Visualizing a Whole-Cell Model

Arne Hoffmann

1 . Introduction

Visualizing biological pathways has long been an important part of understanding biological processes. As computing capacity becomes more cost-efficient and new methods for experimentation are developed, the complexity of biological processes modeled and simulated increases. This has led from simple isolated cell processes over complex interaction between them to modeling a whole cell[1].

The Yeast Cell Model (YCM) project[4] of the Humboldt-University in Berlin builds a whole-cell model of the baker's yeast *Saccharomyces cerevisiae*. To facilitate understanding of the modeled processes and their interactions a visualization is needed that enables exploration of the model.

Most of the cell processes are modeled as ordinary differential equations (ODEs). This provides a significant difference to the whole-cell model project[2] of Karr *et al* which follows an agent based approach to model *Mycoplasma genitalium*. This difference makes WholeCellVis[3], which is used to visualize the project of Karr *et al*, incompatible with the YCM project.

In the following the visualization of the YCM as an interactive node-link graph shall be explored and required features dicussed.

2. Problem Description

The goal of this project is to develop a visualization for the model build in the Yeast Cell Model (YCM) project [4]. The aim of the visualization is to allow to explore the model.

The model consists of a number of modules written in python each describing a subprocess of the cell cycle. Each module introduces multiple species, reactions and algebraic species. A species represents an amount of molecules of the same type. A reaction describes a chemical reactions between species. Algebraic species represent a variety of interactions described by a simple mathematical formula.

Each species, reaction and algebraic species has information connected to it. For species this includes, next to a name, a state of phosphorylation or electrical charge, an affiliation with a compartment and one or more annotations. Annotations usually refer to an entry in the CHEBI[5], SBO[6] or SGD[7] database though some annotations are still placeholders. Compartment is the term used in the YCM project to describe a physical subsection of the cell. Reactions are attributed with a name, a list of substrates, products and modifiers and a rate. Algebraic species have a name, an algebraic equation and an annotation. All species, reactions and algebraic species are also indirectly attributed with the modules they are introduced in.

Due to the inherent relations between species, reactions and algebraic species a node-link graph is chosen to to represent the model.

3. Layouts

Finding a good layout for a graph is a common problem. There are several commonly used aesthetic criteria for a layout that help estimating its quality and selecting between multiple layouts [8]. Firstly visual anomalies should be avoided unless they encode information. This includes for example edge crossings as well as sharp bends in edges. Secondly edges should be kept short to strengthen visual grouping of nodes that are connected. Thirdly symmetry is usually considered visually pleasing.

While a manual placement of all nodes would be possible this project is required to anticipate changes in the YCM project's model. Therefore an automated layout algorithm must be chosen. As nodes are limited to a few hundred and no data is introduced by the user, the computation of the layout can be done separate from and prior to its display. This means that the speed of the layout algorithm is of low importance.

A circular, a radial, a force-directed and GraphViz dot layout algorithm [9] have been compared.

The major concern with the circular layout is that as long as straight edges are used, short edges become hard to read. In a local environment the bend of the circle becomes neglegtable causing short edges to cross nodes. This can for example be mitigated by using curved edges. As a circular layout always creates circles the amount of symmetry concerning nodes is high.

The radial layout causes an especially high amount of edge crossings. A radial layout tries to build a tree-like structure with all nodes being placed on concentric circles around the root. The circle a node is placed on depends on the distance of the node to the root. While this causes most of the edges to be between neighboring circles, many of these edges cross lower radius circles as the connected nodes are on opposing sides of their respective circles.

The force-directed approach causes ADP, ATP and Pi, which are all nodes of high degree that share many of their neighbors, to be strongly clustered. This causes a very high number of edge crossings around those nodes. The force-directed layout produces only few long edges. The short edges also cause high edge crossing to be a very local limited problem.

The dot algorithm form GraphViz produces many long edges even though creating short edges is supposed to be one of its optimization criteria. This is especially evident for nodes of high degree. ATP for example has far more long than short edges connected to it. This effect may be caused by dot trying to order the nodes such that a general direction in the flow of the graph is revealed while the graph does not offer such a flow.

In conclusion the force-directed layout was considered to be the most aesthetic.

The Gestalt Laws [10] of visual perception, proximity, closure and good form, when applied to a node-link graph all depend on position of nodes and the course of the edges. Therefore the selection of layout does not only influence how aesthetic and therefore easy to read the graph is but also its likely interpretation. A radial layout for example inadvertently places focus on the selected root node. As the purpose of the visualization is to explore the YCM project's model, restricting it to one layout and therefore favoring one interpretation may work against the intended purpose.

