Protein domain embeddings for fast and accurate similarity search

Benjamin Giovanni Iovino[1], Haixu Tang[1], and Yuzhen Ye[1, *]

[1]Luddy School of Informatics, Computing and Engineering, Indiana University, 700 N. Woodlawn Avenue, Bloomington, IN 47408.
[*]Corresponding author: yye@indiana.edu

Abstract

Recently developed protein language models have enabled a variety of applications of the protein contextual embeddings. Per-protein representations (each protein is represented as a vector of fixed dimension) can be derived via averaging the embeddings of individual residues, or applying matrix transformation techniques such as the discrete cosine transformation to matrices of residue embeddings. Such protein-level embeddings have been applied to enable fast searches of similar proteins, however limitations have been found; for example, PROST is good at detecting global homologs but not local homologs, and knnProtT5 excels for proteins of single domains but not multi-domain proteins. Here we propose a novel approach that first segments proteins into domains (or subdomains) and then applies discrete cosine transformation to the vectorized embeddings of residues in each domain to infer domain-level contextual vectors. Our approach called DCTdomain utilizes predicted contact maps from ESM-2 for domain segmentation, which is formulated as a *domain segmentation* problem and can be solved using a *recursive cut* algorithm (RecCut in short) in quadratic time to the protein length; for comparison, an existing approach for domain segmentation uses a cubic-time algorithm. We showed such domain-level contextual vectors (referred as *DCT fingerprints*) enable fast and accurate detection of similarity between proteins that share global similarities but with undefined extended regions between shared domains, and those that only share local similarities.
Keywords: protein language model (PLM), ESM-2, domain segmentation, recursive cut, discrete cosine transformation (DCT), DCT fingerprint, homology detection

Introduction

Homology detection is one of the most performed computations in the field of biology. Every homology detection task begins with a protein sequence of interest that is queried against a collection of sequences with the goal of returning the most similar sequence as the top result. Despite the simplicity of this task in its conception, it can be incredibly difficult in practice due to the lack of similarity between two proteins that are considered homologous. Sequences with less than 10-12% similarity (in terms of character identity) have been found to contain similar structures [1], and thus it is crucial to detect proteins, or "remote homologs", within this realm. Many methods have been developed to correctly and efficiently perform this task, ranging from simple (albeit heuristic heavy) sequence-sequence comparisons such as BLAST [2], to profile methods that consider groups of proteins like PSI-Blast [3] and CS-BLAST [4], to profile hidden Markov models (pHMMs) based on probabilistic models like HMMER [5] and HHsearch [6] which are considered state-of-the-art for homology detection.

Recent methods have been developed using contextualized embeddings generated by neural networks, such as neural protein language models (pLMs), for homology detection. pLMs are trained for the purpose of learning about the nature of proteins beyond their sequence representation [7]. Because the sequence of a protein is constrained by the structure it folds into, each residue in a protein has more meaning than its character identity as each residue plays a role in the protein's overall function. This contextual information is learned by pLMs when they are trained on large sequence databases, like UniProt [8] and BFD [9] [10]. Many different pLMs have been trained and they can take on various architectures. ProtTrans [7], which contains a host of different models, was trained with the goal of producing informative embeddings as input

for downstream tasks, such as predicting secondary structure and sub-cellular localization. ESM-2 [11] was trained with the purpose of producing embeddings that ESMFold could use to predict the 3D structure of a sequence. Embeddings from both models have been successfully applied to many tasks, including homology detection, such as knnProtT5 [12] and PROST [13].

One tradeoff with using pLM embeddings to represent protein sequences is the increase in dimensionality compared to the original character representation. For example, by embedding each position in a 500 residue-long protein sequence, each position will be replaced by a vector of N length, where N can be thousands of numbers. Even when using simple distance metrics to calculate the difference between the residual embeddings, it will be computationally demanding to compare a query protein represented as such against every similarly represented protein in a database, because proteins are of various lengths and residual embeddings need to be aligned. Protein-level embeddings can offset this increase. A typical approach of deriving protein-level embedding for a protein is to use the mean of the embeddings of all its residues. The mean embeddings have been used in applications, such as in knnProtT5 for remote homology detection using nearest neighbor search on protein-level embedding spaces [12]. The advantage of using protein-level embeddings is that proteins are now represented as a single vector of a fixed length, so similar proteins can be found by searching for proteins sharing similar embeddings, which can be computed quickly (L1-distance or other metrics can be used). When combined with other methods (MMSeq2 [14] or Smith-Waterman [15]), knnProtT5 achieved comparable performance with sequence-based approaches for homology detection, but it was not competitive for comparison of multi-domain proteins [12].

Another technique that has been applied with success is the DCT vector quantization of embeddings. Introduced by WARP [16] and used in PROST [13], this method uses the Discrete Cosine Transform (DCT) [17], a commonly used technique in image and video compression, to reduce protein embeddings to smaller dimensions while maintaining key information. This method has been shown to be effective for global homolog detection [16] [13] where the entirety of two proteins correspond to one another. However, it performed worse for the cases where two proteins share all of their domains, but with extended undefined regions between the domains [13]. For local homology detection, where only certain regions of two proteins are similar, this method won't work well due to the course-grained nature of the representation (as we show in our Results). To remedy these issues we developed a new method (ESM2-RecursiveCut) for predicting domains using the predicted contact maps from ESM-2, and predicted domains are then used for deriving domain-level fingerprints based on DCT.

Protein domains are subunits that can fold and function independently [18]. The protein universe contains many multi-domain proteins, involving a great diversity of domain architectures [19]. A few methods have been developed for protein domain segmentation given protein sequences, including a more recently developed method FUpred (for Folding Unit predictor) [20], which detects domain boundaries from protein sequences based on contact map prediction. FUpred aims to find domains that maximizes the number of intra-domain contacts, while minimizing the number of inter-domain contacts. For contact map prediction, FUpred generates a multiple sequence alignment (MSA) using the DeepMSA program [21], and the generated MSA is used as the input for contact map prediction using ResPRE [22], a method that couples evolutionary precision matrices with deep residual neural networks. FUpred was shown to outperform existing approaches for contact map prediction, including ConDo [23] and DNN-dom [24]. We found that ResPRE-FUpred pipeline is computationally demanding for creating domain-level DCT fingerprints, and therefore we proposed a new method ESM2-RecCut for this purpose. Our ESM2-RecCut uses contact map prediction from ESM-2 (together with contextual embeddings of individual residues), taking advantage of its capability of generating contextual embeddings without using MSA so no time-consuming iterative searches of similar sequences. In addition, we developed a quadratic time RecCut-based algorithm for domain segmentation given contact map predictions. Predicted domains can then be used for generating domain-level fingerprints to facilitate fast and accurate similarity detection.

