BIOINFORMATICS

Vol. 17 no. 5 2001
Pages 461–467

A graph layout algorithm for drawing metabolic pathways

Moritz Y. Becker* and Isabel Rojas

Scientific Databases and Visualization Group, European Media Laboratory, Schloss-Wolfsbrunnenweg 33, D-69118 Heidelberg, Germany

Received on September 9, 2000; revised on December 6, 2000; accepted on January 16, 2001

ABSTRACT

Motivation: A large amount of data on metabolic pathways is available in databases. The ability to visualise the complex data dynamically would be useful for building more powerful research tools to access the databases. Metabolic pathways are typically modelled as graphs in which nodes represent chemical compounds, and edges represent chemical reactions between compounds. Thus, the problem of visualising pathways can be formulated as a graph layout problem. Currently available visual interfaces to biochemical databases either use static images or cannot cope well with more complex, non-standard pathways.

Results: This paper presents a new algorithm for drawing pathways which uses a combination of circular, hierarchic and force-directed graph layout algorithms to compute positions of the graph elements representing main compounds and reactions. The algorithm is particularly designed for cyclic or partially cyclic pathways or for combinations of complex pathways. It has been tested on five sample pathways with promising results.

Availability: On request from the authors.

Contact: mywyb2@cam.ac.uk

INTRODUCTION

Today, a large amount of information on metabolic pathways is available in various databases. Pathways are typically modelled as complex networks of chemical compounds and reactions. It is evident that a graphical representation of such networks is useful for managing the intrinsic complexity of the data. Powerful research tools could be built which dynamically query a database and visualise the resulting pathway. The visualised pathway could be used to refine the query and to navigate through the database.

An example of such a system is KEGG (Kyoto Encyclopaedia of Genes and Genomes), an online database system for querying information on metabolic and regulatory pathways and genome sequences (Kanehisa and Goto, 2000). As in most currently available systems, KEGG visualises pathways in a static way. Pathway diagrams are manually drawn and stored as bitmap image files. These diagrams are displayed as interactive image maps with links to additional information on enzymes and to adjacent pathways.

Another example for static visualisation of pathways is the ExPASy Molecular Biology Server (Appel *et al.*, 1994) which gives online access to the scanned-in version of the Boehringer Mannheim 'Biochemical Pathways' map (Michal, 1993). The map is partitioned into 115 rectangular pieces. Keywords entered by the user are matched against entries on the map, and the corresponding pieces of the map can be displayed.

As Brandenburg *et al.* (1998) pointed out, static visualisation has many severe disadvantages. Whenever the data has been updated, the corresponding images have to be edited manually to reflect the changes. Furthermore, there is no way to specify the amount of detail to be displayed or to hide parts of the pathway. Finally, when it comes to visualising user defined or novel pathways, static visualisation is not applicable at all.

Therefore the visualisation process should be performed dynamically at runtime, based on the information provided by the database. Dynamic visualisation in contrast to static visualisation provides high flexibility, which is necessary for complex queries and the construction of novel pathways.

Metabolic pathways are commonly modelled as directed graphs. A pathway is a collection of interconnected biochemical reactions. Main reactants and products (the compounds that constitute the 'backbone' of the pathway) are represented as nodes and the reactions as edges of the graph. Usually the enzymes catalysing the reaction are displayed as edge labels. Side substrates are drawn near the edge, connected to the edge by curved arcs. Therefore, the problem of dynamically drawing a pathway is a graph layout problem. Given as input a combinatorial description of a graph, a graph layout algorithm should compute geometric positions for the graph elements

*To whom correspondence should be addressed at: Trinity College, University of Cambridge, Cambridge CB2 1TQ, UK.

according to a set of rules.

There are a number of standard graph layout algorithms. Examples include algorithms for circular, orthogonal or planar drawing, and force-directed layout heuristics (Di Battista *et al.*, 1994, 1999; Brandenburg *et al.*, 1997). However, none of these algorithms give satisfactory results when applied to pathway networks.

