ELSEVIER

Schematic representation of residue-based protein context-dependent data: An application to transmembrane proteins

F. Campagne and H. Weinstein

Department of Physiology and Biophysics, Mount Sinai School of Medicine, New York, New York, USA

An algorithmic method for drawing residue-based schematic diagrams of proteins on a 2D page is presented and illustrated. The method allows the creation of rendering engines dedicated to a given family of sequences, or fold. The initial implementation provides an engine that can produce a 2D diagram representing secondary structure for any transmembrane protein sequence. We present the details of the strategy for automating the drawing of these diagrams. The most important part of this strategy is the development of an algorithm for laying out residues of a loop that connects to arbitrary points of a 2D plane. As implemented, this algorithm is suitable for real-time modification of the loop layout. This work is of interest for the representation and analysis of data from (1) protein databases, (2) mutagenesis results, or (3) various kinds of protein context-dependent annotations or data. © 2000 by Elsevier Science Inc.

Keywords: schematic representation of proteins, transmembrane proteins

INTRODUCTION

We present here an automated method to build representations of residue-based, schematic diagrams of proteins. These diagrams are already familiar to biochemists, biologists, and modelers as they are frequently used to represent properties of sets of residues in the context of a sequence and a secondary structure hypothesis. They consist of a representation of the residues of a protein on a two-dimensional page, as shown in Figure 1.

The manual construction of this kind of diagrams is error prone and time consuming. For instance, the task of building a diagram with classic vectorial drawing programs for a protein of about 600 residues, may require more than 4 h, depending on the experience of the person who is volunteered to complete the tedious task. This time must be reinvested for each new sequence. Considering these factors, it seems plausible that these representations are not used as often as their intrinsic interest would demand. In a previous work[1], a method has been described to build algorithmically this kind of diagram for G protein-coupled receptor sequences and some other transmembrane proteins, given the sequence and the positions of transmembrane helices. The Viseur tool, which implements the method, has been extensively used by a number of laboratories worldwide and by GPCR databases for presentation of their data[2]. We capitalize here on the experience gained from the Viseur project and discuss the issues involved in generalizing the algorithmic construction of residue-based diagrams of any kind of protein. The most important issues can be summarized as follows:

1. The diagram can be split into regions corresponding to the secondary structure elements and the residues that connect them. Disulfide bridges are indicated in the diagram by a line joining the designated residues.
2. Each element represents the residues of the sequence in the context of a low-level model that highlights some properties considered to be important. For instance, the representation can have the regularity of a helical arrangement and build on the relation between the residue at position i and the residues $i + 4$ and $i + 3$. Such a diagram can visualize secondary structure predictions or hypotheses, but there is no unique presentation for a given secondary structure element. The choice of the presentation depends on the intention of the display.
3. The residue representation can display properties that can be identified using colors and other signs. This can be used to show residues of a sequence that had been mutated, to

Corresponding author: F. Campagne, Department of Physiology and Biophysics, Box 1218, Mount Sinai School of Medicine, One Gustave L. Levy Place, New York, NY 10029-6574.

E-mail address: campagne@inka.mssm.edu (F. Campagne)