Methods

Overview of the methods

Our approach DCTdomain uses the protein embedding and contact map prediction from ESM-2, detects domains, and represents each protein as one or more DCT fingerprints, including a DCT fingerprint for

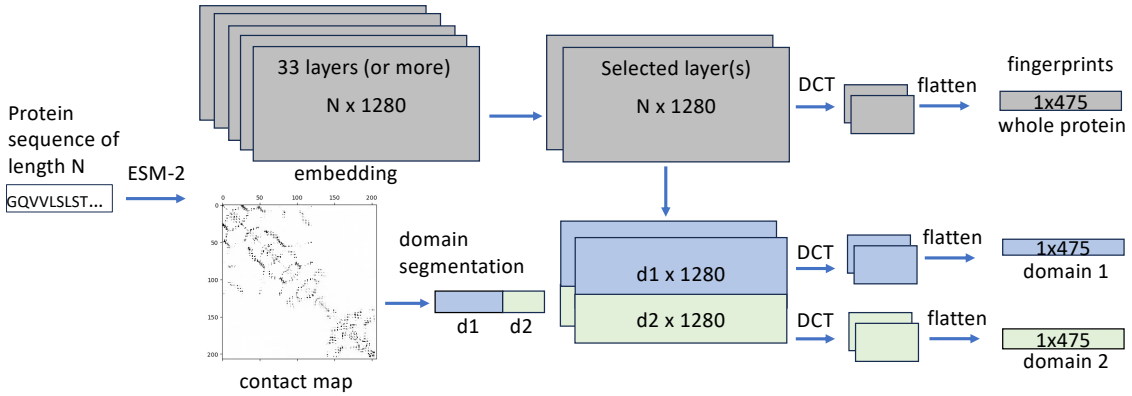


Figure 1: A diagram showing the inference of domain-based embeddings (DCT fingerprints). For the toy example protein which has two domains, a total of three DCT fingerprints will be derived, one representing the whole protein, and the other two are the representations of the domains. This diagram uses two-domain protein for demonstration purpose without loss of generality.

the whole protein, and if applies, a fingerprint for each of the predicted domains in the protein. The DCT fingerprints are then used for computing the similarity between the proteins. Our method reports two similarity scores: one based on the DCT fingerprints of the whole proteins (denoted as DCTglobal), and the other one based on all DCT fingerprints of whole proteins and of individual domains (denoted as DCTdomain). Figure 1 shows the overview of our approach.

Protein language model embedding

We used the output from two layers of the ESM2-t33-650M model (layer 14 and layer 26). It has been shown that each layer of an encoder is able to capture different contextual information about a protein sequence [13]. To find the most useful layer for our particular task, we tested the prediction performance of each layer on one of our data sets. We found that embeddings from layers 14 through 26 (after being quantized to 5x44 matrices with DCT) shared roughly the same performance as each other, meaning they are all effective for our task of homology detection. See details in Results.

Protein domain segmentation based on ESM-2 contact map prediction

We used both FUpred [20] and a new recursive cut algorithm (RecCut) we proposed here to predict domains given predicted contact maps and compared their performance. To distinguish the two versions, we refer them as ESM2-FUpred and ESM2-RecCut. Comparing to the original FUpred pipeline (which uses ResPRE to predict contact maps, so the whole pipeline is referred as ResPRE-FUpred in this paper), here we used ESM-2's contact map prediction as input to FUpred or RecCut for domain segmentations.

We noticed that FUpred/RecCut using ESM-2 contact map prediction results in high precision of the prediction of single domain proteins and high recall for prediction of multi-domain proteins, but tends to split domains into smaller units (see Results). So in some cases, proteins may be segmented into domains or *sub-domains*. For simplicity, we don't distinguish domains and subdomains, and call them domains throughout this paper.

Recursive cut algorithm for domain segmentation based on contact map

We present the computational problem to partition a given single-chain protein into multiple contiguous or non-contiguous domains, given its contact map (predicted) as the input. Notably, in some cases, the N-terminal and C-terminal of a protein chain are proximal in 3D space, and thus the residues in both terminus can form a single protein domain (e.g., protein 1agr chain A has two domains, one domain contains residues

57 to 177, and the other domain contains two discontinuous regions, 1-56 and 178-350). To account for such cases, herein, we represent the protein as a circular string.

Formally, given a circular string $S[1, \ldots, n]$ of length n, we define a *segmentation* of the string as a sequence of k indices $(c_1, c_2, \ldots, c_k)$, where $1 \leq c_i \leq n$ represents the indices dividing the string into a set of k segments $S[c_i + 1, \ldots, c_{i+1}]$ for $i = 1, 2, \ldots, k$, and $c_{k+1} = c_1 + n$ to handle the circular nature of the string. We further define a *domain segmentation* of the string as an annotation of each segment by one of its d domains, i.e., $l[i] \in \{1, 2, ..., d\}$ so that all residues in a segment are all assigned to the same domain. So, given a weight (contact) matrix $W[i, j]$ representing the probability of contact between a pair of residues i and j (from ESM-2), our goal is to find a maximum domain segmentation (i.e., with the maximum number of domains) in which the number of contacts any two domains is smaller than a threshold. We formulate this problem as the following *domain segmentation problem.*

Problem 1 (Domain Segmentation Problem)**.** ***Input:*** *A circular string $S[1, \ldots, n]$, a weight matrix $W[i, j]$ for $\forall 1 \leq i, j \leq n$, and a threshold W_m.* ***Output:*** *The maximum domain segmentation of S with d domains, $(c_1, c_2, \ldots, c_k)$ $(c_{k+1} = c_1 + n)$ and $l[i] \in \{1, 2, ..., d\}$, for $0 \leq i \leq k$, in which the contact between residues in any two domains is below v, i.e., for any $i \neq j$, $\sum_{c_{i-1} < u \leq c_i} \sum_{c_{j-1} < v \leq c_j} W[u, v] \leq W_m$.*

FUpred [20] was previously developed to tackle the computational problem: in each step, an input circular string is partitioned into two contiguous domains (substrings) with the minimum contact, and the procedure continues recursively (i.e., each of the two partitioned substrings forms a circular string that is subject to further partition) until the minimum contact between the two partitioned domains becomes less than the threshold. It was shown that this heuristic approach can reconstruct the known protein domains in most proteins in the curated protein structure classification database SCOP [25], with the exception of rare cases of nested discontinuous multi-domain proteins [20].