Little previous work has been done on developing graph layout algorithms for drawing biochemical networks. Karp and Paley (1994) have pointed out that rather than searching for one single, all-purpose graph layout algorithm, different algorithms should be applied to parts of the pathway with different topologies. They devised an algorithm for drawing metabolic pathways which breaks the graph into cyclic, linear and tree-structured components and then applies different layout methods to each of these individually. Their algorithm has been implemented in the EcoCyc system, an electronic encyclopaedia that allows scientists to visualise a collection of biochemical information (Karp *et al.*, 2000).

Takai-Igarashi and Kaminuma (1998) developed a Cell Signalling Network Database (CSNDB) with an interface for dynamic visualisation of pathway data. A new system, PaF-CSNDB, also allows users to find and construct novel pathways (Takai-Igarashi and Kaminuma, 1999). Pathway diagrams are constructed dynamically by a modified version of an algorithm first implemented in the ACEDB software (A. C. elegans database) (Durbin and Mieg, 1991). The original algorithm (S.Letovsky, personal communication) is very similar to the one by Karp and Paley. Acyclic and cyclic components of the graph are identified and laid out using a hierarchic and a circular algorithm, respectively.

PathDB, a pathway database focused on plant metabolism, takes a similar approach. The information stored in the database is converted into a graph structure. If the structure contains cycles, the visualisation front-end lets the user choose between a hierarchical or a circular layout method to calculate the co-ordinates of the nodes (J.Blanchard, personal communication).

In this paper we propose a new algorithm for drawing graphs representing metabolic pathways. Based on the one proposed by Karp and Paley, it takes into account the topological structure of the graph. The algorithm is supplemented by a special force-directed layout algorithm and additional layout heuristics.

We will concentrate on the placement of the graph nodes and edges representing main reactants and products only. The problem of placement of labels that contain information on enzymes and other information related to the biochemical reactions is also being addressed by our research group and will be the topic of a publication currently in preparation. Readers interested in this topic are encouraged to contact the authors for more information.

SYSTEM AND METHODS

The algorithm was implemented in Java 1.3 and executed on a Pentium II workstation running Windows NT. The Java-based graph library YFiles (Wiese *et al.*, 2000) was used to create, manipulate, and view the graph. YFiles provides Java classes representing data structures for graphs, nodes, and edges. These data structures do not only contain information about the abstract graph but also graphical information such as location, sizes and labels of nodes and edges. The library also offers a number of standard graph layout modules, in particular for circular and hierarchic layout, both of which are used in the presented algorithm. The YFiles graphical user interface provides functions for viewing, navigation and interactive editing.

The algorithm does not depend on the choice of the underlying graph library. There are many other packages which could have been used for implementation instead of YFiles. For example, Automatic Graph Drawing (AGD) from Algorithmic Solutions Software, the Graph Drawing Toolkit from Integra Sistemi or the Graph Layout Toolkit from Tom Sawyer Software all offer features which are similar to YFiles.

AESTHETIC GOALS

Common graph layout algorithms draw a graph in such a way that it satisfies certain well-defined aesthetic criteria and constraints such as planarity, minimal edge crossings (edges intersecting with other edges or nodes), minimal drawing area, and maximal symmetry. In the case of metabolic pathways it is difficult, if not impossible, to state such a set of clear cut constraints. Apart from meeting the aesthetic criteria stated above it seems to be important to adhere to well-established, albeit not well-defined, conventions as can be found in relevant biochemistry textbooks (Michal, 1999).

In the figures of such textbooks one can identify two different structures that are used noticeably frequently: directed, hierarchic components and circular components. Whenever such a circular subgraph exists the remaining components are laid out around the circle in such a way that the components are near the nodes of the circle to which they are connected. Sufficiently small components which are connected to the circle by a relatively large number of edges are often placed inside the circle in order to avoid edge crossings. Our algorithm is based on these observations.

