Supplementary Information

Table of Content

Supporting Figures

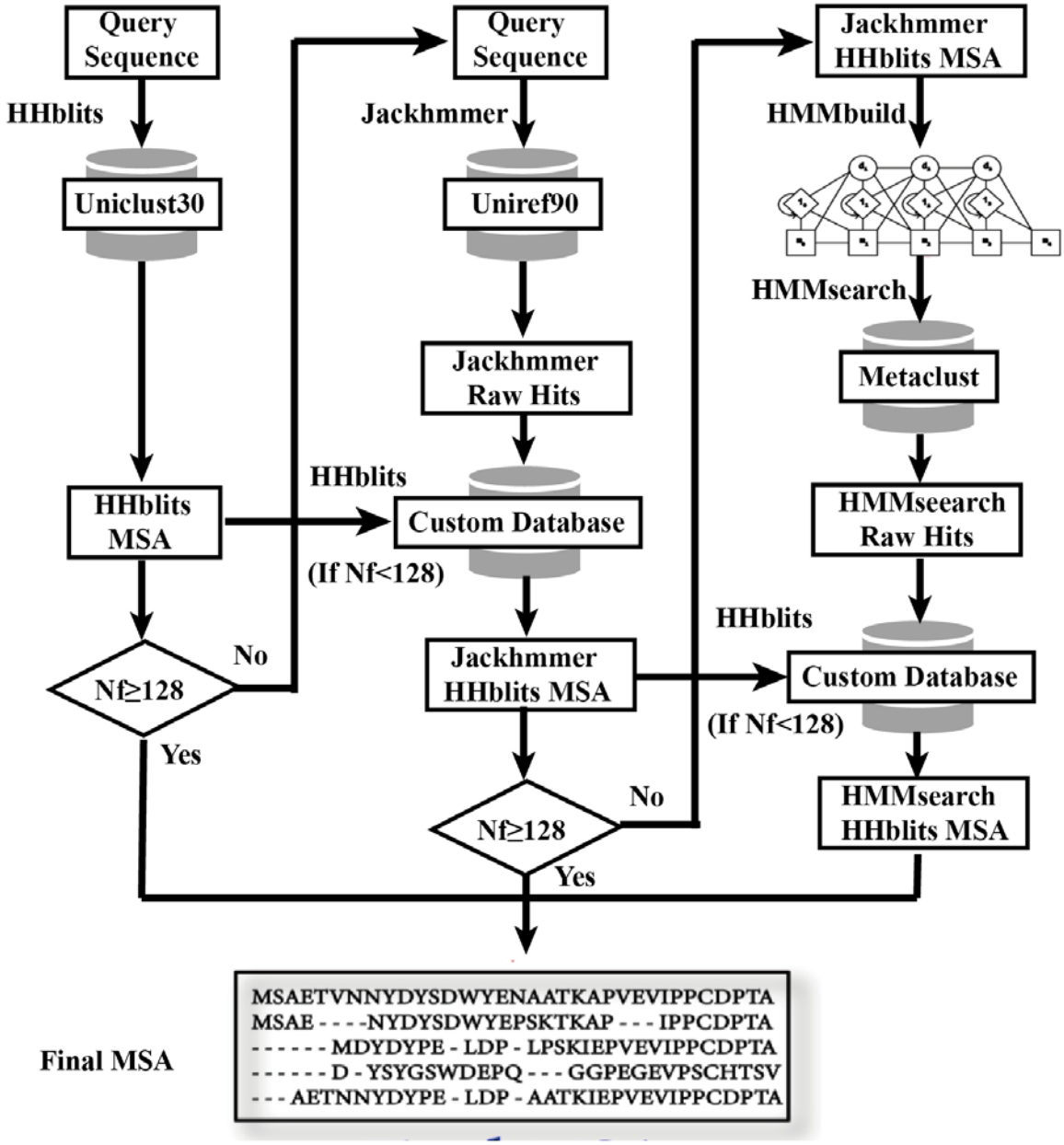

Figure S1. Flowchart of the deep multiple sequence alignment (MSA) construction method, DeepMSA, including three stages of MSA generation based on sequences from HHblits search against Uniclust30 (first column), Jackhmmer search through UniRef (second column) and HMMsearch through Metaclust (third column).

