of all threading alignments against the template library to normalize the threading scores. The mean $\hat{\mu}_{seq}$ and the standard deviation $\hat{\sigma}_{seq}$ can be estimated from the *raw scores* s of one target sequence seq versus the set of template structures STR_{seq} as

$$\hat{\mu}_{seq} = \frac{1}{|STR_{seq}|} \sum_{str \in STR_{seq}} s(seq, str) \tag{20}$$

$$\hat{\sigma}_{seq} = \frac{1}{|STR_{seq} - 1|} \sum_{str \in STR_{seq}} (s(seq, str) - \hat{\mu}_{seq})^2 \quad . \tag{21}$$

Then the *raw scores* s are normalized into $z-scores$ according

$$z(seq, str) = \frac{s(seq, str) - \hat{\mu}_{seq}}{\hat{\sigma}_{seq}} \quad . \tag{22}$$

We have experimentally determined that a $z-score$ value below -2.5 is very probable corresponding with a good sequence-to-structure alignment.

The third significance measure we are using is the *raw score gap* introduced by Sommer *et al.* [30],

$$sg(seq, str) = s(seq, str) - s(seq, next(str)) \tag{23}$$

that measures for a target sequence seq the difference of the target raw score of a template structure str and the next best raw score of a template structure belonging to a different fold class, $next(str)$. This measure can only be used if the template structures can be classified into different folding classes. Therefore, we are using the SCOP [31] classification scheme.

3 Implementation

3.1 Data Sources

The template fold library is built on SCOP [31] domains, which are available as ASTRAL [32] PDB-style files. Additionally, single chains of protein structures in PDB format can be included. Often it is not useful to align a target sequence against all available protein domains, so we use the SCOP40 non-redundant subset as standard template library. Here, all domains have at most a pairwise sequence homology of 40%. The secondary structure of the target sequence is predicted using PSIPRED [23]. The secondary structures of the template structures were obtained by the DSSP [21] method. The sequence-derived profiles for the target sequence and the template structures are both constructed by using PSI-BLAST [8]. This is done with five iterations against the non-redundant version of the UNIPROT [33] database. The database was filtered to remove low-complexity, transmembrane, and coiled-coil regions. The solvent accessibility of the template structures were determined with DSSP but using the quaternary structure files available from the PQS [34] database instead of the ASTRAL PDB-style files, because most of the domains are consisting of multiple domains. Quaternary structure is defined as that level of protein structure in which units of tertiary structure aggregate to form homo- or hetero-multimers. The template library and all preprocessed data is stored in a MySQL database.

3.2 Parallel Threading

Fold recognition by threading can be parallelized by assigning each of a subset of template structures to a different process. Our parallel threading core is implemented in C++ and uses MPI for message passing and POSIX threads. Two kind of parallel architectures are designed:

1. a Master-Slave (MS) version, and
2. a Single-Program-Multiple-Data (SPMD) version.

In the MS architecture the central component is the MySQL database. A master process or POSIX thread distributes each outstanding template structure to a slave process waiting for work. Based on a first-come-first-serve protocol a dynamic load balancing scheme can be realized. In the SPMD architecture the content of the MySQL template structure database is dumped into a binary file which is cloned on each compute node on a Linux cluster. The template structures are distributed in a static scheme amongst the MPI processes, i.e., each MPI process performs its own subset. Having all template structures processed, one MPI process gathers all results from the remaining concurrent MPI processes.

The SPMD approach is significant faster over the MS architecture (shown in Fig. 5: the squares relate to the SPMD and the diamonds to the MS architecture). The drawback of the MS architecture is the time determining database connections: the central database server can not timely satisfy the requests from all the slave processes. The SPMD architecture has the extra advantage of parallel I/O. To show the time efficiency of our implementation, we can process a protein sequence consisting of 573 amino acids against 37556 templates structures representing the whole SCOP template database in about 36 minutes on 32 CPUs on a IA32 Myrinet Linux cluster .

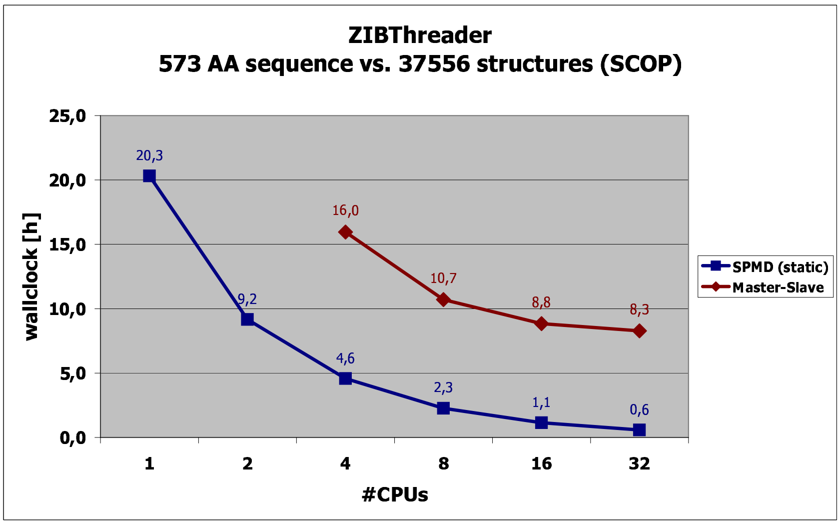


Figure 5: Performance of the two parallel threading architectures.

