Protein domain decomposition using spectral graph partitioning

Steffen Lang

April 17, 2007

Studienarbeit am ITI Wagner
Fakultät für Informatik
Universität Karlsruhe (TH)

und

Computer Science Department
Carnegie Mellon University
Pittsburgh

Acknowledgement

I would like to express my gratitude to all those who gave me the possibility to complete this report. I want to thank my advisors Professor Dorothea Wagner and Professor Gary Miller for their support. Furthermore, I have to thank David Tolliver and Christopher Langmead for their suggestions and ideas. Finally I want to thank the interACT center for funding my stay in Pittsburgh.

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig angefertigt habe und nur die angegebenen Hilfsmittel und Quellen verwendet wurden.

Karlsruhe, den 17. April 2007

Abstract

Proteins usually consist of several structural domains which play an important role in the classification and analysis of them. As the number of known protein structures grows at an exponential rate, a computer-aided domain decomposition of proteins becomes desirable. We present three different decomposition methods in this report. They are based on a graph representation of the proteins. These graphs are recursively decomposed into domains using the Fiedler vector. Our methods correctly decompose 89.1% and 90.9% of the 55 proteins in our test data set. We also show the problems with the incorrectly decomposed proteins in a detailed analysis.

Contents

1. Introduction

Large proteins usually consist of several structural domains. These domains are described broadly as parts of the protein structure that can fold independently and form compact and stable structures. They are not necessarily continuous in the amino acid sequence. There are several databases containing domain assignments like *CATH* [9] or *SCOP* [7], which play an important role in the classification and analysis of proteins. Many of these assignments have been done by human experts like crystallographers. As the number of known protein structures grows at an exponential rate, a computer-aided domain decomposition of proteins becomes desirable. The decompositions provided by such a method should be similar to those provided by the crystallographers.
Many known automatic decomposition methods rely on a graph representation of the protein structure. Some of them use network flow problems [11] or spectral graph partitioning [5] to identify the domains. In this report we describe and test three different methods based on spectral graph theory. All of them cut the protein recursively, until some stop criteria based on domain features are reached. The first one uses the values of the eigenvector corresponding to the lowest non-zero eigenvalue (also known as the *Fiedler vector*) of the graph to cut the protein. The second method uses the values of a certain number of eigenvectors of the graph as an input for known clustering algorithms like K-means. The last method creates a flow network with the help of the Fiedler vector and then uses a max-flow/min-cut algorithm. All of these methods perform well compared to known domain assignments. Two of them correctly decompose 89.1% and one 90.9% of our test data set.
Section 2 of this report contains some general definitions of a protein graph as well as a definition of the decomposition problem. The next section roughly describes a few known protein domain decomposition methods with good results. Section 4 then introduces our basic algorithm and the three different methods. The following section gives an overview of the results of our methods and a detailed analysis of incorrect decompositions. The appendix includes definitions of graph related matrices and a physical interpretation of some of their eigenvectors.

2. Protein decomposition as a graph problem

2.1. General protein graph definition

In this report a *graph* G is an undirected, weighted graph defined as a triple (V, E, w), where V is the set of vertices or nodes, E the set of edges and w the weight function. Each e in E is a pair $\{u, v\}$ with u, v in V, and w is a mapping from E to $\mathbb{R}$.

Proteins consist of folded amino acid chains forming three-dimensional structures. Each amino acid is composed of several atoms like the alpha carbon C_α. The amino acids in a chain form a sequence where every amino acid has an unique number and is connected with its neighbors by peptide bonds. As structural domains occur as parts of one chain, we are only interested in representing amino acid chains as graphs. There is one vertex per amino acid named with the sequence number of the amino acid, and an edge between two vertices if the distance between their amino acids is below some threshold value T. This value usually lies between 4 Angstrom ($\AA$) and 13 Å. The distance between two amino acids can be defined in one of the following ways:

- the distance between their C_α atoms,
- the distance between their C_β atoms or
- the minimum distance between any two atoms of the two amino acids.

All these distances can be derived for proteins with known structures from the coordinates of their atoms. A useful resource for such coordinates is the Protein Data Bank (*PDB*) [1].
The weight of an edge can depend on many factors like

- the secondary structures of the protein,
- the distances between the atoms of the amino acids or
- the order in the amino acid sequence.

Secondary structures of a protein are generalized forms of its local segments. The most common secondary structures are *α-helices* and *β-sheets*. The latter are very important for domain decomposition and consist of β-strands connected by hydrogen bonds. A *β-strand* is a sequence of amino acids whose backbones are almost fully extended.

These graph representations of proteins are also known as *contact maps* or *contact graphs*, and the edges are also called *contacts*. A small example how to create a protein graph can be found in 4.1.1.

2.2. Problem definition

The problem of decomposing a protein into its domains becomes the problem of cutting a graph into small partitions having properties of structural domains. Domain properties used by us and others [5][11] include

- the *domain size*: the number of amino acids in a domain,
- the *average fragment length* of a domain: the number of amino acids divided by the number of continuous sequence fragments in a domain,
- the *interface size* between two domains: the ratio between the minimum number of contacts in one of the domains and the number of contacts between the domains,
- the *compactness* of a domain: the ratio between the number of contacts in the domain and the domain size and
- the fact that a β-sheet usually belongs to only one domain.

These properties can be used as stop criteria for recursive algorithms: the graph is cut into two pieces and these pieces are used as new inputs for the algorithm until a cut produces a piece which does not match the domain properties.

3. Examples of known decomposition methods

3.1. Automatic Domain Decomposition of Proteins by a Gaussian Network Model (GNM method)

In [5] Kundu *et al.* proposed a protein domain decomposition method based on the Gaussian Network Model (called *GNM method* in this report). Their protein graph definition is equivalent to our definition in 2.1. They use a threshold T of 11 Å and a weight of 1 for every edge. The distance between amino acids is defined as the distance between their C_α atoms. Like all methods in this report the GNM method uses a recursive approach: the graph is cut into two parts, and the parts are then cut recursively until a stop criterion is fulfilled. To find a cut they compute the Fiedler vector (Appendix B includes a definition and interpretation of this vector) and assign nodes with a value of less than 0 in this vector to one part and the remaining nodes to the other part. After the cut they do some *refinement*: for every continuous sequence segment with the same part assigned to every node in it and a size less than 10, the assignment is changed to the assignment of the biggest neighbor segment. Stop criteria used by the GNM method are a minimum domain size of 40 amino acids, a minimum average fragment length of 35 and a prevention of β-sheet breaks with some exceptions. They achieved a correct assignment for 90.1% of the proteins in the test data set (see 5.1 for a detailed description of the test data set).

3.2. DomainParser

DomainParser is a protein domain decomposition method proposed by Xu *et al.* in [11]. It is based on a flow network and a minimum cut of it. There is one node per amino acid, and an edge between two nodes if the distance between the closest atoms of their corresponding amino acids is less than 4 Å. The *capacity* or weight of the edges depends on atom distances, β-strands and β-sheets. Edges between close amino acids and those in the same β-strand get a bigger capacity to prevent cutting them. The necessary source and sink for the flow problem are artificially defined nodes. Again, they use a recursive approach. The cut is found by a max-flow/min-cut algorithm and refined mostly the same way as in 3.1. Stop criteria for DomainParser are a minimum domain size of 40 amino acids, a minimum average fragment length of 35, a small interface size, a high compactness and the prevention of β-sheet breaks with some exceptions. The authors correctly decomposed 78.2% of the test data set (see 5.1 for a detailed description of the test data set).

4. Our domain decomposition methods

4.1. Basic algorithm

All our methods are based on the same basic algorithm with few adjustments for the specific methods. The methods differ in the weights of the edges in the contact graph and the method to cut the graph. The steps of the basic algorithm are described in the following subsections and illustrated by some contrived examples as well as by a pseudo-code listing.

4.1.1. Graph definition

First we have to define our protein graph. This is done accordingly to 2.1 with these parameters:

- the threshold T is 11 Å,
- the distance between two amino acids is defined as the distance between their C_α atoms and
- the weight of an edge is 1 with an addition for *neighboring* amino acids (i. e. amino acids adjacent in the sequence) in β-strands, depending on the specific method.

The additional weight hinders the cut of β-sheets and β-strands.

There is a small example of four amino acids with the coordinates of their C_α atoms in Table 1. The last three ones are in the same β-strand. Table 2 shows the resulting protein graph as an Adjacency matrix (see Appendix A for a description of the Adjacency matrix). There are contacts between all amino acids except the first and the last one, and an addition of 9 for neighboring amino acids in a β-strand.

