Computation and Visualization of Protein Topology Graphs Including Ligand Information

Tim Schäfer[1], Patrick May[2], and Ina Koch[1]

1 **Institute of Computer Science, Department of Molecular Bioinformatics, Johann Wolfgang Goethe-University Frankfurt (Main), Robert-Mayer-Straße 11–15, 60325 Frankfurt (Main), Germany,**
`ina.koch@bioinformatik.uni-frankfurt.de`

2 **Luxembourg Centre for Systems Biomedicine, University of Luxembourg, Campus Belval, 7 Avenue des Hauts-Fourneaux, L–4362 Esch-sur-Alzette, Luxembourg**

Abstract

Motivation: Ligand information is of great interest to understand protein function. Protein structure topology can be modeled as a graph with secondary structure elements as vertices and spatial contacts between them as edges. Meaningful representations of such graphs in 2D are required for the visual inspection, comparison and analysis of protein folds, but their automatic visualization is still challenging. We present an approach which solves this task, supports different graph types and can optionally include ligand contacts.

Results: Our method extends the field of protein structure description and visualization by including ligand information. It generates a mathematically unique representation and high-quality 2D plots of the secondary structure of a protein based on a protein-ligand graph. This graph is computed from 3D atom coordinates in PDB files and the corresponding SSE assignments of the DSSP algorithm. The related software supports different notations and allows a rapid visualization of protein structures. It can also export graphs in various standard file formats so they can be used with other software. Our approach visualizes ligands in relationship to protein structure topology and thus represents a useful tool for exploring protein structures.

Availability: The software is released under an open source license and available at `http://www.bioinformatik.uni-frankfurt.de/` in the *Software* section under *Visualization of Protein Ligand Graphs*.

1998 ACM Subject Classification J.3 Life and Medical Sciences

Keywords and phrases protein structure, graph theory, ligand, secondary structure, protein ligang graph

Digital Object Identifier 10.4230/OASIcs.GCB.2012.108

1 Introduction

Our knowledge of proteins expands rapidly with the high-throughput technologies and more and more 3D structures are available in databases like the RCSB Protein Data Bank (PDB, [1]). Thus, computational tools to automatically search, compare and classify protein structures are needed. Proteins are complex macromolecules and representing their structure in a way which allows for fast visual analysis requires significant simplification. This can be achieved by choosing the secondary structure level as an abstraction level and then drawing a 2D cartoon image of the protein. After the pioneering work of Jane Richardson, who first defined protein topology cartoons [23] and the first visualization methods, e.g. based on hydrogen bonds [10], it is not surprising that many secondary structure databases and

German Conference on Bioinformatics 2012 (GCB'12).
Editors: S. Böcker, F. Hufsky, K. Scheubert, J. Schleicher, S. Schuster; pp. 108–118

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

software tools to perform these tasks on all levels of protein structure exist today. The databases CATH [22], SCOP [21] and TOPS [19] all operate on the secondary structure level, i.e., abstracting from the atomic level they consider secondary structure elements (SSEs) and their relations in proteins to describe protein topology. Both CATH and SCOP split proteins into structural domains and then use a combination of automated and manual techniques to classify them into a hierarchic system that reflects structural as well as evolutionary relationships. In contrast, GRAST [7] and TOPS are fully automated methods. They all use graphs to represent protein topologies. GRATH abstracts SSEs as vectors and uses geometric relationships between all pairs of vectors to construct a graph of the protein structure. VAST [6, 16] also uses graph algorithms to detect similar spatial orientation and connectivity between SSEs.

Protein structure databases usually compare protein topologies based on atom coordinates from PDB files, but all methods listed above ignore the ligand atoms stored in those files. TOPS+ [28, 29] is the first method which includes ligand information in the comparison of proteins on the secondary structure level. It works on the domain level, uses a string-based description of protein secondary structure and includes additional biochemical and structural features like length of SSEs. String alignment methods are used to find similarities between domains.