ALGORITHM

Some chemical reactions have more than one main reactant or product. In these cases the reaction has to be represented as a hyperedge, i.e. an edge with multiple source and target nodes. The YFiles library does not

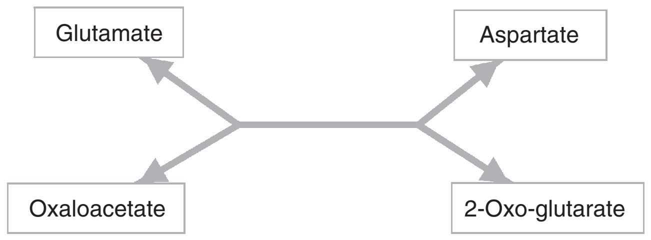

Fig. 1. Hyperedges model chemical reactions with multiple main reactants or products. Zero size dummy nodes are inserted at the forking positions.

support hyperedges, so hyperedges were simulated by inserting a dummy node at the front of an edge which forks out to the multiple target nodes. Similarly, a dummy node is inserted at the back of the edge which is connected to all source nodes. The sizes of the dummy nodes were set to zero (Figure 1).

A simple cycle is a cyclic path in which each node of the path is visited exactly once. The algorithm starts by traversing the directed connected graph to look for the longest simple cycle contained in the graph. The cycle is found by breaking the graph into strongly connected components (subgraphs in which every two nodes are reachable from each other) and then using a depth first search on these components.

Base cases

Two trivial base cases can now be identified: if no cycle could be found at all, the top-to-bottom hierarchic layout algorithm provided by the graph library is applied. The hierarchic layout algorithm partitions the nodes into layers such that nodes in one layer can only be connected to nodes in adjacent layers. If this is not possible, edges crossing several layers are split into several shorter edges, and dummy nodes are inserted. Then the nodes within a layer are permuted to minimise edge crossings. Finally the nodes are positioned to give a balanced layout (Sugiyama *et al.*, 1981; Eades and Sugiyama, 1990).

The other base case applies when the longest cycle is in fact the entire graph. In this case the circular layout algorithm provided by the graph library is used.

General case

If none of the two base cases apply, a longest cycle must have been found, and this cycle must be a proper subgraph of the given graph, so there exist nodes which do not belong to the cycle. These nodes are now grouped into connected components, i.e. sets of connected nodes. The resulting components are by definition not connected to each other. Since the original graph was a connected graph, each of the components must have at least one connection to the cycle. We can distinguish two kinds of components. The *inner components* are those consisting of only one node and that are connected to the cycle by at least two edges, and will be placed inside the circle. All other components are *outer components*, and will be placed around the circle. The choice of one node as threshold for the inner components was made in a somewhat arbitrary manner, since it appeared to be aesthetically the best value. There can be more than one inner node in a cycle but only if they are not connected to each other.

Next, the circular layout algorithm is applied to the cycle, and the minimum radius is chosen such that the resulting circle is large enough to accommodate all inner components.

The outer components are then laid out by recursively applying the same algorithm to each of them individually. The recursion is guaranteed to terminate since one of the two base cases will eventually hold. After that, each component is collapsed into a supernode. The supernode is set to the same location and extent as the subgraph it represents. The supernode–subgraph relationship is stored in a hash table so that later the supernodes can be expanded quickly.

A customised spring embedding layout algorithm (Quinn and Breuer, 1979; Eades, 1984) is applied to the remaining graph, now consisting of the circular cycle, the inner components and the outer supernodes. The spring embedding algorithm places the inner components inside and the supernodes around the circle. The placement is done in such a way that the inner components and the supernodes lie near the circle nodes they are connected to but without overlapping each other. The details of this algorithm are discussed in the next section.

Finally, the supernodes are expanded and replaced by the corresponding subgraphs.