Amino Acid	Atom	Coordinates in Å	β-strand
1	C_α	5 3 2	-
2	C_α	5 6 5	1
3	C_α	13 9 2	1
4	C_α	15 7 5	1

Table 1: Coordinates of C_α atoms

0	1	1	0
1	0	10	1
1	10	0	10
0	1	10	0

Table 2: Contact map for the amino acids in Table 1

Depth	Input	Output
0	1-150	1-90
		91-150
1	1-90	1-43
		44-90
	91-150	*91-109*
		110-150
2	1-43	*1-20*
		21-43
	44-90	*44-70*
		71-90
Result		1 -43
		44-90
		91-150

Table 3: Example of a recursive decomposition

4.1.2. Recursive decomposition

The next step is to cut the graph. We do this by first cutting the whole graph into two parts and then cutting this parts recursively, as long as no stop criterion is fulfilled. The parts are subgraphs of the protein graph. A *subgraph* is a graph (V_1, E_1, w) where V_1 is a subset of V and E_1 is the biggest subset of E with both nodes in V_1 for every element. When a stop criterion is reached, the last cut is not valid and the corresponding part is a domain in the result. The stop criteria used by us are described in 4.1.3.
Table 3 shows an example decomposition of a graph with 150 nodes. The first column indicates the recursion depth, the second one the nodes in the subgraphs and the last one the output of the algorithm. Parts rejected by the stop criteria are marked italic. The result of the decomposition is in the last row.
After each cut, some refinement has to be done in order to avoid too short amino acid sequences in the domains. To achieve this the assignment has to be changed for every continuous sequence segment with the same part assigned to every node in it and a size less than 10. We first delete the assignments for these segments and join neighboring segments without an assignment. Then we assign these segments to the part of their biggest neighbor segment. If all neighbor segments are in parts separated from the *current parts* (i. e. the output of the last cut) in an earlier recursion step, we don't want to change this earlier cut and assign the segment to the part with the most nodes in the original assignment. If just one neighbor segment is in a part separated from the current parts, we assign it to the part of the other neighbor.
The refinement is illustrated by an example in Table 4. In this example, the current parts are part 2 and part 3. The original assignment of the cut, an intermediate state and the refined assignment are depicted in the columns.

Original		Intermediate		Refined	
Segment	Part	Segment	Part	Segment	Part
1-15	2	1-15	2	1-28	2
16-20	3	16-20	-		
21-28	2	21-28	-		
29-40	1	29-40	1	29-40	1
41-44	3	41-44	-	41-50	3
45-47	2	45-47	-		
48-50	3	48-50	-		
51-70	1	51-70	1	51-70	1
⋮	⋮	⋮	⋮	⋮	⋮

Table 4: Example refinement of parts 2 and 3

4.1.3. Stop criteria

The stop criteria we used for our methods are

- a minimum domain size of 40,
- a minimum average fragment length of 35,
- a minimum interface size of 1.9,
- a minimum compactness of 8.85 and
- no *breaks* of β-sheets are allowed (i. e. nodes of a β-sheet in different parts).

See 2.2 for a more detailed explanation of these criteria. Each stop criterion is checked for the refined cuts and not for the original cuts.

4.1.4. Pseudo-code listing

Procedure Decomposition(P)
  begin
    Cut P into P_1 and P_2
    Refine the cut
    if P_1 or P_2 fulfill a stop criterion
      return $\{P\}$
    else
      return Decomposition(P_1) $\cup$ Decomposition(P_2)
  end

---

node	1	2	3	4
value of Fiedler vector	-0.2	0.2	0.3	0.1

t	S	ncv(S)
0.1	{1}	0.7
0.2	{1,4}	0.6
0.3	{1,4,2}	0.65

Table 5: Example for the TC method

4.2. Threshold Cut Method

Our *Threshold Cut Method* (referred to as *TC method* in this report) uses the graph definition of 4.1.1 with an additional weight of 49 for neighboring amino acids in β-strands. The cut is based on an ordering of the nodes by their value in the Fiedler vector (see Appendix B for a definition of this vector). We try to cut the graph by finding a optimal threshold value t and assigning all nodes with a lower value to one part of the graph and the rest to the other part. The threshold t is *optimal* if the corresponding cut locally optimizes an objective function. We used the *normalized cut value* introduced by Shi and Malik in [10] as our objective function. The normalized cut value for a graph $G = (V, E, w)$ and a cut $S \subset V$ is defined as

$$ncv(S) := \frac{1}{2} \times \left(\frac{w(S, V \backslash S)}{w(S, V)} + \frac{w(V \backslash S, S)}{w(V \backslash S, V)} \right) \tag{1}$$

where

$$w(V_1, V_2) := \sum_{\{u,v\} \in E, u \in V_1, v \in V_2} w(\{u, v\}). \tag{2}$$

The local optimum is the minimum *ncv* of all possible threshold cuts.
In the example in Table 5 the threshold 0.2 optimizes the *ncv* and the resulting cut S is {1,4}.

The TC method is very similar to the GNM method. The main differences are the additional weights for amino acids in β-strands, other stop criteria and the more sophisticated determination of t, as the GNM method just uses 0 as threshold.
You can see in Fig. 1 that the TC method is able to assign domains in the same way crystallographers did. The figure shows the values of the Fiedler vector of the nodes in the protein graph of the protein with PDB code 1g6n (see 5.2 for a description of the PDB codes) and the domain for each node. A threshold of 0.029 optimizes the *ncv* and the TC method assigns all domains correctly.

4.3. Spectral Clustering with K-means

Our *Spectral Clustering with K-means method* (referred to as *SCK method* in this report) is inspired by [8]. It uses the graph definition of 4.1.1 with an additional weight of 99 for

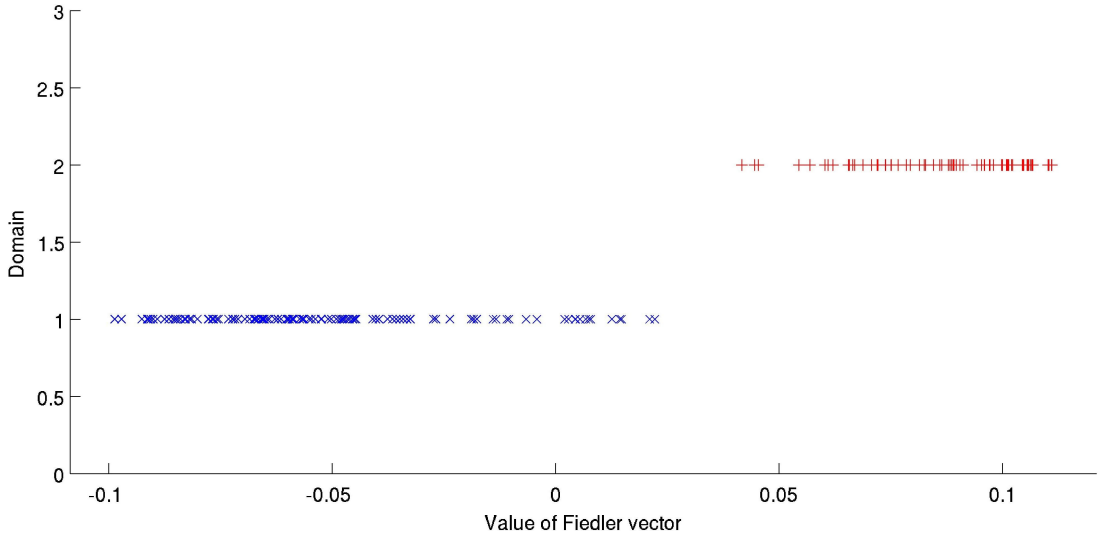

Figure 1: Fiedler vector of 1g6n

Original	v_1	1	1	1	...
	v_2	2	-3	1	...
Normalized	v_1	$\frac{1}{\sqrt{5}}$	$\frac{1}{\sqrt{10}}$	$\frac{1}{\sqrt{2}}$	...
	v_2	$\frac{2}{\sqrt{5}}$	$\frac{-3}{\sqrt{10}}$	$\frac{1}{\sqrt{2}}$	...

Table 6: Normalization example

neighboring amino acids in β-strands. The cut is found by computing the eigenvectors corresponding to some of the smallest eigenvalues of the graph (see Appendix B for a definition of this eigenvectors) and normalizing their values for each node to unit length. Then the values are interpreted as points in $\mathbb{R}^n$ where n is the number of eigenvectors used and these points can be used as input for known cluster algorithms.
We used the K-means algorithm to cut the graph into two parts and got the best results with the two smallest eigenvalues. Then all points are on the graph of the function $f(x) = \sqrt{1-x^2}$ (or $f(x) = -\sqrt{1-x^2}$) and the order along the axis of eigenvector 2 is the same as in the sorted Fiedler vector since all values in eigenvector 1 are the same (see Appendix B).
Table 6 shows an example of the normalization for two eigenvalues and the corresponding eigenvectors, and Fig. 2 the input data of K-means for the protein 1g6n when the two smallest eigenvalues are used.