Journal of Molecular Graphics and Modelling 17, 207–213, 1999

655 Avenue of the Americas, New York, NY 10010

1093-3263/99/$–see front matter
PIIS1093-3263(99)00032-7

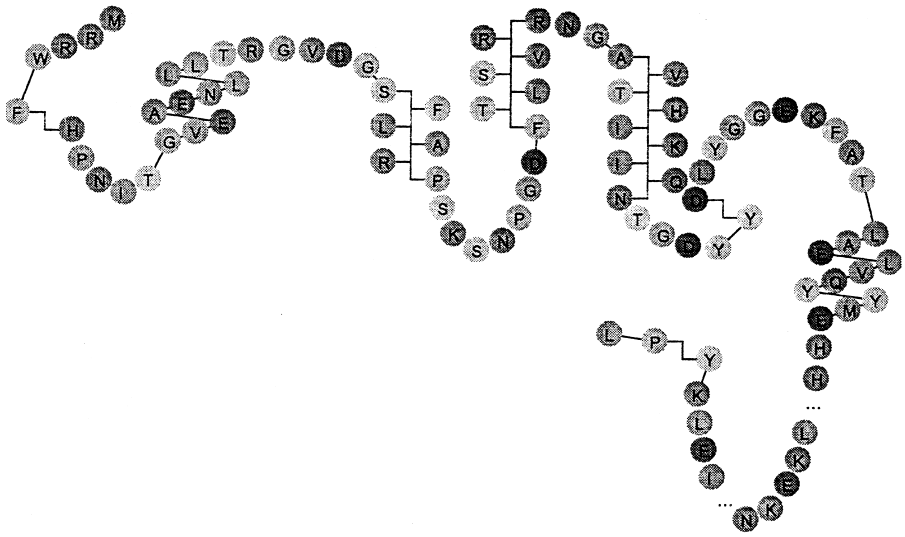


Figure 1. SH2 domain residue-based diagram.

display hypotheses regarding the involvement of certain residues in the binding mechanism of some ligands, etc.

4. To a first approximation, the relative position of the secondary structure elements in the 2D representation can be considered a common property of a family of sequences. For instance, GPCRs are usually represented as a row of transmembrane helices connected by loop regions. The organization of the secondary structure elements on the page depends on the amount of knowledge one wants to express about the supposed packing of these elements, and restrictions imposed by the limits of the page. For the purpose of displaying residue properties in an automatic way, the importance of fitting the diagram on a page takes precedence.

The diagrams we are describing are particularly interesting when no structure is known for the protein that is being sketched. In such cases, a rough knowledge of the secondary structure elements of a sequence can be obtained by the application of prediction algorithms (e.g., see Refs. 3 and 4). These methods provide for each residue an estimate of the probability that this residue is part of a helix, extended, or loop segment, which is the level of detail that our construction method is going to assume at this stage.

METHODS

This section formalizes the requirements described in the introduction for the development of the 2D representation, and describes the solutions we designed.

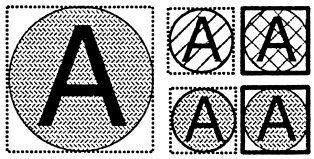
Elements of an Effective Representation

Residue Glyphs The most basic graphical element of a residue-based diagram is the representation of a residue. We use the concept of *residue glyph* to represent residues of a biological sequence by analogy with character glyphs that are the visual representations of characters. The term *residue glyph* names a visual representation of a residue that carries information about the properties of this residue. Character glyphs are used for different representations of a character. All of them represent the same character but each conveys some additional information. For example, a, *a*, a are three glyphs representing the character a. As illustrated in Figure 2, a residue property can be displayed by glyphs of different colors, different shapes, size, circled glyphs, labeled glyphs, etc. Properties that involve several residues, for instance disulfide bridges, can be shown as lines of colors connecting these residues.

Secondary structure elements and representations The representation of protein sequences will be supplemented with information regarding elements of secondary structure. Such identifiers of secondary structure elements (SSE) can be used to tag some particular segments of a protein sequence as helix, β strand, or loop. Figure 3[5,6] illustrates the limited level of detail about the secondary structure that such diagrams convey. Residue-based diagrams can be constructed combining secondary structure representations (SSRs) of two main types: fixed-layout representations (helices, β strands) and free layout representations (loops).

SSRs are intended to display graphically relations among

Figure 2. A residue glyph is shown on the left. The shape of the residue and the letter that identifies its "type" are the main characteristics of the glyph. Among the attributes that can be changed to encode properties at the level of a residue are the color of the background, all the attributes of the character glyph, plus any visual annotation that can be superimposed on the glyph without masking the attributes already shown. A bold face border around the shape of the glyph or inside the limits of its bounding box are also shown. Black and white patterns can substitute for colors.