Despite the effectiveness of FUpred, it takes $O(dn^3)$ running time (where d is the number of domains in the protein), which is slow when executed on a large set of proteins, especially those relatively long proteins (e.g., for $n > 1000$). Here, we present the improved algorithm called RecCut (for recursive cut), which follows the heuristic approach to give the same desirable output as FUpred, but running in only $O(dn^2)$ time. Our results showed that RecCut significantly reduces the running time of FUpred, especially for long proteins (see Results).

The RecCut algorithm is illustrated in Algorithm 1 and 2. Given an input string $S[1, \ldots, n]$, RecCut considers the 1-cut and the 2-cuts partitions of the protein, where the 1-cut partition at position i results in two contiguous, one-segment domains, $C[1, \ldots, i]$ and $C[i+1, \ldots, n]$, respectively, while the 2-cuts partition at positions j and k results in one contiguous, one-segment domain, $C[i+1, \ldots, j]$, and one non-contiguous domain consisting of two segments, $C[j+1, \ldots, n]$ and $C[1, \ldots, i]$, respectively. Following FUpred [20], RecCut aims to first find the 1-cut and 2-cuts partitions of the protein with the minimum contact between two partitioned domains, respectively, and then selects one of them based on two given thresholds (denoted as W_{1-cut} and W_{2-cuts}, instead of a single threshold W_m) to proceed the recursion. Specifically, assuming the between-domain contacts of the 1-cut and 2-cuts partitions are V_{1-cut} and V_{2-cuts}, respectively, if $V_{1-cut} > W_{1-cut}$ and $V_{2-cuts} > W_{2-cuts}$, the recursion is terminated and the entire segment is output as a single domain; otherwise, RecCut selects the 1-cut partition (if $V_{1-cut} - W_{1-cut} \leq V_{2-cuts} - W_{2-cuts}$) or the 2-cuts partition (if $V_{1-cut} - W_{1-cut} > V_{2-cuts} - W_{2-cuts}$) for recursion.

A naive algorithm to find the 1-cut protein partition with the minimum between-domain contact takes $O(n^3)$ time (i.e., for each putative cut site, one computes the contact between the resulting two domains). FUpred exploited a dynamic programming algorithm to reduce the running time to $O(n^2)$ by updating the vector V, in which $V[i]$ represents the contact when the two segments $C[1, \ldots, i]$ and $C[i+1, \ldots, n]$ [20]. Apparently, V can be computed iteratively in $O(n^2)$ time: $V[0] = \sum_{1 \leq j < k \leq n} W[j, k]$, and $V[i] = V[i-1] - \sum_{1 \leq j < i} W[j, i] + \sum_{i < j \leq n} W[i, j]$, for $i \geq 1$. We followed the same algorithm in RecCut (see the OneCutPartition function in Algorithm 1 in Appendix). On the other hand, a naive algorithm to find the 2-cuts protein partition with the minimum between-domain contact takes $O(n^4)$ time (i.e., for every two putative cut sites, one computes the contact between the resulting two domains), and FUpred exploited a dynamic programming algorithm to reduce it to $O(n^3)$. In RecCut, we further optimized the dynamic programming algorithm to reduce the running time to $O(n^2)$ by pre-computing three matrices and two vectors. The three matrices are $E[i, j]$ representing the sum of the contacts between the residue i and all residues within the segment $S[1, \ldots j]$, $F[i, j]$ representing the sum of the contacts between the residue i and

all residues within the segment $S[i+1, \ldots j]$ and $T[i, j]$ representing the sum of the contacts between the residues within the segment $S[i, \ldots j]$. The two vectors are $N_1[i]$ representing the sum of the contacts between all pairs of residues within the segment $S[0, \ldots j]$ and $N_2[i] = N - N_1[i]$, where $N = N_1[n]$ represents the sum of the contacts between all pairs of residues within the entire input protein (see the TwoCutsPartition function in Algorithm 2 in Appendix). After the optimization, RecCut reduces the running time from $O(dn^3)$ by FUpred to $O(dn^2)$ to compute the domain segmentation, where d is the number of domains (typically a small number) and n is the protein length.

Compression of protein embedding matrices using DCT

The DCT vector quantization method, inspired by DCT-based compression methods, was developed to homogenize the length of vectorized protein sequences while also retaining its sequential pattern and as much information from the original data as possible [16]. It is also a relatively quick computation to perform as the DCT and its inverse are both fast Fourier transforms. This method can reduce both dimensions of an input embedding matrix: 2d-DCT is first applied to the input embedding matrix to compute frequency coefficients, and then an inverse DCT (iDCT) is applied to only low frequency coefficients (discarding the high frequency ones) to produce a dimension reduced matrix, thus achieving compression of the embedding matrix. For example, given an embedded protein sequence from ESM2-t33-650M that has dimensions 250 x 1280 (250 residues with each residue embedded as a vector of 1280 dimensions), we can reduce both dimensions (e.g., to 5 and 44) that give a compact yet representative representation of proteins for both efficient and accurate homology detection. The compressed matrix is then flattened to produce 1D vector (referred as a *DCT fingerprint*), which represents a protein allowing for the usage of simple vector operations.

For proteins with multi-domains, DCT transformation is applied to each domain to generate a DCT fingerprint for the domain. By doing this, each protein is presented as a DCT fingerprint for single-domain proteins, or $(d+1)$ DCT fingerprints for proteins with d domains. The scipy.fftpack library is used for DCT transformation, and after DCT transformation, each number in the vector is multiplied with 127 to change the scale.

Protein similarity measure based on DCT fingerprints

Given two DCT fingerprints DCT_i and DCT_j (each of 475 dimensions), their similarity score is defined as $S(DCT_i, DCT_j) = 1 - L1(DCT_i, DCT_j)/c$, where $L1$ is the L1-distance between two vectors and c is a normalization constant such that the similarity score is transformed to the range of [0, 1] (in our case, $c = 17000$). Accordingly, given two proteins i and j each represented as one (for single-domain proteins) or multiple DCT fingerprints (for multi-domain proteins), their global similarity is computed as the similarity of the DCT fingerprints of the whole proteins, and their local similarity is computed as the maximum similarity of any pairs of DCTs (including those for the whole protein and those for individual domains) from the two proteins. We note that L1-distance was used in PROST to quantify the distance between DCT matrices. Our similarity score is based on L1-distance, but it is transformed and normalized to be in the range of [0, 1] (0 for no similarity).

