Supporting Information

Table of Contents

Supporting Texts

- **Text S1.** The normalized number of effective sequences in an MSA
- **Text S2.** Eigen-decomposition of a contact-map
- **Text S3.** Optimization of the number of eigenvectors used by CEthreader
- **Text S4.** Optimization of the scoring function used for aligning contact maps
- **Text S5.** Optimization of the component scores used to match the secondary structure and sequence profiles of two proteins

Supporting Tables

- **Table S1.** Summary of the threading alignments produced by CEthreader for the 614 test proteins in Benchmark Set-I.
- **Table S2.** Threading results by different methods for the 403 Easy targets from Benchmark Set-I.
- **Table S3.** Threading results for different methods using Benchmark Set-I separated based on sequence identity to the ResPRE training set.
- **Table S4**. Fold-recognition performance of CEthreader, HHsearch and MUSTER for 116 Fold families in Benchmark Set-II.
- **Table S5**. List of Fold pairs in Benchmark Set-II with an average TM-score >0.4.
- **Table S6.** Summary of models built by MODELLER based on different threading methods for all 614 proteins in Benchmark Set-I.
- **Table S7.** Summary of models built by CEthreader/C-I-TASSER, CEthreader/I-TASSER and ResPRE/CNS for all 614 proteins in Benchmark Set-I.
- **Table S8.** CEthreader threading alignment results for the 905 query-template pairs using native contact maps or contact maps predicted using 18 different contact predictors.

Supporting Figures

- **Figure S1.** CEthreader template information for 211 Hard targets from Benchmark Set-I.
- **Figure S2.** CEthreader performance and time-cost using different numbers of eigenvectors in the greedy search step.
- **Figure S3.** Comparison of CEthreader's performance to map_align and EigenThreader on Benchmark Set-I.
- **Figure S4.** Optimization of the number of predicted contacts to be used by CEthreader.
- **Figure S5.** The diagrammatic illustration of eigen-decomposition of a contact-map.
- **Figure S6.** Optimization of the number of eigenvectors used by CEthreader.
- **Figure S7.** Optimization of the contact map matching score S_{cm}.
- **Figure S8.** Illustration that alignment of contact eigenvectors between query and template sequences results in the match of global contact maps for the two sequences.

References

Supporting Texts

Text S1. The normalized number of effective sequences (Neff) in an MSA

The depth of a multiple sequence alignment (MSA) can be measured by the normalized number of effective sequences (*Neff*):

$$Neff = \frac{1}{\sqrt{L}} \sum\nolimits_{n=1}^{N} \frac{1}{1 + \sum_{m=1,m\neq n}^{N} I\left[S_{m,n} \geq 0.8\right]} \qquad (S1)$$

where L is the length of a query protein, N is the number of sequences in the MSA, and $S_{m,n}$ is the sequence identity between the m-th and n-th sequences. $I\left[S_{m,n} \geq 0.8\right]$ is equal to 1 if $S_{m,n} \geq 0.8$, or zero otherwise. Therefore, *Neff* is essentially equal to the number of non-redundant sequences (sequence identity<0.8) in the MSA normalized by the query length.

Text S2. Eigen-decomposition of a contact-map

For a given protein P with length L, its contact map $\boldsymbol{M}$ can be represented by an $L \times L$ binary and symmetric matrix, where residues that are in contacts (C_β-C_β distance < 8 Å) are designated as 1 and non-contacting residues are set to 0. As the contact map $\boldsymbol{M}$ is a real-valued $L \times L$ symmetric matrix, it can be decomposed into L eigenvectors and their associated eigenvalues. Assuming that λ_i represents the i-th eigenvalue of $\boldsymbol{M}$ and $\vec{V_i} = (v_{1,i}, v_{2,i}, \ldots, v_{L,i})^T$ is the corresponding eigenvector, we have

$$\begin{cases} \boldsymbol{M}\vec{V_i} = \lambda_i \vec{V_i} \\ \boldsymbol{M} = V\Delta V^{-1} \end{cases} \qquad (S2)$$

where

$$\begin{cases} \mathrm{V} = \begin{bmatrix} v_{1,1} & \cdots & v_{1,L} \\ v_{2,1} & & v_{2,L} \\ \vdots & \ddots & \vdots \\ v_{L,1} & \cdots & v_{L,L} \end{bmatrix} = \left[\vec{V_1}, \vec{V_2}, \ \ldots\ldots, \vec{V_L}\right] \\ \Delta = \begin{bmatrix} \lambda_1 & \cdots & 0 \\ 0 & & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_L \end{bmatrix} \end{cases} \qquad (S3)$$

Again, since $\boldsymbol{M}$ is a real-valued symmetric matrix, $V^{-1} = V^T$. Therefore, we can infer that

$$\begin{aligned} \boldsymbol{M} &= V\Delta V^{-1} = V\Delta V^T \\ &= \begin{bmatrix} v_{1,1} & \cdots & v_{1,L} \\ v_{2,1} & & v_{2,L} \\ \vdots & \ddots & \vdots \\ v_{L,1} & \cdots & v_{L,L} \end{bmatrix} \begin{bmatrix} \lambda_1 & \cdots & 0 \\ 0 & & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_L \end{bmatrix} \begin{bmatrix} v_{1,1} & \cdots & v_{L,1} \\ v_{1,2} & & v_{L,2} \\ \vdots & \ddots & \vdots \\ v_{1,L} & \cdots & v_{L,L} \end{bmatrix} \\ &= \begin{bmatrix} v_{1,1} & \cdots & v_{1,L} \\ v_{2,1} & & v_{2,L} \\ \vdots & \ddots & \vdots \\ v_{L,1} & \cdots & v_{L,L} \end{bmatrix} \left(\sum_{i=1}^{L} \begin{bmatrix} 0 & \cdots & 0 \\ 0 & & 0 \\ \vdots & \lambda_i & \vdots \\ 0 & \cdots & 0 \end{bmatrix} \right) \begin{bmatrix} v_{1,1} & \cdots & v_{L,1} \\ v_{1,2} & & v_{L,2} \\ \vdots & \ddots & \vdots \\ v_{1,L} & \cdots & v_{L,L} \end{bmatrix} \\ &= \sum_{i=1}^{L} \lambda_i \vec{V_i} * \vec{V_i}^T \end{aligned} \qquad (S4)$$

Based on the above inference, the initial contact map $\boldsymbol{M}$ can be reconstructed by L eigenvalues and their associated eigenvectors, where eigenvalues are sorted in descending order. Typically, eigenvalues with greater absolute values are more important for reconstructing a contact map that is identical to the initial one. Thus, an approximate contact map can be constructed by considering only a few of its eigenvectors with large eigenvalues. Also, we do not consider the eigenvectors with negative eigenvalues because it requires introducing complex values into the following computation, which increases computational complexity significantly. In this way, the contact map ***M*** can be approximated by

$$
\begin{aligned}
\boldsymbol{M} &\approx \sum_{i=1}^{k} \lambda_i \vec{V_\imath} * \vec{V_\imath}^T \\
&= \begin{bmatrix} v_{1,1} & \cdots v_{1,k} & \cdots & 0 \\ v_{2,1} & v_{2,k} & \cdots & 0 \\ \vdots & \ddots & & \vdots \\ \vdots & \vdots & & \vdots \\ v_{L,1} & \cdots v_{L,k} & \cdots & 0 \end{bmatrix}
\begin{bmatrix} \lambda_1 & \cdots & 0 \\ 0 & & 0 \\ \vdots & \lambda_k & \vdots \\ 0 & \cdots & 0 \end{bmatrix}
\begin{bmatrix} v_{1,1} & \dots & v_{1,L} \\ \vdots & & \vdots \\ v_{k,1} & \ddots & v_{k,L} \\ \vdots & \vdots & \vdots \\ 0 & \cdots & 0 \end{bmatrix} \\
&= \begin{bmatrix} v_{1,1} & \cdots v_{1,k} & \cdots & 0 \\ v_{2,1} & v_{2,k} & \cdots & 0 \\ \vdots & \ddots & & \vdots \\ \vdots & \vdots & & \vdots \\ v_{L,1} & \cdots v_{L,k} & \cdots & 0 \end{bmatrix}
\begin{bmatrix} \sqrt{\lambda_1} & \cdots & 0 \\ 0 & & 0 \\ \vdots & \sqrt{\lambda_k} & \vdots \\ 0 & \cdots & 0 \end{bmatrix}
\begin{bmatrix} \sqrt{\lambda_1} & \cdots & 0 \\ 0 & & 0 \\ \vdots & \sqrt{\lambda_k} & \vdots \\ 0 & \cdots & 0 \end{bmatrix}
\begin{bmatrix} v_{1,1} & \cdots & v_{1,L} \\ \vdots & \vdots & \vdots \\ v_{k,1} & \ddots & v_{k,L} \\ \vdots & \vdots & \vdots \\ 0 & \cdots & 0 \end{bmatrix} \\
&= \begin{bmatrix} \sqrt{\lambda_1}v_{1,1} & \cdots\sqrt{\lambda_k}v_{1,k} & \cdots & 0 \\ \sqrt{\lambda_1}v_{2,1} & \sqrt{\lambda_k}v_{2,k} & \cdots & 0 \\ \vdots & \ddots & & \vdots \\ \vdots & \vdots & & \vdots \\ \sqrt{\lambda_1}v_{L,1} & \cdots\sqrt{\lambda_k}v_{L,k} & \cdots & 0 \end{bmatrix}
\begin{bmatrix} \sqrt{\lambda_1}v_{1,1} & \cdots\sqrt{\lambda_k}v_{1,k} & \cdots & 0 \\ \sqrt{\lambda_1}v_{2,1} & \sqrt{\lambda_k}v_{2,k} & \cdots & 0 \\ \vdots & \ddots & & \vdots \\ \vdots & \vdots & & \vdots \\ \sqrt{\lambda_1}v_{L,1} & \cdots\sqrt{\lambda_k}v_{L,k} & \cdots & 0 \end{bmatrix}^T \\
&= \begin{bmatrix} \sqrt{\lambda_1}v_{1,1} & \cdots\sqrt{\lambda_k}v_{1,k} & \cdots & 0 \\ \sqrt{\lambda_1}v_{2,1} & \sqrt{\lambda_k}v_{2,k} & \cdots & 0 \\ \vdots & \ddots & & \vdots \\ \vdots & \vdots & & \vdots \\ \sqrt{\lambda_1}v_{L,1} & \cdots\sqrt{\lambda_k}v_{L,k} & \cdots & 0 \end{bmatrix}
\begin{bmatrix} \sqrt{\lambda_1}v_{1,1} & \cdots\sqrt{\lambda_k}v_{1,k} & \cdots & 0 \\ \sqrt{\lambda_1}v_{2,1} & \sqrt{\lambda_k}v_{2,k} & \cdots & 0 \\ \vdots & \ddots & & \vdots \\ \vdots & \vdots & & \vdots \\ \sqrt{\lambda_1}v_{L,1} & \cdots\sqrt{\lambda_k}v_{L,k} & \cdots & 0 \end{bmatrix}^T
\end{aligned}
\qquad (S5)
$$