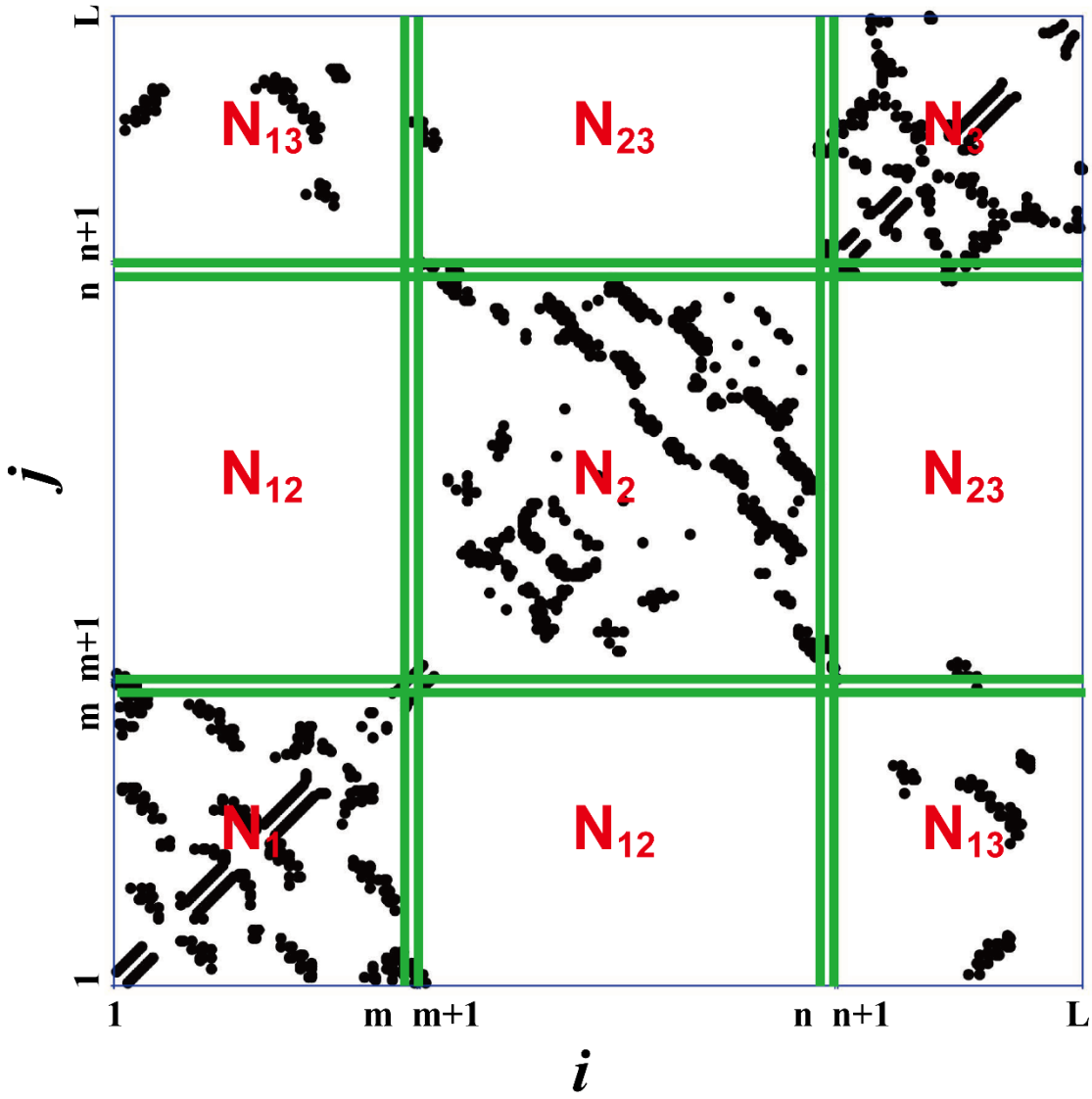

Figure S2. Illustration of the fast algorithm for calculating the FUscore for discontinuous two-domain proteins. There are two domain boundaries for discontinuous two-domain proteins: the first domain is $[(1, m), (n + 1, L)]$ and the second domain is $[m + 1, n]$, where L is the length of the protein.

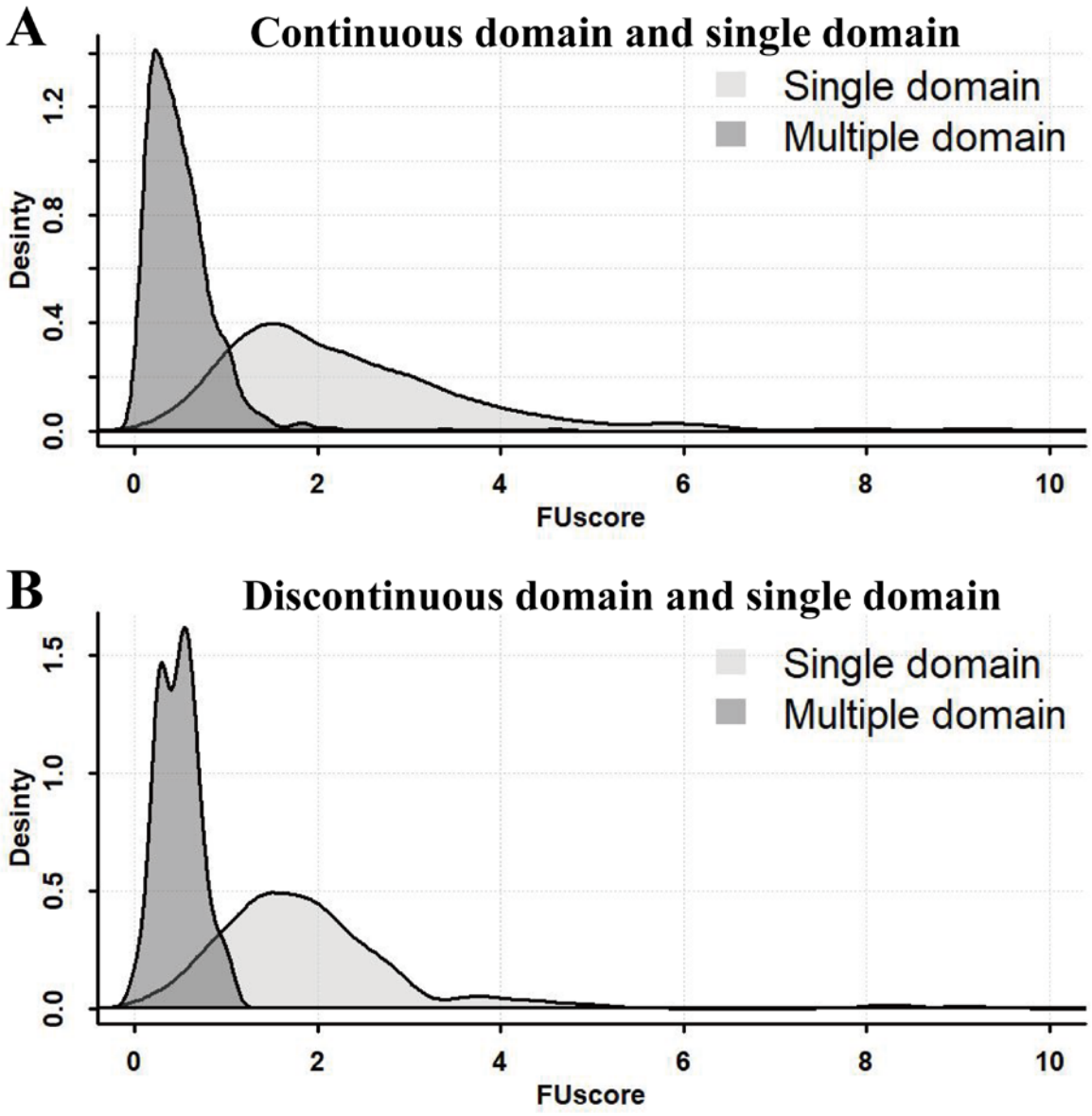

Figure S3. The differences between the distributions of FUscore for multi- and single-domain proteins in the training dataset. (A) The distributions of $FUscore_{2c}$ for continuous multi- and single-domain proteins in the training dataset. (B) The distributions of $FUscore_{2d}$ for discontinuous multi- and single-domain proteins in the training dataset.