The Protein Topology Graph Library (PTGL, [17, 18]) is a database of protein topologies that provides a web interface to compare and visualize protein folds based on SSEs. It uses a graph-based protein model related to earlier work by Koch *et al.* [12, 15, 14] and finds structural similarities between proteins by detecting maximum common substructures in their graph representations, allowing abstraction from the order of the SSEs in the amino acid sequence. This approach has recently been used to explore super-secondary structure patterns in proteins [13].

Images of protein topologies should contain information on the SSEs, their contacts and relative orientations and should still be clear and unequivocal. Automatically arranging the SSE symbols in the plane and fulfilling these constraints is not trivial, but some programs which address this task exist. TOPS cartoons [30] use circles and triangles to represent helices and strands. Pro-Origami [25] also draws protein cartoons and finds a balance between structural information and a clear layout. Additionally, it provides interactive diagrams and a search interface via a website. With the exception of TOPS+, all these programs consider α-helices and β-strands, but ignore ligands.

2 Approach

Our method works on the level of the secondary structure for protein chains. The 3D atom coordinates of all protein residues of a chain as well as the ligand molecules associated with the chain are parsed from PDB files. Each amino acid of the protein chain is assigned a SSE type by the DSSP algorithm [11]. After a pre-processing step which includes filtering of certain residues, the contacts between all remaining residues and ligand molecules are calculated. These residue level contacts are analyzed and SSE level contacts are defined according to specific rules. For each SSE pair which is in contact, we compute the spatial orientation. This leads to a contact matrix which can be used to create a *protein graph* (see Figure 1). These protein graphs are not necessarily connected and can thus be split into one or more connected components, which we call *folding graphs*. Both protein graphs and folding graphs can be visualized as 2D cartoons which allow for a rapid visual inspection and comparison.

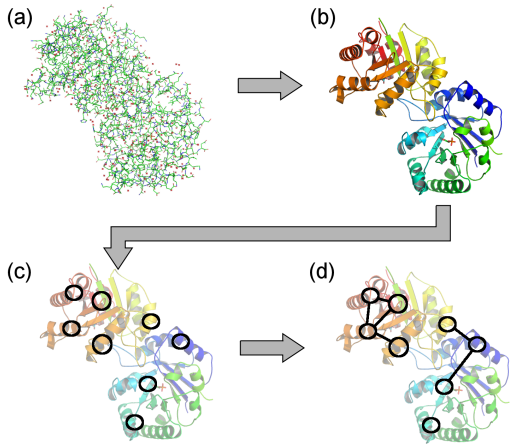

Figure 1 Computation of the protein graph from 3D atom data. Contacts are calculated on atom level from the 3D data in a PDB file (a). All residues of the considered protein chain are assigned to SSEs (b), which become the vertices of the protein graph (c). The atom contact information is used to determine the spatial relationships between the SSEs represented by edges in the graph (d).

3 Methods

3.1 Preprocessing and secondary structure assignment

Meta data, data on protein residues and atom coordinates is first parsed from the PDB file. During this step, PDB files which contain only very short protein chains with less than 30 residues or no protein chains at all are ignored. A list of all helices and strands of a protein chain in sequential order is created from the DSSP output according to the rules described below. Then all protein residues and ligands are assigned to one of the classes listed in Table 1.

Table 1 Description of the SSE classes used in protein ligand graphs.

SSE class	DSSP types	Description
H	H, G, I	residue which is part of an α-helix
E	E, B	residue which is part of a β-strand
C	S, T, none	residue which is not part of an α-helix or β-strand
L	n/a	ligand molecule
O	n/a	other non-protein molecules (solvent, polymers)

Solvent molecules like H_2O as well as polymers like DNA and RNA are filtered at this stage. Protein residues which are not part of a helix or strand are also ignored by default. This leads to a list of SSEs which is ordered by appearance in the primary structure from the N- to the C-terminus. An example for the residues with DSSP numbers 212–247 of chain A of PDB ID 7TIM is given below.

```
DSSP pos. | 212     220       230       240    247
Residue   | NGSNAVTFKDKADVDGFLVGGASLKPEFVDIINSRN
DSSP SSE  |  TTTGGGGTT TT  EEEESGGGGSTTHHHHHHTT
SSE class | CCCCHHHHCCCCCCCEEEECHHHHCCCHHHHHHCCC
```

Consecutive residues of the classes E or H form an SSE, which means that the example above contains four SSEs: a β-strand (227–230) and three α-helices (216–219, 232–235, 239–244). Each ligand is treated as a single SSE of type L and appended to the SSE list at the end of the chain. The contacts between the residues and ligands of the classes H, E and L are computed in the following step.