Assuming $C = \begin{bmatrix} \sqrt{\lambda_1}v_{1,1} & \cdots\sqrt{\lambda_k}v_{1,k} & \cdots & 0 \\ \sqrt{\lambda_1}v_{2,1} & \sqrt{\lambda_k}v_{2,k} & \cdots & 0 \\ \vdots & \ddots & & \vdots \\ \vdots & \vdots & & \vdots \\ \sqrt{\lambda_1}v_{L,1} & \cdots\sqrt{\lambda_k}v_{L,k} & \cdots & 0 \end{bmatrix}$, we can rewrite $\boldsymbol{M} \approx C * C^T$, and thus the initial contact between residue *i* and *j* can be approximated by

$$
\begin{aligned}
M_{i,j} \approx &\left(\sqrt{\lambda_1}v_{i,1}, \sqrt{\lambda_2}v_{i,2}, \dots, \sqrt{\lambda_k}v_{i,k}, 0, \dots, 0\right) \\
&* \left(\sqrt{\lambda_1}v_{j,1}, \sqrt{\lambda_2}v_{j,2}, \dots, \sqrt{\lambda_k}v_{j,k}, 0, \dots, 0\right)^T
\end{aligned}
\qquad (S6)
$$

Which means that we can describe the *i*-th and *j*-th residues of a protein by the contact eigenvectors $\vec{U_\imath} = \left(\sqrt{\lambda_1}v_{i,1}, \sqrt{\lambda_2}v_{i,2}, \dots, \sqrt{\lambda_k}v_{i,k}, 0, \dots, 0\right)$ and $\vec{U_\jmath} =$

$\left(\sqrt{\lambda_1}v_{j,1}, \sqrt{\lambda_2}v_{j,2}, \dots, \sqrt{\lambda_k}v_{j,k}, 0, \dots, 0\right)$, respectively. A protein can be represented by a k-dimensional contact eigenvector sequence.

The highest 7 eigenvalues and their corresponding eigenvectors can reconstruct a contact map that is comparable to the native contact map, as shown in **Fig. S5**. Additionally, the selection of 7 eigenvalues/eigenvectors is computationally efficient, as discussed later, and hence we consider the largest 7 eigenvalues and their corresponding eigenvectors to reconstruct the contact maps.

Two proteins with similar structures have similar contact maps, and thus have similar contact eigenvector sequences. Therefore, we perform the reverse process, where we first compare two contact eigenvector sequences in order to compare their corresponding structures. Using this technique, we decompose the native contact maps for the templates in our Benchmark Set-I and II, and obtain contact eigenvectors for each template. Since the goal of threading is to identify templates for query proteins that do not have experimentally solved structures, there is no native contact map information for the query proteins. Therefore, we use ResPRE [1] to predict each query protein's contact map starting from its sequence. The predicted contact maps are then decomposed to obtain the corresponding contact eigenvector sequences.

Using a semi-global dynamic programming algorithm, which does not penalize gaps in the terminal regions, we align the contact eigenvector sequences for the query and each template (**Fig. S5E**). It is noted that the sign of the eigenvectors has no influence on **Eq. (S4)**, i.e. $\vec{V_i} * \vec{V_i}^T = (-\vec{V_i}) * (-\vec{V_i}^T)$. Therefore, in order to consider all possible combinations, 2^K alignments are required, where K is the maximum number of eigenvectors used to reconstruct the contact maps. Moreover, when aligning K eigenvectors, we compute all alignments with 1 to K eigenvectors. As a result, the procedure must perform a total of $\sum_{k=1}^{K} 2^k = 2^{K+1} - 2$ alignments. For example, if we select K=7, we will evaluate $(2^1 + 2^2 + 2^3 + 2^4 + 2^5 + 2^6 + 2^7)$=254 alignments. In order to choose the best alignment, we utilize the *CMOq* index, which is defined as:

$$CMOq = \frac{O(CM^Q, CM^T)}{N(CM^Q)} \qquad (S7)$$

where $N(CM^Q)$ is the number of contacts in the contact map for a query, and $O(CM^Q, CM^T)$ is the number of overlapped contacts between the aligned query and template. Note that the range of the *CMOq* is [0,1]. We observed a strong correlation (mean Pearson correlation coefficient = 0.945) between *CMOq* and TM-score, which is a widely used measurement to estimate the quality of the alignment between the query and template. Such a strong correlation indicates that the use of *CMOq* is a good index to choose the best alignment and template.

In order to evaluate the precision of predicted contact maps by different methods for all ranges, we calculate the *CMOacc* index using the following equation:

$$CMOacc = \frac{O(CM^{pred}, CM^{native})}{N(CM^{native})} \qquad (S8)$$

where $N(CM^{native})$ is the number of native contacts for the query, and $O(CM^{pred}, CM^{native})$ is the number of overlapped contacts between the native contact map and predicted contact map. Here, we first sort predicted contacts in descending order of the confidence scores, and then select the top $N(CM^{native})$ predicted contacts. Note that the definition of *CMOacc* is similar to top L precision, except we replace L with the number of native contacts, and the value of *CMOacc* is close to 1 when the precision of the predicted contacts is 100%, while it is zero when the prediction precision is zero percent.

Text S3. Optimization of the number of eigenvectors used by CEthreader

Here, we justify the number of eigenvectors that we select in this study by analyzing the effect of the number of selected eigenvectors on the alignment accuracy and the time required for an alignment.

We tested two strategies: a fixed length model and a cumulative length model on the query-template pair dataset, where 335 query sequences were aligned to 905 templates, in order to determine the optimum number of eigenvectors. The fixed length model considered only K eigenvectors, while the cumulative length model considered 1 to K eigenvectors, where K ranged from 1 to 18. The TM-score based on the cumulative length model was always greater than that by the fixed length model, as shown in **Fig. S6A**. This is understandable because the search space for picking up the best alignment is relatively larger in the former model. Regardless of which model was being considered, the average TM-score increased as the number of selected eigenvectors increased. In particular, the average TM-score between the query and template increased rapidly from 0.533 to 0.633 when the selected number of eigenvectors increased from 1 to 7 when considering the cumulative length model. However, the improvement in the alignment quality was not very significant when more than 7 eigenvectors were used, since the average TM-score increased only an additional 2% when 8 to 18 eigenvectors were chosen. We note that, here, only contact information (S_{cm}) was used in the scoring function of the dynamic programming algorithm.

Although the alignment quality improved as the number of selected eigenvectors, K, was increased, the time complexity likewise increased logarithmically as the number of selected eigenvectors increased (**Fig. S6B**). For example, when we set K=7 in the cumulative model, the average time required to align a query sequence to the 905 templates was 1.55 hours ($e^{8.63}$ seconds). On the other hand, 6,710 hours (e^{17} seconds) were required when K was set to 18. While searching a query composed of 200 amino acids through the whole SCOPe database comprising 23,000 templates takes ~10 hours using K=7, it would require an enormous amount of time to perform threading with a higher value of K. Therefore, achieving the highest alignment quality level by increasing the number of eigenvectors is not feasible when only contact information is used in the scoring function.

When contact information was combined with the profile and secondary structure information (see **Text S5**), the average TM-score of the templates from a similar query-template pool was 0.66 based on K=7 using the cumulative model, as shown by the red circle in **Fig. S6A**. Additionally, the average TM-scores reached a plateau beyond 7 eigenvectors. Therefore, combining contact information with profile and secondary structure is more reasonable than increasing the number of eigenvectors. In order to further illustrate this point, we highlight an example from the alignment between the query Human ENPP4 with a Cleavable ATP-Analogue (SCOPe ID: d4le5a3) and the template S-adenosylmethionine synthetase (SCOPe ID: d1mxaa3) using different numbers of eigenvectors, where the results are presented in **Fig. S6C**. Here, the selection of 7 or 18 eigenvectors resulted in TM-scores of 0.81 or 0.90, respectively, when only contact information was used. On the other hand, with the combination of contacts, profile and secondary structure information, using K=7 eigenvectors resulted in a TM-score of 0.904. The inset in **Fig. S6C** shows that the overlap between the query and template, defined by $CMOq$ (**Eq. S7**), became flat when more than 7 eigenvectors were used, indicating optimum alignment between the query and template was achieved at *K*=7. Additionally, the eigenvalues did not decrease significantly when more than 7 eigenvectors were considered. Overall, the benchmark results indicate that the selection of 7 eigenvectors is sufficient and time efficient for reconstructing the query and template contact maps and for generating a high-quality alignment between the two. Therefore, we select 7 eigenvectors to convert the two-body contact map information into a single-body potential, which is combined with the profile and secondary structure information for alignment and threading.