THE CUSTOMISED SPRING EMBEDDING ALGORITHM

The purpose of this sub-algorithm is to lay out the inner components inside the given circle and the supernodes outside the circle. We add the further constraints that nodes must not overlap and that nodes not belonging to the circle are positioned near the nodes on the circle to which they are connected. A force-directed approach seems to be the most natural solution to this problem. Force-directed layout algorithms model the graph as a system of particles with forces acting on them and attempt to find a minimum energy configuration of this system. The spring embedding algorithm is the best known force-directed layout algorithm (Quinn and Breuer, 1979; Eades, 1984). Each edge acts as a spring with a preferred length and exerts a repulsive or attractive force on the nodes connected by it. Nodes are considered as mutually repulsive charges. The total energy of the system is minimised by iteratively letting the nodes move in direction of the forces exerted on them, starting from their initial positions.

The YFiles library implements a version of the spring embedding algorithm. However, it was found to be unsuitable for our purpose, mainly because it does not consider node sizes. We developed a customised implementation with additional heuristics.

As in the original algorithm, the force strength is dependent on the distances between two nodes, but now the distance is computed not as the distance between the centres of the two nodes but rather as the distance between the boundaries of the nodes, thus taking node sizes into account. This is essential because the size of a supernode is the extent of the bounding box of the corresponding subgraph.

Furthermore, each supernode in the system can be assigned a *centre of mass* location. We define the centre of mass of a supernode to be the average position of those nodes inside the supernode which are connected to the circle. In the original spring embedding algorithm, the mutually repulsive charge forces act as if the entire charge was concentrated at the geometric centre of the node. In our model the charge is concentrated at the centre of mass. Also, the ends of the springs are modelled as if they were attached not to the geometric centre of a supernode but to its centre of mass. Using the centre of mass location rather than the geometric centre takes into account the internal structure of a supernode without adding too much overhead.

Each node is associated with a value for its inertia, i.e. its resistance to move, and each edge with an individual preferred length. Clearly the nodes of the circular subgraph should be fixed, hence these nodes are assigned an infinite inertia.

The energy configuration of the system may have multiple local minima, and depending on the initial placement of nodes the algorithm will converge to one of these minima. Therefore care must be taken to compute appropriate initial positions for the nodes.

Computing initial positions

The inner components can simply be placed at the centre of the circle initially. The spring embedding algorithm will automatically find appropriate positions for these nodes within the circle.

For the outer supernodes we can define a *preferred radial angle* relative to the circle. This angle is the average of radial angles of those nodes of the circle to which the supernode is connected.

Suppose the centre of mass of a supernode lies in its upper region. That means that the edges connecting

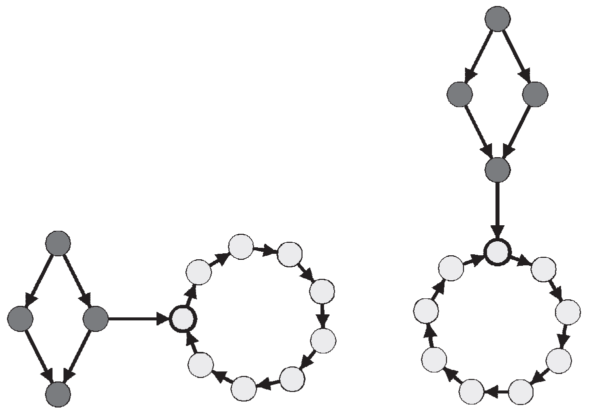

Fig. 2. In the left-hand panel the only node of the outer supernode (darkly shaded) which is connected to the circle lies on the right, so the centre of mass of the supernode lies on the right, hence the preferred orientation is west. The circle is rotated such that the supernode lies west of the circle. Similarly, in the right-hand panel, the centre of mass of the outer supernode is at its bottom, hence the preferred orientation is north of the circle.

it to the circle are incident with internal nodes of the supernode which are located mainly in the upper region of the supernode. So if the supernode is initially positioned such that its centre of mass is south of the circle (at $-90°$ relative to the circle) and if the circle is then rotated in such a way that the resulting preferred radial angle of the supernode is $-90°$, it is likely that fewer edge crossings occur. In this case we say that the *preferred orientation* of the supernode is South. Note that the preferred orientation of a supernode depends only on the displacement of its centre of mass relative to its geometric centre.