3.3 Structure Prediction Pipeline

We have integrated the THESEUS threading environment into a complete protein structure prediction pipeline to provide full-atom 3D models for protein sequences. The pipeline has been designed and implemented for the 6th CASP experiment in 2004 [35, 36] (see also section 4.3). In order to provide a fully automated protein prediction tool, the pipeline integrates various prediction and analysis steps. The whole pipeline is designed modular, so that improved methods and tools can be substituted in, as they become available. Fig. 6 shows the global pipeline architecture. The first

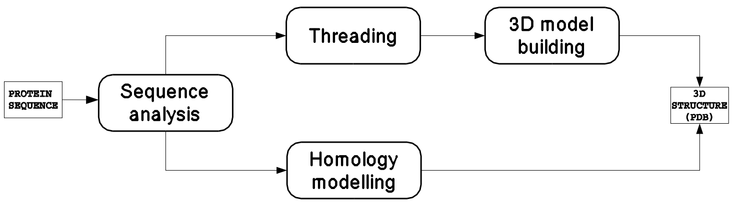


Figure 6: Schematic representation of the structure prediction pipeline.

step in the workflow is the identification of suitable template structures for homology modeling (Fig. 7, top). A sequence analysis sub-workflow is passed to search for homologous sequences with known structures. Successive PSI-BLAST searches are performed in order to find suitable templates. If no suitable template structure has been found in the PDB (Protein Data Bank [37]) database, a second PSI-BLAST search in the UNIPROT [33] database is initiated followed by parallel PSI-BLAST searches in the PDB database starting from the hits found in UNIPROT. If a structural template has been found, an atomic structural model will be generated with a suitable homology modeling tool. Here, we can plug-in any available method for homology modeling.

Currently, we are using the MODELLER [38] tool. For the CASP6 experiment (see section 4.3) methods developed in the Preissner group (Charité, Berlin [36]) were applied. If no suitable structural template is detectable the structure will be predicted through our protein threading implementation. The threading procedure (Fig. 7, bot-

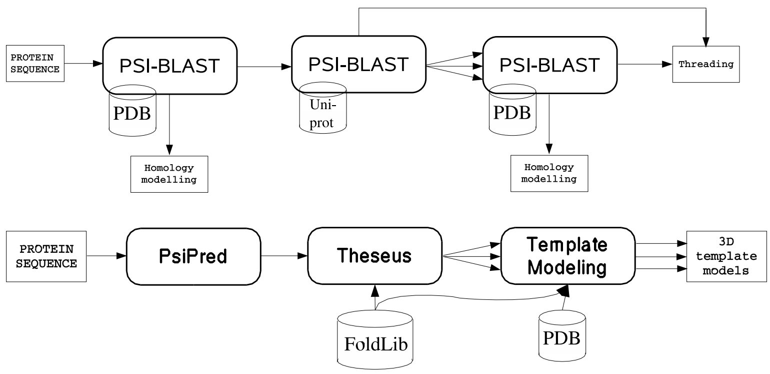


Figure 7: Sub-workflows: (top) Sequence analysis, (bottom) Threading

tom) starts with secondary structure prediction using PSIPRED [23]. PSIPRED provides a 3-state prediction (helix, strand, loop) together with a reliability score for every sequence position. Then THESEUS predicts all sequence-to-structures alignments for the template database. From the best scoring template structures the most probable templates are selected and submitted to the loop modelling procedure, where different 3D models for each template are generated in parallel. The loop and side-chain modelling step finalizes the threading part of the prediction pipeline. Suitable methods for loop and side-chain modeling are utilized: currently, MODELLER is also used to model the loop regions and the amino acid side chain atoms of the given backbone model. In the CASP6 contest, the LIP tool [39] was applied for loop modeling and the Swiss PDB Viewer [40] for side-chain placement. At the end, a full-atomic structure model of the target sequence is provided.

The whole workflow runs on a IA32 Myrinet Linux cluster available at our site[1]. The resources of the compute cluster complex are managed by a job management system (batch system) providing a single point of control (job submission and job control). The overall structure prediction pipeline is implemented as a fully-automated Perl script, which manages the orchestration of the workflow components and the communication with the local batch system. Additionally, we have implemented a Web service-based version of the structure prediction pipeline using Triana [41] as workflow engine [42].

[1] http://elfie.bcbio.de

4 Results

The fold recognition performance of THESEUS was tested on two commonly used benchmark sets for threading methods: the Fischer [13] and the Lindahl [43] benchmark set. Additionally, the THESEUS results from the CASP6 [35] experiment in 2004 and detailed performance measurements are discussed.

4.1 Fischer Benchmark

The Fischer [13] benchmark set consists originally of 68 target sequences and 301 template structures. For every target sequence there is at least one template structure with similar fold type. All pairs were hand selected showing high structure similarity but low sequence similarity. Only 67 of the 68 Fischer pairs are possible to find for THESEUS because of the constraint, that the template structure sequence have to be smaller than the target sequence. A match occurs when the expected matching template structure or a template structure with the same fold is found on the top rank, or within the top five, or within the top ten ranks. In Fig. 8 the results for all 67 target sequences are shown ordered according the SCOP [31] classification CLASS level (Mainly Alpha, Mainly Beta, Alpha/Beta, Alpha+Beta, Others).