4.4. MidFlow Rounding Method

Our last method is the *MidFlow Rounding Method* (referred to as *MFR method* in this report). It uses the MidFlow rounding introduced in [6]. This method is based on a

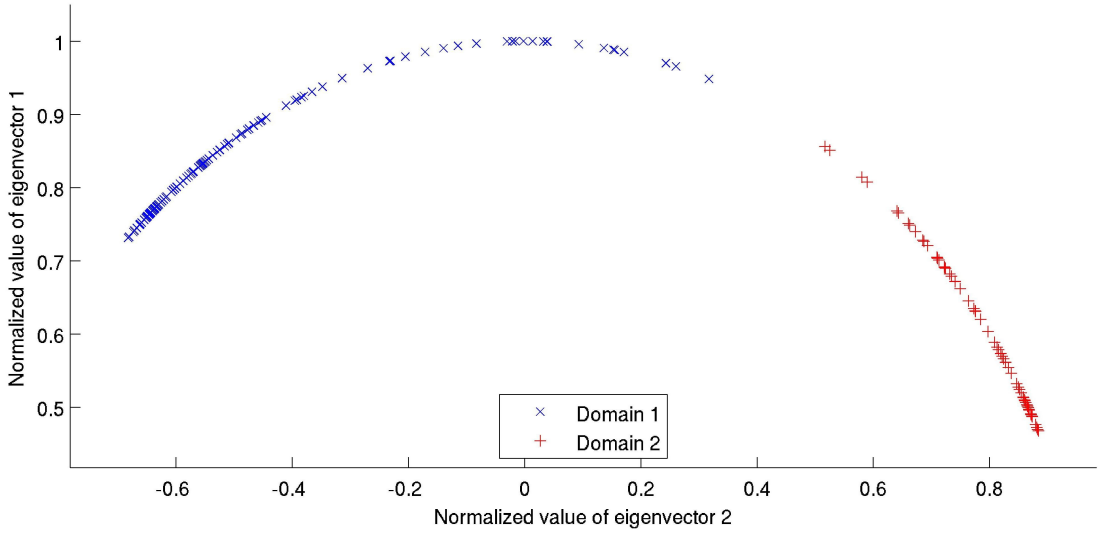

Figure 2: Normalized eigenvectors of 1g6n

network flow problem. To construct the network we need an ordering of the nodes which we obtain from the values of the Fiedler vector (see Appendix B for a definition of this vector) of the protein graph. The protein graph is defined as in 4.1.1 with an additional weight of 9 for neighboring amino acids in β-strands. MidFlow rounding looks for a *β-balanced cut*, i. e. a cut S with

$$\begin{aligned} S &\subset V \\ |S| &\geq \beta \times |V| \\ |V \backslash S| &\geq \beta \times |V| \end{aligned}$$

The method partitions V into three sets F, L and U. F contains the first $\beta \times |V|$ nodes (ordered by the values of the Fiedler vector), L the last $\beta \times |V|$ nodes and U the remaining nodes. The source s and the sink t are two additional nodes. The capacities between the nodes are the same as the edge weights in the protein graph, with additional edges of infinite capacity between the source and the nodes in F as well as between the sink and the nodes in L. Then the resulting cut is the minimum cut of the flow network, calculated by a max-flow algorithm. In each recursion step we compute the cut for the values $\frac{1}{4}, \frac{1}{3}$ and $\frac{2}{5}$ of β and take the cut with the best normalized cut value (see Eq. 1). Fig. 3 and Fig. 4 show an example for a graph and its MidFlow network.

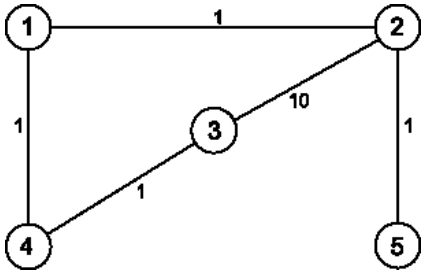

Figure 3: A graph with the Fiedler vector v_2 = (-0.4 0.01 -0.03 -0.4 0.8).

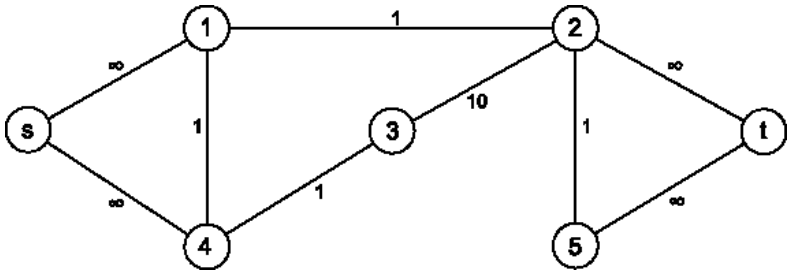

Figure 4: The corresponding MidFlow network for $\beta = \frac{2}{5}$.

5. Results

5.1. Test data set

We tested our methods on a data set of 55 proteins proposed by Jones *et al.* [3]. The GNM method and DomainParser have also been tested on this data set. It consists of 30 single-domain proteins, 20 two-domain proteins, 2 three-domain proteins and 3 four-domain proteins. We obtained the necessary data from the Protein Data Bank [1] and used DSSP [4] to find the β-sheets where the secondary structure information was not available in the Protein Data Bank.
A domain decomposition is considered *correct* if the number of domains is the same as in the manual assignment by crystallographers and the amino acid assignment to the domains is at least 85% in agreement with the crystallographers.

5.2. Test results

Tables 7-10 show the results for our methods. We tried all our methods with standard eigenvectors and generalized eigenvectors (see Appendix B for a detailed description). The SCK method has worse results for standard eigenvectors and there is no significant difference for the MFR method, so we only show the results for the eigenvectors with better results for these two methods.
There is one row per protein with multiple domains in the tables. Single domain proteins only occur if they are incorrectly decomposed since they always have an agreement of 100% when decomposed correctly. The first columns shows the PDB code of the protein.

It consists of four letters (a number followed by three alpha-numeric characters) and a chain identifier in uppercase if the protein has multiple chains. The second and third columns show the domain assignment by crystallographers and by the corresponding method. Fragments in a domain are delimited by a ';', domains by a '/'. The last column shows the *agreement* of the assignments, i. e. the percentage of the assigned amino acids assigned to the same domain by both decompositions. A decomposition is called *overcut* if it consists of more domains than the crystallographers' decomposition, and *undercut* if it consists of less.
The TC method and the SCK method correctly decomposed 49, the MFR method 50 of the 55 proteins in our test data set. A detailed explanation of incorrect decompositions grouped by their reasons follows in the next subsections.

Protein	Crystallographers	TC method	Agreement
1 domain:			
1ace	4-535	4-325;403-516/326-402;517-535	overcut
2 domains:			
1ezm	1-134/135-298	1-81;98-134/82-97;135-298	94.6%
1fnr	19-161/162-314	19-152/153-314	97.0%
1gpb	19-489/490-841	19-165;179-494;812-841/166-178;495-811	94.2%
1lap	1-150/171-484	1-127;138-155/128-137;156-484	97.8%
1pfkA	0-138;251-301/139-250;302-319	0-141;252-306/142-251;307-319	97.2%
1ppn	1-10;112-208/21-111;209-212	1-212	undercut
1rhd	1-158/159-293	1-157/158-293	99.7%
1sgt	22-123;234-245/129-233	16-121;236-245/122-235	98.2%
1vsgA	1-29;92-251/42-75;266-363	1-33;86-255/34-85;256-362	100.0%
1wsyB	9-52;86-204/53-85;205-393	17-45;91-188/3-16;46-90;189-394	90.6%
2cyp	3-145;266-294/164-265	2-144;263-294/145-262	98.5%
2had	1-155;230-310/156-229	1-310	undercut
3cd4	1-98/99-178	1-98/99-178	100.0%
3gapA	1-129/139-208	7-129/130-206	100.0%
3pgk	1-185;403-415/200-392	1-188;404-415/189-403	99.7%
4gcr	1-83/84-174	1-82/83-174	99.4%
5fbpA	6-201/202-335	6-201;275-291;302-313/202-274;292-301;314-335	91.2%
8adh	1-175;319-374/176-318	1-177;318-374/178-317	99.2%
8atcA	1-137;288-310/144-283	1-147;287-310/148-286	98.7%
8atcB	8-97/101-152	8-100/101-153	100.0%
3 domains:			
1phh	1-175/176-290/291-394	1-72;97-180;269-344/73-96;181-268;345-394	undercut
3grs	18-157;294-364/158-293/365-478	18-64;104-161;290-364/65-103;162-289/365-478	89.8%
4 domains:			
1atnA	1-32;70-144;338-372/33-69/145-180;270-337/181-269	0-136;339-372/137-181;271-338/182-270	undercut
2pmgA	1-188/192-315/325-403/408-561	1-194;252-261/195-251;262-303;389-419/304-388;475-484/420-474;485-561	88.6%
8acn	2-200/201-317/320-513/538-754	2-99;120-209;231-315;515-540;572-582/100-119;210-230;316-514/541-571;583-754	undercut