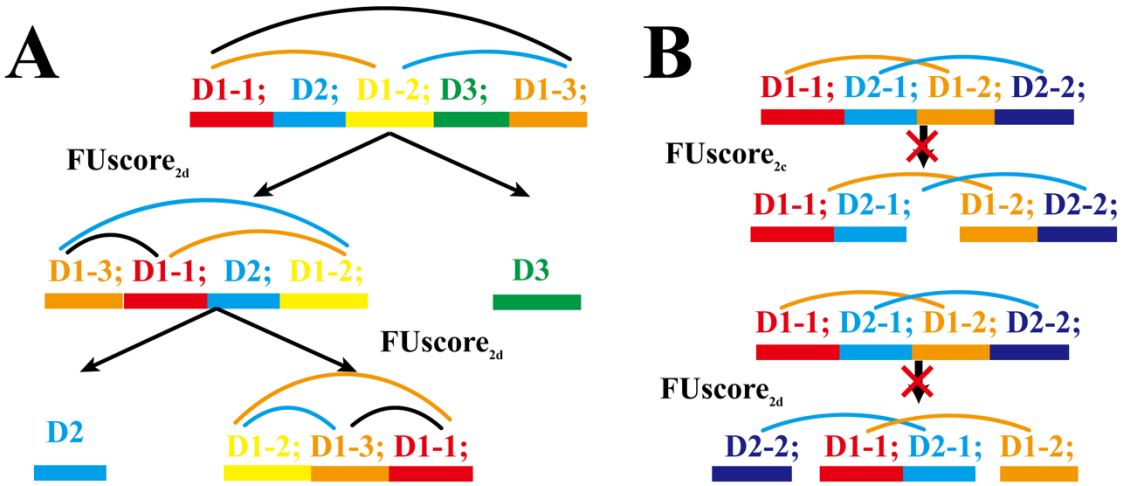

Figure S4. Case illustration for two special domain patterns, “D1-1, D2, D1-2, D3, D1-3” and “D1-1, D2-1, D1-2, D2-2”. (A) Iterative domain boundary detection of FUpred for pattern “D1-1, D2, D1-2, D3, D1-3”. (B) Iterative domain boundary detection of FUpred for pattern “D1-1, D2-1, D1-2, D2-2”.

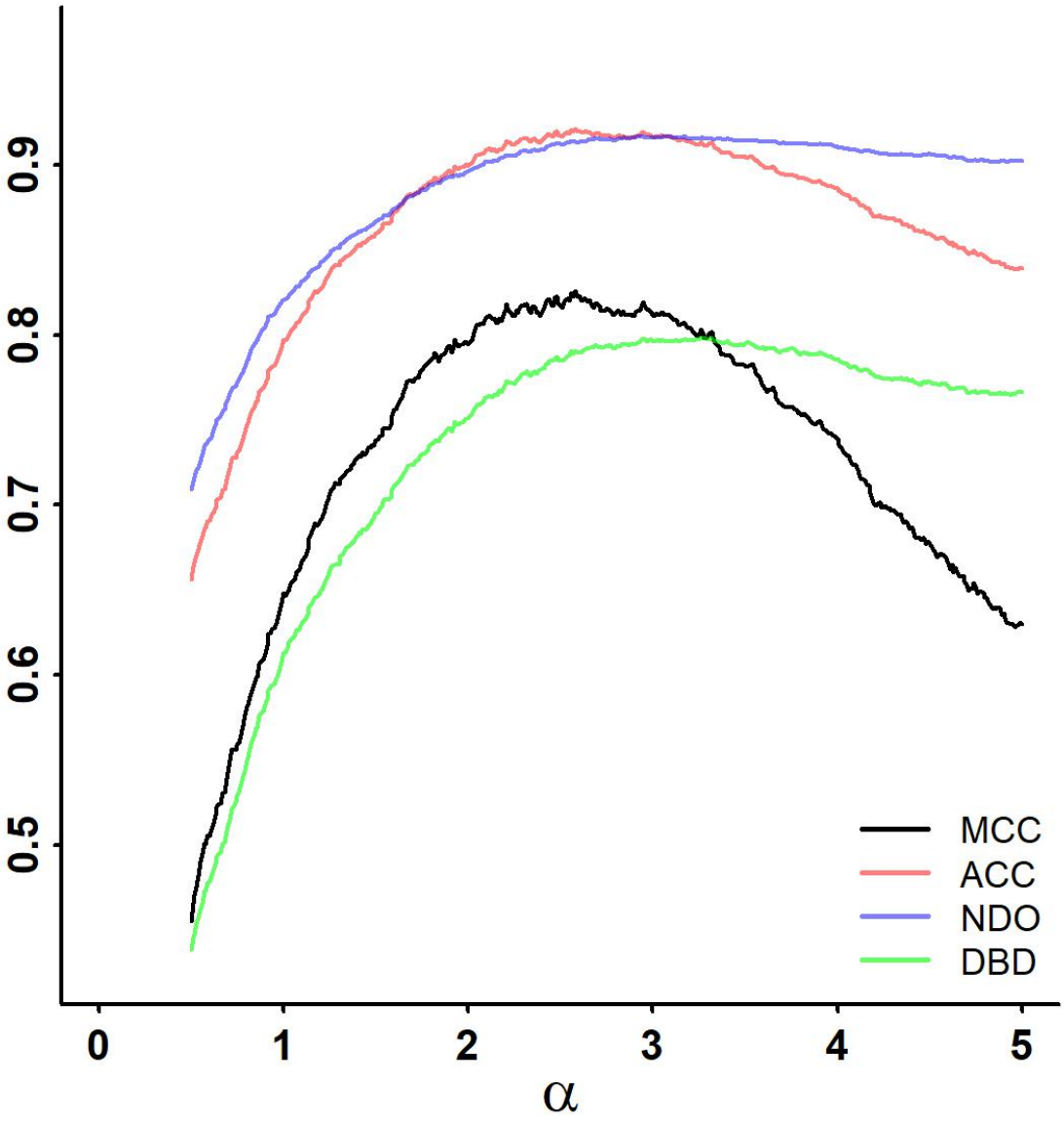

Figure S5. Optimization of the parameter α, which determines the top α*L* contact pairs used to form the final contact map for an input sequence, where *L* refers to length of the query protein.