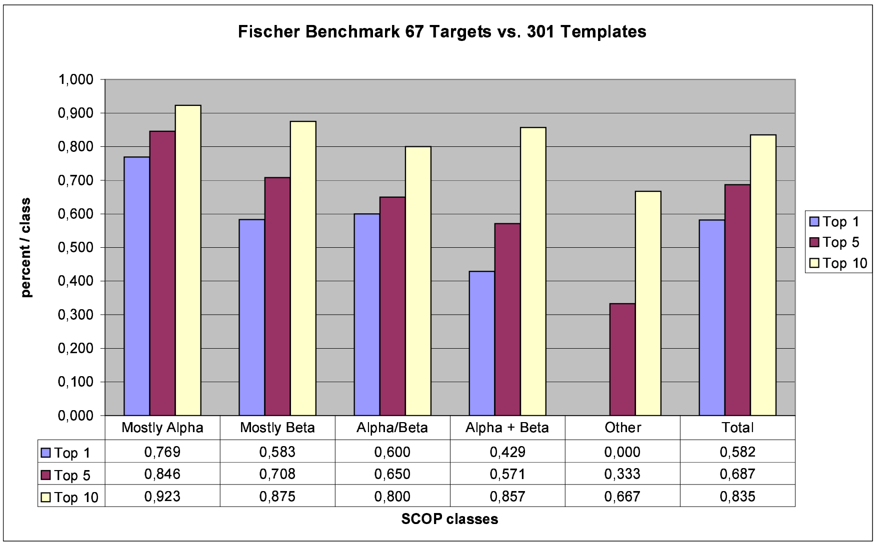


Figure 8: Results from the Fischer benchmark sets.

Totally, THESEUS is able to predict 56 (83%) of the 67 correct pairs within the top ten ranks. The best recognition is given for the 'Mainly Alpha' class representing protein folds that consists of helices mainly. For the 'Other' class the recognition is worst because this class contains only 3 target sequences consisting of only few secondary structure elements with few pairwise contacts in between.

4.2 Lindahl Benchmark

The Lindahl [43] benchmark set consists of 976 protein sequences of SCOP [31] domains. All these 976 protein domains were aligned all against all. SCOP is a hierarchi-

Table 3: Performance of various fold recognition methods for the Lindahl Benchmark [43]: Percentage of recognized folds at different SCOP levels within top ranked results. All results except for THESEUS are from Zhou et al. [44]. The methods are ordered according the TOP5 ranks for the 'Fold only' category.

	Family only			Superfamily only			Fold only		
Method	Top1	Top5	Top10	Top1	Top5	Top10	Top1	Top5	Top10
RAPTOR	89.3	91.6		59.2	74.4		39.7	61.2	
PROSPECT II	84.1	88.2		52.6	64.8		27.7	50.3	
SPARKS	82.9	90.1		56.5	72.4		23.1	43.6	
THREADER	49.2	58.9		10.8	24.7		14.6	37.7	
THESEUS	59.8	77.0	78.4	13.0	43.1	51.9	9.0	36.2	49.7
FUGUE	82.3	85.8		41.9	53.2		12.5	26.8	
SAMT98-PsiBlast	70.1	75.4		28.3	38.9		3.4	18.7	
HMMER-PsiBlast	67.7	73.5		20.7	32.5		4.4	14.6	
PSIBLAST	71.2	72.3		27.4	27.9		4.0	4.7	

cal classification of protein domains, so relationships between proteins can be studied on different levels:

- FAMILY: Protein domains with clear evolutionary relationship.
- SUPERFAMILY: Protein domains that have low sequence identities, but whose structural and functional features suggest that a common evolutionary origin is probable.
- FOLD: Protein domains are defined as having a common fold if they have the same major secondary structure arrangement and with the same topological connections. Protein domains placed together in the same fold category may not have a common evolutionary origin.

Since all methods are much better in recognizing closely related proteins all hits from lower SCOP levels are ignored, otherwise the score on, e.g., the fold level would be the sum of fold, superfamily and family level dominated by the easiest family level. There are 555, 434, and 321 pairs of protein domains in the same family, superfamily, and fold, respectively. THESEUS is tested whether or not the method can recognize the correct pair as first rank or within the top five or top ten ranks. The results of THESEUS in comparison to several well-established methods for protein structure prediction are shown in Table 3. The comparison of the results can only be approximate, because the structure and sequence databases used for previous methods at the time publishing the results have become larger.

Generally, THESEUS has problems to rank the correct pair within the top five hits, but the results for the top ten ranks are comparable to methods with comparable algorithms and energy functions like RAPTOR, PROSPECT, FUGUE, or THREADER. The performance in all three categories is better than those for methods based on profile alignments only, like HMMER-PSIBLAST, SAMT98-PSIBLAST, or PSIBLAST. The overall performance is for the most difficult category FOLD better than for the two other categories, where sequence similarity is detectable.

4.3 THESEUS in CASP6

4.3.1 The CASP6 Experiment

CASP [35] is a blind prediction experiment where crystallographers give their solved protein structures to the organizers before they are published in the PDB. Predictors can submit their predictions within a given period or until the structure is published. The predicted data is then passed to the assessors who analyse the data and try to derive general conclusions. All models and results are available on-line[2].

We participated with our structural prediction pipeline (see section 3.3) in the 6th Critical Assessment of Techniques for Protein Structure Prediction (CASP) 2004 [36] together with the Preissner group from Charité – Universitätsmedizin Berlin who were responsible for the homology modeling part of the structure prediction pipeline. Totally, in CASP6 201 so called human expert groups were registered and 65 fully automated servers. 87 target sequences were released, from which 11 were cancelled before the end and 12 after the end of the prediction season, so that totally 64 targets representing 90 domains were assessed. After the experiment the target domains were classified from the assessors into five categories:

- CM/easy, where similarity can be detected by simple BLAST [7] searches,
- CM/hard, where similarity only can be detected by iterated PSI-BLAST [8] searches,
- FR/homolog, where similarity is detectable by profile-profile alignments,
- FR/analog, where only structure similarity but no sequence similarity is present,
- New Fold representing new protein fold types that are not present in current PDB [37] database.