Text S4. Optimization of the scoring function used for aligning contact maps

As mentioned before, we use a semi-global dynamic programming algorithm to align the contact eigenvectors for contact-map matching. The semi-global algorithm does not penalize gaps at the termini of the sequences, as gaps only incur penalties in the middle of the sequences. Here, we utilize an affine penalty schemes, $G_{cm} = g_o(cm) + g_e(cm) * l$, with gap opening penalty $g_o(cm) = -1.0$ and gap extension penalty $g_e(cm) = -0.1$, where l is the length of the gap in the contact-map (cm) alignment.

To align two contact-maps, Di Lena et al. [2] used the following dot product (dp) scheme:

$$S_{dp}(i,j) = \overrightarrow{U_i} \cdot \overrightarrow{P_j} = \sum_{l=1}^{k} \sqrt{\lambda_l} u_{il} \sqrt{\lambda_l} p_{jl} \qquad (S9)$$

where $\overrightarrow{U_i} = \left(\sqrt{\lambda_1}u_{i,1}, \sqrt{\lambda_2}u_{i,2}, \dots, \sqrt{\lambda_k}u_{i,k}\right)$ is the contact eigenvector of the i-th residue of a query and $\overrightarrow{P_j} = \left(\sqrt{\lambda_1}p_{j,1}, \sqrt{\lambda_2}p_{j,2}, \dots, \sqrt{\lambda_k}p_{j,k}\right)$ is the contact eigenvector of the j-th residue of the template.

In our study, we propose a revised version of the dot product scoring function, which shows an improved alignment performance, as discussed later. The modified dot product scoring function is written as:

$$S_{cm}(i,j) = \begin{cases} \dfrac{\overrightarrow{U_i} \cdot \overrightarrow{P_j}}{\max\,(|\overrightarrow{U_i}|, |\overrightarrow{P_j}|)^{\alpha}} & \text{if } \overrightarrow{U_i} \neq \vec{0} \textit{ and } \overrightarrow{P_j} \neq \vec{0} \\ 0 & \text{if } \overrightarrow{U_i} = \overrightarrow{P_j} = \vec{0} \end{cases} \qquad (S10)$$

Here, we use the $\max\,(|\overrightarrow{U_i}|, |\overrightarrow{P_j}|)^{\alpha}$ to nomalize the dot product scoring function. In order to optimize the parameter α in the scoring function, we set it to a value ranging from [0.5, 4.0] at an interval of 0.1, and calculated the mean TM-score and mean *CMOq* from the alignment results based on different α values. As shown in **Fig. S7**, $\alpha \approx 2.0$ results in the largest *CMOq* and TM-score based on the gap penalty. Therefore, we selected α=2.0 in our final scoring function:

$$S_{cm}(i,j) = \begin{cases} \dfrac{\overrightarrow{U_i} \cdot \overrightarrow{P_j}}{\max\,(|\overrightarrow{U_i}|, |\overrightarrow{P_j}|)^{2}} & \text{if } \overrightarrow{U_i} \neq \vec{0} \textit{ and } \overrightarrow{P_j} \neq \vec{0} \\ 0 & \text{if } \overrightarrow{U_i} = \overrightarrow{P_j} = \vec{0} \end{cases} \qquad (S11)$$

The selection of α=2 has further mathematical significance, which is illustrated by the examples in **Fig. S7C**. As shown in the top left portion of **Fig. S7C**, two contact eigenvectors with lengths 2 and 10 are at an angle of 30^0. However, on the top right side of the figure, there are two contact eigenvectors with lengths 2 and 1.8, which are at a similar included angle. Based on the definition of the dot product **(S_{dp})** scoring function by Di Lena et al., the former group of vectors has a score of 17.3, while the latter group has a score of 3.06, indicating the contact eigenvector of length 2 tends to match with the contact eigenvector of length 10 during dynamic programming alignment. However, since the aim of our method is to align two similar contact eigenvectors, we should design a scoring function in a way that it gives a higher score to the second group of vectors than the first group in the example, since the second group of vectors are more similar. To this end, we designed an extended dot product **(S_{cm})** scoring function, which has the ability to scale different contact eigenvectors into a unit circle with a radius of 1 according to the larger contact eigenvector. Based on our scoring function, the first and second groups have a score of 0.173 and 0.765, respectively, illustrating the scoring function has a better mathematical significance.

To further examine the performance of the normalized scoring function, we calculate the average *CMOq* values and average TM-scores for 905 query-template pairs based on our scoring function and the one used by Di Lena et al. and present the findings in **Table S8**. Since the dataset is selected uniformly, we perform paired one-sided Wilcoxon single-ranked tests instead of Student t-tests to evaluate the significance of the difference between the dot product scoring function, S_{dp}, and our improved dot product scoring function, S_{cm}. Based on the same sets of contact map predictions, the average CMOq values (TM-scores) using S_{dp} and S_{cm} were 0.4165 (0.5853) and 0.4802 (0.6334), respectively, with a p-value of 1.54E-115 (2.48E-82). This indicates that the normalized dot product scoring function is more efficient than the simple dot product scoring function for detecting templates of similar folds as the query.

Text S5. Optimization of the component scores used to match the secondary structure and sequence profiles of two proteins

In addition to the contact-map alignment guided by **Eq. (S11)**, CEthreader includes two other energy terms that depend on sequence profile and secondary structure matching, which have been shown to be essential for improving alignment accuracy for both close- and distant-homology proteins [3]. Inspired by MUSTER, the normalized profile term is defined by [4]:

$$S_{prof}(i,j) = \sum_{k=1}^{20} P(i,k) * L(k,j)/max_{m,n}(|\sum_{k=1}^{20} P(m,k) * L(k,n)|), \qquad (S12)$$

where $P(i,k)$ is the frequency of the *k*-th amino acid at the *i*-th position in a multiple sequence alignment (MSA) obtained by PSI-BLAST search [5] of the query sequence through the NR database (ftp://ftp.ncbi.nlm.nih.gov/blast/db), where Henikoff [6] weighting is used to reduce the redundancy in the MSA. $L(k,j)$ is the log-odds profile (Position-Specific Substitution Matrix in PSI-BLAST with an *E*-value of 0.001) of the template sequence for the *k*-th amino acid at the *j*-th position. Here, we use a similar affine penalty scheme as used by the contact-based term, where the gap penalty $G_{prof} = g_o(prof) + g_e(prof) * l$, with gap opening penalty $g_o(prof) = -1.0$ and gap extension penalty $g_e(prof) = -0.1$, where *l* is the length of the gap in the profile (prof) guided alignments.

The secondary structure (SS) alignment term in CEthreader is defined by

$$S_{ss}(i,j) = \delta(s_i, s_j) \qquad (S13)$$

where s_i is the secondary structure of the *i*-th residue of the query as predicted by PSSpred [7] and s_j is the secondary structure of the *j*-th residue of the template as assigned by the DSSP program [8]. To align the query and template secondary structures, we follow a very similar protocol as MUSTER [4], where $S_{ss}(i,j)$ is equal to 0.093 if $s_i = s_j$, and -0.093 otherwise. Again we employ an affine penalty scheme, where the gap penalty $G_{ss} = g_o(ss) + g_e(ss) * l$, with gap opening penalty $g_o(ss) = -1.0$ and gap extension penalty $g_e(ss) = -0.077$, where l is the gap length. Moreover, no gaps are allowed inside the continuous secondary structure regions (helices and strands), i.e., with gap open and gap extension penalties of -100 in these regions.

In summary, the final multi-feature scoring function and gap penalties used in CEthreader can be written as:

$$\begin{cases} S_{cm+ss+prof}(i,j) = \omega_1 S_{cm}(i,j) + \omega_2 S_{prof}(i,j) + \omega_3 S_{ss}(i,j) + \omega_4 \\ g_o = \omega_1 g_o(cm) + \omega_2 g_o(prof) + \omega_3 g_o(ss) \\ g_e = \omega_1 g_e(cm) + \omega_2 g_e(prof) + \omega_3 g_e(ss) \\ \omega_1 + \omega_2 + \omega_3 = 1 \\ \omega_4 > 0 \end{cases} \quad (S14)$$

Here, we utilize a weighting strategy for the three scoring function terms and gap penalties. Also, we set a bonus term for matching identical residues to each other. The optimized values of the four parameters are $\omega_1 = 0.5, \omega_2 = 0.4, \omega_3 = 0.1$ and $\omega_4 = 0.1$, based on the 905 query-template pairs dataset.

Supporting Tables

Table S1. Summary of the threading alignments produced by CEthreader for the 614 test proteins in Benchmark Set-I, where different component scores were used. P-values were calculated between $S_{cm+ss+prof}$ and the other score functions based on Wilcoxon signed-rank tests. N_{st} is the number of cases with a TM-score >0.5 in each category.