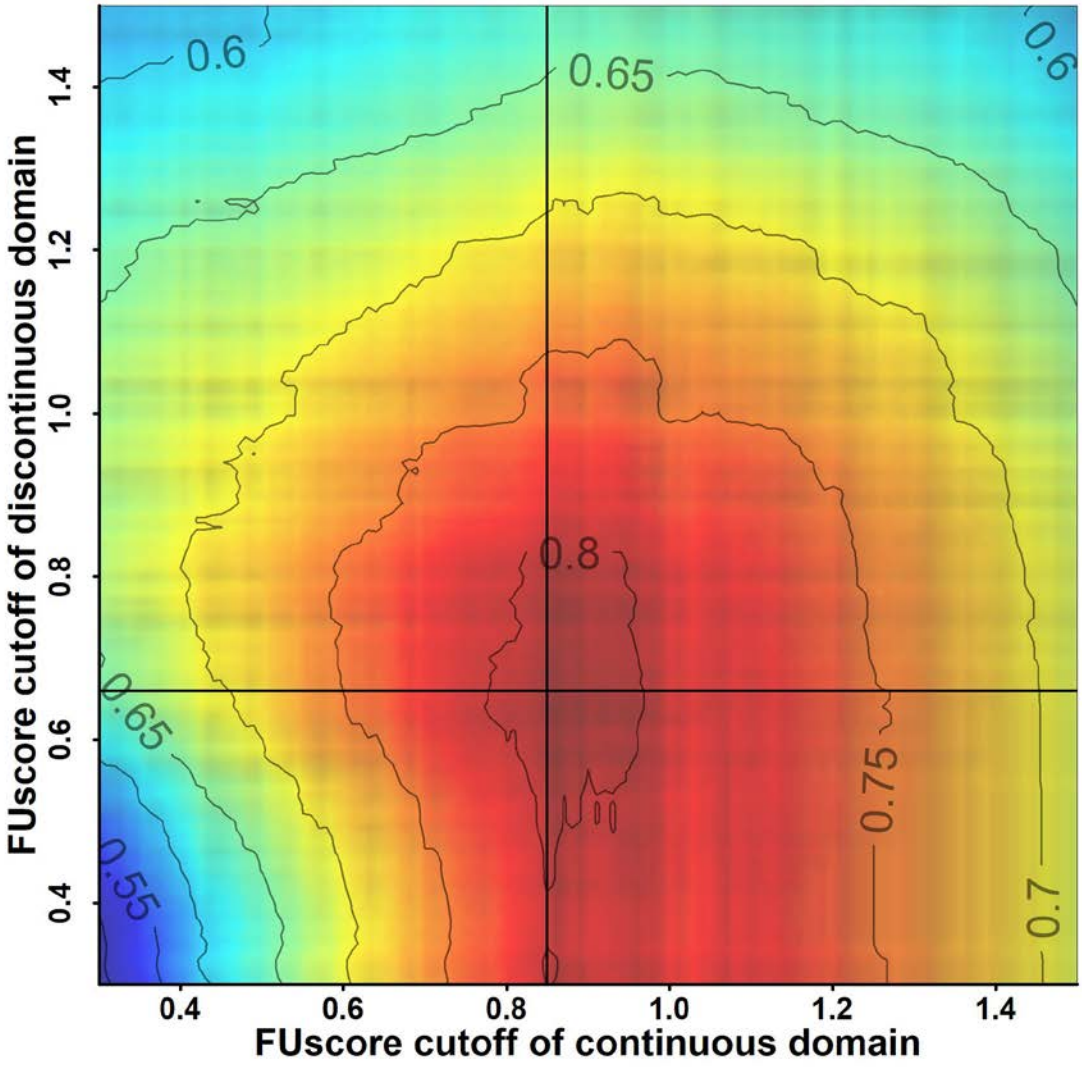

Figure S6. Optimization of parameters $Cutoff_{2c}$ and $Cutoff_{2d}$, which are used to distinguish between continuous multi- and single-domain proteins, as well as discontinuous multi- and single-domain proteins, respectively. The heat map value corresponds to the MCC.