The main assessment method in the CASP experiments is the global distance test total score (GDT_TS) [45], which evaluates the structural similarity of a model to the experimental structure at four distance thresholds 1, 2, 4, and 8Å. A large sample of possible structure superpositions of the model on the experimental structure is generated by superposing all sets of three, five and seven consecutive C_α atoms along the backbone. Each of these initial superpositions is then iteratively extended by including all residue pairs under the threshold in the next iteration, until no new residues can be added. For each threshold the superposition that includes the maximum number of residues (as a percentage of the total residues this is Nn, where n is the threshold distance) is selected. Note that superimposed residues are not required to be continuous in the sequence. The resulting score is then obtained by averaging over all thresholds:

$$GDT\_TS = \frac{1}{4}[N1 + N2 + N4 + N8] \ . \tag{24}$$

4.3.2 Overall Performance

THESEUS have been used to build 91 models for 39 of the totally 64 assessed CASP6 targets that have been assigned as threading targets through the prediction pipeline. In contrast to all other groups, who have submitted five different models for every target sequence, we have just submitted one model in different modeling stages, e.g., the first

[2] http://predictioncenter.org/casp6/Casp6.html

model including only the backbone coordinates of the threading templates, the second model including additionally the backbone coordinates of the loop residues, the third model adding the side chains the atoms, and if possible the fourth model after model refinement. Consecutively, our best model is not the first model, but the fourth one. Most of the automatically CASP evaluation are done just on the first model or an all models.

In the automated assessment of CASP6 models by CASP6-servers and human predictors conducted by the Skolnick-Group[3] the THESEUS structure prediction method was ranked on position 88 of 174 evaluated prediction groups. One have to note, that again the first model of the five possible models was evaluated.

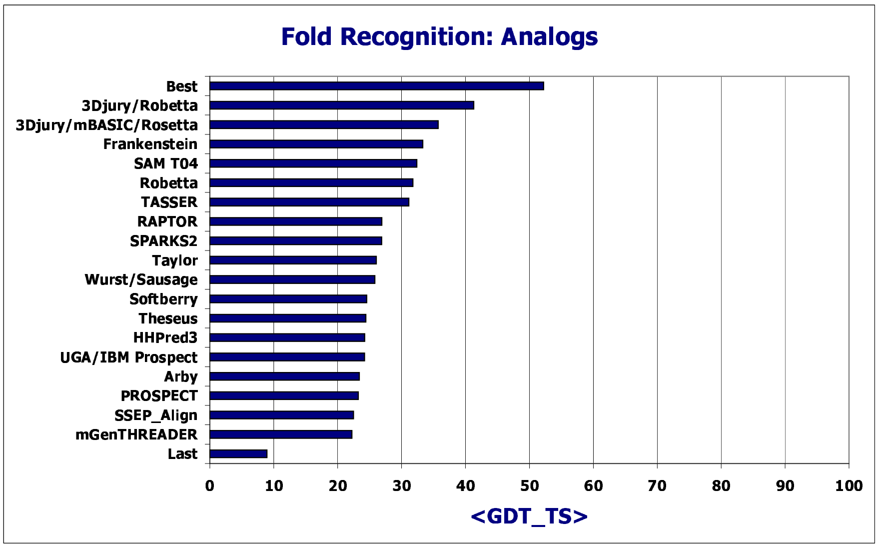


Figure 9: Results for the FR/analog category.

In Fig. 9 the results for the FR/analog category for 18 selected methods out of the 201 registered predicting groups are shown. Here, for every predicting group the GDT_TS scores for all targets in that category are averaged. The two groups 'Best' and 'Last' denote the particular best and respectively worst predicted model for every target. The results show that THESEUS is competitive to groups using similar threading implementations like Raptor [11], Prospect [9], Genthreader [46] or Wurst [47] but there are some groups with a significant better averaged GDT_TS score greater than 30. These methods are based either consensus methods like Tasser/3D-Jury [48, 49] or fragment-based like Robetta and Rosetta [50, 51].

Fig. 10 shows the results for two targets from the FR/analog category. On the left the target structure (PDB code 1TD6, thicker line) and our model structure (thinner line) are superimposed. The structures are colored according their alignment quality. Are the aligned residues within a certain distance cutoff they are colored in green, yellow, and orange, or, when residues are not aligned, the residues are colored purple for the target structure and red for the model structure. On the right the results for all models of all groups for the two targets are presented. The diagram shows the percentage of residues within a certain distance cutoff. A perfect prediction would correspond with