Similarly, the preferred orientation of a supernode is North if the centre of mass lies in its bottom region, East if the centre of mass lies on its left-hand side and West if it lies on its right-hand side (Figure 2).

It is clear that this preference cannot be satisfied for all supernodes simultaneously. In our algorithm the circle is initially rotated in such a way that at least the largest supernode has its preferred orientation.

After the rotation has been performed, the preferred radial angle is computed for all other supernodes, and each supernode is placed around the circle accordingly (Figure 3). The preferred edge lengths are set to the actual current lengths of the supernode edges. This ensures that the radial angles are conserved during the spring embedding layout process, if possible.

The customised spring embedding algorithm is then applied to the circle nodes, the inner components and the outer supernodes.

For each outer supernode, its new position is analysed, and it is checked whether its centre of mass can be

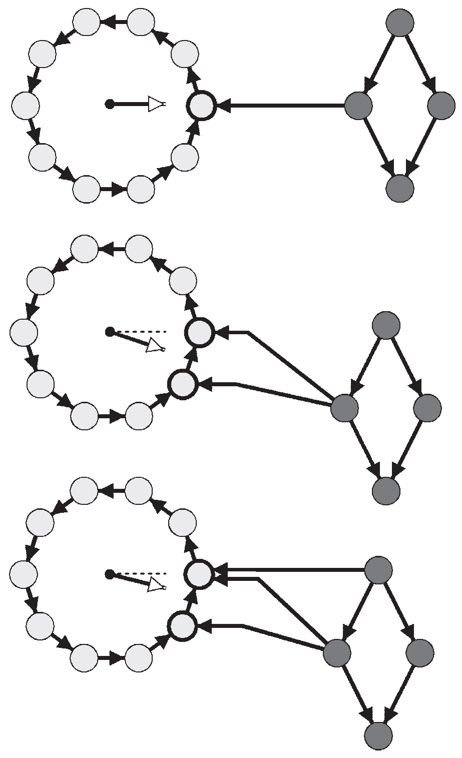

Fig. 3. The preferred radial angle of the centre of mass of an outer supernode (darkly shaded) is computed by taking the weighted average over the angles of the connected circle nodes (bold borders). The preferred angle (0°, −18° and −12°, respectively) is indicated by the arrow in the centre of the circle.

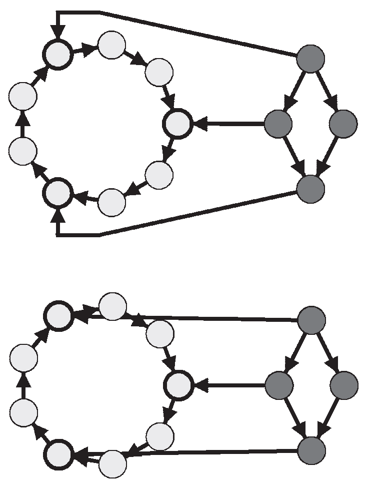

Fig. 4. In the top panel, bends are inserted at the edges connecting the circle to the outer supernode (darkly shaded) in an attempt to avoid edge crossings with the circle. In the bottom panel, the same graph without bends. The edges clearly intersect with the circle.