4. Nodes and Edges

The nodes and edges of a graph provide several ways to visually encode attributes of data. This includes color and opacity as well as symbol choice and symbol size for nodes and stroke thickness and stroke pattern for edges.

In case of dynamic visualization additional encoding options are gained through animation. For edges usually

an animation is chosen which displays particles moving along the edges. Data attributes can than be encoded onto the symbol, color, opacity and size of the particle as well as the frequency, pattern and speed with which the particle moves along an edge.
Beyond encoding data attributes studies have shown that the selection of pattern and animation of an edge also influences how easily a graph can be read [11].
One of the major obstacles in visualizing the model of the YCM Project is that the nodes of certain species, for example ADP, ATP and Pi, have a high degree causing a high density of edges in their surroundings. This makes reading the graph in those surroundings difficult. Especially edge symbols providing edge direction tend to become unreadable as they overlap near the node.
In the following section methods to reduce this problem will be discussed.

4.1. Replacing Edge Symbols

As the original intention for the visualization was to follow the Process Description Language of SBGN [12] whenever possible, edges encode information about the kind of connection between two nodes by the kind and position of a symbol on the edge. An arrow at the end of an edge pointing towards the node of a reaction means that the species belonging to the other connected node is a substrate for that reaction. The reverse, an arrow pointing towards a node of a species, means that the species is a product of the connected reaction. If the species is a modifier to the reaction a circle is used on the reaction side of the corresponding edge. If either or both connected node represents an algebraic species no symbol is used.
The implemented prototype showed that in cases where multiple edges converge on a node from similar direction the overlap of the symbols causes problems in distinguishing between symbol types (arrow or circle) as well as to which edge a symbol belongs. Therefore the information intended to be encoded into the edge symbol is lost. As nodes with high degrees usually also have more edge symbols overlapping this problem is more grave near them. An exception to this would be a node with only outgoing edges as there would be no symbols to overlap.
To reduce this loss in information the stroke pattern of the edges could be replaced with a repetitive pattern. For example instead of a single symbol at the end of an edge that symbol could be repeated over the length of the edge. This would allow to read the information previously encoded into the end symbol at multiple points of the edge making overlaps in some places less severe. This also provides a basis for commonly used edge animation which could be used to encode additional data attributes or simply improve highlighting [13].

4.2. Edge Reduction

Another way to improve the readability of the graph would be to reduce the number of edges that must be displayed. There are two principle ways to achieve such a reduction: Edge compression and edge bundling. In the following chapter both will be described and their relevance for visualizing the YCM project discussed.

4.2.1. Edge Compression

Edge compression as discussed by Tim Dwyer, Nathalie Henry Riche, Kim Marriott and Christopher Mears [14] could provide such a reduction. The general idea of edge compression is to cluster nodes with similar edges together and have any shared edges replaced by one edge connected to the resulting cluster. In the paper of Dwyer *et al.* [14] three types of edge compression are introduced: Neighbor Matching, Modular

Decomposition and Power-Graph Decomposition. In Neighbor Matching only nodes with identical edges are clustered, in Modular Decomposition nodes of a cluster are allowed to have edges between each other and in Power-Graph Decomposition there may also be nodes that have edges to a node outside their cluster. The paper evaluated Neighbor Matching and Modular Decomposition to be faster for finding shortest path in a graph while causing no significant difference in accuracy. While Power-Graph Decomposition achieves the highest reduction in in edges, edges from inside a cluster to a different cluster tend to be difficult to read.

The graph of the YCM project's model provides difficult conditions for edge compression.

First there are four different type of edges per node, though most edges belong to two of the four edge types. To conserve the information encoded into these edges, edges of different types must be looked at separately when considering to cluster nodes. This for example prohibits the clustering of ADP and ATP. While ADP and ATP share most of the nodes they are connected to, their edges to these nodes are usually inverse.

There are three types of nodes: species (circles), reactions (rectangles) and algebraic species (triangles). Due to the information the node types encode there cannot be an edge between two species or two reactions. Since most nodes are either a species or a reaction this makes neighboring nodes unlikely to share edges to other nodes. Therefore Modular Decomposition is unlikely to provide more or larger cluster than Neighbor Matching.

In addition there is also already a meaningful clustering representing subsections of the cell called compartments in context of the YCM project. This inhibits edge compression as nodes inside different compartments cannot be clustered together. Furthermore introducing additional clusters through edge compression might cause confusion between these new clusters and compartments.