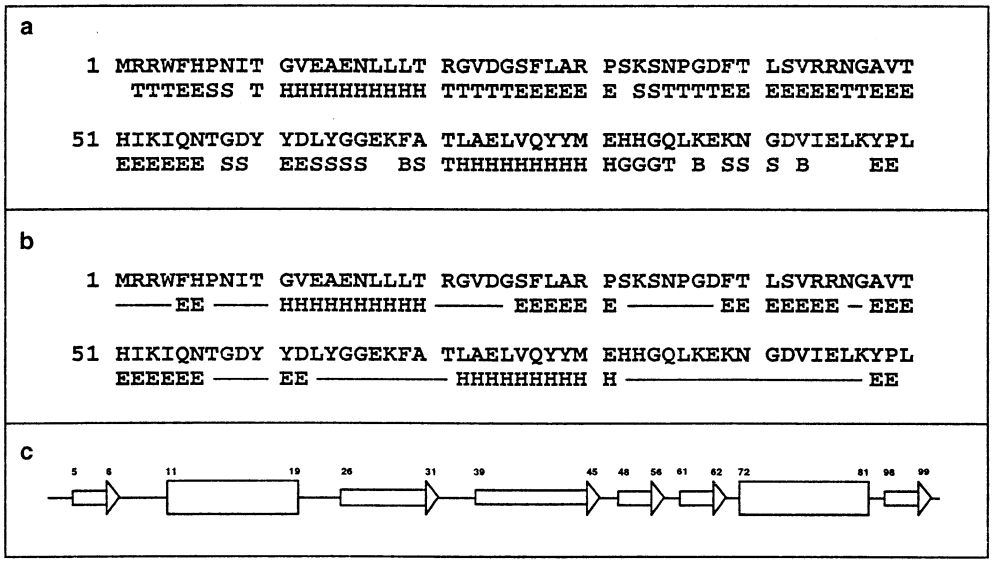


Figure 3. (a) Secondary structure information derived from the sequence of the SH2 domain, 1AYC chain A structure (as obtained from Ref. 5). (b) Secondary structure elements considered for the drawing of the snake diagrams, as calculated from the structure with the program DSSP.[6] (c) The SSE on one line with the indices of their limits on the sequence. For (a) and (b), H tags residues that appear in a helical conformation in the structure, E tags residues that appear in β strands. For (a) T tags β-turn residues, S bend residues, and B residues in isolated β bridges. For (b) and (c), a line tags the residues considered part of weakly defined conformations, or loops.

the residues they contain. These relations encode on a 2D page the relations observed in 3D structures. Unlike the TOPS diagrams,[7,8] where 2D relations between SSRs represent the 3D distance relations with a good accuracy on a page, we give up this accuracy in order to display the residues of the sequence in the context of each SSE. However, the accuracy of the translation is not definitively lost, because a set of different 2D representations of the same sequence contains the whole set of relations (consider how photographs of the same object from different angles allow for a reconstruction of the 3D geometry of the object). Some SSRs are likely to allow only a limited number of layouts, namely those that transpose on the page the relations observed between the residues of the helices or strands. The loop SSRs generally represent only the order of the residues in the protein. Therefore, loop SSRs can be used to adjust the relative orientation of the fixed SSRs on the page. The variability of the structure of loops provides a degree of freedom for the representation on the 2D page where the only considerations that remain are driven by the esthetics of the diagram and the connectivity of loops and SSRs.

Connecting SSRs to build a diagram To construct the diagram of a protein, the secondary structure representations need to be assembled so that the connectivity of the sequence is easily seen. To this end, we impose the constraint that the first residue glyph for each SSR is at the position of the last residue glyph of the previous SSR, if such an SSR exists.

An SSR contains at least one residue glyph. We define the first, respectively last, residue glyph of an SSR as the representation of the residue that comes first, respectively last, in the secondary structure element of this sequence. When the SSR contains only one glyph, first and last residue glyphs are the same. The vector whose origin is at the position of the first glyph of an SSR and destination at the position of the last residue glyph of the same SSR is called the *spacer* of the SSR. These graphic elements related to an SSR are defined in Figure 4.

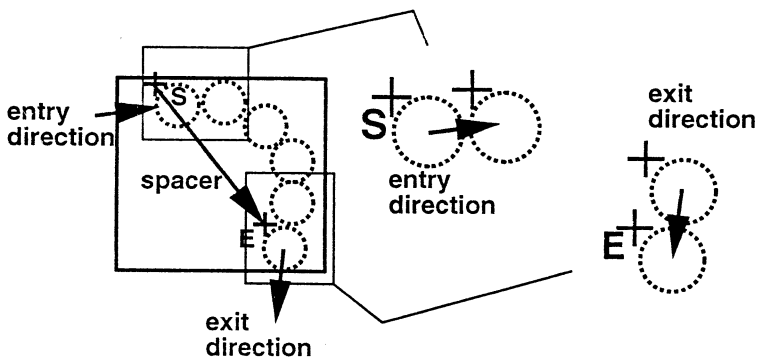


Figure 4. Schema of a secondary structure representation (SSR) of a loop. The large rectangle is the bounding box of the SSR, defined by the maximum rectangular area spanned by its rendering. The loop is rendered as a sequence of residue glyphs (dashed circles). The point S shows the position of the first residue glyph of the SSR, the point E is the position of the last residue glyph. The entry direction and exit direction of the loop representation are given by the vectors $\mathbf{SS}_{+1}$ and $\mathbf{EE}_{-1}$.