[3]http://cssb.biology.gatech.edu/skolnick/files/casp6/index.shtml

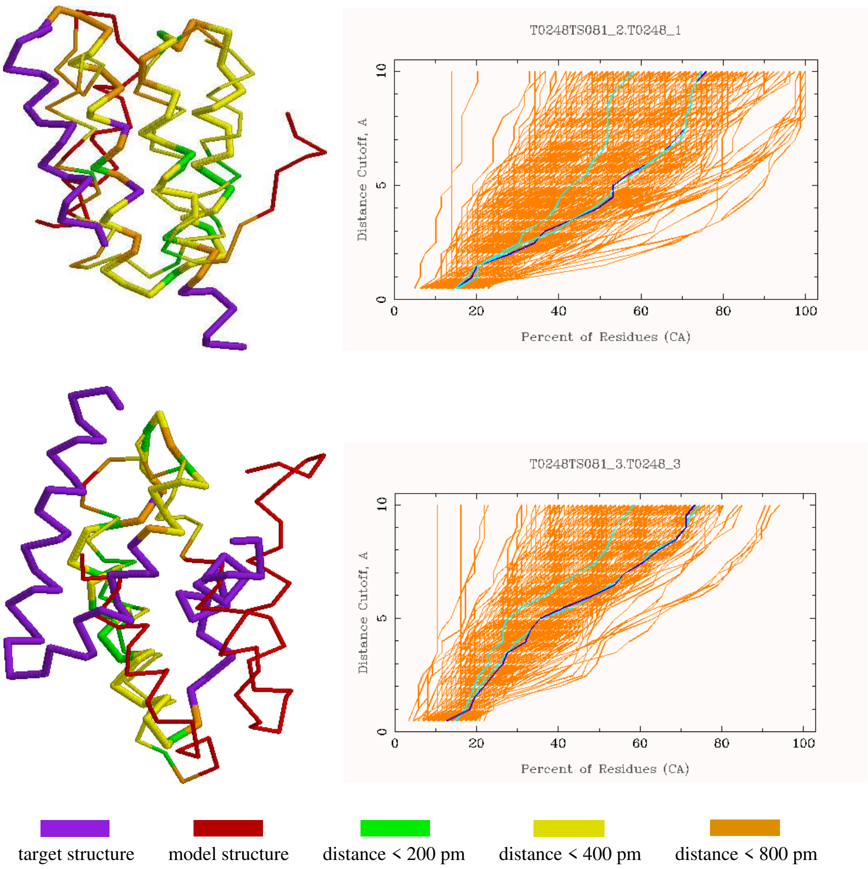


Figure 10: Results for the CASP targets T0248_1 and T0248_3. Left: Superposition of target structure and predicted model. The structures are visualized with RASMOL (100 pm = 1Å. Right: Results of all models from all groups for one target.

a horizontal line near 1Å. The highlighted lines are marking the THESEUS models, the dark blue line shows the model visualized on the left. All model and result files are downloadable from the CASP6 website[4]. In the upper alignment (target T0248_1) we have predicted three helices within a cutoff of 2 or 4Å, whereas the most left helix has switched two the wrong side. The lower alignment (target T0248_3) shows only one well predicted helix, the other helices are shifted or switched in wrong positions. Nevertheless, for this target no group has predicted a model with all residues within a cutoff of 10 Å. Totally our best model for T0248_3 was ranked on position 18 among all number one models.

4.4 Runtime Analysis

In this section the performance of the Branch-and-Bound algorithm is tested against two integer linear programming (ILP) approaches for solving the optimal protein

[4]http://predictioncenter.org/casp6/Casp6.html

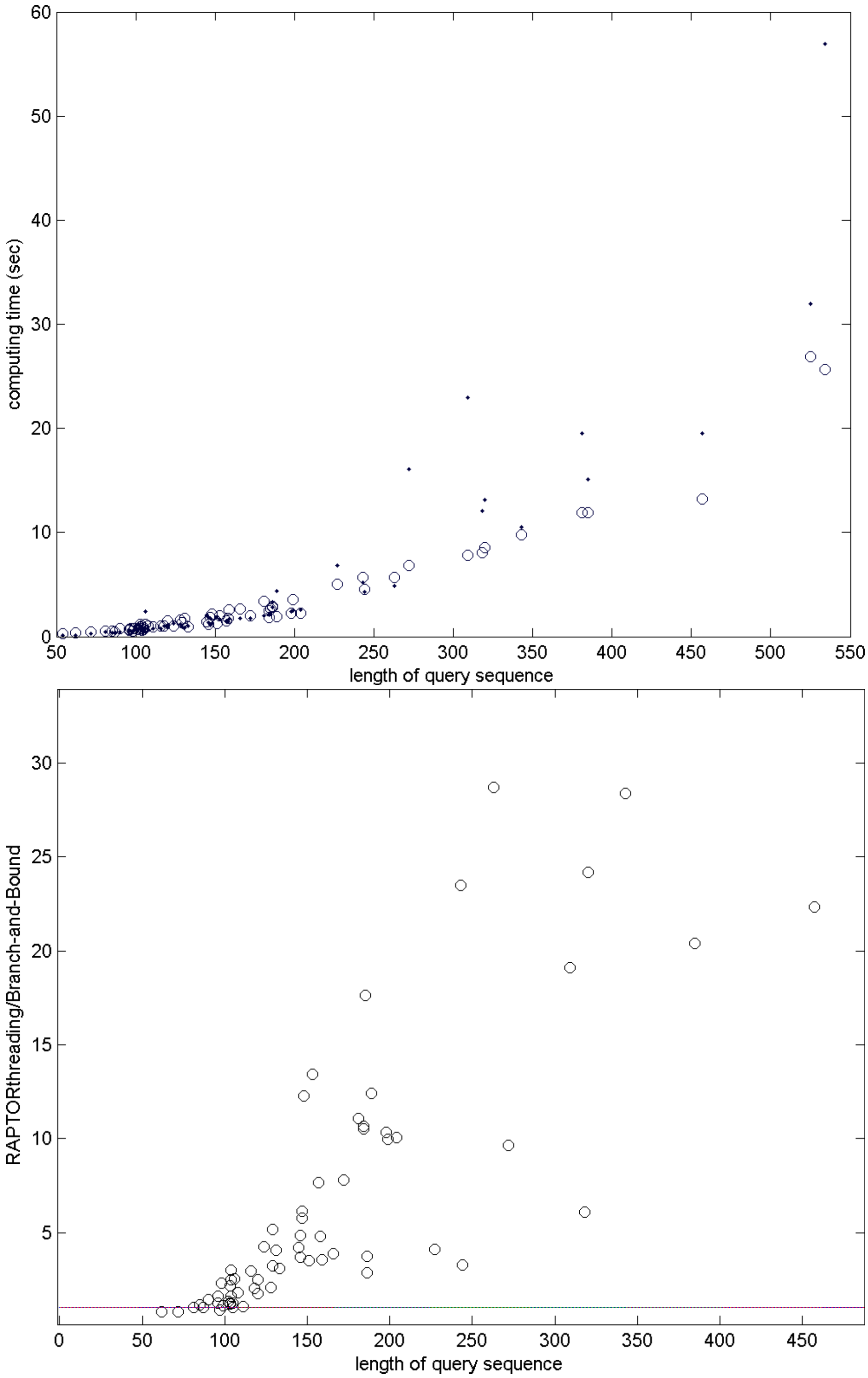


Figure 11: Runtime analysis. Top: Runtime (in seconds) for RAPTORthreading (dots) and THESEUS (circles) (68 sequences against template 1aaj (4 core segments). Bottom: Runtime ratio RAPTORthreading/THESEUS (68 sequences against template structure 1a45_1 (8 core segments))

Table 4: Template proteins.