3.2 Computation of atom level and residue level contacts

A hard-sphere model is used to determine atom contacts: atoms are treated as hard spheres, and the atom coordinates defined in a PDB file are assigned to *collision spheres*. The default collision sphere radius r_{atom} is 2 Å for protein atoms and 3 Å for ligand atoms. Depending on the experiment which was used to generate the data, a PDB file may or may not contain data on hydrogen atoms, so we ignore them if they are listed. By definition, a contact between two atoms exists if their collision spheres overlap and a contact between two residues exists if the collision spheres of any of their atoms overlap.

According to its type, each atom can be assigned to one of the following classes: protein backbone atom, protein side chain atom or ligand atom. Thus, a contact between a pair of residues can be the result of multiple contacts of different types on the atom level. We distinguish the following contact types:

- backbone–backbone (BB) contacts
- backbone–side chain (BC) contacts
- side chain–side chain (CC) contacts
- ligand–backbone (LB) contacts
- ligand–side chain (LC) contacts
- ligand–ligand (LL) contacts
- ligand–non-ligand (LX) contacts, i.e., LX = LB || LC

For each residue pair which is in contact, the number of contacts of all these classes is saved for SSE contact determination.

To speed up the calculation, collision spheres are also assigned to residues by determining the central atom c and the maximal distance d_{max} between c and any other atom of the same residue. For protein residues, the C_α-atom is considered to be the residue center. This rule cannot be applied to ligand molecules though, so we choose the atom with minimal maximal distance to all other atoms of the molecule in this case. Thus, the collision sphere for a residue has the radius $r_{res} = d_{max} + r_{atom}$. If the collision spheres of two residues do not overlap, no contact is possible and the atom level comparison can be omitted.

Furthermore, multiple consecutive residues may be skipped entirely under certain circumstances (see Figure 2): if two residues with DSSP numbers n and m are very far from each other, the residues n and $m+1$ are far from each other as well, and it may be possible to skip the residues $m+1, m+2, \ldots, m+k$. The maximum distance between the central atoms of consecutive residues in the chain, $dist_{max}$, the maximum collision sphere radius of all residues, and the distance $dist_{n,m}$ between the central atoms of the residues n and m are required to determine k, the number of residues that can be skipped. Integer division[1] is used to compute $k = dist_{n,m} \backslash dist_{max}$ and skip residues during the pairwise contact calculation.

[1] Integer division can be defined as $a \backslash b \equiv \lfloor a/b \rfloor$.

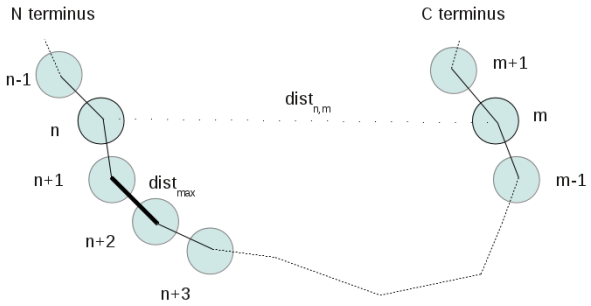

Figure 2 Residue skipping during the determination of residue level contacts. The maximum distance between the central atoms of consecutive residues in the chain is $dist_{max}$, and $dist_{n,m}$ is the distance between the central atoms of the residues n and m. Then, $dist_{n,m+1}$ is at least $dist_{n,m} - dist_{max}$.

3.3 Computation of SSE level contacts and relative orientation

We apply a rule set to determine whether two SSEs are in contact. This is the case if enough atom level contacts exist between residues of the considered pair. Determining the number of atom level contacts that is considered to be sufficient for an SSE contact is a difficult and time-consuming task though, because different settings have to be tested on a number of proteins and their results have to be verified manually by visual inspection using a viewer program like PyMol [24]. The rules depend on the type of both SSEs, see Table 2.