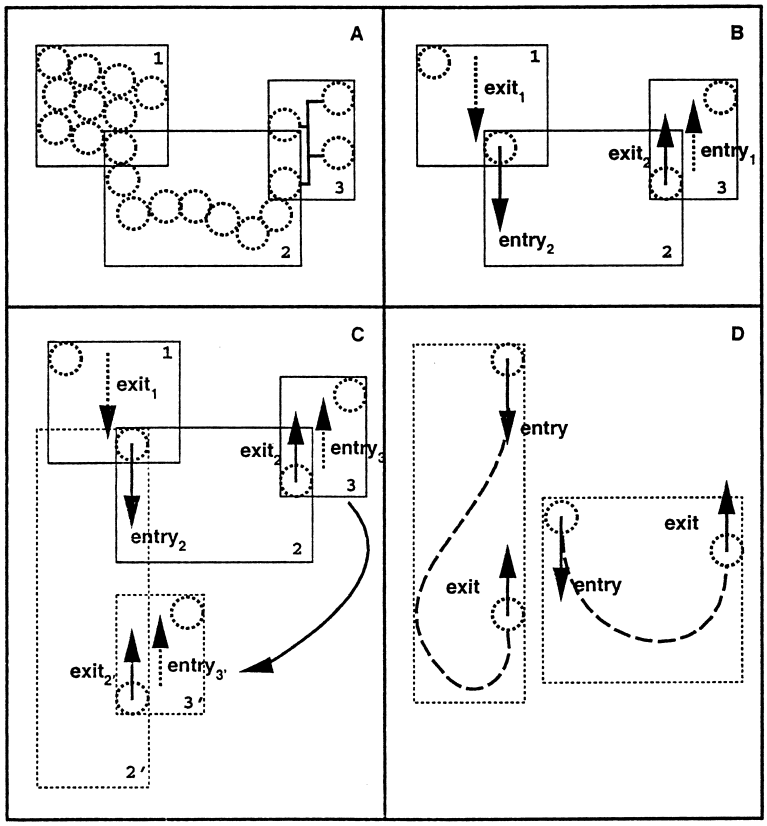


Figure 5. (a) Three SSRs used consecutively to render a small sequence. SSR_1 is a helical SSR, SSR_2 is a loop, and SSR_3 is a β-strand representation. These are the basic types of representation of SSE necessary to build residue-based diagrams for any protein. One representation is required for each kind of secondary structure element. In practice several representations can be developed for each type of SSE, each specialized to show a certain kind of relation among the residues of the SSE. (b) is a simplification of (a), where most of the residue glyphs were hidden and the entry and exit directions of each SSR are shown. (b) illustrates that a loop can be constructed so that its entry direction is equal to the exit direction of the SSR that comes before it in the sequence, and its exit direction is equal to the entry direction of the following SSR. (c) illustrates that to achieve any relative orientation of the fixed-layout $SSR_{1,3}$ the loop layout must be adapted to any set of entry and exit directions and spacer. (d) Sketch of a sensible path for laying out residue glyphs for the two sets of parameters shown in (c) for the $SSRs_{2,2'}$.

Laying Out on a Page

A strategy and a limited set of layout conventions are described below, addressing the presentation of residue-based informations and data.

A solution for the problem of laying out a sequence on a page is facilitated by the consideration that if the representation of loops is mainly undetermined, this freedom can be used to position precisely the SSRs connected by a loop. Figure 5 shows how the complexity of laying out a diagram is reformulated into the problem of laying out residues of loops when their first and last residue representations can be moved to any position on the page.