Benchmarks

We used the FUpred benchmarks [20] for domain segmentation test using ESM-2 contact maps. Similarly, we used the train set (with 2549 proteins) to tune the parameters (W_{1-cut} and W_{2-cuts}, corresponding to Cutoff2c and Cutoff2d in FUpred [20], for distinguishing between continuous multi- and single-domain proteins, as well as discontinuous multi- and single-domain proteins, respectively), and reported the performance on the test collection with the same number of proteins. The train and test collections don't share proteins. The benchmark was downloaded from https://zhanggroup.org/FUpred/.

For homolog detection, we first used a curated benchmark of protein pairs [26] which labels proteins as homologous if their domains, in consecutive order, are in the same family or clan, and non-homologous if no domain in the first protein is a part of the same family (e.g., PF03946) or clan (e.g., CL0188) as any domain in the second. All proteins were derived from a given specific set of genomes (16 species including three prokaryotes and 13 eukaryotes). Like in PROST [13], we used two groups of this benchmark:

max50, where the maximum distance between two domains is 50 residues, and nomax50, where there is no limit between domains. See Table 1 for a summary of the max50 and nomax50 benchmarks and example cases of homologs in each group. We re-tested each homolog detection method (including HHsearch [6], UBLAST [27], FASTA [28], phmmer [5], BLAST [2], and CS-BLAST [4]) on each dataset and found nearly identical Area Under Curve (AUC) values as originally reported [26]. The benchmark was downloaded from http://sonnhammer.org/download/Homology_benchmark.

Based on the nomax50 benchmark, we further created a benchmark for testing detection of local similarities between proteins (refered as pfam-local). This benchmark contains homologs that share at least some of their domains, but not all. An example is shown in Table 1: the two proteins each have two domains and they share one domain (PF00319).

For performance evaluation of domain segmentation on the FUpred benchmark, we focused on the accuracy of classifying proteins into single-domain versus multi-domain proteins, and used various metrics for evaluation, including accuracy and Matthew's correlation coefficient. For evaluation of homology detection, we used accuracy, as well as the ROC plots to show the true positive rates vs false positive rates for the different methods. Each of the benchmarks contains similar number of homologs (positives) and non-homologs (negatives).

Table 1: Protein benchmarks for homology/similarity detection.

Benchmark *	# pairs** (homologs)	homolog definition	example homologs (domain architecture)
pfam-max50	10450 (5228)	identical domain architecture; < 50 aa between domains	Q9VFJ2 [PF03946,PF00298] P53875 [PF03946,PF00298]
pfam-nomax50	71988 (36278)	identical domain architecture; no constraint on the aa between domains	Q15149 [PF03501,CL0188,CL0188,PF00681] Q9QXS1 [PF03501,CL0188,CL0188,PF00681]
pfam-local	15273 (7602)	share some domains, but not all	P40791 [PF00319,**PF12347**] Q8VWM8 [PF00319,**PF01486**]

* The benchmarks are denoted as pfam-max50 and so on to indicate that pfam domains found in proteins are used for defining homologs. The benchmarks include full length proteins. ** We didn't include the proteins that are longer than 6000 aa in our benchmarks due to the substantial memory requirement for embedding such long proteins by ESM-2.

Availability of data and code

Our programs (RecCut for domain domain segmentation in C++, and Python scripts for sequence embedding, DCT fingerprint generation, and sequence comparison and similarity search based on DCT fingerprints) are available as open source on github at https://github.com/mgtools/DCTdomain. Benchmarks and the results are also available in the same github repo.

Results

Domain predictions using predicted contact maps from ESM-2

Comparing to FUpred predicted domains using predicted contact maps from ResPRE (ResPRE-FUpred), our approaches that use contact maps from ESM-2 (ESM2-FUpred, and ESM2-RecCut) gave very good recall of the predictions for multi-domain protein predictions and precision for single-domain protein predictions, as shown in Table 2, however, their overall accuracy was lower. The results also suggested that approaches based on ESM-2 contact maps tend to split domains into smaller units (some single-domain proteins were predicted to be multi-domain proteins and more domains are predicted for multiple-domain proteins). Proteins in the test dataset of 2549 proteins each contains 2.47 domains on average, and ESM2-FUpred and ESM2-RecCut predicted 2.96 and 2.80 domains per protein on average, respectively. Figure 2 shows two examples of ESM2-RecCut predictions. In the first case, 1a04a, ESM2-RecCut gave almost identical domain predictions with 3D structure based domain segmentation in SCOP (with the boundary of the domains shifted by one residue). Visualization of the contact map from ESM-2 shows that this protein has two domains. By contrast, ESM2-RecCut predicted two domains in d1wd3a1, although SCOP considers it a single-domain protein.

We were unable to directly compare the running time of the different methods, as we couldn't run ResPRE to predict contact maps (the FUpred package lacks some of the important files needed for running the pipeline). It was reported in the paper [20] that it takes a few hours on average to predict domains for a protein which is longer than 400 amino acids. The ResPRE-FUpred method is time consuming because it involves two slow steps: the contact map prediction step (ResPRE) and the domain prediction (FUpred) in cubic time. Our method ESM2-RecCut reduces the running time significantly by utilizing ESM-2's contact map prediction (it was shown that by bypassing the generation of MSA, ESMFold achieved an order-of-magnitude acceleration comparing to ALphaFold2 [11]). In addition, RecCut reduces the time complexity for domain segmentation from cubic time to quadratic, with respect to the protein length.

Here we showed the comparison of running time spent on the domain segmentation step (not including the contact map prediction) by the different approaches. For the 2549 FUpred benchmark proteins, FUpred used 158 seconds, whereas RecCut used 40 seconds. The running time difference is more significant on the homolog detection benchmarks (with longer proteins than the FUpred benchmark). For example, for a total of 13342 proteins from the nomax50 benchmark, given their predicted contact maps (by ESM-2), it took FUpred 32 hours to predict the domains, whereas it took RecCut only 18 mins.