Table 2 Rules for the determination of SSE contacts. The class X represents an arbitrary SSE class, i.e., X = E || H || L.

SSE 1	SSE 2	Required contacts
E	E	$BB > 1 \,\|\|\, CC > 2$
H	E	$(BB > 1 \,\&\&\, BC > 3) \,\|\|\, CC > 3$
H	H	$BC > 3 \,\|\|\, CC > 3$
L	X	$LX >= 1$

Since a polypeptide chain has a direction, multiple spatial relationships can be defined between two adjacent sections: they can be *parallel*, *anti-parallel* or something in between, which we call *mixed* orientation. Each edge in the protein graph is labeled with the spatial relation of the SSEs represented by its vertices, which is computed as follows:

Let u and v be two spatially adjacent, non-ligand SSEs of a protein chain. Let A and B be the sets of the DSSP residue numbers of the SSEs u and v respectively. Let S be the set of all sums of the DSSP residue numbers of the residue pairs (a_i, b_j) with $a_i \in A$ and $b_j \in B$ which form a contact and D the set of all differences of these pairs. The double difference is then defined as $DD = (S_{max} - S_{min}) - (D_{max} - D_{min})$. The spatial relation between the SSEs u and v is determined from DD using four thresholds for which $T_{antip} < T_{mixedLower} < 0 < T_{mixedUpper} < T_{parallel}$ holds. By definition, the SSEs are (1) mixed, if $T_{mixedLower} < DD < T_{mixedUpper}$, (2) parallel, if $DD > T_{parallel}$ and (3) anti-parallel, if $DD < T_{antip}$. If one of them is a ligand and thus has no direction, the contact type is defined as (4) *ligand*.

During the computation of contacts between the n SSEs of a protein chain and the relative orientation of all SSE pairs which are in contact, an orientation matrix M of size n^2

is generated. At position (i, j), the matrix is 0 if the SSEs i and j are not in contact, 1 if they are anti-parallel, 2 if they are mixed, 3 if they are parallel and 4 if one of them is a ligand.

3.4 Computation of the different protein graph types

The orientation matrix M can be interpreted as the adjacency matrix of an undirected protein graph $G = (V, E)$ with the vertex set V and the edge set E. Each vertex v of the graph represents an SSE and each edge $e = (v_i, v_j)$ a contact between the vertices v_i and v_j it connects. An edge $e = (v_i, v_j)$ is added to E if and only if $M_{i,j} > 0$.

Depending on what the user is interested in, a subset of the SSE types can be ignored when creating the graph. For example, a protein graph which only consists of helices and the contacts between them is called an *alpha-graph*. We support the following graph types:

- the **alpha-graph** based on the α-helices (H),
- the **beta-graph** consisting of all β-strands (E),
- the **albe-graph** with all SSEs of types E and H,
- the **alphalig-graph** consisting of all SSEs of types H and L (ligands),
- the **betalig-graph** with all SSEs of types E and L,
- the **albelig-graph** based on all β-strands, helices, and ligands.

Note that two consecutive SSEs in the SSE list may be separated by a variable region in the amino acid sequence and thus an SSE is not necessarily spatially adjacent to its primary structure predecessor and successor in the protein graph. This means that protein graphs are not necessarily connected in the graph-theoretic sense, i.e., protein graphs may consist of several connected components. We call these connected components the *folding graphs* of a protein graph. They can be computed using the algorithm described in [9] and drawn separately. Optionally, folding graphs under a certain size can be ignored during this process. It turned out that many protein graphs consist of one large connected component and several isolated vertices which often represent helices on the protein surface.