Diagram Layout Engines

Assuming that the loop layout problem is solved, as described below, laying out a diagram consists of

1. Fixing the spacers for each loop SSR involved in the representation (N-term and C-term are treated as loop SSRs for this purpose)
2. Fixing the entry and exit direction for the fixed-layout SSRs (helices, strands)

These choices will directly control the relative orientations and positioning of the SSRs on the page. They should be determined by the knowledge of the secondary structure element proximities in the protein or by any available approximation of them. We abstracted the process of fixing these diagram layout parameters such that these choices, which depend on a knowledge about the family of the protein the diagram is being drawn for, can be easily modified or replaced by others. In the object paradigm we used during the design, the process of fixing spacer and directions for the SSRs of the diagram is accomplished by a diagram layout engine.

Specialized Layout Engine for Integral Membrane Proteins

We designed this engine for the superfamily of transmembrane proteins. Such proteins contain different numbers of transmembrane helices, ranging from 1 to 12 and more (the upper limit is difficult to know as the transmembrane segments are predictions based on a limited set of experimental results[9]). Because the structural knowledge about transmembrane proteins is limited, we will represent each SSE that is predicted to be inside the membrane vertically, while the other SSE will be shown horizontally. Consecutive transmembrane segments of the sequence are rendered antiparallel. Given this scheme, one need only fix the entry direction of the *N*-term SSR (a loop SSR that takes the entryDirection as a parameter instead of obtaining it from the SSR that precedes it in the sequence). The diagram appearance is completely determined with the choice of a particular spacer length for loop SSRs.

Flexible Loops and Paths

The rendering of loop SSRs is achieved by a method that lays out the residues on a sensible path (e.g., as introduced informally in Figure 5D). If the number of residues in a loop conflicts with the chosen path, a specific set of conventions can be adopted for the representation, as described below. A path could be thought of as the line that goes through each succes-

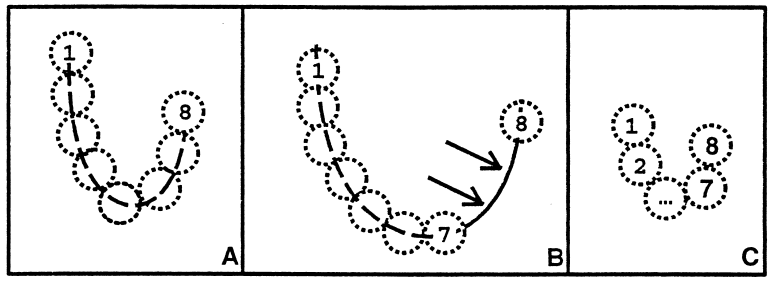


Figure 6. (a) Loop contains eight residues in a path that provides eight positions. Every residue glyph is displayed. (b) The spacer of this SSR has been enlarged and the path could provide up to 10 positions. As the loop still contains eight residues, the first seven are displayed at the first positions of the path and the last one at the last position of the path. Two arrows show the positions of the path that are ignored when rendering residue glyphs. (c) The spacer of the SSR has been shortened such that the path contains only five positions. The two first residue glyphs are displayed at positions 1 and 2 of the path. At position 3 of the path, the ellipsis glyph is shown. This insertion of the special glyph signals that some residues have been hidden (four residues in this case). Residue 7 and 8 are placed at positions 4 and 5 of the path.

sive position of the residue glyphs. We describe a sensible loop path as a path along which residue glyphs can be rendered such that

- They do not overlap with other residues of the same loop
- The path is smooth (without rough angles, which are non-esthetic)
- The beginning of the path is tangent to the entry direction of the loop SSR and its extremity is tangent to the exit direction
- The first and last positions of the path define a vector that exactly matches the spacer of the loop SSR

We do not enforce any constraint on the number of residues that can be rendered on the path. This facilitates the development of methods to automatically build sensible loop paths, but forces specific consideration of the following.

The length of the path matches the length of the loop. This is the ideal case, with all the residue glyphs of the loop SSR represented at the positions given by the path.

The path is longer than the loop. All the residue glyphs of the loop, but the last one, are rendered at the position obtained from the beginning of the path. The last residue glyph of the loop is placed at the last position of the path. To minimize the visual disruption of the diagram a line is drawn through the positions of the path that were not assigned to residue glyphs. See Figure 6b.