Target type (# proteins)	**Component scores**	**TM-score**	p**-value**	**RMSD**	**Coverage**	$\boldsymbol{N_{st}}$
Hard (211)	$S_{cm+ss+prof}$	0.453	-	9.531	0.875	80
	S_{cm}	0.439	2.31E-03	9.815	0.873	71
	$S_{ss+prof}$	0.284	7.75E-30	14.781	0.832	19
Easy (403)	$S_{cm+ss+prof}$	0.687	-	4.795	0.909	365
	S_{cm}	0.658	4.02E-22	5.270	0.907	339
	$S_{ss+prof}$	0.656	1.14E-15	5.286	0.883	347
All (614)	$S_{cm+ss+prof}$	0.607	-	6.423	0.898	445
	S_{cm}	0.583	1.64E-19	6.832	0.895	410
	$S_{ss+prof}$	0.528	2.14E-45	8.549	0.866	366

Table S2. Threading results by different methods on the 403 Easy targets from Benchmark Set-I. P-values were calculated between the CEthreader alignment TM-scores and the other methods' TM-scores using pairwise one-sided Wilcoxon signed-rank tests; coverage is the number of aligned residues divided by length of the query; N_{st} denotes the number of targets whose templates were correctly identified with a TM-score >0.5.

Methods	TM-score	p-value	RMSD	Coverage	N_{st}
CEthreader	0.687	-	4.795	0.909	365
HHsearch	0.682	2.24E-03	4.783	0.893	363
MUSTER	0.667	2.67E-11	5.171	0.895	348
PPA	0.656	1.14E-15	5.286	0.883	347
map_align	0.641	3.23E-23	5.724	0.902	328
EigenThreader	0.634	5.10E-39	5.670	0.904	324
SAM-T99	0.631	6.14E-29	5.020	0.839	330
PROSPECT2	0.618	1.46E-27	6.898	0.907	306
FFAS03	0.524	9.59E-49	7.350	0.810	258

Table S3. Threading results for different methods using Benchmark Set-I separated based on sequence identity to the ResPRE training set. "<30%" corresponds to the subset of 149 Easy targets and 90 Hard targets which have sequence identities <30% to the ResPRE training set. "≥30%" represents the subset of 254 Easy targets and 121 Hard targets that have sequence identities ≥30% (and <40%) to the ResPRE training set. *P*-values were calculated between CEthreader alignment TM-scores and other methods' TM-scores using pairwise one-sided Wilcoxon signed-rank tests.

Target	**Methods**	***<30%***		***≥30%***	
		TM-score	***p*-value**	**TM-score**	***p*-value**
Easy targets	**CEthreader**	0.6729	-	0.6957	-
	HHsearch	0.6711	3.51E-02	0.6878	2.48E-03
	MUSTER	0.6548	7.49E-04	0.6738	2.59E-09
	PPA	0.6410	1.00E-05	0.6655	7.44E-12
	SAM-T99	0.6278	5.02E-10	0.6327	9.14E-21
	EigenThreader	0.6175	1.17E-15	0.6440	2.74E-25
	map_align	0.6073	8.43E-13	0.6610	1.93E-12
	PROSPECT2	0.6036	3.60E-12	0.6256	2.55E-17
	FFAS03	0.4838	3.75E-20	0.5478	2.04E-30
Hard targets	**CEthreader**	0.4372	-	0.4649	-
	EigenThreader	0.4031	1.86E-04	0.4208	1.00E-06
	map_align	0.4008	1.62E-03	0.4241	2.18E-04
	HHsearch	0.3115	1.70E-10	0.3137	7.98E-16
	MUSTER	0.2929	2.98E-12	0.3124	1.22E-17
	PPA	0.2618	2.53E-14	0.2999	1.59E-17
	PROSPECT2	0.2606	9.26E-16	0.2611	1.71E-21
	SAM-T99	0.2077	1.24E-15	0.2086	5.30E-21
	FFAS03	0.1833	3.83E-16	0.1936	1.99E-21

Table S4. Fold-recognition performance of CEthreader, HHsearch and MUSTER on 116 Fold families in Benchmark Set-II. *Nq*: numbers of members in the SCOPe Fold families. *Nce, Nhh,* and *Nmu*: number of cases with a correct Fold template detected by CEthreader, HHsearch, and MUSTER, respectively, after filtering out all templates in the same Superfamily.

Fold	*Nq*	*Nce*	*Nhh*	*Nmu*	Fold	*Nq*	*Nce*	*Nhh*	*Nmu*
a.1	2	0	0	0	b.159	2	1	2	1
a.2	6	0	0	0	c.1	29	29	23	22
a.4	7	4	3	1	c.6	3	0	0	0
a.5	2	0	0	0	c.8	10	9	3	1
a.7	9	6	2	0	c.9	1	0	0	0
a.8	5	0	0	0	c.10	2	2	2	2
a.11	2	2	2	1	c.13	2	1	1	0
a.23	2	0	0	0	c.23	14	9	4	2
a.24	21	14	2	1	c.26	3	0	0	0
a.25	3	2	1	0	c.44	2	0	0	0
a.28	1	1	0	0	c.47	1	0	0	0
a.29	11	8	0	0	c.49	2	0	0	0
a.39	3	1	1	1	c.51	6	3	2	1
a.40	2	0	0	0	c.52	3	1	0	0
a.47	6	1	0	1	c.53	2	0	0	0
a.48	3	0	0	0	c.55	6	6	1	0
a.60	7	1	3	1	c.56	8	7	3	4
a.69	3	0	0	0	c.67	3	2	0	0
a.70	2	0	0	0	c.72	3	1	0	0
a.71	2	1	0	0	c.78	2	0	0	0
a.102	4	3	3	2	c.92	3	0	2	0
a.118	20	17	11	14	c.97	2	1	0	0
a.137	3	0	0	0	c.98	1	1	1	0
a.144	1	0	0	0	d.13	2	0	0	0
a.159	2	0	0	0	d.15	8	8	3	1
a.246	3	1	0	0	d.17	4	3	0	0
b.1	27	24	14	8	d.26	2	1	0	0
b.2	8	2	0	0	d.41	5	2	2	0
b.3	4	3	2	1	d.43	1	0	0	0
b.6	2	0	0	0	d.50	1	0	0	0
b.7	2	1	0	0	d.52	4	3	1	0
b.23	3	0	0	0	d.58	41	32	12	5
b.30	2	1	0	0	d.64	1	0	0	0
b.34	13	8	4	0	d.67	3	0	0	0
b.35	1	0	0	0	d.68	2	1	0	0
b.38	2	1	0	0	d.74	2	2	0	0
b.40	10	9	3	1	d.75	1	0	0	0
b.42	6	5	5	5	d.79	9	8	1	0
b.43	4	1	0	0	d.81	4	3	0	0
b.44	1	1	0	0	d.82	2	0	0	1
b.45	2	2	0	0	d.83	2	1	0	1
b.49	2	0	1	0	d.87	2	0	0	0
b.52	2	1	0	0	d.92	2	0	0	0
b.55	1	0	0	0	d.94	1	0	0	0
b.61	6	4	1	0	d.95	1	0	0	0
b.67	3	1	0	0	d.96	2	0	0	0
b.68	10	3	5	4	d.98	2	0	0	0

b.69	10	9	5	6
b.76	2	1	0	0
b.77	3	3	1	0
b.80	6	6	0	1
b.81	3	1	0	0
b.82	6	6	3	4
b.84	3	1	1	0
b.85	3	1	2	0
b.88	2	2	0	0
b.121	7	5	0	0
b.129	1	0	0	0
d.109	3	2	2	1
d.110	9	8	6	5
d.129	9	4	1	0
d.142	2	0	1	0
d.153	2	2	0	0
d.170	1	1	0	0
d.198	4	3	2	0
d.211	1	0	0	0
d.230	1	0	0	0
d.241	2	0	0	0
d.248	2	2	1	0

Table S5. List of Fold pairs in Benchmark Set-II with an average TM-score >0.4.

Fold	Fold	TM-score	Fold	Fold	TM-score
a.2	a.7	0.578	a.7	a.24	0.474
a.2	a.8	0.506	a.7	a.29	0.481
a.2	a.24	0.535	a.7	a.47	0.583
a.2	a.29	0.550	a.7	a.102	0.456
a.2	a.47	0.597	a.7	a.118	0.423
a.2	a.102	0.485	a.8	a.24	0.492
a.2	a.118	0.449	a.8	a.29	0.512
a.2	c.1	0.408	a.8	a.47	0.455
a.2	c.56	0.411	a.8	a.102	0.400
a.2	d.129	0.405	a.24	a.29	0.452
a.7	a.8	0.459	a.24	a.47	0.446
a.29	a.47	0.443	c.1	c.8	0.401
a.29	a.60	0.405	c.1	c.23	0.427
a.102	a.118	0.427	c.1	d.79	0.402
b.1	b.2	0.518	c.23	c.51	0.421
b.1	b.3	0.464	c.23	c.56	0.444
b.1	b.121	0.428	c.51	c.56	0.425
b.2	b.3	0.419	c.56	d.198	0.405
b.2	b.121	0.412	d.41	d.58	0.439
b.61	d.17	0.419	d.41	d.129	0.416
b.61	d.129	0.404	d.52	d.58	0.417
b.68	b.69	0.600	d.58	d.129	0.418

Table S6. Summary of models built by MODELLER based on different threading methods for all 614 proteins in Benchmark Set-I. P-values were calculated between the TM-scores of the models built using CEthreader alignments and other threading programs using pairwise one-sided Wilcoxon signed-rank tests. N_{st} represents the number of targets whose templates had a TM-score >0.5.