Table 7: Results of the TC method using standard eigenvectors

Protein	Crystallographers	TC method	Agreement
2 domains:			
1ezm	1-134/135-298	1-81;98-134/82-97;135-298	94.6%
1fnr	19-161/162-314	19-314	undercut
1gpb	19-489/490-841	19-165;179-490;812-841/166-178;491-811	94.7%
1lap	1-150/171-484	1-155/156-484	100.0%
1pfkA	0-138;251-301/139-250;302-319	0-141;252-306/142-251;307-319	97.2%
1ppn	1-10;112-208/21-111;209-212	1-212	undercut
1rhd	1-158/159-293	1-157/158-293	99.7%
1sgt	22-123;234-245/129-233	16-121;236-245/122-235	98.2%
1vsgA	1-29;92-251/42-75;266-363	1-33;86-255/34-85;256-362	100.0%
1wsyB	9-52;86-204/53-85;205-393	17-45;91-189/3-16;46-90;190-394	90.9%
2cyp	3-145;266-294/164-265	2-144;263-294/145-262	98.5%
2had	1-155;230-310/156-229	1-310	undercut
3cd4	1-98/99-178	1-98/99-178	100.0%
3gapA	1-129/139-208	7-128/129-206	99.5%
3pgk	1-185;403-415/200-392	1-188;404-415/189-403	99.7%
4gcr	1-83/84-174	1-82/83-174	99.4%
5fbpA	6-201/202-335	6-201;275-291;302-313/202-274;292-301;314-335	91.2%
8adh	1-175;319-374/176-318	1-177;319-374/178-318	99.5%
8atcA	1-137;288-310/144-283	1-147;288-310/148-287	98.7%
8atcB	8-97/101-152	8-100/101-153	100.0%
3 domains:			
1phh	1-175/176-290/291-394	1-74;87-181;268-347;379-394/75-86;182-267;348-378	undercut
3grs	18-157;294-364/158-293/365-478	18-64;104-161;290-364/65-103;162-289/365-478	89.8%
4 domains:			
1atnA	1-32;70-144;338-372/33-69/145-180;270-337/181-269	0-136;339-372/137-181;271-338/182-270	undercut
2pmgA	1-188/192-315/325-403/408-561	1-193/194-303;389-419/304-388;475-484/420-474;485-561	90.6%
8acn	2-200/201-317/320-513/538-754	2-99;122-210;228-316;516-541;572-582/100-121;211-227;317-515/542-571;583-754	undercut

Table 8: Results of the TC method using generalized eigenvectors

Protein	Crystallographers	SCK method	Agreement
2 domains:			
1ezm	1-134/135-298	1-82;98-134/83-97;135-298	95.0%
1fnr	19-161/162-314	19-152/153-314	97.0%
1gpb	19-489/490-841	19-164;181-484;813-841/165-180;485-812	93.9%
1lap	1-150/171-484	1-162/163-484	100.0%
1pfkA	0-138;251-301/139-250;302-319	0-141;253-305/142-252;306-319	97.2%
1ppn	1-10;112-208/21-111;209-212	1-212	undercut
1rhd	1-158/159-293	1-156/157-293	99.3%
1sgt	22-123;234-245/129-233	16-121;236-245/122-235	98.2%
1vsgA	1-29;92-251/42-75;266-363	1-30;89-253/31-88;254-362	100.0%
1wsyB	9-52;86-204/53-85;205-393	3-53;83-204/54-82;205-394	99.0%
2cyp	3-145;266-294/164-265	2-145;255-294/146-254	96.0%
2had	1-155;230-310/156-229	1-310	undercut
3cd4	1-98/99-178	1-98/99-178	100.0%
3gapA	1-129/139-208	7-125/126-206	97.9%
3pgk	1-185;403-415/200-392	1-192;403-415/193-402	100.0%
4gcr	1-83/84-174	1-82/83-174	99.4%
5fbpA	6-201/202-335	6-200/201-335	99.7%
8adh	1-175;319-374/176-318	1-174;323-374/175-322	98.7%
8atcA	1-137;288-310/144-283	1-130;290-310/131-289	97.0%
8atcB	8-97/101-152	8-98/99-153	100.0%
3 domains:			
1phh	1-175/176-290/291-394	1-394	undercut
3grs	18-157;294-364/158-293/365-478	18-64;102-161;290-364/65-77;91-101;162-222;235-289/78-90;223-234;365-478	87.6%
4 domains:			
1atnA	1-32;70-144;338-372/33-69/145-180;270-337/181-269	0-137;337-372/138-182;269-336/183-268	undercut
2pmgA	1-188/192-315/325-403/408-561	1-561	undercut
8acn	2-200/201-317/320-513/538-754	2-99;121-210;231-316;516-540/100-120;211-230;317-515/541-754	undercut

Table 9: Results of the SCK method using generalized eigenvectors

Protein	Crystallographers	MFR method	Agreement
2 domains:			
1ezm	1-134/135-298	1-81;98-134/82-97;135-298	94.6%
1fnr	19-161/162-314	19-152/153-314	97.0%
1gpb	19-489/490-841	19-166;179-490;813-841/167-178;491-812	94.9%
1lap	1-150/171-484	1-158/159-484	100.0%
1pfkA	0-138;251-301/139-250;302-319	0-142;252-306/143-251;307-319	96.9%
1ppn	1-10;112-208/21-111;209-212	1-212	undercut
1rhd	1-158/159-293	1-157/158-293	99.7%
1sgt	22-123;234-245/129-233	16-121/122-245	93.6%
1vsgA	1-29;92-251/42-75;266-363	1-34;86-255/35-85;256-362	100.0%
1wsyB	9-52;86-204/53-85;205-393	18-45;91-189/3-17;46-90;190-394	90.6%
2cyp	3-145;266-294/164-265	2-144;266-294/145-265	99.6%
2had	1-155;230-310/156-229	1-310	undercut
3cd4	1-98/99-178	1-98/99-178	100.0%
3gapA	1-129/139-208	7-127/128-206	99.0%
3pgk	1-185;403-415/200-392	1-188;404-415/189-403	99.7%
4gcr	1-83/84-174	1-82/83-174	99.4%
5fbpA	6-201/202-335	6-201;275-291;302-313/202-274;292-301;314-335	91.2%
8adh	1-175;319-374/176-318	1-177;319-374/178-318	99.5%
8atcA	1-137;288-310/144-283	1-147;286-310/148-285	98.7%
8atcB	8-97/101-152	8-100/101-153	100.0%
3 domains:			
1phh	1-175/176-290/291-394	1-74;86-180;269-347;375-394/75-85;181-268;348-374	undercut
3grs	18-157;294-364/158-293/365-478	18-64;104-161;290-364/65-103;162-289/365-478	89.8%
4 domains:			
1atnA	1-32;70-144;338-372/33-69/145-180;270-337/181-269	0-136;339-372/137-182;268-338/183-267	undercut
2pmgA	1-188/192-315/325-403/408-561	1-194;252-261/195-251;262-303;389-419/304-388/420-561	90.5%
8acn	2-200/201-317/320-513/538-754	2-99;122-318;516-541;572-582/100-121;319-515/542-571;583-754	undercut

Table 10: Results of the MFR method using standard eigenvectors

Figure 5: Crystallographers' assignment for 1atnA

Incorrect decompositions due to stop criteria

The protein 1atnA consists of four domains according to the crystallographers. One of the domains has only 37 amino acids in it (amino acids 33-69) and cannot be found exactly by our methods since we use a minimum domain size of 40 as a stop criterion. All our methods decompose 1atnA into three domains. The TC method tries to decompose the protein into 0-33;69-136;339-372/34-68/137-181;271-338/182-270, but rejects the second domain because of its size. With a more sophisticated stop criterion which would allow a smaller domain in this case we would achieve an agreement of 96.5%. The SCK method and the MFR method try to decompose the protein into 0-28;73-137;337-372/29-72/138-182;269-336/183-268 and 0-28;96-136;339-372/29-95/137-182;268-338/183-267. In these cases the second domain is big enough, but has to be rejected because of a break of the β-sheet 8-12;16-21;29-32. More sophisticated stop criteria like the ones used by the GNM method could allow this break and we would achieve agreements of 94.9% and 88.7%. Figure 5 shows the assignment of the domains by crystallographers for the protein 1atnA. All protein pictures in this report are generated using Pymol [2] and show a cartoon representation. Different domains are colored with blue, red, green and grey, and some parts with incorrect assignments are in purple. Cyan is used for unassigned amino acids. Figure 6 depicts the β-sheet break by the SCK method and the MFR method. The incorrectly assigned β-strand is colored purple.