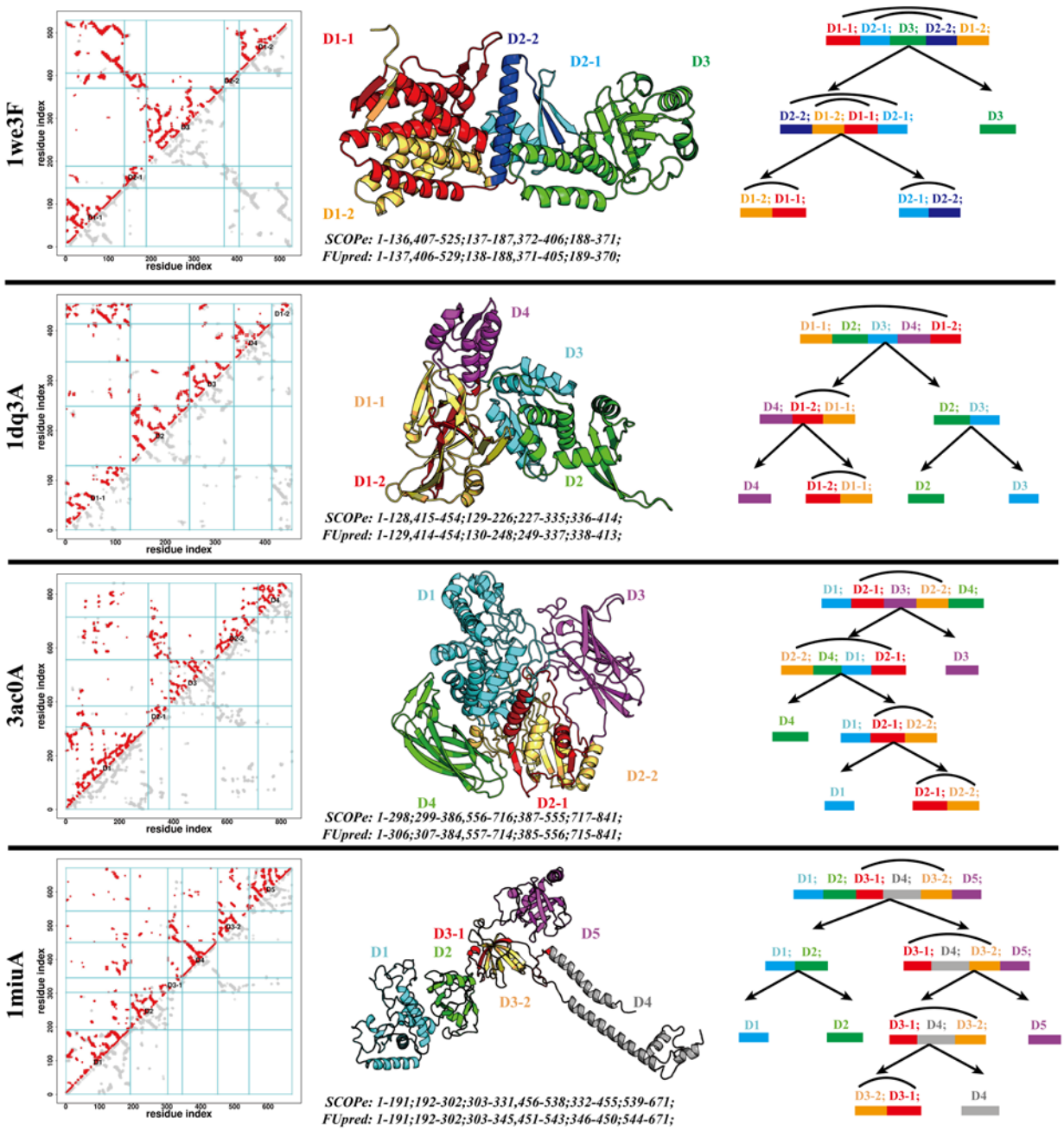

Figure S7. Case Study of domain boundary prediction for the (A) Cpn60 chaperonin (PDB ID: 1we3F), (B) archaeal intein-encoded homing endonuclease PI-PfuI (PDB ID: 1dq3A), (C) Beta-glucosidase from Kluyveromyces marxianus (PDB ID: 3ac0A), and (D) Breast Cancer type 2 susceptibility protein (PDB ID: 1miuA). The left panels show the native (grey) and ResPRE-predicted (red) contact maps for the target proteins, where cyan lines indicate the domain boundaries for the native structures. The middle panels give the experimental structures for each protein. The right panels show the iterative recursion procedure for domain boundary detection used by FUpred. Different domains are marked by distinct colors.

Supporting Tables

Table S1. Summary of domain boundary prediction for 38 three-domain proteins, where the total length of the two adjacent domains is less than the length of the third one. The values in parentheses are *p*-values between the results of FUpred and the other control methods calculated using pairwise two-sided Student's t-tests. FUpred^{s} (FUpred^{b}) forces the algorithm to search for the domain split point between the two adjacent small domains (or between the adjacent small domain and large domain) in the first iteration round, where the split point is located where the local minimum FUscore occurs around the SCOPe2.07 domain boundary definition ($\pm$20 residues).

Methods	NDO	DBD
FUpred	0.937	0.748
FUpred^{s}	0.937 (0.88)	0.756 (0.89)
FUpred^{b}	0.939 (0.94)	0.729 (0.41)

Table S2. Information for the discontinuous multi-domain patterns present in the SCOPe2.07 database.

Pattern	Number of patterns	Percent in all discontinuous patterns	Percent in SCOPe2.07 database	Can be solved by FUpred
ABA	2137	65.37%	0.7736%	Yes
ABAC	466	14.26%	0.1687%	Yes
ABCBA	201	6.15%	0.0728%	Yes
ABCB	162	4.96%	0.0586%	Yes
ABACDCA	70	2.14%	0.0253%	Yes
ABAB	61	1.87%	0.0221%	No
ABCDC	25	0.76%	0.0091%	Yes
ABCBDE	24	0.73%	0.0087%	Yes
ABCBD	21	0.64%	0.0076%	Yes
ABCA	18	0.55%	0.0065%	Yes
ABACD	17	0.52%	0.0062%	Yes
ABACDEF	12	0.37%	0.0043%	Yes
ABCDCAEFGHGE	9	0.28%	0.0033%	Yes
ABACAD	8	0.24%	0.0029%	Yes
ABCADE	8	0.24%	0.0029%	Yes
ABCDEFEG	6	0.18%	0.0022%	Yes
ABCDEB	5	0.15%	0.0018%	Yes
ABCBC	4	0.12%	0.0014%	No
ABCDCE	4	0.12%	0.0014%	Yes
ABACDE	3	0.09%	0.0011%	Yes
ABCDEBF	3	0.09%	0.0011%	Yes
ABCBDEF	1	0.03%	0.0004%	Yes
ABCDA	1	0.03%	0.0004%	Yes
ABCDEDFGH	1	0.03%	0.0004%	Yes
ABCDEFGHIFJKLKMN	1	0.03%	0.0004%	Yes
ABCDEFGHIFJKLKMNO	1	0.03%	0.0004%	Yes

Each identical character indicates one domain. The same character in a pattern means the separated fragments of discontinuous domains. For example, "ABA" means a protein with two domains where domain A is a discontinuous domain with two separated fragments.

Table S3. Single- and multi-domain classification results on 491 test proteins which are non-redundant to the training datasets of ResPRE, ThreaDomEx and ConDo. 'Pre', 'Rec', 'ACC' and 'MCC' are the precision, recall, accuracy and Matthew's correlation coefficient, respectively, as defined by **Eq. (5)**. Bold values indicate the best performer in each category.

Methods	Multi		Single		All	
	Pre	**Rec**	**Pre**	**Rec**	**ACC**	**MCC**
FUpred	**0.770**	0.882	0.952	**0.899**	**0.894**	**0.751**
ThreaDomEx	0.681	0.912	0.961	0.837	0.857	0.693
ConDo	0.704	0.699	0.885	0.887	0.835	0.587
DOMpro	0.564	0.647	0.857	0.808	0.764	0.438
DoBo	0.395	**0.971**	**0.975**	0.431	0.580	0.385

Table S4. Summary of domain boundary prediction results for the 136 multi-domain proteins which are non-redundant to the training datasets of ResPRE, ThreaDomEx and ConDo. The values in parentheses are *p*-values between the FUpred results and the other control methods results calculated using one-sided Student's t-tests. Bold values indicate the best performer in each category.

Methods	NDO	DBD
FUpred	**0.747**	**0.430**
ThreaDomEx	0.689 (2.26E-03)	0.332 (2.89E-03)
ConDo	0.687 (1.15E-03)	0.266 (7.76E-06)
DOMpro	0.594 (4.92E-11)	0.110 (4.08E-14)
DoBo	0.544 (5.92E-14)	0.169 (1.47E-11)

Table S5. Performance of different methods for both continuous and discontinuous multi-domain proteins. One-sided Student's t-tests were adopted here.

Target	Method	NDO	*p*-value	DBD	*p*-value
Continuous domain (716)	FUpred	0.791	*	0.494	*
	ThreaDomEX	0.776	2.33E-02	0.480	1.94E-01
	ConDo	0.765	4.81E-04	0.388	7.85E-09
	DOMpro	0.600	3.14E-54	0.094	1.64E-66
	DoBo	0.596	2.18E-52	0.211	1.78E-45
Discontinuous Domain (133)	FUpred	0.788	*	0.521	*
	ThreaDomEX	0.672	2.22E-08	0.421	2.29E-03
	ConDo	0.620	2.05E-11	0.312	9.38E-09
	DOMpro	0.500	2.31E-19	0.052	1.86E-19
	DoBo	0.417	2.00E-21	0.171	2.93E-15

Table S6. The best template quality comparison for ThreaDomEx for different data groups. One-sided Student's t-tests were adopted here.