Lastly there is the risk of such clusters to be misinterpreted as cliques as it happened to one participant in the experiments of the paper by Dwyer *et al* [14].

Analyzing the model for it's compatibility with edge compression leads to the following result: There are only three cluster that could be created with Neighbor Matching. The first cluster would include three nodes with three edges, the second two nodes with two edges and the third three nodes with two edges. Neither of these nodes is connected with one of the nodes of high degree. This does not mean however that the surroundings of these nodes can not profit from a reduction in edges.

4.2.2. Edge Bundling

Edge bundling is a generic term for a number of different algorithms that reduce edge crossing by merging multiple edges into one edge when they are close to each other. For example Holten and van Wijk [15] achieves this for general node-link graphs with straight edges by modeling edges as flexible springs with an attracting force between them. Lambert *et al*. [16] introduced an edge bundling algorithm where edges of a graph are routed along the edges of a mesh using a shortest path algorithm. Ganser *et al*. [17] propose a multilevel clustering algorithm that tries to reduce the amount of 'ink' needed to represent the edges.

There are several problems with the introduction of edge bundling into the visualization of the YCM project's model.

Any edge that becomes part of a bundle will in most cases become difficult to distinguish from other edges of the same bundle. It follows that in order to preserve the information encoded in the edges edge bundles cannot be simply formed based on the proximity. Both the information about which node is connected to which and what kind of connection exists between two nodes could be lost. To avoid losing information about the kind of connection only edges of the same type can be bundled. In order to preserve connection information one end of a bundle must be connected to a node. This way the connection information encoded

in the bundle is the same as the connection information originally encoded in all the bundled edges for that end of the edges and a distinction between the edges at that end becomes unnecessary.
Lastly as with edge compression there is the risk of the bundling of edges to be wrongly interpreted as more than a visualization technique. For example if a number of edges from species that are products to the same reaction are bundled together the bundling could be interpreted as these products forming a new species before partaking in the actual reaction.
Even under the given conditions many edge bundles can be created. Especially in the surroundings of nodes of high degree, which are all species nodes, are many edges close together. These edges are also mostly of the same type because species nodes usually have a strong bias towards being either a substrate or a product to reactions. While algebraic species are usually rather limited in their number of edges they also have only one type of edge connected to them. This makes it relatively easy to form small edge bundles near nodes of algebraic species.
Since edge bundling manipulates the shape of edges while edge compression manipulates the number of edges both could be applied together.

5. Color

Applying different colors to elements of a graph exploits the Gestalt Law of similarity. Same colored elements are grouped together while different colored elements are perceived as separate. This makes coloration especially useful to encode information about shared attributes of elements that are not already grouped by proximity in a layout.
In this project color is used to encode a nodes affiliation with a certain module. As modules are of small number and not likely to increase by a large amount in the course of the YCM project only a small color palette is needed. To avoid confusion between the different modules, colors should be chosen such that they have high contrast between them. There are two well known color palettes that provide colors of high contrast. The first is Kelly’s 22 colors of maximum contrast [18] hereafter called Kelly Colors. The second is a palette proposed by Boynton [19] hereafter called Boynton Colors. Kelly Colors are ordered as such that if the first n colors are selected there is a maximum contrast between these n colors. In addition the first 9 colors are easy to distinguish for people with defective color vision. Boynton Colors provide no such order.

6. Interaction and Navigation

To overcome the limits of layout algorithms for visualizing the required information features to interact with and navigate the graph are required. They allow to focus on certain parts of the graph and display additional data on demand that would cause to much clutter if displayed continuously.

6.1. Pan and Zoom

Pan and zoom are traditional tools of visualization to focus on parts of a graph. Zoom allows to select a subsection of the graph defined by proximity of its elements while pan allows to change the view between those subsections.

6.2. Highlighting

A different technique to focus on only part of a graph is to highlight or downplay certain nodes and edges. Different to pan and zoom the part of the graph that can be put into or out of focus is not limited to the proximity of the elements in the graph layout.
Highlighting an element is usually achieved by enlarging the element, changing its color, adding an additional graphical element (e.g. an arrow glyph or a label) to it or any combination thereof.
Meaningful highlights for this project include but are not limited to the neighborhood of a selected node and interface points between different modules.