The path is shorter than the loop. There is not enough room in the path to position all the residue glyphs. We call this case an *ellipsis* case, in which some residues must be hidden. When this occurs, we need a mechanism to signal to the user that some part of the sequence has not been rendered. Therefore, we introduce an *ellipsis glyph* to mark the positions of the loop where we hide parts of the sequence. See Figure 6c. Locations on the sequence where the ellipsis glyph can substitute for actual residues are defined as *ellipsis seeds.* A loop can contain several seeds. Each seed can be moved along the loop to prevent hiding motifs or annotations in the sequence. The maximum number of residues that can be ellipsed at the position of a seed is user defined. When an ellipsis hides a residue to which a line must be drawn to show a disulfide bridge, the line starts at the ellipsis position. This provides a smooth transition when the spacers are adjusted interactively and cysteines in disulfide bridges are shown and hidden to accommodate the length of the path.

Path Generation

We choose to calculate loop paths using an iterative algorithm based on the Bernstein polynomial development of the Bézier parametric equations[10]. In two dimensions, a Bézier curve is defined by a parametric function of a set of control points (P_i). The positions of the point along the curve can be described by two functions of a parameter t ranging from zero to one.

$$P(t) = \sum_{i=0}^{n} P_i B_{i,n}(t)$$

where

$$B_{i,n}(t) = \binom{n}{i} t^i (1 - t)^{n-i}$$

Interesting properties of Bézier curves for our purpose are (1) that the tangent line to the curve at the point P_0 is the vector P_0P_1 and (2) the tangent line at the point P_n is the vector $P_{n-1}P_n$. Choosing the positions of the first two control points selects the entryDirection whereas the last two control points control the exitDirection of the path. Varying the value of the parameter t with a constant increment, produces positions of points along the Bézier curve that are located at different distances from one another. We build a path made of equally spaced positions iteratively generating points with a small δt until the distance between two successive positions is at least the diameter of a residue glyph. The last position of the path is always set to match the spacer of the SSR. This creates a small visual separation between the last two residues of the loop to be displayed, at most the size of a residue glyph diameter. This procedure produces sensible paths as a function of a list of control points. In the following two sections, we describe two methods for generating the control points for the loop paths.

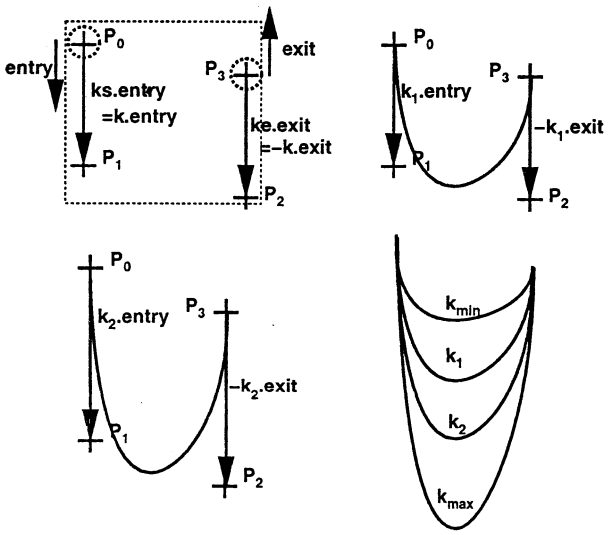


Figure 7. Location of the control points in the algorithmic construction of the loop path. P_{0-3} are the control points and k is the parameter that is optimized in order to match the length of the path with the number of residues of the loop. k_{min} and k_{max} are the limits of the optimization. With the spacer, they define the minimum and maximum lengths of the path. During an interactive change to the spacer of the loop SSR, the previous value of the parameter k is used as the initial value for the new optimization.

Algorithmic Generation of Control Points

Path control points can be generated from the spacer, entry, and exit direction of a loop SSR. Typically, four points (P_{0-3}) are required with positions constructed as follows: P_0 is placed at the location of the first residue of the SSR. P_3 is placed at the location of the last residue. From the properties of the Bézier curve, $P_0P_1 = k_s.entryDirection$ and $P_2P_3 = k_e.exitDirection$, so that we would need to determine two parameters. These parameters will influence the geometry of the path and are directly related to its length. We determine the value of k_s and k_e such that the length of the path is as close as possible to the actual number of residues in the loop. We choose $k = k_s = -k_e$ as this allows for symmetrically shaped paths and a simple optimization scheme: k is considered valid when k is greater than k_{min} and k is lower than k_{max}. When k is increased, the length of the path increases, while it diminishes when k is reduced. When the optimization of k ends with an invalid value, the path length differs from the loop residue number and the cases we described in the section Flexible Loops and Paths (see above) apply. Figure 7 illustrates these points.