3.5 Visualization of protein graphs and folding graphs

Both protein and folding graphs can be visualized in the same way. The vertices are ordered and labeled by the position of the SSE they represent in the amino acid sequence (S) and by their position in the graph (G). Helices are shown as red, filled circles, strands are represented by black, filled squares, and ligands by magenta rings. The contacts are drawn as arcs, and their color encodes the spatial relation between the two SSEs (red $\rightarrow$ parallel; blue $\rightarrow$ anti-parallel; green $\rightarrow$ mixed; magenta $\rightarrow$ ligand contact). The protein graph of triosephosphate isomerase is depicted in Figure 3 as an example.
To visualize a graph $G = (V, E)$ consisting of the n vertices $v_0, v_1, \ldots, v_{n-1}$, the visualization function first computes the required image dimensions from the number of vertices. Ignoring the page margins of fixed size, the drawing area can be split into three parts vertically:

1. The **header section** is used to print information on the depicted graph, including the graph type, PDB ID and chain ID. Its width w_h is determined from the font size and the number of characters of the printed string. The header section is not shown in the figures of this paper.

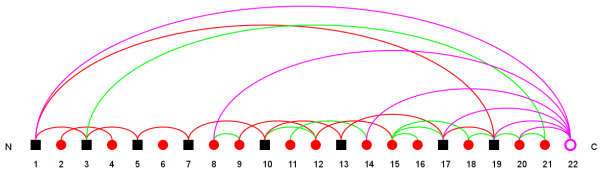

Figure 3 The albelig-graph of the β-chain of triosephosphate isomerase. The α-helices are shown as red, filled circles and the β-strands as black, filled squares. Ligands are represented by magenta circles. From left to right, the vertices are labeled by their position in the amino acid sequence. The arcs mark spatial contacts (red for parallel; blue for anti-parallel; green for mixed; magenta for ligand contact).

2. The **graph section** contains the visualization of the protein graph. Its width w_g is $|V| * dist_v + 2 * r$, where $dist_v$ is the distance of adjacent vertices in the visualization ($dist_v$ is fixed and set to 50 pixels by default) and r is the vertex radius. Its height h_g is $h_{max} + r$, where h_{max} is the height of the largest arc that could possibly occur in the image, the one between vertices v_0 and v_{n-1}.
3. The **footer section** is used to label the vertices from N- to C-terminus. The labels are written directly underneath the vertices, so the width of the footer also is w_g.

The total width of the image thus is $margin_{left} + max(w_h, w_g) + margin_{right}$ and its height is $margin_{top} + h_h + h_g + h_f + margin_{bottom}$.

Once the canvas has been prepared, the header and footer are printed and the graph is depicted. Before drawing a vertex or edge, the color is set based on the vertex or edge type. The position (v_x^1, v_y^1) of the first vertex in the list is determined and all other vertices are drawn relative to it, i.e., the n^{th} vertex is drawn at position $(v_x^n, v_y^n) = (v_x^1 + (n-1) * dist_v, v_y^1)$.

An arc is then used to visualize contacts between a pair of SSEs. The required values for an arc connecting a vertex pair (n, m) can be computed from the position of these vertices. All edges are drawn and the resulting image is written to the output directory in in pixel-based (PNG or JPG) and vector-based (SVG) formats. Additionally, a text file representing the protein graph is saved to the same directory in our own PLG file format, GML format [8], GraphML format [26], and DOT language format [27]. These text files can be used to get more detailed information on the SSEs shown in the image, i.e., their length and amino acid sequence. They can also be opened in other graph editing and visualization software like GraphViz [3]. Furthermore, our software can read PLG files and visualize them directly without having to re-compute any SSE data. In that case, no PDB and DSSP files are required.

The software comes with a graphical user interface but can also be used on the command line, which is useful in batch mode. Options exist to write the results to a database and to include coiled regions in the protein graphs. Our software also includes an option to filter ligands based on their atom count.

4 Discussion

We have applied our method to a copy of the PDB retrieved on May 2011 and saved the results in a database. The database contains entries on 139,923 protein chains stored in 60,880 PDB files, the average PDB file in the database thus contains 2.3 chains. These proteins contain a total of 1,165,616 α-helices, 1,322,639 β-strands and 357,311 ligands. The average SSE consists of about 7 residues, and in general helices are longer than β-strands.