Measures	NDO>=0.4	NDO<0.4	*p*-value	DBD>=0.25	DBD<0.25	*p*-value
TM-score	0.642	0.584	1.36E-02	0.652	0.604	2.18E-04

Table S7. Accuracy comparison for contacts that were used in FUpred for different data groups.

Measures	NDO>=0.4	NDO<0.4	*p*-value	DBD>=0.25	DBD<0.25	*p*-value
$\mathbf{PRE_{intra}^{+}}$	0.502	0.469	4.33E-02	0.514	0.466	2.46E-07
$\mathbf{PRE_{inter}^{+}}$	0.035	0.024	1.00E-03	0.038	0.036	8.08E-01
$\mathbf{PFP_{intra}^{-}}$	0.364	0.368	1.00E-00	0.358	0.381	1.97E-03
$\mathbf{PFP_{inter}^{-}}$	0.096	0.139	1.84E-02	0.092	0.115	5.34E-04

'+' indicates that one-sided Student's t-tests were used to test whether the PRE_{intra}/PRE_{inter} values of the NDO>=0.4 group were statistically greater than those values for the NDO<0.4 group, while '-' indicates that one-sided Student's t-tests were used to test whether the PFP_{intra}/PFP_{inter} values of the NDO>=0.4 group were statistically less than those values for the NDO<0.4 group.

Supporting Texts

Text S1. Deep multiple sequence alignment (MSA) construction

Starting from the input protein sequence, a deep MSA (Zhang, et al., 2019) is generated by iterative sequence homology searches against multiple sequence databases. This deep MSA construction process can be divided into three stages (**Fig. S1.**). In stage 1, HHblits (Remmert, et al., 2011) from HH-suite is used to search the query sequence against UniClust30 (Galiez, et al., 2016) to generate the first-level MSA. If stage 1 does not generate enough sequences, stage 2 will be performed, where Jackhmmer from the HMMER (Johnson, et al., 2010) package is used to search the query sequence against UniRef90 (the UniProt, et al., 2014) to extract full-length sequences (hits) and HHblits is used to convert the full-length sequences into a custom HHblits format database. Starting from the first-level MSA, HHblits is again applied to search this custom database to generate the second-level MSA. If the MSA from stage 2 still does not have enough sequences, stage 3 will be performed, where the second-level MSA is converted by hmmbuild from the HMMER package into a Hidden Markov Model (HMM) and the HMM is then searched against the Metaclust (Steinegger and Söding, 2018) metagenome sequence database by HMMsearch from the HMMER package to extract full-length hits. Similar to stage 2, hits from HMMsearch are built into a custom HHblits database. The second-level MSA is used to jump-start an HHblits search against this new custom HHblits database to get the third-level MSA. Based on these three stages, we generate a deep MSA to obtain a higher number of sequences.

Text S2. Fast Algorithm for calculating the FUscore for discontinuous two-domain proteins

As shown in **Fig. S2**, there are two domain boundaries for discontinuous two-domain proteins: the first domain is $[[1, m],\ [n+1, L]]$ and the second domain is $[m+1, n]$, where L is the length of the protein. Note that $1 \le m < n \le L-1$. In order to calculate the FUscore_{2d}, we need to calculate the number of contacts in each block, i.e., $N_1, N_2, N_3, N_{12}, N_{13}, N_{23}$ which is shown as follows:

$$N_1(m,n) = \sum_{i=1}^{m}\sum_{j=1}^{m} CM(i,j)$$
$$N_2(m,n) = \sum_{i=m+1}^{n}\sum_{j=m+1}^{n} CM(i,j)$$
$$N_3(m,n) = \sum_{i=n+1}^{L}\sum_{j=n+1}^{L} CM(i,j)$$
$$N_{12}(m,n) = \sum_{i=1}^{m}\sum_{j=m+1}^{n} CM(i,j)$$
$$N_{13}(m,n) = \sum_{i=1}^{m}\sum_{j=n+1}^{L} CM(i,j)$$
$$N_{23}(m,n) = \sum_{i=m+1}^{n}\sum_{j=n+1}^{L} CM(i,j)$$

However, the complexity of the above strategy for calculating each block is $O(L^2)$. By iterating over all pairs of splitting points m and n, the total running time will be $O(L^2 \times L \times L) = O(L^4)$, which is time-consuming. In order to improve the efficiency of our algorithm, we propose a dynamic programming algorithm to speed up the procedure by reducing the time complexity of calculating each block from $O(L^2)$ to $O(L)$. Thus, the total time complexity will be reduced to $O(L \times L \times L) = O(L^3)$. The dynamic programming strategy is shown as follows:

1. The recursion relationship for N_1

When the domain splitting point shifts from m to $m+1$ and the domain shifting point n does not change, i.e., the domains are $[[1, m+1], [n+1, L]]$ and $[m+2, n]$, the value of $N_1(m+1, n)$ can be calculated based on $N_1(m, n)$ as follows:

$$\begin{aligned} N_1(m+1,n) &= \sum_{i=1}^{m+1}\sum_{j=1}^{m+1} CM(i,j) \\ &= \sum_{i=1}^{m}\sum_{j=1}^{m} CM(i,j) + CM(m+1,m+1) + \sum_{i=1}^{m} CM(i,m+1) \\ &\quad + \sum_{j=1}^{m} CM(m+1,j) \\ &= N_1(m,n) + CM(m+1,m+1) + 2 * \sum_{i=1}^{m} CM(i,m+1) \end{aligned}$$

When the domain shifting point shifts from n to $n+1$ and the domain splitting point m does not change, i.e., the domains are $[[1,m],[n+2,L]]$ and $[m+1,n+1]$, N_1 is not affected by the change, i.e.,

$$N_1(m,n+1) = N_1(m,n)$$

$$N_1(m+1,n+1) = N_1(m+1,n)$$

2. The recursion relationship for N_3

Similarly, when the domain shifting point shifts from n to $n+1$ and the domain splitting point m does not change, i.e., the domains are $[[1,m],[n+2,L]]$ and $[m+1,n+1]$, the value of $N_3(m,n+1)$ can be calculated based on $N_3(m,n)$ as follows:

$$\begin{aligned} N_3(m,n+1) &= \sum_{i=n+2}^{L}\sum_{j=n+2}^{L} CM(i,j) \\ &= \sum_{i=n+1}^{L}\sum_{j=n+1}^{L} CM(i,j) + CM(n+1,n+1) - \sum_{i=n+1}^{L} CM(i,n+1) \\ &\quad - \sum_{j=n+1}^{L} CM(n+1,j) \\ &= N_3(m,n) + CM(n+1,n+1) - 2 * \sum_{i=n+1}^{L} CM(i,n+1) \end{aligned}$$