This method of loop path generation usually gives good results but may fail to accommodate every end user-specific need. For instance, the N terminus and C terminus of a sequence are similar to loop SSRs, but a path constructed by the previous method may produce a geometry that appears inadequate. Another example would be the loop 3 of some G protein-coupled receptors, which is long but needs to be displayed in its entirety. Manually constructed diagrams represent these loops in a number of ways that we would like to mimic, as discussed in the next section.

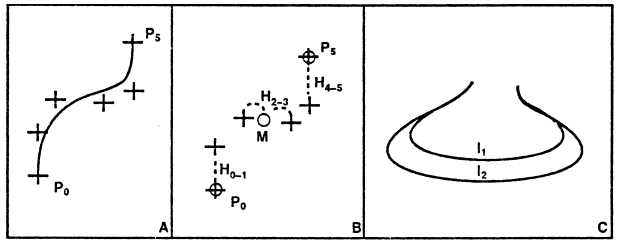


Figure 8. (a) The control points used to render the N-terminus part of a sequence. The points are labelled from zero to five. (b) The scalings (centers shown as circles) that adapt the control points to the number of residue of the SSR. H_{0-1} is centered on P_0 and transforms the points P_{0-1}. H_{4-5} is centered on the last control point and transforms the points P_{4-5}. The last scaling is centered on the middle of the line P_0P_5 and transforms all the points not transformed by H_{0-1} and H_{4-5}. (c) The result of this complex transformation on a loop SSR. The entry and exit directions are unchanged, the spacer is not changed, but the path length is altered. Because the geometry can change significantly as a function of the l parameter, the lower and upper borns of l are under the control of the template.

Templates of Control Points

To provide for more flexibility in the representation we allow the user to select a specific list of control points, which we call the "loop template." New templates can be defined as needed and the choice of the template can be driven by the spacer, entry, and exit direction, or the number of residues of the loop. (It can also be overridden by a specific choice.)

Given a loop template, we need to adapt the template to the specifics of the loop, i.e., spacer, entry and exit direction vectors, and number of residues of the loop. This adaptation is important because a small number of templates must be applicable to a large number of specific sequences. The adaptation to the spacer is relatively straightforward: the control points of the template are scaled homogeneously to give P_0P_n the same norm as the loop spacer. No adaptation is done for the entry and exit directions: different templates are expected to be provided for different relative orientations of these vectors.* The adaptation is realized as follows.

The problem of adapting the template to the number of residues of the loop is similar to optimizing the positions of algorithmically derived control points. Consequently, we developed a generic algorithm that solves both these problems. This algorithm was defined in the previous section; we describe here the specifics of template adaptation: We define the parameter l to be a scaling factor used to transform the control points from the template to the position actually used to build the path. For the purpose of the optimization of the path length, l plays the same role as k: its values range from one to l_{max}, the maximum value is fixed when the template is developed.

To obtain the control points for path construction, l is used as a scaling factor as illustrated in Figure 8. The points are constructed by three uniform scalings such that the properties

*In practice, a tolerance of 10 to 15 degrees is acceptable, allowing a single template to be used in similar situations.

of a sensible path are conserved by the transformation while the length of the path changes with *l*. This procedure adapts the control points to best accommodate the residues of the loop and does not change the spacer. Hence, the adaptation to the spacer can be combined with the adaptation to the residue number without conflict.