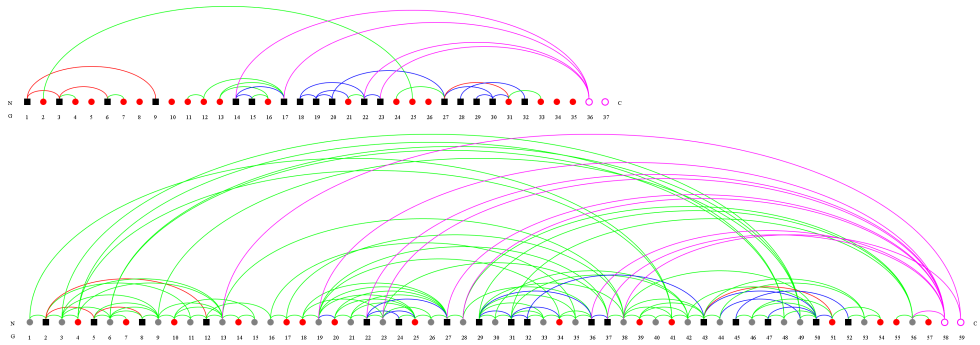

Figure 4 Coiled regions in protein graphs. **(a)** The protein graph of chain A of biotin carboxylase, PDB identifier 2W70. The chloride ion (SSE #37) is isolated because it only has contacts with residues in coiled regions. **(b)** A modified version of the protein graph which includes coiled regions as a new SSE type (shown as gray, filled circles). The contacts between the chloride ion (SSE #59) and protein residues in two coiled regions of the chain (SSE #13 and #38), become visible in this representation.

Note that not all currently available PDB files are included in the database. Many of the ignored files contain only RNA or DNA, others contain very short polypeptides or cannot be processed properly by DSSP.

More than 2,500,000 contacts between SSEs and about 400,000 ligand contacts were detected in the data set. This means that the average α-helix or β-strand has 1.0 contacts to other SSEs of the protein while the average ligand is in contact with 1.1 SSEs or other ligands. These numbers include 319,962 SSEs and 133,216 ligands that do not have any SSE contacts at all, i.e., they are isolated vertices in the protein graph. The higher percentage of isolated ligands compared to SSEs may be related to the fact that ligands usually are small molecules bound to the surface area of a protein. Note that some ligands like Mg^{2+} only consist of a single atom. This means they cannot be surrounded by a large number of SSEs like an α-helix or β-strand deep within the protein core. Additionally, the majority of contacts to coiled regions are ignored by our method, because coiled regions are not made up of regular spatial patterns.

The phosphate ion PO4-147, which is associated with chain D of hemoglobin in the PDB structure 2HHB [5], is such an example: with a collision sphere setting of 2 Å, it is in contact only with VAL-1, but this residue is part of a coiled region. A situation like this is not directly related to the contact computation algorithm, but rather is a consequence of the protein model. A possible solution could be to increase the collision sphere radius of ligand atoms. Visual inspection did indeed show that in many cases, only a subset of the residues forming the binding pocket overlaps with the collision sphere of small ligands. Therefore the default collision sphere radius of ligand atoms was increased to 3 Å. An alternative solution could be to include coiled regions as a new SSE type and to consider all protein residues which are neither an α-helix nor a β-sheet to be part of a coiled region. This would mean that *all* protein residues are represented by an SSE in the protein graph and ligands which only have contacts to coiled regions are not longer isolated.

The large protein graph for chain A of PDB entry 2W70 [20] is depicted in Figure 4, with and without consideration of coiled regions. In the graph which includes coiled regions, the last ligand, the chloride ion CL-1448 is not longer represented by an isolated vertex, but the number of vertices and edges in the graph is dramatically increased. A problem is that

allowing coiled regions as SSEs may split other SSEs into multiple parts. Another approach would be to determine the SSE which is spatially closest to such ligands and introduce a new contact type with a new spatial relation "close to".