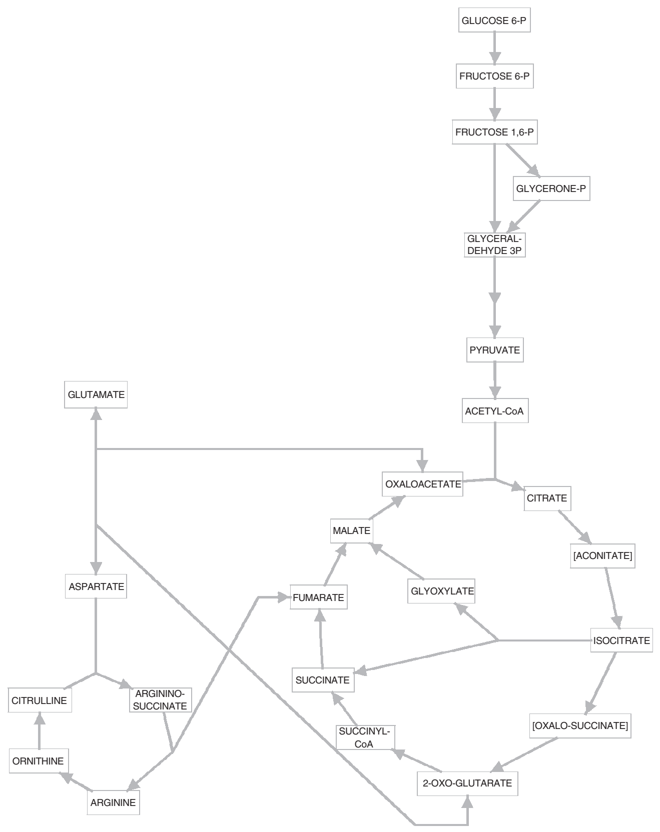

Fig. 5. Combination of TCA cycle, Glycolysis and Urea cycle.

brought closer to the circle by mirroring or flipping it. This heuristic attempts to further reduce the number of edge crossings.

Finally, bends are attached to edges connecting the circle to outer components so that the edges emerge orthogonally from the circle. This is done in an attempt to reduce edge crossings with the circle (Figure 4).

EXPERIMENTAL RESULTS

The algorithm has been tested on five different pathways or combinations of pathways.

The tested pathways were all relatively complex in that they contain cyclic as well as hierarchical components. In all five cases good results were produced. The layout is clear and easy to understand because it emphasises the topological structures of different parts of the graph and keeps logically connected units together. The algorithm managed well to avoid overlapping components and to reduce the required drawing area and unnecessary edge crossings.

Due to space limitations we only show two of the resulting images. Figure 5 shows the TCA cycle connected to Glycolysis and the Urea cycle. The TCA cycle is

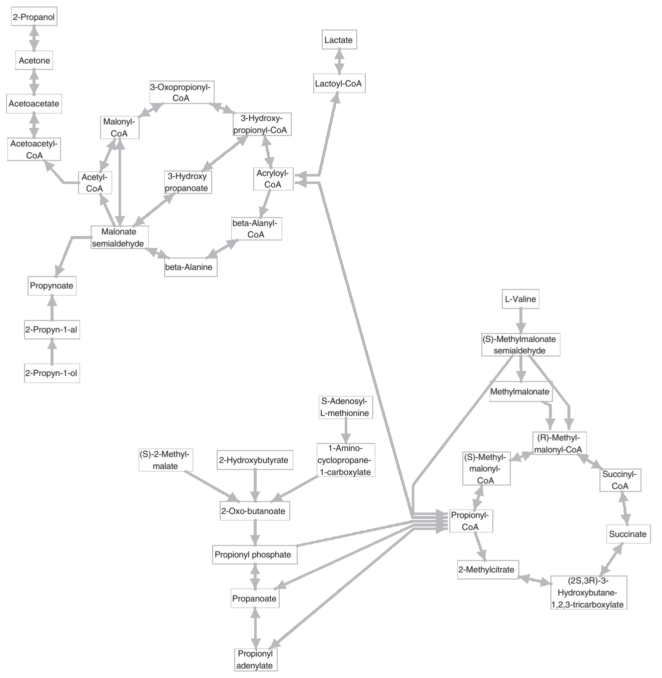

Fig. 6. Propanoate metabolism.

the longest cycle found and has two outer components: Glycolysis, and the Urea cycle together with Aspartate and Glutamate. Glyoxylate is the only inner component of the TCA cycle and hence placed inside the circle. The hierarchical layout algorithm is applied to Glycolysis since it does not contain a cycle. The second outer component, the Urea cycle connected to Aspartate and Glutamate, is laid out recursively. The Urea cycle is found as the only cycle of the subgraph, and Glutamate and Aspartate as the only outer component of the cycle.