IMPLEMENTATION AND PERSPECTIVES

The method we presented here allows automatic drawing of residue-based diagrams of proteins. The complete goals of an implementation are

- To achieve a representation of properties in the context of the diagram. This requires standard ways to represent information on the diagrams
- To allow biological database curators to provide residue-based diagrams of the sequences. A generic and robust engine is needed to render any kind of protein sequence as a diagram. As this goal could be difficult to achieve, several complementary engines could be developed, each tuned for a specific superfamily or family of proteins
- To allow the generation of precise and custom annotated diagrams, for presentation or publication purposes. A diagram editor is required that will allow fine tuning of the location of each SSR on a page. Ideally, the fine-tuned presentations could be reused as diagram templates to render similar sequences (thanks to the loop flexibility central to our method)

We implemented the method as a Java package (named **edu.mssm.crover.domain2d**). The Java 2 platform was chosen for portability reasons. The performance drawbacks that we expected as a consequence of this choice were not important. The interactive adjustment of two SSRs of a diagram (the most intensive calculation performed that involve several optimizations of the loop length, one per value of the spacer) is completed easily on a personal laptop computer. Figure 1 shows the complete diagram that was generated with a small command line program that transforms a SwissProt file to a series of diagram images or PostScript files. This program is distributed with the package as part of the test set.

In addition to the example shown here, the present stage of the approach we describe can be used to develop custom diagrammatic representations of proteins that highlight relations among groups of residues in the sequence. For example, helical wheel representations of sequence alignments can be laid out as described here to depict the spatial relation of the helices. Our efforts will now be targeted to the development of a web service and an interactive, stand-alone diagram editor, both built on top of the domain2d package. The HTML editor should provide a convenient access to residue-based diagrams of proteins, but will be limited (mainly by the current state of the web technology) to basic customizations of the final diagram. The interactive editor will give more freedom to the end user to position the SSRs and annotate the diagrams.

CONCLUSION

We have devised a new method to automate the drawing of residue-based protein diagrams. A novel element in this method is the automatic layout of the loop segments. The method described here provides a foundation on which a number of tools can be built to present a protein as a schematic, two-dimensional diagram. The importance of developing such tools is best understood by considering the quantity of data that is easier to interpret when presented in the context of the secondary structure and topology of a protein. Examples of such data are mutagenesis results, prediction of contacts between the residues, motif, patterns and signals, knowledge of the active site, prediction of turns or of transmembrane domains, etc. Currently, the package we are developing can be used as the basis for the production of image and PostScript diagrams. Readers interested in extending the package, for research purposes, are invited to contact the authors to obtain a (free) license. The package documentation is freely available on-line. Other types of licenses are available on agreement. End users interested in the tools and their application to their work are invited to consult our web site **http://transport.physbio.mssm.edu:8080/joe.**

ACKNOWLEDGMENTS

This work was supported by National Institutes of Health Grants DA09083, DA12408, and DA00060. The authors thank Karel Konvicka for the initial implementation of β-strand SSR.

REFERENCES

1 Campagne, F., Jestin, R., Reversat, J.L., Bernassau, J.M., and Maigret, B. Visualisation and integration of G protein-coupled receptor related information help the modelling: Description and applications of the Viseur program. *J. Comput. Aided Mol. Design* 1999, **6,** 625–643

2 Horn, F., Weare, J., Beukers, M.W., Hörsch, S., Bairoch, A., Chen, W., Edvardsen, Ø, Campagne, F., and Vriend, G. GPCRDB: An information system for G protein-coupled receptors. *Nucleic Acids Res.* 1998, **1,** 275–279. See also URL:http://www.gpcr.org/7tm]

3 Rost, B. and Sander, C. Combining evolutionary information and neural networks to predict protein secondary structure. *Proteins* 1994, **19,** 55–72

4 Rost, B. and Sander, C. Progression of 1D protein prediction at last. *Proteins* 1996, **23,** 295–300

5 Bernstein, F.C., Koetzle, T.F., Williams, G.J., Meyer, E.E., Jr., Brice, M.D., Rodgers, J.R., Kennard, O., Shimanouchi, T., and Tasumi, M. The Protein Data Bank: A computer-based archival file for macromolecular structures. *J. Mol. Biol* 1977, **112,** 535

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

7 Flores, T.P., Moss, D.S., and Thornton, J.M. An algorithm for automatically generating protein topology cartoons. *Protein Eng.* 1994, **7,** 31–37

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

9 Gerstein, M. Patterns of protein-fold usage in eight microbial genomes: A comprehensive structural census. *Proteins* 1998, **33,** 518–534

10 University of California, Davis. On-line computer graphics notes. URL: http://graphics.cs.ucdavis.edu/GraphicsNotes/Graphics-Notes.html, December 1996