5 Conclusion

In the following we demonstrate the results of our approach using the structure of triosephosphate isomerase (TIM) with PDB identifier 7TIM [2]. TIM is a glycolytic enzyme which catalyzes the interconversion of the three-carbon sugars dihydroxyacetone phosphate and D-glyceraldehyde 3-phosphate. Each of its two chains consists of a central parallel eight-strand β-barrel surrounded by α-helices. The albelig-graph consists of 22 SSEs and is depicted in Figure 3. The betalig-graph makes it easier to spot the β-barrel (see Figure 5). The ligand, phosphoglycolohydroxamic acid, has contacts to several β-strands and α-helices.

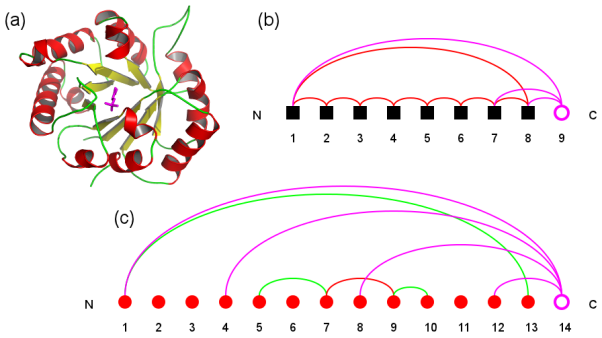

Figure 5 Examples for different types of protein graphs and the structure of triosephosphate isomerase (a). The betalig-graph (b) and the alphalig-graph (c) of the β-chain of triosephosphate isomerase, PDB identifier 7TIM. The α-helices are shown as red, filled circles and the β-strands as black, filled squares. Ligands are represented by magenta circles. From left to right, the vertices are ordered by their position in the amino acid sequence. The arcs mark spatial contacts (red, parallel; blue, anti-parallel; green, mixed; magenta, ligand contact). Note that the β-barrel is clearly visible in the betalig-graph as a series of parallel β-strands.

We have presented a new method to describe and visualize the structure of proteins on the secondary structure level including ligand information. This method is based on a unique graph-theoretic description of protein structure topology. In this paper, we explain the description and how ligand information is considered in the graph representation as well as in the topology cartoons. The visualization of these cartoons is described in detail using examples for illustration.

The related software is baded on PDB structures and DSSP files and produces high quality cartoons which include ligand data. These cartoons can be used for exploring protein structure topology with ligand information. Additionally, the graphs can be stored in a database or exported in various standard graph file formats for usage in other software tools.

The protein model defined by the software has been evaluated on the atom and SSE levels. Limits of both the input data and our method have been discussed. We demonstrate that our approach produces a clear, unique and meaningful representation of protein structure topology. The description as well as the visualization can be applied in proteomics, drug design, medicine and biology. Our software can also be used to extend the existing PTGL [18] database by adding ligand information. The protein graphs could be used for similarity searching within the database using graph-based or string-based methods. A web server

with links to other databases, including sequence and pathway databases as well as ligand databases like the PDB Ligand Expo [4], is planned.

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 Acid Research*, 28:235–242, 2000.

2 R.C. Davenport, P.A. Bash, B.A. Seaton, M. Karplus, G.A. Petsko, and D. Ringe. Structure of the triosephosphate isomerase-phosphoglycolohydroxamate complex: an analogue of the intermediate on the reaction pathway. *Biochemistry*, 30:5821–6, 1991.

3 J. Ellson, E.R. Gansner, E. Koutsofios, S.C.> North, and G. Woodhull. Graphviz and dynagraph – static and dynamic graph drawing tools. In M. Junger and P. Mutzel, editors, *Graph Drawing Software*, pages 127–148. Springer-Verlag, 2004.

4 Z. Feng, L. Chen, H. Maddula, O. Akcan, R. Oughtred, H.M. Berman, and J. Westbrook. Ligand depot: a data warehouse for ligands bound to macromolecules. *Bioinformatics*, 20:2153–5, 2004.

5 G. Fermi, M.F. Perutz, B. Shaanan, and R. Fourme. The crystal structure of human deoxyhaemoglobin at 1.74 Å resolution. *Journal of Molecular Biology*, 175:159–174, 1988.