Figure 6 shows the Propanoate metabolism. Here the longest cycle is the one containing Acetyl-CoA, having one inner component and four outer components, only one of which requires the recursive step.

The results of the other experiments, images depicting the Pentose phosphate cycle, the Urea cycle together with the metabolism of amino groups, and Phenylalanine, Tyrosine and Tryptophan biosynthesis can be found on the web at the EML research site (http://www.eml.villa-bosch.de).

DISCUSSION AND CONCLUSION

Potentially, the most time-consuming part of the algorithm is the search for the longest cycle since the corresponding decision problem is NP-complete. However, we observed that the depth first search algorithm has good performance if the displayed graph is sparse and not too large (a few hundred nodes and an edge to node ratio of less than 1.5) as it is usually the case with metabolic pathway networks for display purposes. For larger or more highly connected graphs the search for the longest cycle becomes the major bottleneck, and heuristic methods will be necessary. The customised spring embedding algorithm has a time complexity quadratic in the number of nodes to be laid out and linear in the number of edges. Since it can be assumed that the number of distinct connected components outside the longest cycle in the graph is small, the spring embedding algorithm is efficient enough for our purposes. Furthermore, it worked well with a relatively low maximum iteration bound of less than 500. The overall performance of the pathway layout algorithm is good enough for use in interactive applications.

The algorithm proposed by Karp and Paley (1994) does not only compute positions for main compounds but also deals with the problem of placing side substrates and enzymes. This last point has been left out in our algorithm, where we concentrated on the placement of the main compounds. Apart from that, their algorithm differs from ours mainly in that they use a special tree layout algorithm in which nodes are packed as compactly as possible (Karp *et al.*, 1994), whereas in our case a customised spring embedding algorithm is used. Our algorithm gives better results for topologically more complex graphs, e.g. if a combination of different pathways is to be visualised simultaneously. The heuristic computation of initial positions of nodes and the subsequent spring embedding process helps to reduce the number of edge crossings and to place interconnected components at more appropriate locations.

It should be noted that the results produced by the algorithm often differ in many aspects from the conventional drawings in biochemistry textbooks. For instance, in some cases a sequence of reactions is conventionally not drawn as a circle although a cycle exists. In large pathways, it is often the case that individual compounds have a high degree of connectivity. If the algorithm is given such data as input, ‘unconventional’ cycles will be discovered and presumably many edge crossings will occur. This problem can be avoided if highly connected compounds are allowed to appear in multiple instances of graph nodes, thereby reducing the edge to node ratio. In fact, this is what is usually done in conventional manual drawings.

Details such as the relative placement of well-known pathways in a bigger pathway map or the traditional orientation of specific hierarchical structures (e.g. left-to-right or top-to-bottom) also belong to the established conventions in biochemistry.

In order to comply with the above-mentioned conventions the algorithm would need additional layout information that is specific to the pathway or reactions to be drawn. We assumed that such information is not available to the algorithm, which is the case if the graphs to be drawn are novel or computer generated pathways, or the results of complex queries, and not only historically important pathways.

In the future, work has to be done to develop graph libraries and graph layout algorithms tailored to graphs with hyperedges. Simulating hyperedges by inserting dummy nodes did not always give optimal results because the current algorithms treat them like normal nodes.