1ppn is a two-domain protein and is undercut into just one domain by all our methods. This incorrect assignment is due to the break of a β-sheet. This break also occurs in the crystallographers' assignment and we would need more sophisticated stop criteria to allow it. Then we would achieve agreements between 87.6% and 97.0%.

Figure 6: The β-sheet break of our assignment for 1atnA

Incorrect decompositions due to eigenvectors

For some proteins the approach using our protein graph definition and then using the Fiedler vector is not feasible since the domains are not separated in the vector. One of these proteins is the three-domain protein 1phh. Our methods undercut it into one or two domains with little agreement with the crystallographers' assignment. Figure 7 shows the Fiedler vector values of the domains of 1phh. It can easily be seen that no good decomposition is possible by our methods, which try to find a cut based on the Fiedler vector. Neither the GNM method nor DomainParser with their similar graph definitions decompose 1phh correctly, so maybe a totally different approach is needed for this protein.

The same arguments hold for the proteins 2had and 8acn, which could not be decomposed correctly by any method in this report. Figures 8 and 9 show their Fiedler vectors. It is possible to find domain 4 for protein 8acn, but Figure 10 shows that the remaining three domains cannot be found in their subgraph.

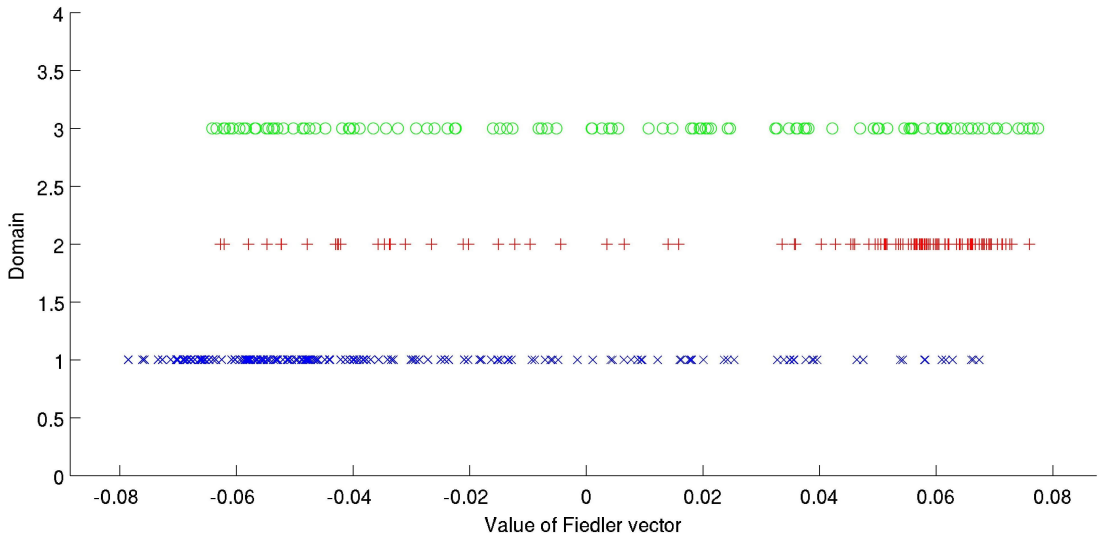

Figure 7: Fiedler vector of 1phh

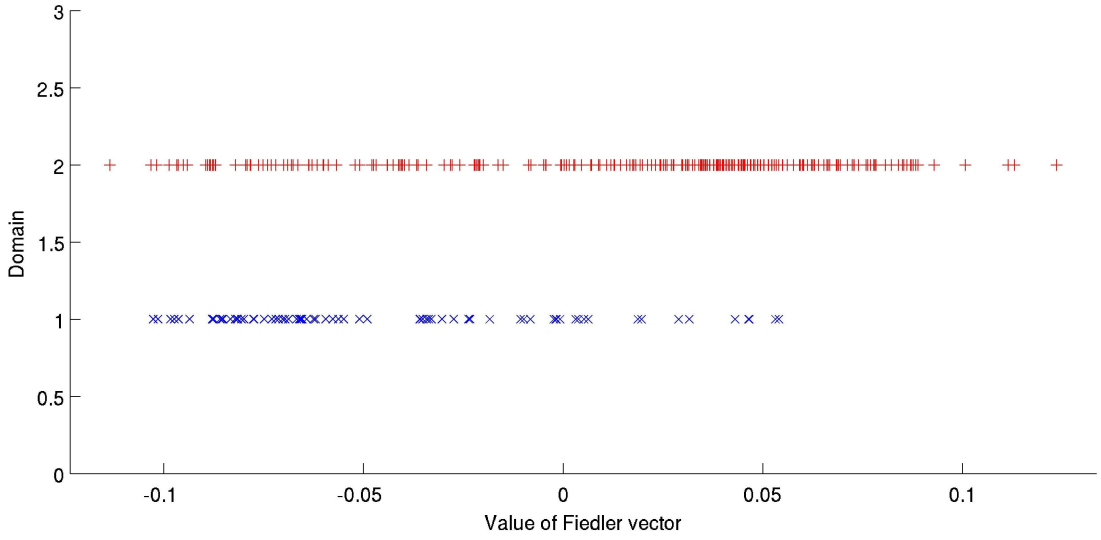

Figure 8: Fiedler vector of 2had

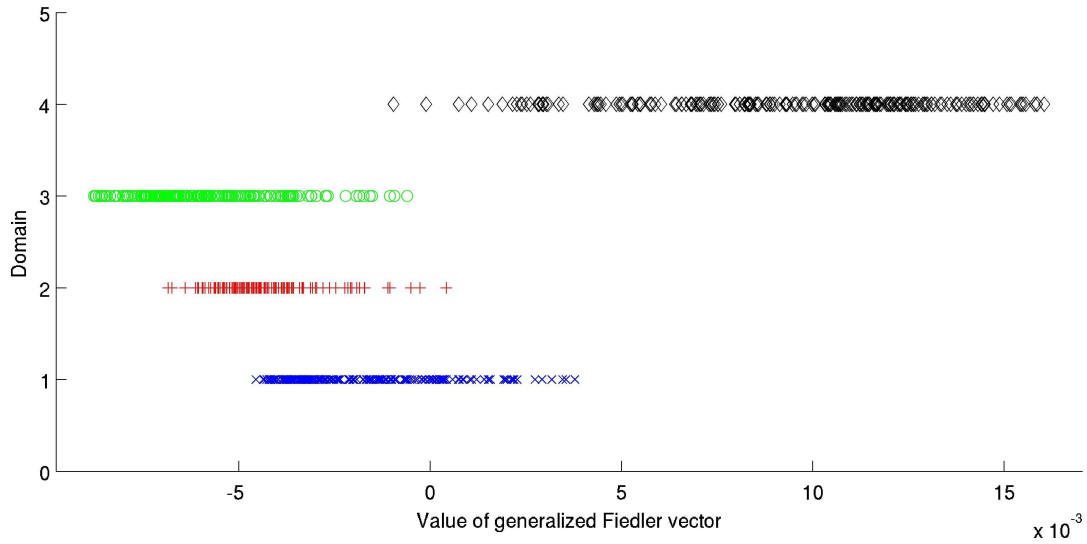

Figure 9: Generalized Fiedler vector of 8acn

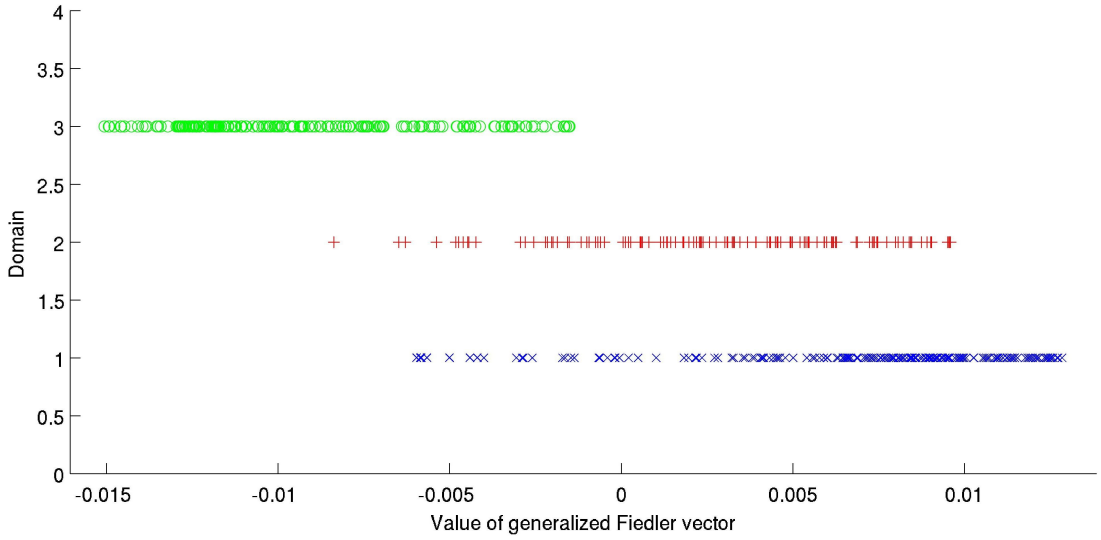

Figure 10: Generalized Fiedler vector of the subgraph of domains 1-3 of 8acn

Incorrect decompositions due to bad cuts

The last reason for incorrect decompositions is a *bad cut*, i. e. a cut which results in an incorrect decomposition whereas the domains are visible in the Fiedler vector and match our stop criteria. One of these bad cuts is the decomposition of 2pmgA by the SCK method. 2pmgA is a four-domain protein decomposed correctly by all our methods except for the SCK method, which undercuts it into one domain. This undercut is due to a β-sheet break caused by a bad cut. Figure 11 shows the crystallographers' assignment for 2pmgA, Figure 12 the bad cut of the SCK method. The blue part in this cut is too big compared to the grey domain 4 in the correct assignment. Figures 13 and 14 illustrate the same error. A better first cut like the one used by the MFR method is depicted in Figure 15.