Target type (# proteins)	**Methods**	**TM-score**	***p*-value**	**RMSD (Å)**	N_{st}
All (614)	CEthreader	0.628	-	4.999	476
	EigenThreader	0.585	3.05E-40	4.952	418
	map_align	0.584	7.26E-16	5.163	412
	HHsearch	0.571	1.77E-24	4.934	403
	MUSTER	0.564	1.08E-32	4.860	394
	PPA	0.548	2.43E-42	5.043	381
	PROSPECT	0.533	5.31E-49	5.018	352
	SAM-T99	0.510	1.64E-57	4.827	351
	FFAS03	0.425	1.25E-85	5.281	270
Easy (403)	CEthreader	0.708	-	4.749	372
	EigenThreader	0.656	1.62E-37	4.932	343
	map_align	0.663	2.03E-14	5.016	337
	HHsearch	0.698	4.42E-05	4.830	366
	MUSTER	0.694	6.61E-08	4.779	363
	PPA	0.677	2.58E-14	4.913	358
	PROSPECT	0.666	8.86E-15	5.024	339
	SAM-T99	0.658	7.63E-22	4.837	340
	FFAS03	0.541	7.39E-49	5.150	262
Hard (211)	CEthreader	0.476	-	5.475	104
	EigenThreader	0.449	4.21E-07	4.992	75
	map_align	0.432	6.04E-05	5.444	75
	HHsearch	0.330	3.72E-24	5.133	37
	MUSTER	0.316	4.68E-28	5.014	31
	PPA	0.301	2.01E-29	5.289	23
	PROSPECT	0.278	1.85E-34	5.006	13
	SAM-T99	0.228	3.32E-34	4.807	11
	FFAS03	0.205	9.68E-36	5.532	8

Table S7. Summary of model quality for models built by CEthreader/C-I-TASSER, CEthreader/I-TASSER and ResPRE/CNS for all 614 proteins in Benchmark Set-I. P-values were calculated between the model TM-scores for CEthreader/C-I-TASSER and the other modeling approaches using pairwise one-sided Wilcoxon signed-rank tests. N_{st} represents the number of targets whose templates had a TM-score >0.5.

Target type (# proteins)	Methods	TM-score	p-value	RMSD (Å)	N_{st}
All (614)	CEthreader/C-I-TASSER	0.686	-	4.575	547
	CEthreader/I-TASSER	0.653	2.29E-21	4.769	502
	ResPRE/CNS	0.490	2.14E-91	4.707	312
Easy (403)	CEthreader/C-I-TASSER	0.734	-	4.319	384
	CEthreader/I-TASSER	0.721	6.36E-05	4.481	370
	ResPRE/CNS	0.518	1.19E-65	4.804	230
Hard (211)	CEthreader/C-I-TASSER	0.595	-	5.063	163
	CEthreader/I-TASSER	0.524	1.01E-21	5.319	132
	ResPRE/CNS	0.436	2.65E-26	4.523	82

Table S8. CEthreader threading alignment results for the 905 query-template pairs using native contact maps or contact maps predicted using 18 different contact predictors [1,9-23]. Different contact predictors are sorted in descending order of contact accuracy, *CMOacc*. 'cm' and 'dp' correspond to alignments generated using the contact map score from **Eq. (6)** and the dot-product score from **Eq. (S9)** as proposed by Di Lena et al. [2], respectively.

Methods	**CMOacc**	**TM-score**		**CMOq**	
		cm	**dp**	**cm**	**dp**
native	1	0.6492	0.5975	0.5814	0.5021
ResPRE	0.8462	0.6334	0.5853	0.4802	0.4165
DeepContact	0.7634	0.6114	0.5724	0.4269	0.3775
DeepCov	0.6561	0.5830	0.5473	0.3726	0.3349
NeBcon	0.6244	0.5738	0.5412	0.3489	0.3140
PconsC2	0.5796	0.5632	0.5287	0.3276	0.2908
MetaPSICOV	0.5731	0.5700	0.5406	0.3350	0.3011
GREMLIN	0.3586	0.4931	0.4714	0.1956	0.1767
CCMpred	0.3579	0.4910	0.4676	0.1940	0.1752
SVMcon	0.3573	0.4684	0.4713	0.2242	0.2127
BETAcon	0.3319	0.4658	0.4712	0.2173	0.2074
SVMSEQ	0.3316	0.4606	0.4664	0.2234	0.2122
plmDCA	0.3160	0.4699	0.4567	0.1691	0.1550
PSICOV	0.3033	0.4750	0.4615	0.1738	0.1598
NNcon	0.2897	0.4342	0.4435	0.2002	0.1889
PSpro.beta	0.2633	0.4215	0.4374	0.1847	0.1754
DNcon	0.1986	0.4590	0.4599	0.2134	0.1987
FreeContact	0.1497	0.3127	0.3053	0.1287	0.1208
PSpro	0.1314	0.4277	0.4317	0.3266	0.3053

Supporting Figures

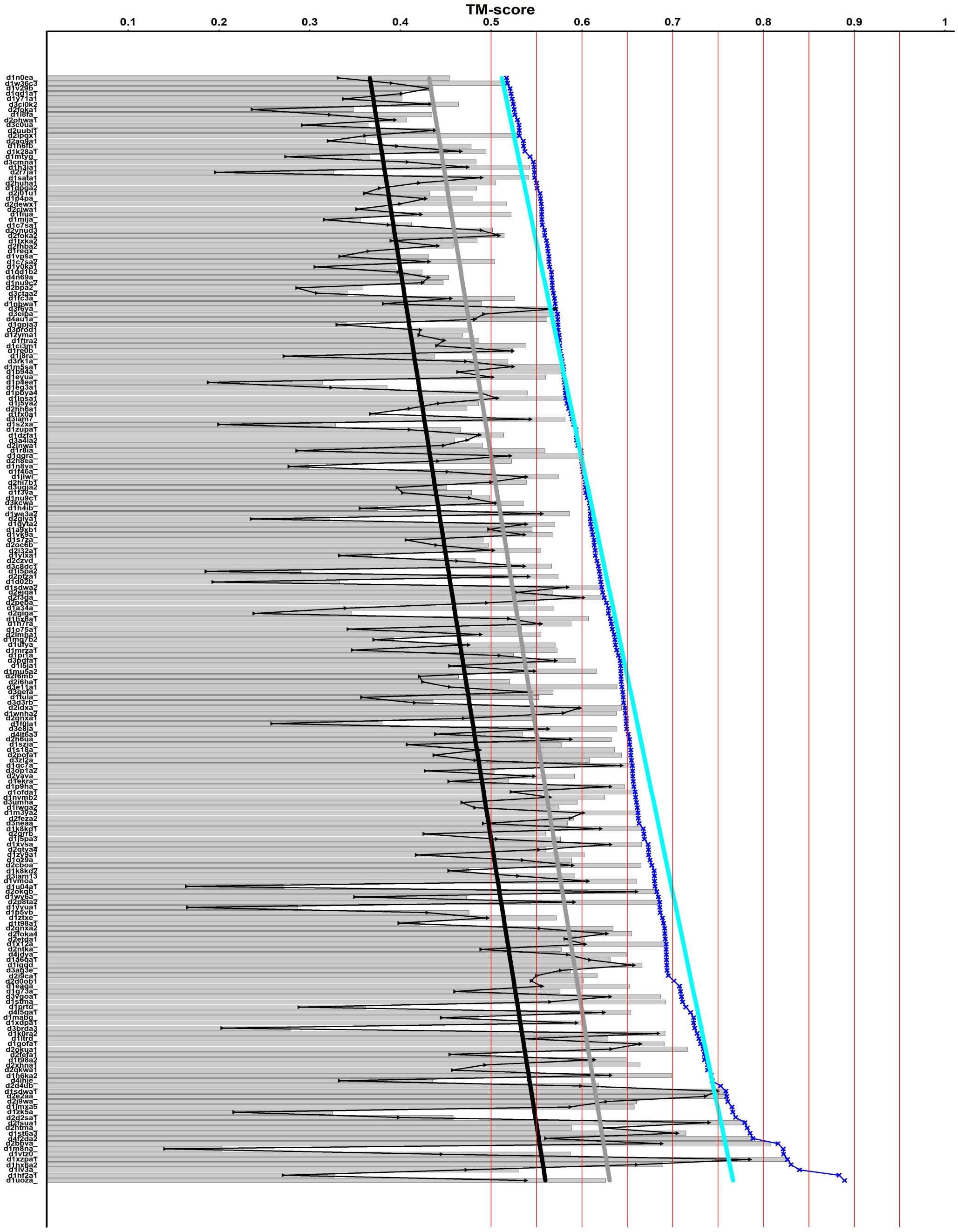

Figure S1. CEthreader template information for the 211 Hard targets from Benchmark Set-I. The blue cross points are the TM-scores of the best possible templates aligned to the corresponding target by the structure-based alignment method, TM-align; the gray bars mean the TM-scores of the first template detected by CEthreader but aligned by TM-align; the black triangles represents the TM-scores of the first templates detected and aligned by CEthreader. The linear regression is used to fit each corresponding set of TM-scores, where the fitted relationships for the cyan line, gray line and black line are y=0.501+ 0.001x, y=0.421+0.0008x, and y=0.356+0.0008x, respectively. Here we varied x from 1 to 211, which corresponds to the order of each target ranked by the TM-score of the best possible template.