When the domain splitting point shifts from m to $m+1$ and the domain shifting point n does not change, i.e., the domains are $[[1,m+1],[n+1,L]]$ and $[m+2,n]$, N_3 is not affected by the change, i.e.,

$$N_3(m+1,n) = N_3(m,n)$$

$$N_3(m+1,n+1) = N_3(m,n+1)$$

3. The recursion relationship for N_2

When the domain splitting point shifts from m to $m+1$ and the domain shifting point n does not change, i.e., the domains are $[[1,m+1],[n+1,L]]$ and $[m+2,n]$, the value of $N_2(m+1,n)$ can be calculated based on $N_2(m,n)$ as follows:

$$\begin{aligned} N_2(m+1,n) &= \sum_{i=m+2}^{n}\sum_{j=m+2}^{n} CM(i,j) \\ &= \sum_{i=m+1}^{n}\sum_{j=m+1}^{n} CM(i,j) + CM(m+1,m+1) - \sum_{i=m+1}^{n} CM(i,m+1) \\ &\quad - \sum_{j=m+1}^{n} CM(m+1,j) \\ &= N_2(m,n) + CM(m+1,m+1) - 2 * \sum_{i=m+1}^{n} CM(i,m+1) \end{aligned}$$

When the domain shifting point shifts from n to $n+1$ and the domain splitting point m does not change, i.e., the domains are $[[1,m],[n+2,L]]$ and $[m+1,n+1]$, the value of $N_2(m,n+1)$ can be calculated based on $N_2(m,n)$ as follows:

$$\begin{aligned}N_2(m,n+1) &= \sum_{i=m+1}^{n+1}\sum_{j=m+1}^{n+1} CM(i,j)\\ &= \sum_{i=m+1}^{n}\sum_{j=m+1}^{n} CM(i,j) + CM(n+1,n+1) + \sum_{i=m+1}^{n} CM(i,n+1)\\ &+ \sum_{j=m+1}^{n} CM(n+1,j)\\ &= N_2(m,n) + CM(n+1,n+1) + 2 * \sum_{i=m+1}^{n} CM(i,n+1)\end{aligned}$$

When the domain splitting point shifts from m to $m+1$ and the domain shifting point n shifts from n to $n+1$, i.e., the domains are $[[1,m+1],[n+2,L]]$ and $[m+2,n+1]$, the value of $N_2(m+1,n+1)$ can be calculated based on $N_2(m,n)$ as follows:

$$\begin{aligned}N_1(m+1,n+1) &= \sum_{i=m+2}^{n+1}\sum_{j=m+2}^{n+1} CM(i,j)\\ &= \sum_{i=m+2}^{n}\sum_{j=m+2}^{n} CM(i,j) + CM(n+1,n+1) + \sum_{i=m+2}^{n} CM(i,n+1)\\ &+ \sum_{j=m+2}^{n} CM(n+1,j)\\ &= N_2(m+1,n) + CM(n+1,n+1) + 2 * \sum_{i=m+2}^{n} CM(i,n+1)\\ &= N_2(m,n) + CM(m+1,m+1) - 2 * \sum_{i=m+1}^{n} CM(i,m+1)\\ &+ CM(n+1,n+1) + 2 * \sum_{i=m+2}^{n} CM(i,n+1)\end{aligned}$$

4. The recursion relationship for N_{12}

Similar to the calculations of N_1, N_2, and N_3, the recursion relationship for N_{12} is shown as follows:

$$N_{12}(m+1,n) = \sum_{i=1}^{m+1}\sum_{j=m+2}^{n} CM(i,j) = \sum_{i=1}^{m+1}\left[\sum_{j=m+1}^{n} CM(i,j) - CM(i,m+1)\right]$$
$$= \sum_{i=1}^{m+1}\sum_{j=m+1}^{n} CM(i,j) - \sum_{i=1}^{m+1} CM(i,m+1)$$
$$= \sum_{i=1}^{m}\sum_{j=m+1}^{n} CM(i,j) + \sum_{j=m+1}^{n} CM(m+1,j) - \sum_{i=1}^{m+1} CM(i,m+1)$$
$$= N_{12}(m,n) + \sum_{j=m+1}^{n} CM(m+1,j) - \sum_{i=1}^{m+1} CM(i,m+1)$$
$$= N_{12}(m,n) + \sum_{i=m+1}^{n} CM(i,m+1) - \sum_{i=1}^{m+1} CM(i,m+1)$$

$$N_{12}(m,n+1) = \sum_{i=1}^{m}\sum_{j=m+1}^{n+1} CM(i,j) = \sum_{i=1}^{m}\left[\sum_{j=m+1}^{n} CM(i,j) + CM(i,n+1)\right]$$
$$= \sum_{i=1}^{m}\sum_{j=m+1}^{n} CM(i,j) + \sum_{i=1}^{m} CM(i,n+1)$$
$$= N_{12}(m,n) + \sum_{i=1}^{m} CM(i,n+1`)$$

$$N_{12}(m+1,n+1) = \sum_{i=1}^{m+1}\sum_{j=m+2}^{n+1} CM(i,j)$$
$$= \sum_{i=1}^{m+1}\left[\sum_{j=m+2}^{n} CM(i,j) + CM(i,n+1)\right]$$
$$= \sum_{i=1}^{m+1}\left[\sum_{j=m+1}^{n} CM(i,j) - C(i,m+1) + CM(i,n+1)\right]$$
$$= \sum_{i=1}^{m+1}\sum_{j=m+1}^{n} CM(i,j) - \sum_{i=1}^{m+1} CM(i,m+1) + \sum_{i=1}^{m+1} CM(i,n+1)$$
$$= \sum_{i=1}^{m}\sum_{j=m+1}^{n} CM(i,j) + \sum_{j=m+1}^{n} CM(m+1,j) - \sum_{i=1}^{m+1} CM(i,m+1)$$
$$+ \sum_{i=1}^{m+1} CM(i,n+1)$$
$$= N_{12}(m,n) + \sum_{i=m+1}^{n} CM(i,m+1) - \sum_{i=1}^{m+1} CM(i,m+1)$$
$$+ \sum_{i=1}^{m+1} CM(i,n+1)$$