1fnr is a two-domain protein undercut into one domain by the TC method with generalized eigenvectors. All other methods provide correct decompositions. Figure 16 and Figure 17 show the values of the standard and the generalized Fiedler vector of 1fnr. The TC method with standard eigenvectors uses a threshold value of -0.006. The TC method with generalized eigenvectors instead uses a threshold value of -0.015 and cuts the few points at the left side in Figure 17 apart. This cut is rejected because of the domain size stop criterion. The bad cut is colored purple in Figure 18.

The only protein overcut by one of our methods is 1ace. It is a single-domain protein overcut into two domains by the TC method with standard eigenvectors. The method uses a threshold value of 0.0341 for the Fiedler vector depicted in Figure 19. The resulting cut is shown in Figure 20. All other methods try a cut which breaks a β-sheet and is rejected, like the cut tried by the MFR method shown in Figure 21. As the cut found by the TC method with standard eigenvectors has one small part compared to the protein size (only 96 of 527 amino acids are in the second domain), an additional cut constraint like the β-balance in the MFR method could prevent this bad cut.

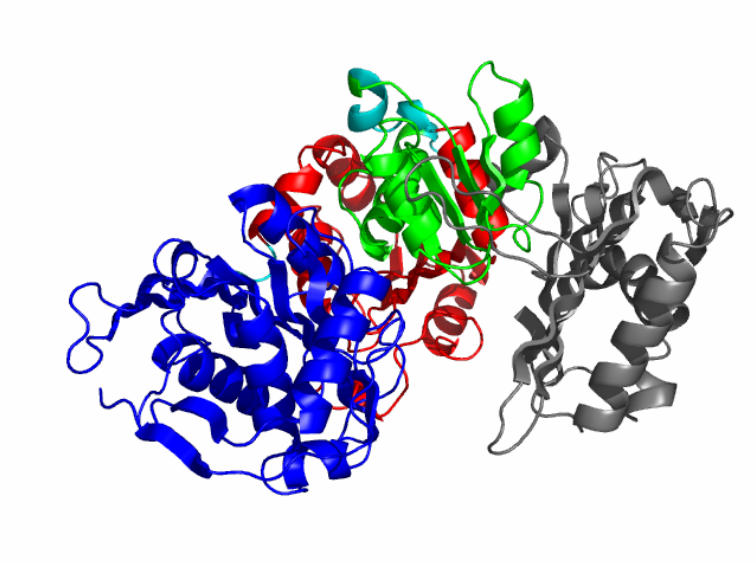

Figure 11: Crystallographers' assignment for 2pmgA

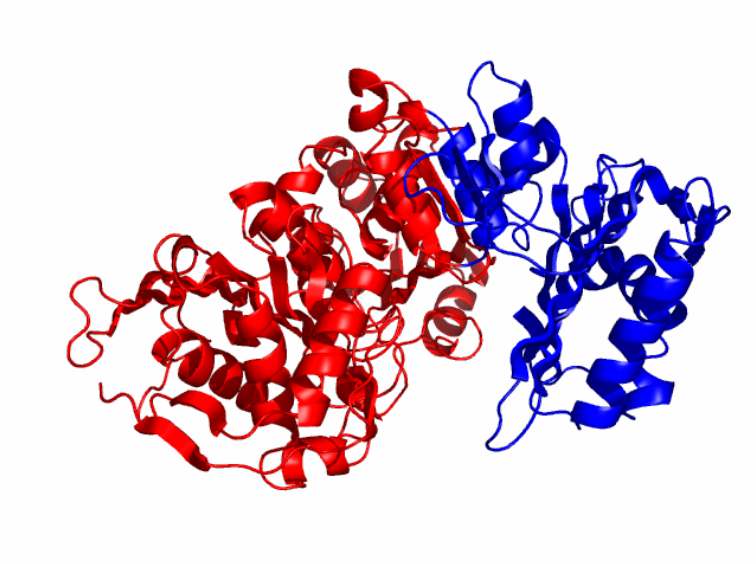

Figure 12: First cut of the SCK method of 2pmgA

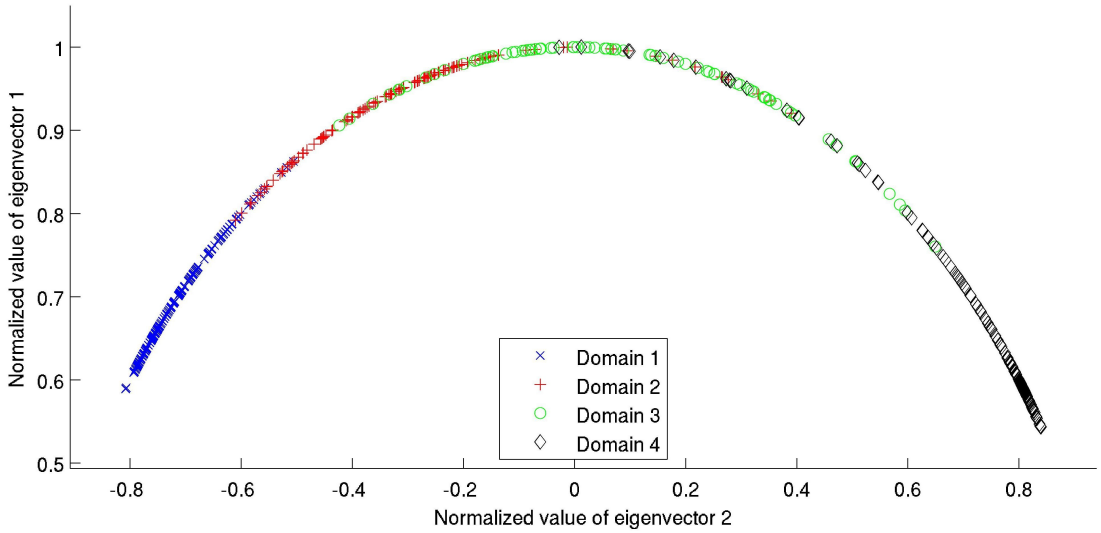

Figure 13: Normalized eigenvectors of 2pmgA with the crystallographers' assignment