6.3. Search

As the graph includes nearly 400 nodes if a certain species, reaction or algebraic species is of interest manually searching all nodes for the one of interest is tedious. To facilitate finding a node a feature should be provided that allows to search for a node by name or annotation. The search result should than be highlighted in the visualization.

6.4. Selective Information Display

Not all information connected with elements of the graph is displayed simultaneously. The reason for this is that the amount of information a human can process at a time is limited and therefore the amount of information displayed must be similar limited. In addition some information is difficult to encode in graphical elements without fully displaying the information itself. This includes for example mathematical formula, longer passages of text or pictures. Never the less this information has to be included. To display this information usually means to provide features that show a separate representation of this information on interaction with a corresponding element of the graph. For example each species has an annotation which in turn stands for an amount of information stored within a database. To display this information would mean to display the database entry. Doing so constantly for every species node in the graph would cause a huge amount of clutter. Instead the information is displayed for only one species at a time on interaction with the matching species node. The information is also not displayed within the graph but in a separate section.

7. Prototype

As part of this project a prototype was implemented. Python and HTML with JavaScript where used for the implementation. The different layouts are build in python and with GraphViz in a preprocessing stage. The graph is visualized in JavaScript with ECharts [20] as a framework.
ECharts has proven to be a disadvantageous choice in framework. While tutorials, examples and documentations are provided they are often incomplete or not up to date. In addition many predefined features, like pan and zoom, tend to be very limited in their working conditions.
The prototype implements the following features:
The prototype can display one layout at a time. A circular, dot, force-directed and radial layout can be selected between. Layouts are precomputed using GraphViz and cannot be changed by the user.
Nodes are colored according to the modules that introduced them. The user can select between using Kelly Colors or Boynton Colors. Nodes that are associated with multiple modules arbitrary select one module for coloration. Edges are colored based on the node they originate from with undirected edges selecting an arbitrary connected node for color.

The display can be panned and zoomed.
Nodes can be searched by their exact or partial name or annotation. Matching nodes are highlighted by enlargement and adding their name as a label to them.
While hovering with the mouse over a node, that node and its neighbors are highlighted while the rest of the graph is downplayed.
Additional information attached to the nodes of the graph like the rate of a reaction can be displayed by clicking a node. This subsections the window, resizing the graph in the process. The newly displayed subsection also displays one entry from a database if the clicked node provides at least one legal annotation.
As ADP, ATP and Pi create the most clutter the option was provided not to display them and their connections to ease focusing on other nodes.
Similar all nodes of a module and their edges can be hidden to better allow exploring single modules or the connections between certain modules.

8. Future Work

There are many directions for future work. Implementing and comparing the effects of edge bundling and edge collapse on the readability of different layouts. While their reduce in clutter generally increases readability it remains to be seen whether this extends to the specific layouts used in this project. Especially how to introduce the clusters of edge collapse without causing confusion with existing clusters demands further examination.
Since exploration is the aim of the visualization any feature that provides the user with more customization options might be examined for its ability to provide additional insight into the model. For example it could be tested whether providing the ability to manipulate the layout by hand, to move, hide or highlight edges and nodes manually leads to an improvement in understanding or confusion.
As the model is simulated as part of the YCM project including simulation data into the visualization would be a natural next step. How to best visualize this additional data aesthetically while not overburden the users ability to process the displayed information requires further research.

9. Sources

[1] Kitano, Systems Biology: A brief overview, Science 2002.Kitano, H. (2002). Systems biology: a brief overview. Science, 295(5560), 1662-1664.

[2] Karr, J. R., Sanghvi, J. C., Macklin, D. N., Gutschow, M. V., Jacobs, J. M., Bolival, B., ... & Covert, M. W. (2012). A whole-cell computational model predicts phenotype from genotype. Cell, 150(2), 389-401.

[3] Lee, R., Karr, J. R., & Covert, M. W. (2013). WholeCellViz: data visualization for whole-cell models. BMC bioinformatics, 14(1), 253.

[4] Yeast Cell Modeling. https://rumo.biologie.hu-berlin.de/tbp/index.php/en/research/topics2/42-research/topics/32-yeast-cell-modeling. Accessed 8 Jun 2018

[5] Hastings, J., de Matos, P., Dekker, A., Ennis, M., Harsha, B., Kale, N., Muthukrishnan, V., Owen, G., Turner, S., Williams, M., and Steinbeck, C. (2013) The ChEBI reference database and ontology for biologically relevant chemistry: enhancements for 2013. Nucleic Acids Res.