Protein	Length	Cores	Contacts	Class
1aaj	105	6	59	Mainly Beta
1aep	153	5	149	Mainly Alpha
1arb	263	22	164	Mainly Beta
1bbt1	208	10	94	Mainly Beta
1caub	184	16	110	Mainly Beta
1chra1	244	17	126	Alpha/Beta
1cid_2	72	7	71	Mainly Beta

threading problem as presented as RAPTOR by Xu et al. [11] and ANDONOVMYZ by Andonov et al. [52]. The implementation of both ILP formulation adopts the original approaches in most instances with a few little changes that had to be done for reasons of compatibility to THESEUS. Both ILP formulations were implemented through C++ interfaces that directly can be plugged into the THESEUS threading environment by exchanging the Branch-and-Bound algorithm module with the corresponding ILP module. The three algorithms work on the same protein template model (see section 2.2) and on the same energy function (see section 2.5). To solve the integer linear program the ILP implementations use the commercial optimization package CPLEX [5]. All tests were performed on a 1.6 GHz AMD64 computer system with 2 GB of main memory. The test set used for evaluation is the Fischer benchmark set with 68 target sequences and 301 template structures [13] (see section 4.1).

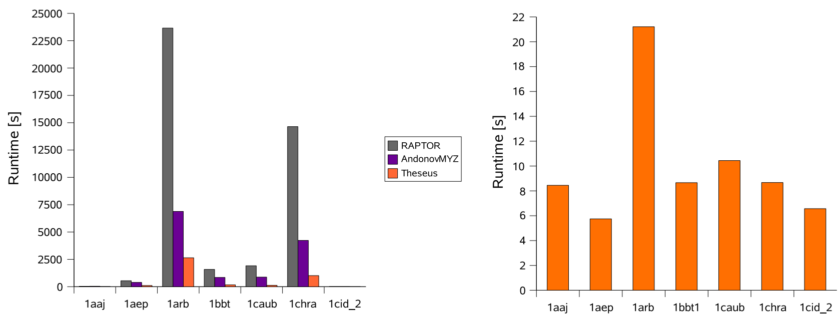


Figure 12: Runtime comparisons. Left: Runtime (in seconds) for threading seven sequences against the Fischer database. Right: Runtime (in seconds) for threading of 1ATNA against seven template structures with THESEUS.

In Fig. 11 (top) we compared runtimes for the Branch-and-Bound algorithm and the RAPTORthreading approach in dependence of the query sequence length. For short query sequences (length < 200) the runtimes of the ILP and the Branch-and-Bound implementation are very close, but for longer query sequences the Branch-and-Bound algorithm is eminently faster than the ILP formulation. This effect is even stronger for templates with more core segments. A graphical presentation of the relation between runtime of both implementations given by the ratio of the CPU times for threadings with RAPTOR vs. Branch-and-Bound with respect to the sequence length is shown

[5] ILOG Inc., http://www.cplex/com

in Fig. 11 (bottom). In this test we used a template with 8 core segments. Here, the Branch-and-Bound algorithm clearly outperforms the RAPTORthreading for query sequences longer than 100 and is up to 30 times faster for sequences longer than 250.

We compared the runtimes of THESEUS, RAPTORthreading, and ANDONOVMYZ for seven sequences against the complete Fischer database (shown in Fig. 12 (left)). The lengths of the target sequences are listed in Table 4. For shorter sequences the three algorithms have similar runtimes, but the longer the target sequence the more THESEUS outperforms the ILP implementations. These results are in conflict with the performance comparison in [52], where it is stated that the ILP formulations would clearly outperform the Branch-and-Bound strategy. Nevertheless, there was not clearly described, how they had implemented the Branch-and-Bound algorithm, i.e., what kind of lower bound they used and if they implemented the multi-queued variant of the algorithm. The runtime ratios between RAPTORthreading and ANDONOVMYZ agree with the results given in [52].

To show that the runtimes of threading alignments depend on a variety of features of the template structure, we performed alignments (shown in Fig. 12 (right)) of the target sequence 1atna consisting of 373 amino acids against seven template structures with different structural features (shown in Table 4) using THESEUS. The template structures are described by the number of amino acids, the number of core segments, the number of interactions between the core segments, and the corresponding SCOP [31] class characterizing the main secondary structure topology of the protein fold. The template structure with the most core segments and the most contacts between core segments shows the worst performance, whereas template structures with few core segments have better runtimes. Interestingly, templates containing α-helices (1chra1 and 1aep) show better runtimes than comparable template structures consisting mainly β-strands, e.g., 1chra1 contains more amino acids, core segments, and contacts than 1caub but shows a better runtime.

5 Summary

An open and extensible framework for protein structure prediction was designed and implemented. The work started with the implementation of reliable fold recognition methods known at the time of the project start in 2001. The implementation of the THESEUS framework focused on the following design features:

- robust to be suitable for production environments,
- flexible and extensible for the evaluation of new scoring functions and different search algorithms,
- fast and efficient through single CPU optimization and parallelization,
- re-usable as building block in complex workflow scenarios through command line and Web service interface.