6 J.F. Gibrat, T. Madej, and S.H. Bryant. Surprising similarities in structure comparison. *Curr Opin Struct Biol.*, 6:377–85, 1996.

7 A. Harrison, F. Pearl, I. Silitoe, T. Slidel, and R. Mott. Recognizing the fold of a protein structure. *Bioinformatics*, 19:1748–1759, 2003.

8 Michael Himsolt. GML: A portable graph file format. Technical report, 1996.

9 John Hopcroft and Robert Tarjan. Algorithm 447: efficient algorithms for graph manipulation. *Communications of the ACM*, 16:372–378, 1973.

10 E.G. Hutchinson and J.M. Thornton. HERA - a program to draw schematic diagrams of protein secondary structures. *PROTEINS: Structure, Function, and Genetics*, 8:203–212, 1990.

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

12 I. Koch, F. Kaden, and J. Selbig. Analysis of protein sheet topologies by graph-theoretical methods. *PROTEINS: Structure, Function, and Genetics*, 12:314–323, 1992.

13 I. Koch, A. Kreuchwig, and P. May. *Protein supersecondary structure*, chapter Hierarchical representation of super-secondary structures using a graph-theoretical approach. Humana Press, New York, 2012. In press.

14 I. Koch and T. Lengauer. Detection of distant structural similarities in a set of proteins using a fast graph-based method. In T. Gaasterland, P. Karp, K. Karplus, C. Ouzounis, C. Sander, and A. Valencia, editors, *Proceedings of the Fifth International Conference on Intelligent Systems for Molecular Biology*, pages 167–178, 1997.

15 I. Koch, T. Lengauer, and E. Wanke. An algorithm for finding maximal common subtopologies in a set of protein structures. *Journal of Computational Biology*, 3:289–306, 1996.

16 T. Madej, J.-F. Gibrat, and S.H. Bryant. Threading a database of protein cores. *PROTEINS: Structure, Function, and Genetics*, 23:356–369, 1995.

17 P. May, S. Barthel, and I. Koch. PTGL - protein topology graph library. *Bioinformatics*, 20:3277–3279, 2004.

18 P. May, A. Kreuchwig, T. Steinke, and I. Koch. PTGL: a database for secondary structure-based protein topologies. *Nucleic Acid Research*, 38:D326–D330, 2010.

19 I. Michalopoulos, G.M. Torrance, D.R. Gilbert, and D.R. Westhead. Tops: an enhanced database of protein structural topology. *Nucleic Acids Research*, 32:D251–D254, 2004.

20 I. Mochalkin, J.R. Miller, L.S. Narasimhan, V. Thanabal, P. Erdman, P. Cox, J.V. Prasad, S. Lightle, M. Huband, and K. Stover. Discovery of antibacterial biotin carboxylase inhibitors by virtual screening and fragment-based approaches. *Acs Chem.Biol.*, 4:473–483, 2009.

21 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. *Journal of Molecular Biology*, 247:536–540, 1995.

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

23 J.S. Richardson. β-sheet topology and the relatedness of proteins. *Nature*, 268:495–500, 1977.

24 Schrödinger, LLC. The PyMOL molecular graphics system, version 1.3r1. August 2010.

25 A. Stivala, M. Wybrow, A. Wirth, J. Whisstock, and P. Stuckey. Automatic generation of protein structure cartoons with Pro-origami. *Bioinformatics*, 27:3315–3316, 2011.

26 The GraphML Team. The GraphML file format. `http://graphml.graphdrawing.org/`.

27 The GraphViz Team. The DOT language. `http://www.graphviz.org/content/dot-language/`.

28 M. Veeramalai. *A Novel Method for Comparing Topological Models of Protein Structures Enhanced with Ligand Information.* PhD thesis, University of Glasgow, 2005.

29 M. Veeramalai and D. Gilbert. A novel method for comparing topological models of protein structures enhanced with ligand information. *Bioinformatics*, 24:2698–2705, 2008.

30 D.R. Westhead, T.W.F. Slidel, T.P.J. Flores, and J.M. Thornton. Protein structural topology: Automated analysis and diagrammatic representation. *Protein Science*, 8:897–904, 1999.