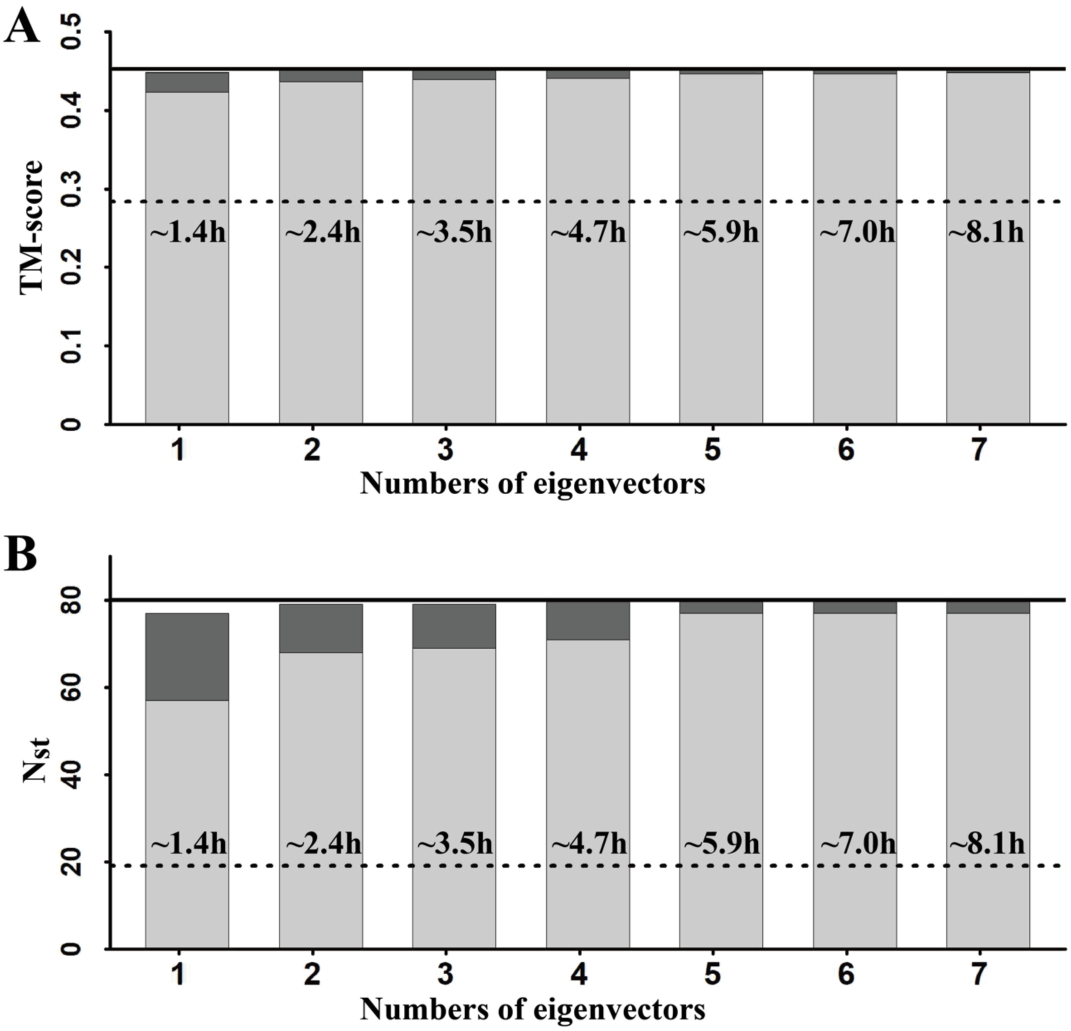

Figure S2. The performance and time-cost by CEthreader using different numbers of eigenvectors in the greedy search step. The height of the grey bar represents the average TM-score of the first template in (A) and the number of targets with identified templates that had a TM-score >0.5 in (B) based on the greedy searching strategy. The grey bar plus the black bar represents the results based on the hybrid searching strategy. The solid lines are the results based on enumerative searching with 7-dimensional eigenvectors using the scoring function $S_{cm+ss+prof}$, where the dotted line represents the results based only on profile and secondary structure information ($S_{ss+prof}$).

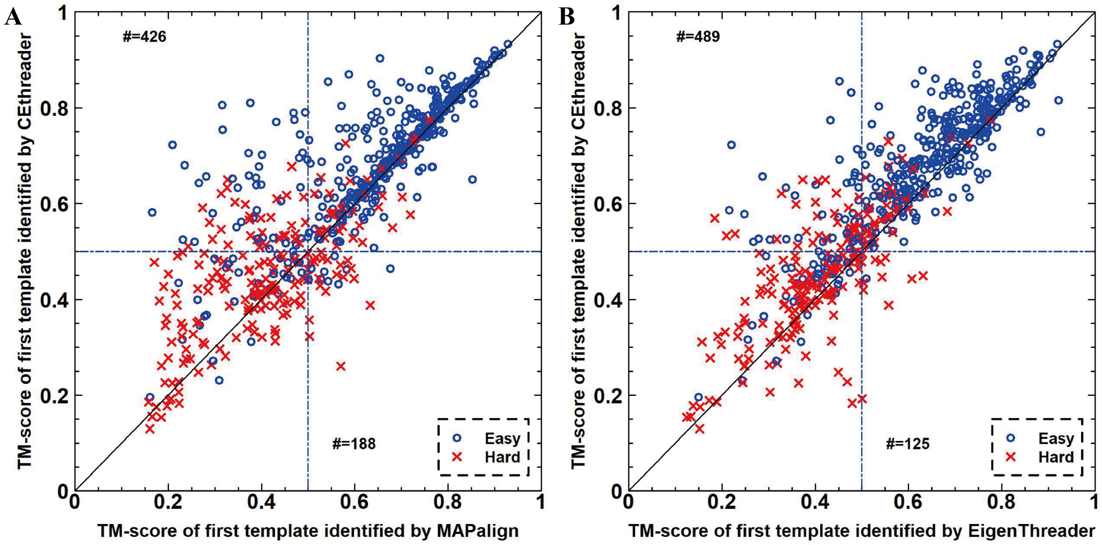

Figure S3. Comparison of CEthreader's performance to map_align and EigenThreader on Benchmark Set-I. (A) TM-score of the first template identified by CEthreader versus that by map_align. (B) TM-score of the first template identified by CEthreader versus that by EigenThreader.

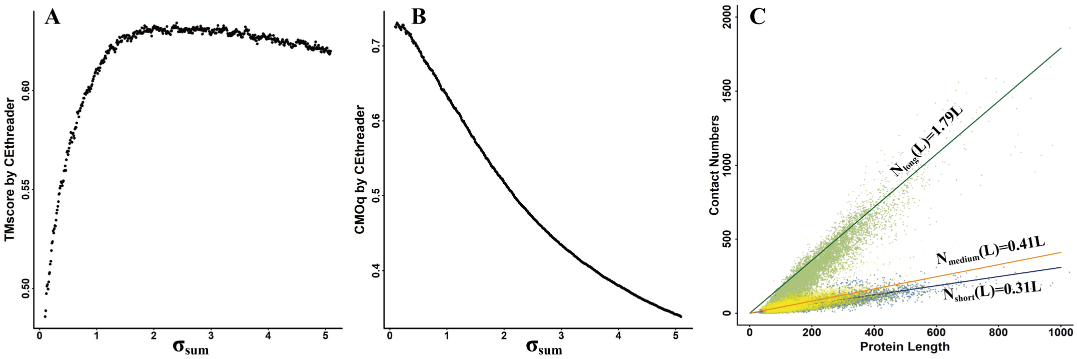

Figure S4. Optimization of the number of predicted contacts to be used by CEthreader. (A, B) TM-score and $CMOq$ as a function of σ_{sum} by CEthreader on 905 training protein pairs. (C) Correlation between the number of contacts and the sequence length for 9,896 non-redundant domains from the SCOPe database with a pair-wise sequence identity cutoff of <30%. The linear regression with fitted σ_{cr} parameters coincides well with the experimental structure data.