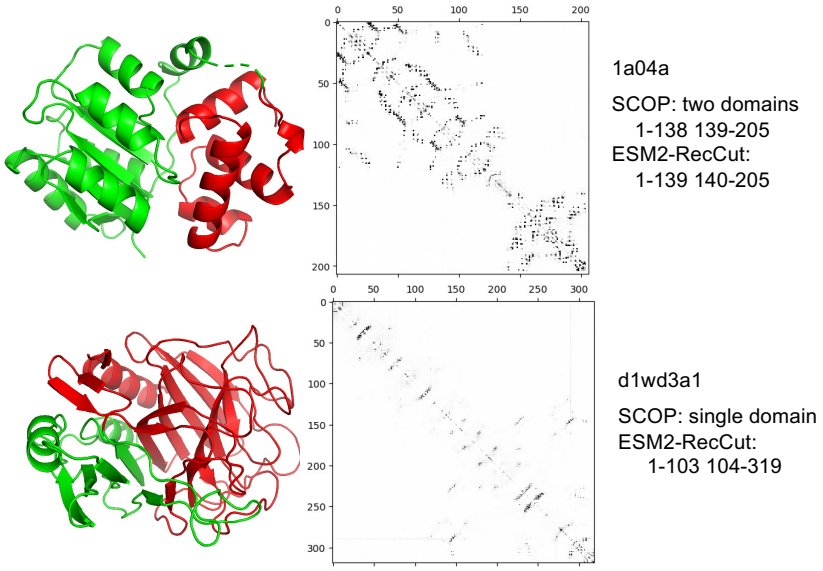


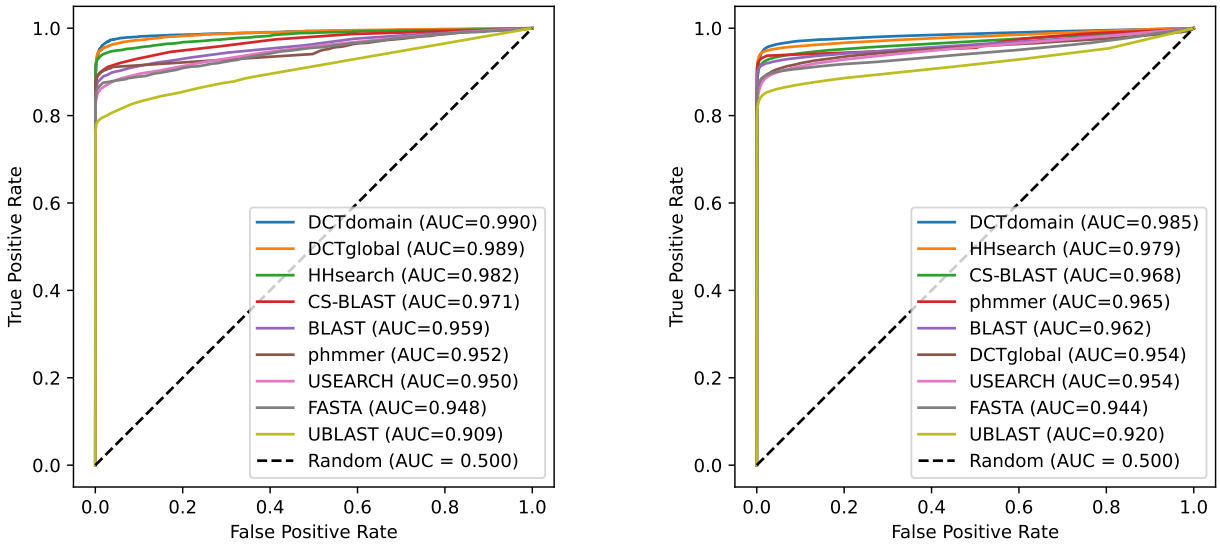
Figure 2: Examples of domain segmentation using ESM2-RecCut. The cartoon visualizations (by PyMol) of the protein structures are shown on the left, with predicted domains (by ESM2-RecCut) shown in different colors: the first domain in green and the second domain in red. The figures in the middle show ESM-2 contact maps of the two proteins.

Combining the accuracy and running time, we chose to use ESM2-RecCut as the approach for predicting domains, and used it for preparing domain level DCT fingerprints for homology detection. We show below that ESM-2 contact map based domain predictions, although not as accurate as those from ResPRE-FUpred, still gave good performance for referring DCT fingerprints for detecting similarities between proteins.

Table 2: Single- and multi-domain classification results on 2549 test proteins from the FUpred benchmark.

Method	Multi-domain		Single-domain		All	
	Precision	Recall	Precision	Recall	ACC	MCC
ResPRE-FUpred*	0.860	0.873	0.936	0.929	0.910	0.799
ESM2-FUpred	0.700	0.939	0.963	0.799	0.846	0.700
ESM2-RecCut	0.729	0.890	0.938	0.835	0.853	0.696

*ResPRE-FUpred results are taken from [20]. In principle, ESM2-FUpred and ESM2-RecCut should result in the same results as they use the same scoring scheme (so called FUscore). However, there are certain technical details that we cannot replicate in ESM2-RecCut so the results vary slightly. 'Pre', 'Rec', 'ACC' and 'MCC' are the precision, recall, accuracy and Matthew's correlation coefficient, respectively.



(a) pfam-max50 **(b)** pfam-nomax50

Figure 3: ROC plots for comparison of the different methods on two benchmarks of global homologs. (a) pfam-max50 benchmark; (b) pfam-nomax50 benchmark. PROST is not included in the plots, as DCTglobal is essentially the same as PROST.

Results on global homolog benchmarks

We applied our method DCTdomain, which computes the similarity between proteins using their DCT fingerprints (of individual domains and whole proteins) to the two global homolog benchmarks, and compared the results with the method that uses the DCT fingerprint of the whole proteins only (DCTglobal, which is essentially the same as the PROST method [13]) and other methods for homology detection. For accuracy and ROC calculation, we used the DCT similarity score (see Methods) for DCTdomain and DCTglobal, and bit-scores for all the other methods that we compared.

Figure 3 shows the comparison of homology detection on the two benchmarks (max50 and nomax50). Our results showed that using DCT of whole proteins underperformed, achieving worse AUCs than a few methods including PHMMER on the nomax50 benchmark (although DCTglobal still achieved a good performance with AUC of 0.954). This result is consistent with [13] that the global DCT based distance (PROST) didn't work well for detecting similarity between protein pairs that share global similarities but with extended, undefined regions between shared domains. By contrast, DCT similarity based on domains (DCTdomain) still gave good results, outperforming all methods that were compared, including DCTglobal.