THESEUS was and is used routinely as core component of a fully-automated pipeline for protein structure prediction in the CASP6 and CASP7 contests, respectively, and thus demonstrates is robustness. For the first time we provide an efficient and parallelized implementation of the Branch-and-Bound search strategy for protein threading as formulated by Lathrop et al [5]. THESEUS uses a new scoring schema that enhances traditionally threading energy functions by averaging over homologs and by additional scoring terms for assessing the alignment of secondary structure elements and scoring loop alignments. Our THESEUS implementation outperforms various threading approaches with respect to the required computational time and the reliability of the fold recognition results as additionally proven in the CASP6 contest. Our results show also that protein threading is superior above sequence-based methods for protein alignment and classification tasks.

Furthermore, THESEUS will be used as part of a complex workflow for protein-protein interaction and functional prediction within the MediGRID [53] project.

Acknowledgement

We thank Michael Meyer for the initial implementation of threading core functions. For their supporting work on the Web service-based pipeline, we like to thank our students Hans-Christian Ehrlich, René Heek, Falko Krause, Jonas Maaskola, and Sascha Willuweit. For the implementation of the two ILP algorithms, we like to thank Sandro Andreotti and Gunnar Klau.
This work was funded by BMBF (Germany, grant no. 031U209A (Berlin Center of Genome Based Bioinformatics).

References

[1] D. Baker and A. Sali. Protein structure prediction and structural genomics. *Science*, 294:93–96, 2001.

[2] Li Z. and H. A. Scheraga. Monte-carlo-minimization approach to the multiple-minima problem in protein folding. *PNAS*, 84:6611–6615, 1987.

[3] J. Skolnick and A. Kolinski. Dynamic monte carlo simulations of a new lattice model of globular protein folding, structure and dynamics. *J. Mol. Biol.*, 221:499–531, 1991.

[4] Godzik A. Fold recognition methods. *Methods Biochem. Anal.*, 44:525546, 2003.

[5] R. H. Lathrop and T. F. Smith. Global optimum protein threading with gapped alignment and empirical pair score functions. *J. Mol. Biol.*, 255:641–665, 1996.

[6] A. Marin, J. Pothier, K. Zimmermann, and J. F. Gibrat. FROST: A filter-based fold recognition method. *Proteins*, 49:493–509, 2002.

[7] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *J. Mol. Biol.*, 215:403–410, 1990.

[8] S. F. Altschul, T. L. Madden, A. A. Schaffler, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nuc. Acids Res.*, 25:3389–3402, 1997.

[9] Y Xu, D. Xu, and E. C. Uberbacher. An efficient computational method for globally optimal threading. *Journal of Computational Biology*, 5(3):597–614, 1998.

[10] G. Lancia, R. Carr, B. Walenz, and S. Istrail. 101 optimal PDB structure alignments: a branch-and-cut algorithm for the maximum contact map overlap problem. In *Proc. of the Fifth Annual International Conference on Computational Biology*, pages 193–202. ACM Press, 2001.

[11] J. Xu, M. Li, D. Kim, and Y. Xu. RAPTOR: optimal protein threading by linear programming. *Journal of Bioinformatics and Computational Biology*, 1:95–117, 2003.

[12] G. Taguchi and S. Konishi. *Orthogonal Arrays and Linear Graphs*. ASI press, 1987.

[13] D. Fischer, A. Elofsson, J. U. Bowie, and D. Eisenberg. Assessing the performance of fold recognition methods by means of a comphrehensive benchmark. In L. Hunter and T. Klein, editors, *Biocomputing: Proceedings of the 1996 Pacific Symposium*, pages 300–318. World Scientific Publishing Co., 1996.

[14] P. J. Munson and R. K. Singh. Multi-body interactions within the graph of protein structure. In *Proceedings of the Fifth International Conference on Intelligent Systems for Molecular Biology*, pages 198–201. AAAI Press, 1997.

[15] G. H. Gonnet, M. A. Cohen, and Benner S. A. Exhaustive matching of the entire protein sequence database. *Science*, 256:1443–1445, 1992.

[16] Steven Henikoff and Jorja G. Henikoff. Amino acid substitution matrices from protein blocks. *Proc. Natl. Acad. Sci.*, 89:10915–10919, 1992.

[17] R. A. Abagyan and S. Batalov. Do aligned sequences share the same fold? *J. Mol. Biol.*, 273:355–368, 1997.

[18] Y. Cui and W. H. Wong. Multiple-sequence information provides protection against mis-specified potential energy functions in the lattice model of proteins. *Phys. Rev. Lett.*, 85:5242–5245, 2000.

[19] B. A. Reva, J. Skolnick, and A. V. Finkelstein. Averaging interaction energies over homologs improves protein fold recognition in gapless threading. *Proteins*, 35:353–359, 1999.

[20] M. J. Sippl and S. Weitcus. Detection of native-like models for amino acid sequences of unknown three-dimensional structure in a data base of known protein conformations. *Proteins*, 13:258–271, 1992.

[21] W. Kabsch and C. Sander. Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, 22:2577–2637, 1983.

[22] D. Fischer, C. J. Tsai, R. Nussinov, and Wolfson. A 3d sequence-independent representation of the protein data bank. *Protein Eng.*, 8:981–997, 1995.

[23] L.J. McGuffin, K. Bryson, and D.T. Jones. The PSIPRED protein structure prediction server. *Bioinformatics*, 16:404–405, 2000.

[24] H. Zhou and Y. Zhou. Single-body residue-level knowledge-based energy score combined with sequence-profile and secondary structure information for fold recognition. *Proteins*, 8:1005–1013, 2004.

[25] D. Fischer and D. Eisenberg. Protein fold recognition using sequence-derived predictions. *Proteins*, 5:947–955, 1996.

[26] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48:443–453, 1970.