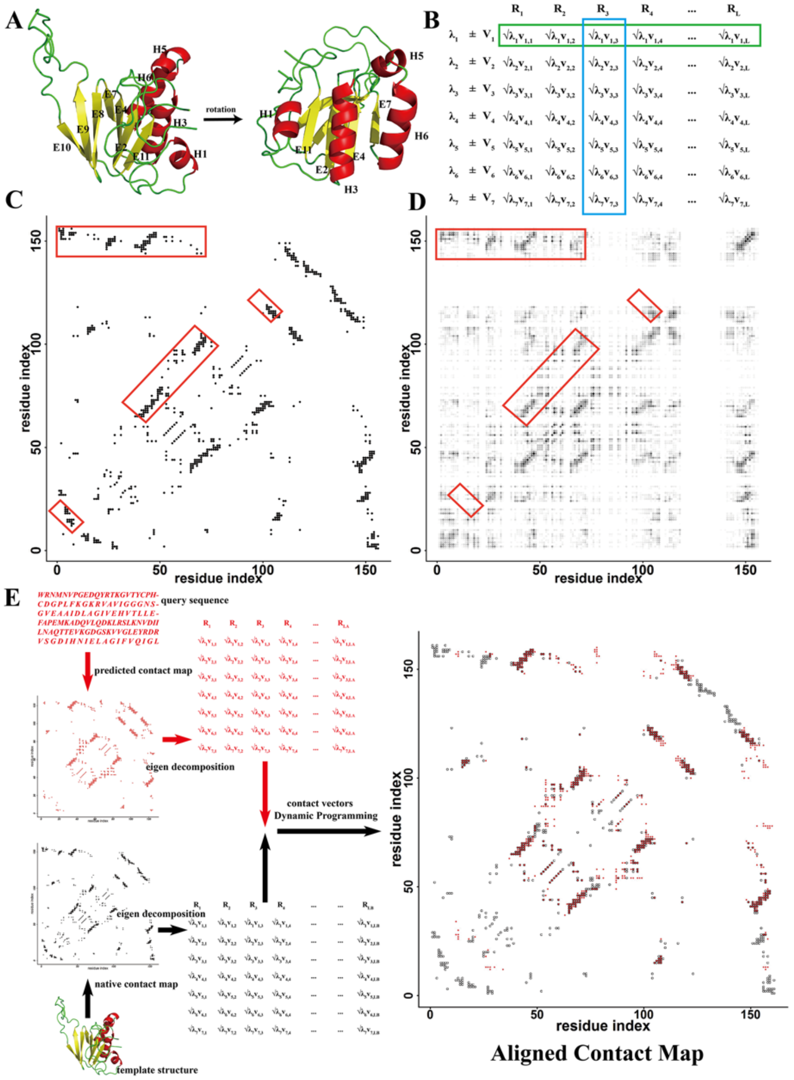

Figure S5. The diagrammatic illustration of eigen-decomposition of a contact-map. (A) 3D structure of the example from the trimethylamine dehydrogenase (SCOPe ID: d1o94a2). (B) the largest 7 eigenvalues and the corresponding eigenvectors decomposed from the native contact map; the green frame represents the *i*-th eigenvector, V_i, weighted by the square root of the corresponding eigenvalue, λ_i, and the blue frame describes the third contact eigenvector. (C) Native contact map for d1o94a2. (D) The contact map reconstructed using the largest 7 eigenvalues and their corresponding eigenvectors. The critical contacting residues in the red frames of (C) and (D) are very similar, indicating that the largest 7 eigenvalues and associated eigenvectors are sufficient for reconstructing the contact map. (E) Diagram of the query-template alignment. The upper left portion of the picture shows the predicted contact map for the query. The predicted contact map is then decomposed, resulting in the contact eigenvector sequence of the query. Similarly, we get the contact eigenvector sequence of the template from its native contact map, as shown in the lower left portion of the figure. Using a semi-global dynamic programming algorithm, we align the two contact eigenvector sequences that represent the query and template sequences. The picture on the right shows the overlapped contact maps for the query (red) and template (black).

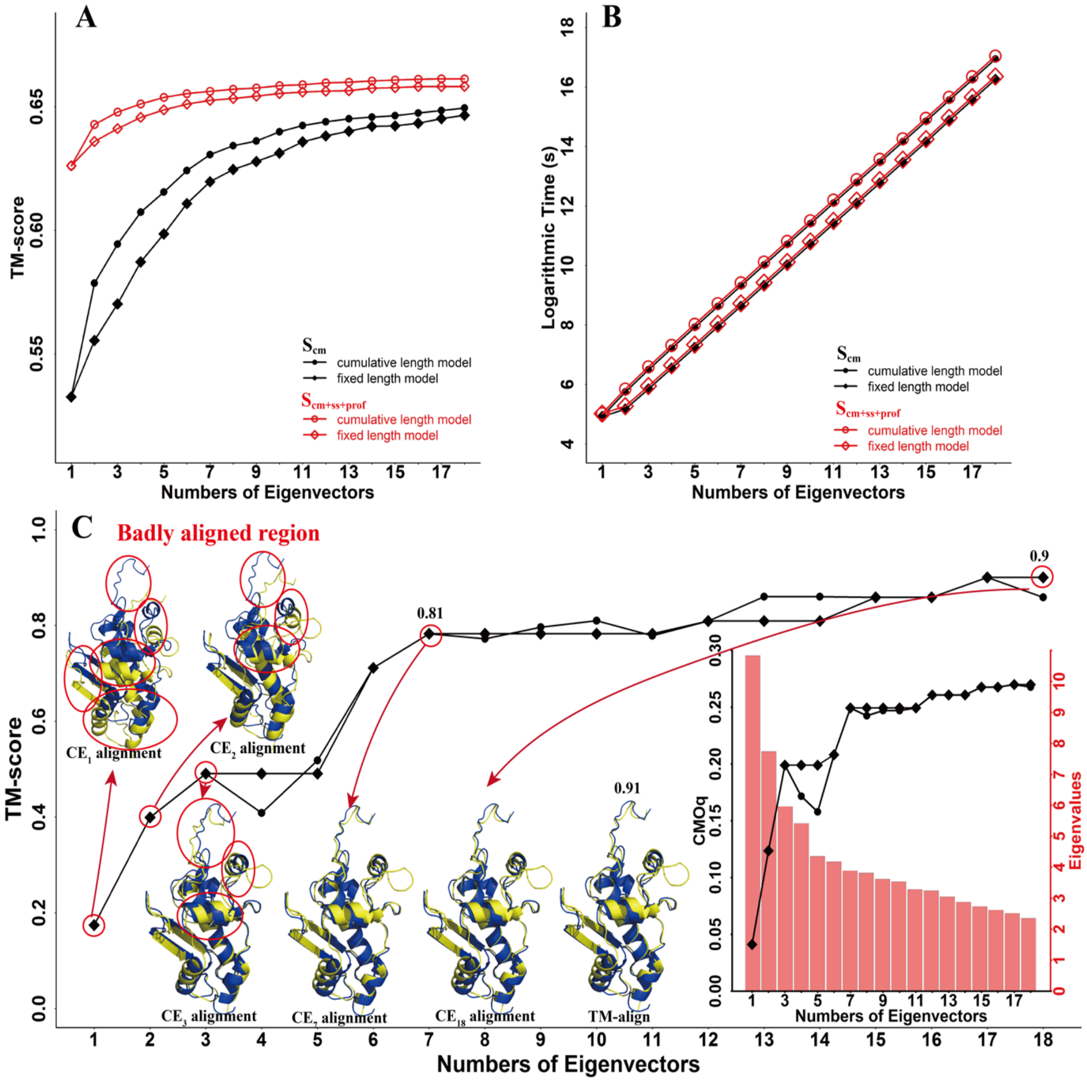

Figure S6. Optimization of the number of eigenvectors used by CEthreader. (A) The average alignment TM-score versus the number of eigenvectors used by CEthreader. Results based on contact information (S_{cm}) are shown using black circles (cumulative length model) and squares (fixed length model), while those based on the combination of contact, secondary structure, and profile information ($S_{cm+ss+prof}$) are highlighted using red circles (cumulative length model) and squares (fixed length model). (B) The logarithmic time required by CEthreader as a function of the number of eigenvectors. (C) TM-score as a function of the number of eigenvectors for the example protein pairs. The Inset shows *CMOq* (black points) and the eigenvalues (red bars) using various numbers of eigenvectors for the example. The superimposed structures of the template from S-adenosylmethionine synthetase (SCOPe ID: d1mxaa3) obtained using different numbers of eigenvectors onto the query Human ENPP4 with a Cleavable ATP-Analogue (SCOPe ID: d4lr5a3) are shown, where badly aligned regions are highlighted with red circles.