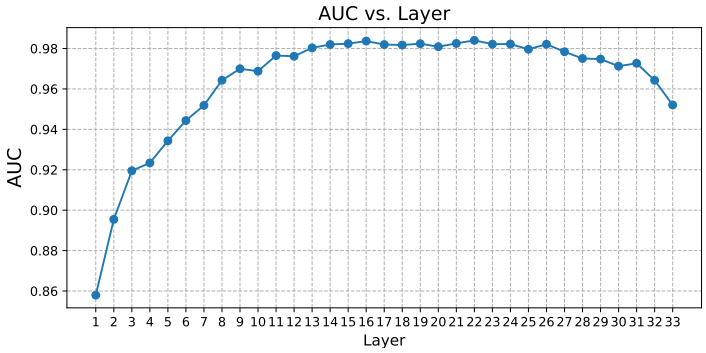


Figure 4: Comparison of the performance on pfam-max50 benchmark when DCT fingerprints (5x44 flattened into a vector of 220 dimensions) were generated from different layers of ESM-2 embeddings.

We note the above results were based on DCT fingerprints from two layers of the ESM-2 embeddings of proteins: layer 14 transformed into 3x85 matrix and layer 26 transformed into 5x44 matrix, and the two reduced matrices are flattened and concatenated into a vector of 457 dimensions. This configuration was shown to perform well in PROST [13]. To confirm

Table 3: AUC and total run time for each method (listed from the fastest to slowest) on pfam-local benchmark with 15273 pairs of proteins.

	DCTdomain*	USEARCH	UBLAST	BLAST	FASTA	phmmer	CS-BLAST	HHsearch
AUC	0.964	0.906	0.840	0.951	0.906	0.861	0.952	0.971
time	5.3 s	156 s	237 s	468 s	749 s	993 s	50.0 m	5.7 hr

*: our program reports both DCTglobal scores and DCTdomain scores and the reported time is for computing both. All programs were run on the same linux computer using one CPU.

their results, we also tested the impact of using different layers and the results (Figure 4) show that using individual layer 13 to 26 all gave similar results.

Results on local homolog benchmark

Figure 5 shows that DCTdomain works well for detecting local similarities between the proteins (but the proteins don't share global similarities). DCTdomain only performed slightly worse than HHsearch on this benchmark, but outperformed all other methods, including DCTglobal. Given that a DCT fingerprint is an averaged representation, it is not surprising that DCTglobal had the worst performance on this benchmark.

Table 3 summarizes the running time of the different approaches along with their performance in AUC. The results show that using DCT fingerprints achieved fast similarity calculation comparing to all other methods. Although our approach DCTdomain performed slightly worse than HHSearch in AUC on this benchmark, our method is more than three orders of magnitude faster. For fair comparison, the times reported in Table 3 don't include the time for construction of the search database that is required by some of the approaches, and protein embedding and DCT fingerprints construction for the DCTdomain and DCTglobal. We acknowledge that ESM-2 embedding and domain segmentation take time, but we only do the calculation once, and the DCT fingerprints of proteins can be saved in numpy NPZ file (compressed) for later applications.

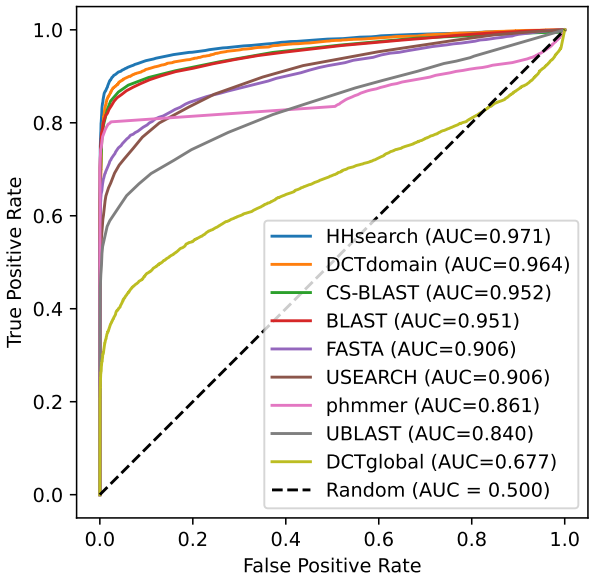
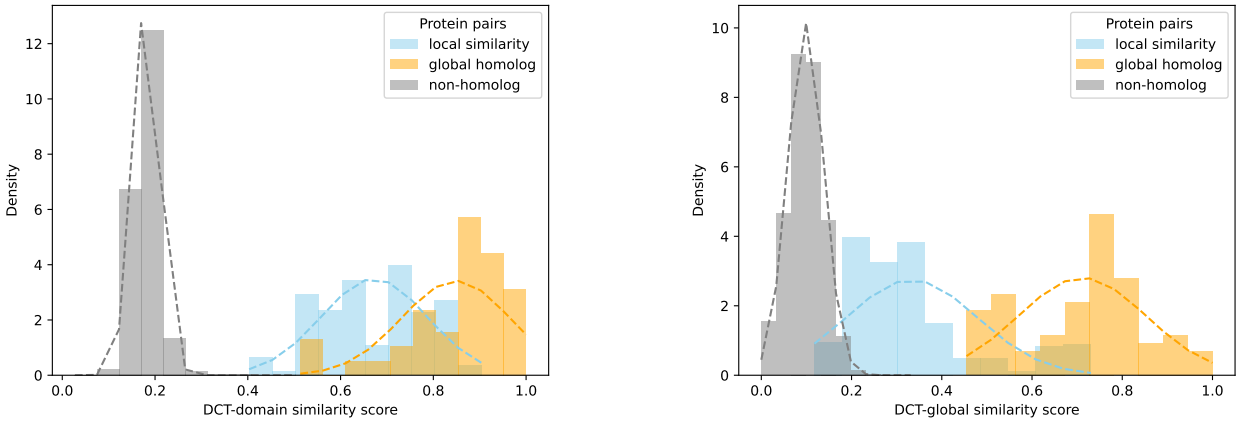


Figure 5: ROC plots for comparison of the different methods on the pfam-local benchmark that contains proteins sharing local similarities.