[27] R. H. Lathrop. The protein threading problem with sequence amino acid interaction preferences is NP-complete. *Protein Eng.*, 7(9):1059–1068, 1994.

[28] R. H. Lathrop, A. Sazhin, Y. Sun, N. Steffen, and S. S. Irani. A multi-queue branch-and-bound algorithm for anytime optimal search with biological applications. *Genome Informatics*, 12:73–82, 2001.

[29] A. Marin, J. Pothier, K. Zimmermann, and J. F. Gibrat. Protein threading statistics: an attempt to assess the significance of a fold assignment to a sequence. In I. F. Tsigelny, editor, *Protein Structure Prediction: Bioinformatic Approach*. International University Line, 2002.

[30] I. Sommer, A. Zien, N. von Öhsen, R. Zimmer, and T. Lengauer. Confidence measures for protein fold recognition. *Bioinformatics*, 18(6):802–812, 2002.

[31] A.G. Murzin, S.E. Brenner, T. Hubbard, and C. Chothia. SCOP: a structural classification of proteins database for the investigation of sequences and structures. *J. Mol. Biol.*, 247:536–540, 1995.

[32] J.M. Chandonia, N.S. Walker, L. Lo Conte, P. Koehl, M. Levitt, and S.E. Brenner. ASTRAL compendium enhancements. *Nucl. Acids Res.*, 30:264–267, 2002.

[33] A. Bairoch, R. Apweiler, C.H. Wu, W.C. Barker, B. Boeckmann, S. Ferro, E. Gasteiger, H. Huang, R. Lopez, M. Magrane, M.J. Martin, D.A. Natale, C. O'Donovan, N. Redaschi, and L.S. Yeh. The universal protein resource (uniprot). *Nucleic Acids Res.*, 1(33):154–159, 2005.

[34] K. Henrick and J. M. Thornton. PQS: a protein quaternary structure file server. *Trends Biochem Sci.*, 23(9):358–361, 1998.

[35] J. Moult. A decade of CASP: progress, bottlenecks and prognosis in protein structure prediction. *Curr. Opin. Struct. Biol.*, 15:285–289, 2005.

[36] E. Michalsky, A. Goede, R. Preissner, P. May, and T. Steinke. A distributed pipeline for structure prediction. In *CASP6 Methods Abstracts, 6th Meeting on the Critical Assessment of Techniques for Protein Structure Prediction*, pages 112–114, Gaeta, Italy, 2004.

[37] H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shyndyalov, and P.E. Bourne. The protein data bank. *Nucl. Acids Res*, 28:235–242, 2000.

[38] M.A. Marti-Renom, A. Stuart, A. Fiser, R. Sanchez, F. Melo, and A. Sali. Comparitive protein structure modeling of genes and genomes. *Annu. Rev. Biophys. Biomol. Struct.*, 29:291–325, 2000.

[39] E. Michalsky, A. Goede, and R. Preissner. Loops in proteins (LIP) - a comprehensive database for homology modelling. *Protein Eng.*, 16:979–985, 2003.

[40] N. Guex and M. C. Peitsch. SWISS-MODEL and the Swiss-PdbViewer: an environment for comparative protein modelling. *Electrophoresis*, 18:2714–2723, 1997.

[41] S. Majithia, M. Shields, I. Taylor, and I. Wang. Triana: A graphical web service composition and execution toolkit. In *IEEE International Conference on Web Services (ICWS 2004)*, 2004.

[42] P. May, H. C. Ehrlich, and T. Steinke. ZIB Structure Prediction Pipeline: Composing a Complex Biological Workflow through Web Services. In W. E. Nagel, W. V. Walter, and W. Lehner, editors, *Euro-Par 2006 Parallel Processing*, LNCS 4128, pages 1148–1158. Springer Verlag, 2006.

[43] E. Lindahl and A. Elofsson. Identification of related proteins on family, superfamily and fold level. *J. Mol. Biol.*, 295:613–625, 2000.

[44] H. Zhou and Y. Zhou. Fold recognition by combining sequence profiles derived from evolution and from depth-dependent structural alignment of fragments. *Proteins*, 58(2):321–328, 2005.

[45] C. Venclovas, A. Zemla, K. Fidelis, and J. Moult. Assessment of progress over the casp experiments. *Proteins*, 53(S6):585–595, 2003.

[46] D. T. Jones. GenTHREADER: an efficient and reliable protein fold recognition method for genomic sequences. *J.Mol.Biol.*, 287(4):797–815, 1999.

[47] A. E. Torda, J. B. Procter, and T. Huber. Wurst: a protein threading server with a structural scoring function, sequence profiles and optimized substitution matrices. *Nucleic Acids Res.*, 32(Web Server issue):W532–535, 2004.

[48] Y. Zhang and J. Skolnick. TASSER: An automated method for the prediction of protein tertiary structures in CASP6. *Proteins*, 61(7):91–98, 2005.

[49] K. Ginalski, A. Elofsson, D. Fischer, and L. Rychlewski. 3d-jury: a simple approach to improve protein structure predictions. *Bioinformatics*, 19(8):1015–1018, 2003.

[50] D. E. Kim, D. Chivian, L. Malmstrom, and D. Baker. Automated prediction of domain boundaries in CASP6 targets using Ginzu and RosettaDOM. *Proteins*, Epub ahead of print, 2005.

[51] C. A. Rohl, C. E. M. Strauss, D. Chivian, and D. Baker. Modeling structurally variable regions in homologous proteins with Rosetta. *Proteins*, 55:656–677, 2004.

[52] R. Andonov, S. Balev, and N. Yanev. Protein threading: from mathematical models to parallel implementations. *INFORMS*, 1(1):393–405, 2004.

[53] MediGRID. http://www.medigrid.de/.