The presented algorithm appears to be an attractive choice for visualising metabolic pathways. It produces good results, even in complex cases where cyclic pathways are to be visualised in the context of connected pathways. Thus, the algorithm could well be used to enhance existing and to build new graphical user interfaces to access biochemical databases.

ACKNOWLEDGEMENTS

We would like to thank Ulrike Wittig and Andreas Kohlbecker for assistance with the acquisition of data sets. The work presented in this paper was carried out thanks to the support of the German Ministry for Education and Sciences (BMBF) (Project Bioregio 0312212) and the Klaus Tschira Stiftung (KTS).

REFERENCES

Appel,R., Bairoch,A. and Hochstrasser,D. (1994) A new generation of information retrieval tools for biologists: the example of the ExPASy WWW server. *Trends Biochem. Sci.*, **19**, 258–260.

Brandenburg,F.-J., Jünger,M. and Mutzel,P. (1997) Algorithmen zum automatischen Zeichnen von Graphen. *Informatik Spektrum*, **20**, 199–207.

Brandenburg,F.-J., Gruber,B., Himsolt,M. and Schreiber,F. (1998) Automatische Visualisierung biochemischer Information. In *Proceedings of the Workshop Molekulare Bioinformatik, GI Jahrestagung,* pp. 24–38.

Di Battista,G., Eades,P., Tamassia,R. and Tollis,I.G. (1994) Annotated bibliography on graph drawing algorithms. *Comput. Geom.-Theor. Appl.*, **4**, 235–282.

Di Battista,G., Eades,P., Tamassia,R. and Tollis,I.G. (1999) *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, New Jersey.

Durbin,R. and Mieg,J.T. (1991) A C. elegans Database. Documentation, code and data available from anonymous FTP servers at lirmm.lirmm.fr, cele.mrc-lmb.cam.ac.uk and ncbi.nlm.nih.gov.

Eades,P. (1984) A heuristic for graph drawing. *Congr. Numer.*, **41**, 149–160.

Eades,P. and Sugiyama,K. (1990) How to draw a directed graph. *J. Inform. Proc.*, **13**, 424–437.

Kanehisa,M. and Goto,S. (2000) KEGG: Kyoto encyclopedia of genes and genomes. *Nucleic Acids Res.*, **28**, 27–30.

Karp,P.D. and Paley,S. (1994) Automated drawing of metabolic pathways. In Lim,H., Cantor,C. and Robbins,R. (eds), *Third International Conference on Bioinformatics and Genome Research*.

Karp,P.D., Lowrance,J.D., Strat,T.M. and Wilkins,D.E. (1994) The Grasper-CL graph management system. *LISP Symb. Comput.*, **7**, 251–290.

Karp,P.D., Riley,M., Saier,M., Paulsen,I.T., Paley,S. and Pellegrini-Toole,A. (2000) The EcoCyc and MetaCyc databases. *Nucleic Acids Res.*, **28**, 56–59.

Michal,G. (1993) *Biochemical Pathways* (poster). Boehringer Mannheim GmbH.

Michal,G. (1999) *Biochemical Pathways*. Spektrum Akadem., Heidelberg.

Quinn,N.R., Jr and Breuer,M.A. (1979) A force directed component placement procedure for printed circuit boards. *IEEE Trans. Circuits Syst.*, CAS **26**, 377–388.

Sugiyama,K., Tagawa,S. and Toda,M. (1981) Methods for visual understanding of hierarchical systems. *IEEE Trans. Syst. Man Cybern.*, **11**, 109–125.

Takai-Igarashi,T. and Kaminuma,T. (1998) A database for cell signaling networks. *J. Comput. Biol.*, **5**, 747.

Takai-Igarashi,T. and Kaminuma,T. (1999) A pathway finding system for the cell signaling networks database. *In Silico Biol.*, **1**, 129–146.

Wiese,R., Eiglsperger,M. and Schabert,P. (2000) The Y-files graph library: documentation and code available at http://www-pr.informatik.uni-tuebingen.de/yfiles.