G6PD containing proteins

As a case study, we applied DCTdomain and DCTglobal to a collection of G6PD containing proteins. G6PD proteins (glucose-6-phosphate dehydrogenase) are enzymes whose main function is to produce NADPH, a key electron donor in the defense against oxidizing agents and in reductive biosynthetic reactions. We used InterPro [29] to look up G6PD_N (PF00479, Glucose-6-phosphate 1-dehydrogenase, NAD-binding domain) containing proteins, which showed that this domain can be found in 163 different domain architectures, among which, the domain architecture (G6PD_N - G6PD_C) is the dominate one found in 44k sequences (the second most frequent domain architecture is found in 1658 proteins). We collected a total of 28 G6PD containing proteins representing different domain architectures, including 8 sequences that have this domain architecture: PF00479 (G6PD_N) - PF02781 (G6PD_C) - PF02781 (G6PD_C) - PF01182 (Glucosamine_iso), 7 sequences with this domain architecture PF00479 - PF02781 - PF13347 (MFS_2), 6 sequences of this architecture PF03446 (NAD_binding_2) - PF00393 - PF00479 - PF02781, and 6 sequences of this domain



(a) **(b)**

Figure 6: Comparison of domain embedding based similarity between G6PD containing proteins. (a) DCTdomain; (b) DCTglobal.

architecture PF00479 - PF02781 - PF08123 (DOT1). Since these proteins all contain G6PD domain, pairs of these proteins can be global homologs (they share the same domain architecture), or share local similarities (some domains are different). We compared the distributions of the DCT fingerprint similarity of the global homologs, partially similar pairs, and non-homologs (from the pfam-nomax50 collection). Figure 6 shows that all similar proteins (sharing global or local similarities) have high DCTdomain similarity, forming distinct distributions that are well separated from non-homolog DCT similarity distribution (Figure 6a). By contrast, if using DCTglobal similarities, there is significant overlap between the DCT similarity distributions of the local homologs and the non-homologs (Figure 6b).

Discussion

Using ESM-2 to predict contact maps for subsequent domain prediction with RecCut, as well as generating embeddings that were compressed using DCT, proved to be an effective method for detecting sequences that are homologous. DCTdomain performed well on every benchmark we tested it against. It performed the best on both global homolog benchmarks relative to every other method we tested and was a close second behind HHsearch on the local homolog benchmark. One possible explanation for the drop in performance on the local benchmark is the prediction of domains based on ESM2-predicted contact maps. Using the FUpred domain prediction benchmark, it is clear that ResPRE-predicted contact maps are more effective than ESM2-predicted contact maps for domain prediction. But our method ESM2-RecCut is significantly faster than the existing approaches that rely on more accurate but slower contact map prediction methods. In addition, as our results showed, the domain segmentations achieved by ESM2-RecCut are already very helpful for generating DCT fingerprints for local homology detection. We anticipate that our RecCut approach can also be used for automatic domain segmentation when 3D structures of proteins are available.

Since multi-domain proteins are prevalent, and various domain architectures found in those proteins have important structural and functional implications, it is important to develop methods that can effectively compare multi-domain proteins. We showed that it is important to have domain or sub-domain level representations such as the DCT fingerprints for homology detection, and we proposed a method that utilizes domain segmentation based on contact maps for this purpose. In this work, we compute the similarity of two proteins as the highest similarity of any two DCT fingerprints of the domains found in the proteins, and we anticipate that other metrics may be developed for more sensible similarity quantification.

We anticipate that DCT fingerprints can be applied to other applications. They can be used to enable fast database searches, where DCT fingerprint based searches can be combined with sequence-based methods (such as MMSeq2 as in knnProtT5 [12]) or alignment methods that utilize contextual embeddings of individual residues (PEbA [30] or vcMSA [31]) for more accurate and sensitive detection of similar proteins. They may also be applied for other structural and functional analysis of proteins.

References

[1] Burkhard Rost. Twilight zone of protein sequence alignments. *Protein Engineering, Design and Selection*, 12(2):85–94, feb 1999.

[2] Stephen F. Altschul, Warren Gish, Webb Miller, Eugene W. Myers, and David J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, oct 1990.

[3] Stephen F Altschul, Thomas L Madden, Alejandro A Schäffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic acids research*, 25(17):3389–3402, 1997.

[4] Andreas Biegert and Johannes Söding. Sequence context-specific profiles for homology searching. *Proceedings of the National Academy of Sciences*, 106(10):3770–3775, 2009.

[5] Sean R Eddy. A new generation of homology search tools based on probabilistic inference. pages 205–211, 2009.

[6] Johannes Söding. Protein homology detection by hmm–hmm comparison. *Bioinformatics*, 21(7):951–960, 2005.

[7] Ahmed Elnaggar, Michael Heinzinger, Christian Dallago, Ghalia Rehawi, Yu Wang, Llion Jones, Tom Gibbs, Tamas Feher, Christoph Angerer, Martin Steinegger, Debsindhu Bhowmik, and Burkhard Rost. ProtTrans: Toward understanding the language of life through self-supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(10):7112–7127, oct 2022.

[8] UniProt Consortium. Uniprot: a hub for protein information. *Nucleic acids research*, 43(D1):D204–D212, 2015.

[9] Martin Steinegger and Johannes Söding. Clustering huge protein sequence sets in linear time. *Nature Communications*, 9(1), jun 2018.

[10] Martin Steinegger, Milot Mirdita, and Johannes Söding. Protein-level assembly increases protein sequence recovery from metagenomic samples manyfold. *Nature Methods*, 16(7):603–606, jun 2019.

[11] Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Salvatore Candido, and Alexander Rives. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, mar 2023.

[12] Konstantin Schütze, Michael Heinzinger, Martin Steinegger, and Burkhard Rost. Nearest neighbor search on embeddings rapidly identifies distant protein relations. *Frontiers in Bioinformatics*, 2:1033775, 2022.

[13] Mesih Kilinc, Kejue Jia, and Robert L Jernigan. Improved global protein homolog detection with major gains in function identification. *Proceedings of the National Academy of Sciences*, 120(9):e2211823120, 2023.

[14] Martin Steinegger and Johannes Söding. Mmseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nature biotechnology*, 35(11):1026–1028, 2017.

[15] Temple F Smith, Michael S Waterman, et al. Identification of common molecular subsequences. *Journal of molecular biology*, 147(1):195–197, 1981.

[16] Daniele Raimondi, Gabriele Orlando, Yves Moreau, and Wim F Vranken. Ultra-fast global homology detection with discrete cosine transform and dynamic time warping. *Bioinformatics*, 34(18):3118–3125, 2018.

[17] John Makhoul. A fast cosine transform in one and two dimensions. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(1):27–34, 1980.

[18] Lijia Yu, Deepak Kumar Tanwar, Emanuel Diego S Penha, Yuri I Wolf, Eugene V Koonin, and Malay Kumar Basu. Grammar of protein domain architectures. *Proceedings of the National Academy of Sciences*, 116(9):3636–3645, 2019.

[19] Yuzhen Ye and Adam Godzik. Comparative analysis of protein domain organization. *Genome research*, 14(3):343–353, 2004.