[6] Courtot M., Juty N., Knüpfer C., Waltemath D., Zhukova A., Dräger A., Dumontier M., Finney A., Golebiewski M., Hastings J., Hoops S., Keating S., Kell D.B., Kerrien S., Lawson J., Lister A., Lu J., Machne R., Mendes P., Pocock M., Rodriguez N., Villeger A., Wilkinson D.J., Wimalaratne S., Laibe C., Hucka M., Le Novère N. (2011), Model storage, exchange and integration. Molecular Systems Biology, 7:543DOI MSB

[7] Cherry JM, Hong EL, Amundsen C, Balakrishnan R, Binkley G, Chan ET, Christie KR, Costanzo MC, Dwight SS, Engel SR, Fisk DG, Hirschman JE, Hitz BC, Karra K, Krieger CJ, Miyasato SR, Nash RS, Park J, Skrzypek MS, Simison M, Weng S, Wong ED (2012) Saccharomyces Genome Database: the genomics resource of budding yeast. Nucleic Acids Res.

[8] Purchase, H. C., Cohen, R. F., & James, M. (1995, September). Validating graph drawing aesthetics. In *International Symposium on Graph Drawing* (pp. 435-446). Springer, Berlin, Heidelberg.

[9] Gansner, E. R., Koutsofios, E., North, S. C., & Vo, K. P. (1993). A technique for drawing directed graphs. *IEEE Transactions on Software Engineering*, *19*(3), 214-230.

[10] Banerjee, J. C. (1994). "Gestalt Theory of Perception". Encyclopaedic Dictionary of Psychological Terms. M.D. Publications Pvt. Ltd. pp. 107–109.

[11] Holten, D., & Van Wijk, J. J. (2009, June). Force□Directed Edge Bundling for Graph Visualization. In Computer graphics forum (Vol. 28, No. 3, pp. 983-990). Blackwell Publishing Ltd.

[12] Nicolas Le Novère, Michael Hucka, Huaiyu Mi, Stuart Moodie, Falk Schreiber, Anatoly Sorokin, Emek Demir, Katja Wegner, Mirit I Aladjem, Sarala M Wimalaratne, Frank T Bergman, Ralph Gauges, Peter Ghazal, Hideya Kawaji, Lu Li, Yukiko Matsuoka, Alice Villéger, Sarah E Boyd, Laurence Calzone, Melanie Courtot, Ugur Dogrusoz, Tom C Freeman, Akira Funahashi, Samik Ghosh, Akiya Jouraku, Sohyoung Kim, Fedor Kolpakov, Augustin Luna, Sven Sahle, Esther Schmidt, Steven Watterson, Guanming Wu, Igor Goryanin, Douglas B Kell, Chris Sander, Herbert Sauro, Jacky L Snoep, Kurt Kohn & Hiroaki Kitano. The Systems Biology Graphical Notation. Nature Biotechnology 27, 735 - 741 (2009). http://dx.doi.org/10.51038/nbt.1558

[13] Ware, C., & Bobrow, R. (2004). Motion to support rapid interactive queries on node--link diagrams. *ACM Transactions on Applied Perception (TAP)*, *1*(1), 3-18.

[14] Dwyer, T., Riche, N. H., Marriott, K., & Mears, C. (2013). Edge compression techniques for visualization of dense directed graphs. *IEEE transactions on visualization and computer graphics*, *19*(12), 2596-2605.

[15] Holten, D., & Van Wijk, J. J. (2009, June). Force□Directed Edge Bundling for Graph Visualization. In Computer graphics forum (Vol. 28, No. 3, pp. 983-990). Blackwell Publishing Ltd.

[16] Lambert, A., Bourqui, R., & Auber, D. (2010, June). Winding roads: Routing edges into bundles. In Computer Graphics Forum (Vol. 29, No. 3, pp. 853-862). Blackwell Publishing Ltd.

[17] Gansner, E. R., Hu, Y., North, S., & Scheidegger, C. (2011, March). Multilevel agglomerative edge bundling for visualizing large graphs. In Visualization Symposium (PacificVis), 2011 IEEE Pacific (pp. 187-194). IEEE.

[18] Kelly, K. L. (1965). Twenty-two colors of maximum contrast. *Color Engineering*, *3*(26), 26-27.

[19] Boynton, R. M. (1989, August). Eleven colors that are almost never confused. In Human Vision, Visual Processing, and Digital Display (Vol. 1077, pp. 322-333). International Society for Optics and Photonics.

[20] ECharts https://github.com/ecomfe/echarts Accessed 20. May 2018