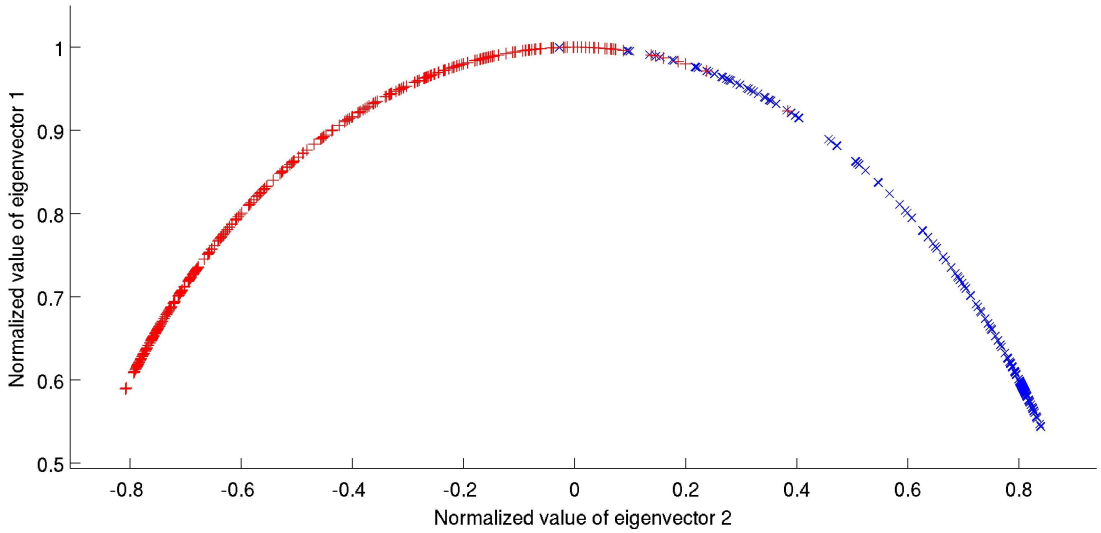

Figure 14: Normalized eigenvectors of 2pmgA with the first cut of the SCK method

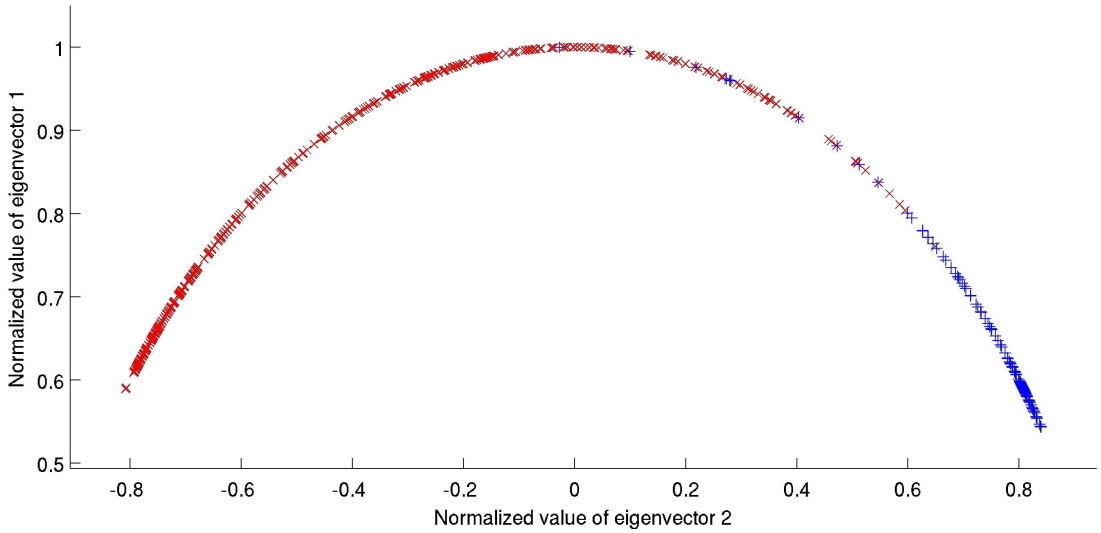

Figure 15: Normalized eigenvectors of 2pmgA with the first cut of the MFR method

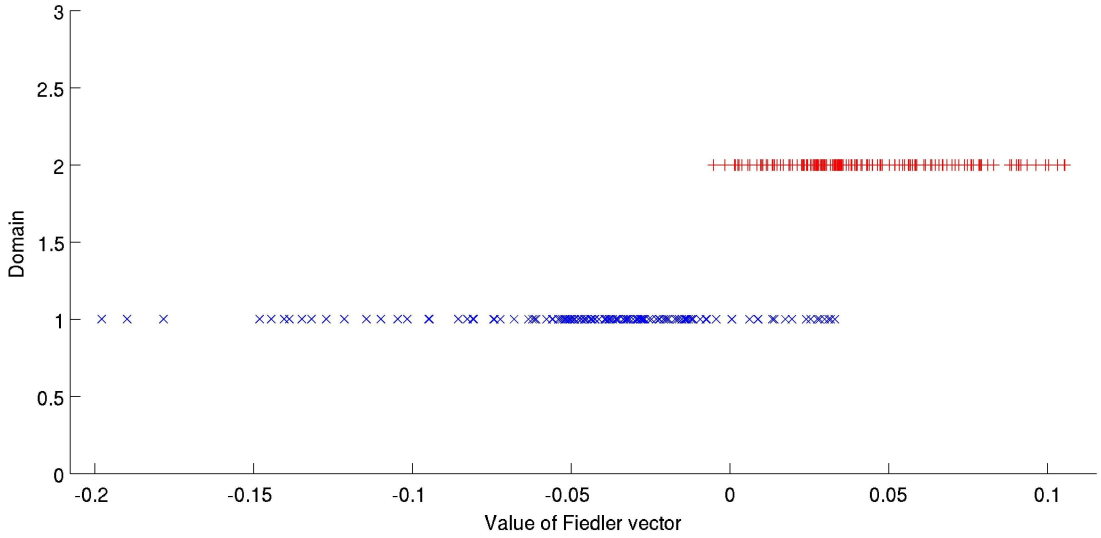

Figure 16: Fiedler vector of 1fnr

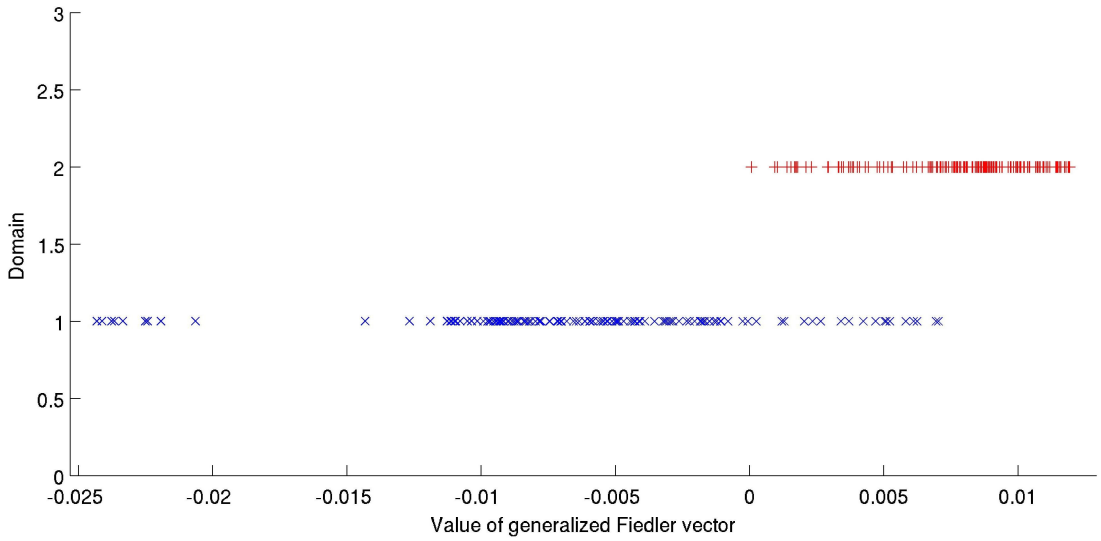

Figure 17: Generalized Fiedler vector of 1fnr

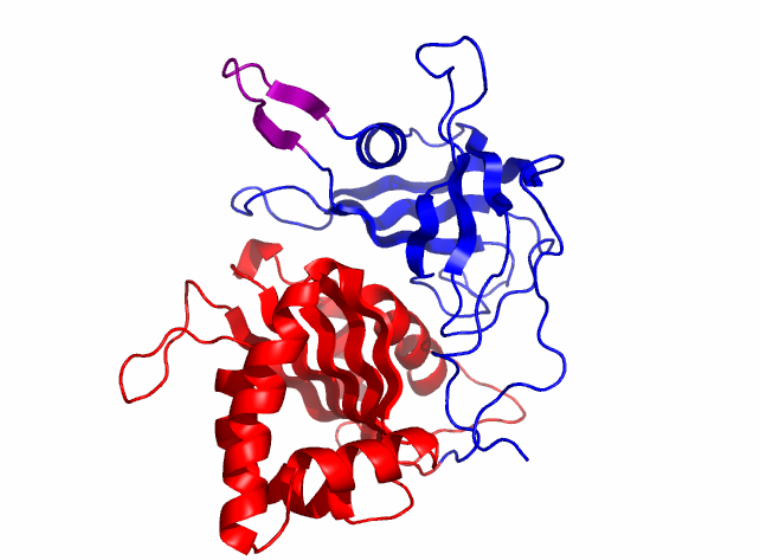

Figure 18: Cartoon representation of 1fnr

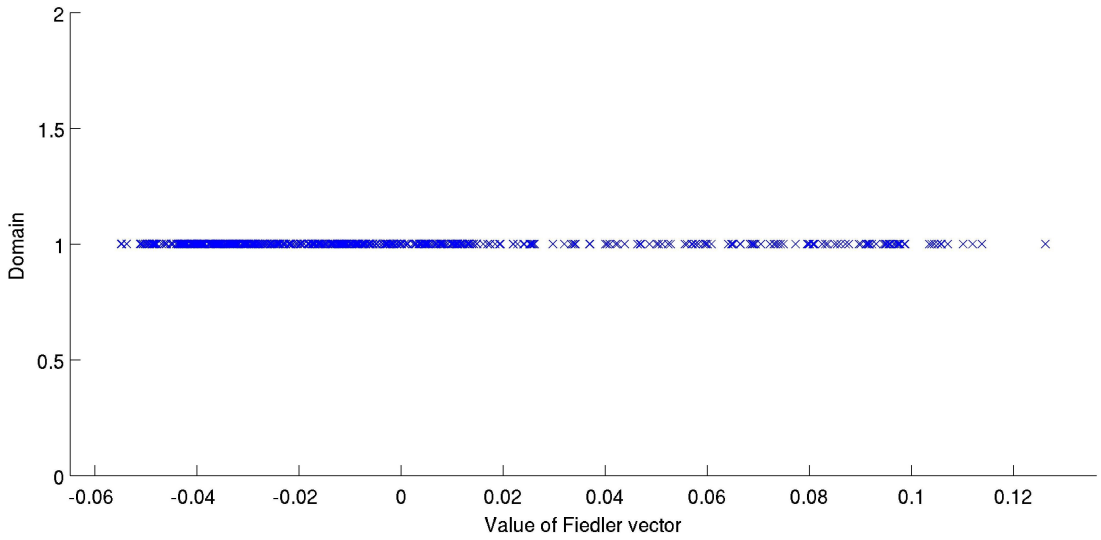

Figure 19: Fiedler vector of 1ace

Figure 20: The bad cut of 1ace by the TC method with standard eigenvectors

Figure 21: The cut of 1ace tried by the MFR method

6. Conclusion

Our methods introduced in this report correctly decompose 89.1% (the TC method and the SCK method) or 90.9% (the MFR method) of our test data set. Other methods achieved the same number of correct decompositions (the GNM method) or less (DomainParser).
Most of the proteins with more than two domains are decomposed incorrectly or with a low agreement with the crystallographers. As our test data set includes only five such proteins, we cannot say if there is a general problem with such proteins. A test on a data set with more proteins with many domains could answer this question. The analysis of our incorrect decompositions shows that some proteins (1phh, 2had, 8acn) cannot be correctly decomposed with spectral graph theory and our graph definition. All other incorrect decompositions could be prevented by more sophisticated stop criteria and cut constraints.
The MFR method performs best compared to our other methods. We think its advantages are the generation of β-balanced cuts and the direct use of the edge weights to find the cut. It just needs an additional weight of 9 for the β-strands. The SCK method and the TC method, which use the edge weights indirectly in the Fiedler vector and the *ncv* to find the cut, need additional weights of 99 and 49 to achieve good results. After all, our methods have very similar results for most of the proteins since they are based on the same graph theoretic ideas, and small differences in the cuts are often undone by the refinement step.

References

[1] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne. The Protein Data Bank. *Nucleic Acids Research*, 28:235–242, 2000.

[2] W. L. Delano. The pymol molecular graphics system on world wide web http://www.pymol.org. 2002.

[3] Susan Jones, Michael Stewart, Alex Michie, Mark B. Swindells, Christine Orengo, and Janet M. Thornton. Domain assignment for protein structures using a consensus approach: Characterization and analysis. *Protein Science*, 7:233–242, 1998.

[4] W. Kabsch and C. Sander. Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, 22(12):2577–2637, December 1983.

[5] Sibsankar Kundu, Dan C. Sorensen, and George N. Phillips, Jr. Automatic Domain Decomposition of Proteins by a Gaussian Network Model. *Proteins: Struct. Funct. Bioinf.*, 57:725–733, 2004.

[6] Kevin Lang. Fixing two weaknesses of the Spectral Method. In *NIPS*, pages 715–722, 2005.

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

[8] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On Spectral Clustering: Analysis and an algorithm. In *NIPS*, pages 849–856, 2001.

[9] C. A. Orengo, A. D. Michie, S. Jones, D. T. Jones, M. B. Swindells, and J. M. Thornton. CATH - a hierarchic classification of protein domain structures. *Structure*, 5:1093–1108, 1997.

[10] Jianbo Shi and Jitendra Malik. Normalized Cuts and Image Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905, 2000.

[11] Ying Xu, Dong Xu, and Harold N. Gabow. Protein domain decomposition using a graph-theoretic approach. *Bioinformatics*, 16(12):1091–1104, 2000.

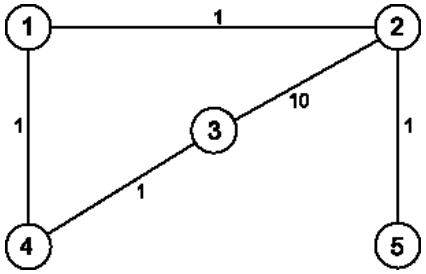

Figure 22: An example graph

A. Graph related matrices

This section contains definitions of useful matrices related to a graph.

Adjacency matrix