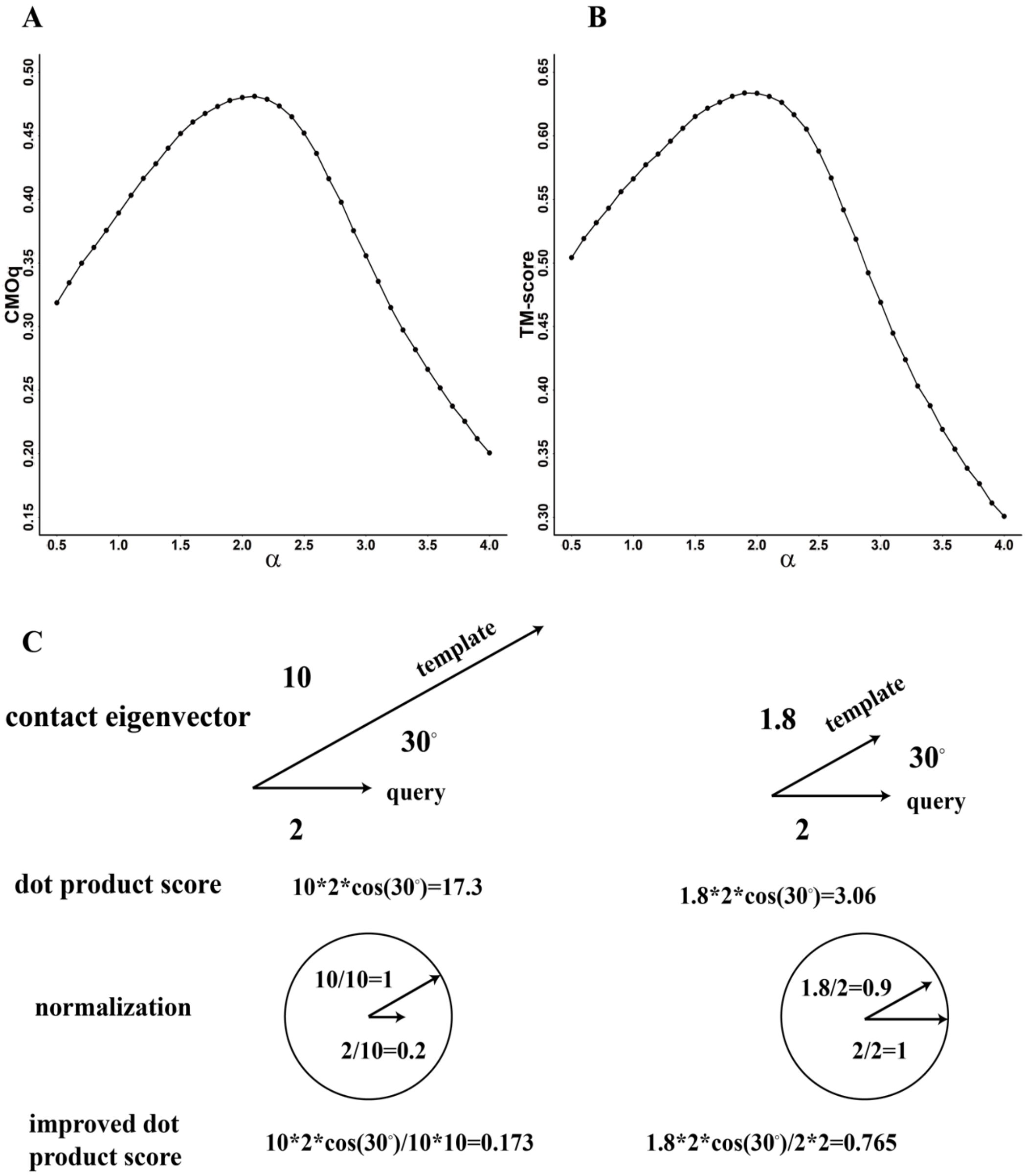

Figure S7. Optimization of the contact map matching score S_{cm}. Based on 905 training query-template pairs, we calculated the average TM-score and average $CMOq$ from the alignment results using different α values in **Eq. (S10)**. (A, B) Average $CMOq$ and TM-score as a function of α. (C) The comparison of dot product scoring function S_{dp} (top panel) from **Eq. (S9)** and the improved dot product scoring function S_{cm} (bottom panel) from **Eq. (S10)**.

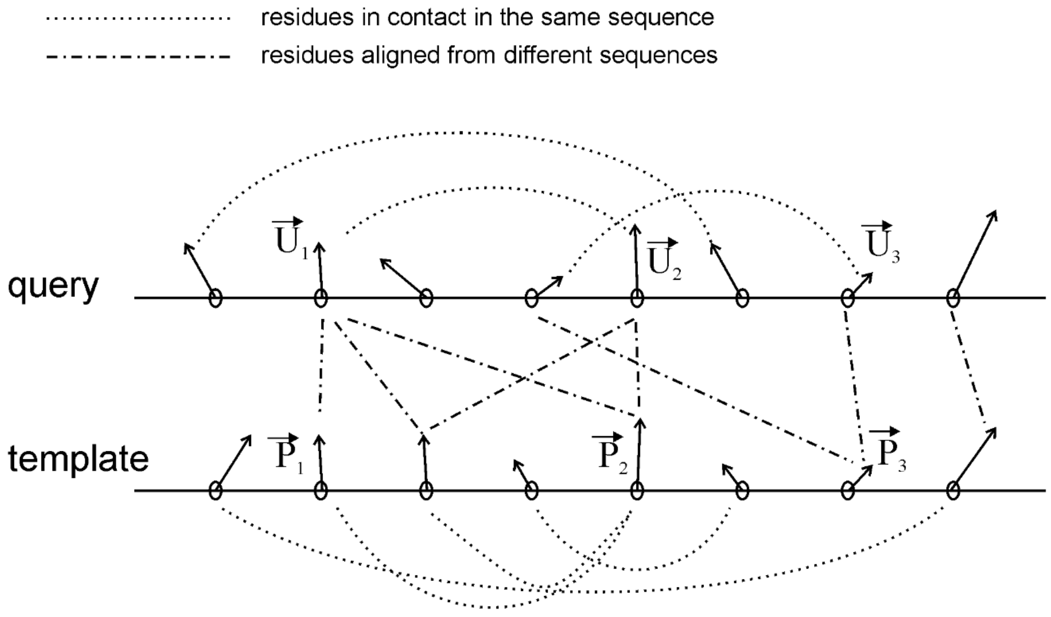

Figure S8. Illustration that alignment of contact eigenvectors between query and template sequences results in the match of global contact maps for the two sequences. Contact eigenvectors of individual residues are represented by the vectors along the sequences. Dotted lines connect the residues that are in contact in the same sequence, while dotted-dashed lines connect those residue pairs across the query and template chains that have high $S_{cm}(i,j)$ scores calculated using **Eq. (6)**. Although the contact vectors are defined by the contact matrix for the same sequence by **Eqs. (3)** and **(4)**, the picture shows that the alignment of $(\overrightarrow{U_\iota}, \overrightarrow{P_j})$ across query and template sequences can enhance the match of the global contact maps between them. In the illustrative example, when the three residue pairs between the query and template are well-aligned, i.e., $\overrightarrow{U_1} \cdot \overrightarrow{P_1} \sim \overrightarrow{U_2} \cdot \overrightarrow{P_2} \sim \overrightarrow{U_3} \cdot \overrightarrow{P_3} \sim 1$, the relationship between $\overrightarrow{U_1}$, $\overrightarrow{U_2}$ and $\overrightarrow{U_3}$, no matter if they are in contact ($\overrightarrow{U_1} \cdot \overrightarrow{U_2} = 1$) or not ($\overrightarrow{U_1} \cdot \overrightarrow{U_3} \ll 1$), will be similar to the relationship between $\overrightarrow{P_1}$, $\overrightarrow{P_2}$ and $\overrightarrow{P_3}$, i.e., the contact maps of the query and template will be similar.

References

1. Li Y, Hu J, Zhang C, Yu D, Zhang Y (2019) ResPRE: high-accuracy protein contact prediction by coupling precision matrix with deep residual neural networks. Bioinformatics.
2. Di Lena P, Fariselli P, Margara L, Vassura M, Casadio R (2010) Fast overlapping of protein contact maps by alignment of eigenvectors. Bioinformatics 26: 2250-2258.
3. Zhang Y (2014) Interplay of I-TASSER and QUARK for template-based and ab initio protein structure prediction in CASP10. Proteins: Structure, Function, and Bioinformatics 82: 175-187.
4. Wu S, Zhang Y (2008) MUSTER: Improving protein sequence profile–profile alignments by using multiple sources of structure information. Proteins: Structure, Function, and Bioinformatics 72: 547-556.
5. S F Altschul, T L Madden, A A Schäffer, J Zhang, Z Zhang, et al. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Res 25(17): 3389–3402.
6. Henikoff S, Henikoff JG (1994) Position-based sequence weights. Journal of Molecular Biology 243: 574-578.
7. Yan R, Xu D, Yang J, Walker S, Zhang Y (2013) A comparative assessment and analysis of 20 representative sequence alignment methods for protein structure prediction. 3: 2619.
8. Kabsch W, Sander C (1983) Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features. Biopolymers 22: 2577-2637.
9. Liu Y, Palmedo P, Ye Q, Berger B, Peng J (2018) Enhancing Evolutionary Couplings with Deep Convolutional Neural Networks. Cell Systems 6: 65-74.e63.
10. Jones DT, Kandathil SM (2018) High precision in protein contact prediction using fully convolutional neural networks and minimal sequence features. Bioinformatics 34: 3308-3315.
11. Skwark MJ, Raimondi D, Michel M, Elofsson A (2014) Improved Contact Predictions Using the Recognition of Protein Like Contact Patterns. PLoS Comput Biol 10: e1003889.
12. Jones DT, Singh T, Kosciolek T, Tetchner S (2015) MetaPSICOV: Combining coevolution methods for accurate prediction of contacts and long range hydrogen bonding in proteins. Bioinformatics 31 (7): 999-1006.
13. Kamisetty H, Ovchinnikov S, Baker D (2013) Assessing the utility of coevolution-based residue-residue contact predictions in a sequence- and structure-rich era. Proc Natl Acad Sci U S A 110: 15674-15679.
14. Seemayer S, Gruber M, Söding J (2014) CCMpred—fast and precise prediction of protein residue–residue contacts from correlated mutations. Bioinformatics.
15. Cheng J, Baldi P (2007) Improved residue contact prediction using support vector machines and a large feature set. BMC Bioinformatics 8: 1-9.
16. Cheng J, Baldi P (2005) Three-stage prediction of protein β-sheets by neural networks, alignments and graph algorithms. Bioinformatics 21: i75-i84.
17. Wu S, Zhang Y (2008) A comprehensive assessment of sequence-based and template-based methods for protein contact prediction. Bioinformatics 24: 924-931.

18. Magnus E, Cecilia L, Yueheng L, Martin W, Erik A (2013) Improved contact prediction in proteins: using pseudolikelihoods to infer Potts models. Phys Rev E Stat Nonlin Soft Matter Phys 87.
19. Jones DT, Buchan DWA, Cozzetto D, Pontil M (2012) PSICOV: Precise structural contact prediction using sparse inverse covariance estimation on large multiple sequence alignments. Bioinformatics 28 (2): 184-190.
20. Tegge AN, Wang Z, Eickholt J, Cheng J (2009) NNcon: improved protein contact map prediction using 2D-recursive neural networks. Nucleic Acids Research 37: W515-W518.
21. Eickholt J, Cheng J (2013) A study and benchmark of DNcon: a method for protein residue-residue contact prediction using deep networks. BMC Bioinformatics 14: S12.
22. Kaján L, Hopf TA, Kalaš M, Marks DS, Rost B (2014) FreeContact: fast and free software for protein contact prediction from residue co-evolution. BMC Bioinformatics 15: 1-6.
23. He B, Mortuza SM, Wang Y, Shen HB, Zhang Y (2017) NeBcon: protein contact map prediction using neural network training coupled with naive Bayes classifiers. Bioinformatics 33: 2296-2306.