[20] Wei Zheng, Xiaogen Zhou, Qiqige Wuyun, Robin Pearce, Yang Li, and Yang Zhang. Fupred: detecting protein domains through deep-learning-based contact map prediction. *Bioinformatics*, 36(12):3749–3757, 2020.

[21] Chengxin Zhang, Wei Zheng, SM Mortuza, Yang Li, and Yang Zhang. Deepmsa: constructing deep multiple sequence alignment to improve contact prediction and fold-recognition for distant-homology proteins. *Bioinformatics*, 36(7):2105–2112, 2020.

[22] Yang Li, Chengxin Zhang, Eric W Bell, Dong-Jun Yu, and Yang Zhang. Ensembling multiple raw coevolutionary features with deep residual neural networks for contact-map prediction in casp13. *Proteins: Structure, Function, and Bioinformatics*, 87(12):1082–1091, 2019.

[23] Seung Hwan Hong, Keehyoung Joo, and Jooyoung Lee. Condo: protein domain boundary prediction using coevolutionary information. *Bioinformatics*, 35(14):2411–2417, 2019.

[24] Qiang Shi, Weiya Chen, Siqi Huang, Fanglin Jin, Yinghao Dong, Yan Wang, and Zhidong Xue. Dnndom: predicting protein domain boundary from sequence alone by deep neural network. *Bioinformatics*, 35(24):5128–5136, 2019.

[25] Antonina Andreeva, Eugene Kulesha, Julian Gough, and Alexey G Murzin. The scop database in 2020: expanded classification of representative family and superfamily domains of known protein structures. *Nucleic acids research*, 48(D1):D376–D382, 2020.

[26] Ganapathi Varma Saripella, Erik LL Sonnhammer, and Kristoffer Forslund. Benchmarking the next generation of homology inference tools. *Bioinformatics*, 32(17):2636–2641, 2016.

[27] Robert C Edgar. Search and clustering orders of magnitude faster than blast. *Bioinformatics*, 26(19):2460–2461, 2010.

[28] William R Pearson and David J Lipman. Improved tools for biological sequence comparison. *Proceedings of the National Academy of Sciences*, 85(8):2444–2448, 1988.

[29] Typhaine Paysan-Lafosse, Matthias Blum, Sara Chuguransky, Tiago Grego, Beatriz Lázaro Pinto, Gustavo A Salazar, Maxwell L Bileschi, Peer Bork, Alan Bridge, Lucy Colwell, et al. Interpro in 2022. *Nucleic acids research*, 51(D1):D418–D427, 2023.

[30] Yuzhen Ye and Benjamin Giovanni Iovino. Protein embedding based alignment. *Authorea Preprints*, 2023.

[31] Claire D. McWhite and Mona Singh. Vector-clustering multiple sequence alignment: Aligning into the twilight zone of protein sequence similarity with protein language models. oct 2022.

Appendix

Algorithm 1 The RecursiveCut algorithm for recursive domain segmentation

```
function RecCut(S[1,...,n], n, W_{1-cut}, W_{2-cuts}, W[1,...n,1,...n])
    V_{1-cut}, i ← OneCutPartition(S[1,...,n], n, W[1,...n,1,...n])
    V_{2-cuts}, i, j ← TwoCutsPartition(S[1,...,n], n, W[1,...n,1,...n])
    if V_{1-cut} > W_{1-cut} and V_{2-cuts} > W_{2-cuts} then
        Output S[1,...,n]
    else if V_{1-cut} − W_{1-cut} ≤ V_{2-cuts} − W_{2-cuts} then
        RecCut(S[1,...,i], i, W_{1-cut}, W_{2-cuts}, W[1,...n,1,...n])
        RecCut(S[i+1,...,n], n − i, W_{1-cut}, W_{2-cuts}, W[1,...n,1,...n])
    else
        RecCut(join(S[j+1,...,n], S[1,...,i]), n − j + i, W_{1-cut}, W_{2-cuts}, W[1,...n,1,...n])
        // join() concatenates two segments
        RecCut(S[i+1,...,j], j − i, W_{1-cut}, W_{2-cuts}, W[1,...n,1,...n])
    end if
end function

function OneCutPartition(S[1,...,n], n, W[1,...n,1,...n])
    G[1] ← 0
    for i ← 2 to n do
        G[i] ← G[i − 1]
        for j ← 1 to i − 1 do
            G[i] ← G[i] + W[j, i]
        end for
    end for
    H[n] ← 0
    for i ← n − 1 downto 1 do
        H[i] ← H[i − 1]
        for j ← i + 1 to i do
            H[i] ← H[i] + W[i, j]
        end for
    end for
    V[1] ← 0
    MinV ← ∞
    for i ← 2 to n do
        V[i] ← V[i − 1] − G[i] + H[i]
        if V[i] < MinV then
            MinV ← V[i]
            MinIndexI ← i
        end if
    end for
    Return MinV, MinIndexI
end function
```

Algorithm 2 The RecCut algorithm for recursive domain segmentation (cont'd)

```
function TwoCutsPartition(S[1,...,n], n, W[1,...n,1,...n])
    G[1] ← 0
    for i ← 2 to n do
        G[i] ← G[i − 1] + W[i − 1, i]
    end for
    N_1[0] ← 0
    for i ← 1 to n do
        N_1[i] ← N_1[i − 1] + G[i]
    end for
    for i ← 1 to n do
        N_2[i] ← N_1[n] − N_1[i − 1]
    end for
    for i ← 1 to n do
        E[i, 0] ← 0
        for j ← 1 to n do
            E[i, j] ← E[i, j − 1] + C[i, j]
        end for
    end for
    for i ← 1 to n do
        for j ← i + 1 to n do
            F[i, j] ← E[i, j] − E[i, i − 1]
        end for
    end for
    for i ← 1 to n do
        for j ← i + 1 to n do
            T[i, j] ← 0
        end for
    end for
    for j ← 1 to n do
        for k ← i + 1 to j do
            T[i, j] ← T[i, j] + F[k, j]
        end for
    end for
    MinV ← ∞
    for i ← 1 to n do
        for j ← i + 1 to n do
            V[i, j] ← N_1[n] − N_1[i] − N_2[j] − T[i, j]
            if V[i, j] < MinV then
                MinV ← V[i, j]
                MinIndexI ← i
                MinIndexJ ← j
            end if
        end for
    end for
    Return MinV, MinIndexI, MinIndexJ
end function
```