The Adjacency matrix A of a graph $G = (V, E, w)$ is defined as a $|V| \times |V|$ matrix where the entry a_{ij} is $w(\{i, j\})$ if there is an edge between node i and j, and 0 otherwise. The Adjacency matrix of the graph in Figure 22 is

$$\begin{matrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 10 & 0 & 1 \\ 0 & 10 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{matrix}$$

Degree matrix

The Degree matrix D of a graph $G = (V, E, w)$ is defined as a $|V| \times |V|$ diagonal matrix where the entry d_{ii} is the sum of the weights of all edges adjacent to node i. The Degree matrix of the graph in Figure 22 is

$$\begin{matrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 12 & 0 & 0 & 0 \\ 0 & 0 & 11 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{matrix}$$

Laplacian matrix

The Laplacian matrix L is defined as $D - A$. Its eigenvectors are useful to partition a graph as described in Appendix B. The Laplacian matrix of the graph in Figure 22 is

$$\begin{bmatrix} 2 & -1 & 0 & -1 & 0 \\ -1 & 12 & -10 & 0 & -1 \\ 0 & -10 & 11 & -1 & 0 \\ -1 & 0 & -1 & 2 & 0 \\ 0 & -1 & 0 & 0 & 1 \end{bmatrix}$$

B. Interpretation of graph eigenvectors

This section contains a physical interpretation of the eigenvectors of the Laplacian matrix.

A graph as a spring-mass system

A graph $G = (V, E, w)$ can be interpreted as a spring-mass system where every node is a mass and every edge a spring. The weight of an edge is the spring constant of the corresponding spring. Let m_i be the mass corresponding to node i, x_i the displacement of mass i from its rest position and k_{ij} the spring constant of the spring between mass i and mass j, or 0 if there is no spring between them. Then, according to Hooke's Law, the following equation holds:

$$m_i \times x_i'' \quad = \quad \sum_{j \in V} k_{ij}(x_j - x_i) \tag{3}$$

This leads to the next equation:

$$-x_i'' \quad = \quad -\sum_{j \in V} \frac{k_{ij}}{m_i} x_j + \left(\sum_{j \in V} \frac{k_{ij}}{m_i} \right) x_i \tag{4}$$

We can write the equations for all nodes in matrix form:

$$\begin{pmatrix} \sum_{j \in V} \frac{k_{1j}}{m_1} & \frac{-k_{12}}{m_1} & \frac{-k_{13}}{m_1} & \cdots \\ \frac{-k_{21}}{m_2} & \sum_{j \in V} \frac{k_{2j}}{m_2} & \frac{-k_{23}}{m_2} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ \vdots \end{pmatrix} \quad = \quad - \begin{pmatrix} x_1'' \\ x_2'' \\ \vdots \end{pmatrix}$$

or shorter as

$$K \times x \quad = \quad -x''$$

It can be shown that the eigenvectors of the matrix K are solutions for this equation

system and the square roots of the corresponding eigenvalues are the frequencies of the harmonic oscillations of the spring-mass system. The values of an eigenvector are the displacements of the masses for the corresponding harmonic oscillation. Oscillations with low frequencies are useful to find strongly connected parts, as the elements of such a part move together then. The vector $v = (c\ c\ \ldots\ c)$ is always an eigenvector of K with eigenvalue 0 for any constant c and represents a translation of all masses. It is the only eigenvector with eigenvalue 0 if the graph is connected, which is the case for our protein graphs.

Relation to the Laplacian matrix

When we set all masses to 1, then K is the same as the Laplacian matrix L of the graph. Another possibility is to set the masses to the degree of the corresponding node, i. e. to the sum of the weights of the adjacent edges. Then K is the same as $D^{-1}L$. We call the eigenvectors of L *standard eigenvectors* and the eigenvectors of $D^{-1}L$ *generalized eigenvectors* of the graph G. The eigenvector corresponding to the smallest positive eigenvalue of L is called *Fiedler vector*, the one of $D^{-1}L$ *generalized Fiedler vector*. It describes the slowest oscillation of the spring-mass system interpretation of the graph, as the smaller eigenvalues are all 0 and thus their eigenvectors are translations. Nodes with similar values in the Fiedler vector are likely strong connected and we can use it to cut a graph.