5. The recursion relationship for N_{13}

Similar to the calculations of N_1, N_2, and N_3, the recursion relationship for N_{13} is shown as follows:

$$N_{13}(m+1,n) = \sum_{i=1}^{m+1}\sum_{j=n+1}^{L} CM(i,j) = \sum_{i=1}^{m}\sum_{j=n+1}^{L} CM(i,j) + \sum_{j=n+1}^{L} CM(m+1,j)$$
$$= N_{13}(m,n) + \sum_{i=n+1}^{L} CM(i,m+1)$$

$$N_{13}(m,n+1) = \sum_{i=1}^{m}\sum_{j=n+2}^{L} CM(i,j) = \sum_{i=1}^{m}\left[\sum_{j=n+1}^{L} CM(i,j) - CM(i,n+1)\right]$$
$$= \sum_{i=1}^{m}\sum_{j=n+1}^{L} CM(i,j) - \sum_{i=1}^{m} CM(i,n+1)$$
$$= N_{13}(m,n) - \sum_{i=1}^{m} CM(i,n+1)$$

$$N_{13}(m+1,n+1) = \sum_{i=1}^{m+1}\sum_{j=n+2}^{L} CM(i,j) = \sum_{i=1}^{m}\sum_{j=n+2}^{L} CM(i,j) + \sum_{j=n+2}^{L} CM(m+1,j)$$
$$= N_{13}(m,n+1) + \sum_{j=n+2}^{L} CM(m+1,j)$$
$$= N_{13}(m,n) - \sum_{i=1}^{m} CM(i,n+1) + \sum_{i=n+2}^{L} CM(i,m+1)$$

6. The recursion relationship for N_{23}

Similar to the calculations of N_1, N_2, and N_3, the recursion relationship for N_{23} is shown as follows:

$$N_{23}(m+1,n) = \sum_{i=m+2}^{n}\sum_{j=n+1}^{L} CM(i,j) = \sum_{i=m+1}^{n}\sum_{j=n+1}^{L} CM(i,j) - \sum_{j=n+1}^{L} CM(m+1,j)$$
$$= N_{23}(m,n) + \sum_{i=n+1}^{L} CM(i,m+1)$$

$$N_{23}(m,n+1) = \sum_{i=m+1}^{n+1}\sum_{j=n+2}^{L} CM(i,j) = \sum_{i=m+1}^{n+1}\left[\sum_{j=n+1}^{L} CM(i,j) - CM(i,n+1)\right]$$
$$= \sum_{i=m+1}^{n+1}\sum_{j=n+1}^{L} CM(i,j) - \sum_{i=m+1}^{n+1} CM(i,n+1)$$
$$= \sum_{i=m+1}^{n}\sum_{j=n+1}^{L} CM(i,j) + \sum_{j=n+1}^{L} CM(n+1,j) - \sum_{i=m+1}^{n+1} CM(i,n+1)$$
$$= N_{23}(m,n) + \sum_{i=n+1}^{L} CM(i,n+1) - \sum_{i=m+1}^{n+1} CM(i,n+1)$$

$$N_{23}(m+1,n+1) = \sum_{i=m+2}^{n+1}\sum_{j=n+2}^{L} CM(i,j)$$
$$= \sum_{i=m+1}^{n+1}\sum_{j=n+2}^{L} CM(i,j) - \sum_{j=n+2}^{L} CM(m+1,j)$$
$$= N_{23}(m,n+1) - \sum_{j=n+2}^{L} CM(m+1,j)$$
$$= N_{23}(m,n) + \sum_{i=n+1}^{L} CM(i,n+1) - \sum_{i=m+1}^{n+1} CM(i,n+1)$$
$$- \sum_{i=n+2}^{L} CM(i,m+1)$$

Each time we update the values, we remember the values of $N_1(m,n), N_2(m,n), N_3(m,n), N_{12}(m,n), N_{13}(m,n)$ and $N_{23}(m,n)$. Then when calculating the new terms $N_1(m+1,n)$, $N_2(m+1,n)$, $N_3(m+1,n)$, $N_{12}(m+1,n)$, $N_{13}(m+1,n)$, $N_{23}(m+1,n)$, $N_1(m,n+1)$, $N_2(m,n+1)$, $N_3(m,n+1)$, $N_{12}(m,n+1)$, $N_{13}(m,n+1)$, $N_{23}(m,n+1)$, $N_1(m+1,n+1)$, $N_2(m+1,n+1)$, $N_3(m+1,n+1)$, $N_{12}(m+1,n+1)$, $N_{13}(m+1,n+1)$, and $N_{23}(m+1,n+1)$, we only need to calculate the increment. Therefore, the complexity of the dynamic programming strategy for calculating each block is only $O(L)$, resulting in a running time for calculating the FUscore_{2d} (as shown below) that is also only $O(L)$.

$$FUscore_{2d}(m,n)$$
$$= 2\big(N_{12}(m,n) + N_{23}(m,n)\big)$$
$$* \left[\frac{1.0}{N_3(m,n) + N_1(m,n) + 2N_{13}(m,n)} + \frac{1.0}{N_2(m,n)}\right]$$

By iterating over all pairs of splitting points m and n, the total running time is $O(L \times L \times L) = O(L^3)$.

References

Galiez, C., *et al.* (2016) Uniclust databases of clustered and deeply annotated protein sequences and alignments, *Nucleic Acids Research*, **45**, D170-D176.
Johnson, L.S., Eddy, S.R. and Portugaly, E. (2010) Hidden Markov model speed heuristic and iterative HMM search procedure, *BMC Bioinformatics*, **11**, 431.
Remmert, M., *et al.* (2011) HHblits: lightning-fast iterative protein sequence searching by HMM-HMM alignment, *Nature Methods*, **9**, 173.
Steinegger, M. and Söding, J. (2018) Clustering huge protein sequence sets in linear time, *Nature Communications*, **9**, 2542.
the UniProt, C., *et al.* (2014) UniRef clusters: a comprehensive and scalable alternative for improving sequence similarity searches, *Bioinformatics*, **31**, 926-932.
Zhang, C., *et al.* (2019) DeepMSA: constructing deep multiple sequence alignment to improve contact prediction and fold-recognition for distant-homology proteins, *Bioinformatics*.