Visualization of Functional Aspects of microRNA Regulatory Networks Using the Gene Ontology*

Alkiviadis Symeonidis[1,2], Ioannis G. Tollis[1,2], and Martin Reczko[1,**]

[1] Computer Science Department, University of Crete
[2] Institute for Computer Science, Foundation for Research and Technology - Hellas, P.O. Box 1385, 711 10 Heraklion, Crete, Greece

Abstract. The post-transcriptional regulation of genes by microRNAs (miRNAs) is a recently discovered mechanism of growing importance. To uncover functional relations between genes regulated by the same miRNA or groups of miRNAs we suggest the simultaneous visualization of the miRNA regulatory network and the Gene Ontology (GO) categories of the targeted genes. The miRNA regulatory network is shown using circular drawings and the GO is visualized using treemaps. The GO categories of the genes targeted by user-selected miRNAs are highlighted in the treemap showing the complete GO hierarchy or selected branches of it. With this visualization method patterns of reoccurring categories can easily identified supporting the discovery of the functional role of miRNAs. Executables for MS-Windows are available under *www.ics.forth.gr/∼reczko/isbmda06*

1 Introduction

MicroRNAs (miRNAs) are very small non coding DNA regions regulating the post-transcriptional activity of other genes [1]. Experimental verification of these regulations is very expensive both in time and money so prediction algorithms are being developed [2,3,4,5,6,7,8,9,10]. In the following we have used the miRanda program ([2,3,4]), where every prediction is also assigned a score that reflects the strength of the regulation. The genes and miRNAs together with a set of predicted regulations form a (predicted) miRNA regulatory network. Information about genes is stored in the Gene Ontology (GO) where terms related to genes' functionalities are hierarchically organized [11].

Since research is focused on predictions little attention has been paid to the produced networks, their structure and properties. Here, we present algorithms for the visualization of miRNA regulatory networks and the GO. We also propose an alternative point of view for miRNA regulatory networks through the GO.

* This work was supported in part by INFOBIOMED code: IST-2002-507585 and the Greek General Secretariat for Research and Technology under Program "ARISTEIA", Code 1308/B1/3.3.1/317/12.04.2002.

** Corresponding author.

The remainder of this paper is organized as follows: In Section 2 we discuss how the biological information is transformed into graphs and review appropriate drawing techniques. In Section 3 we discuss an interactive drawing technique that allows the user to see the portion of the miRNA regulatory network that is of interest to him. In Section 4 we explain how we adopt the use of treemaps for the visualization of the GO. In Section 5 we discuss how these two drawing techniques communicate to support our suggestion for looking at miRNA regulatory networks through the GO. Finally, in Section 6 we discuss future extensions of our work.

2 Background

2.1 From Biology to Graph Theory

Here, we discuss how miRNA regulatory networks and the GO can be mathematically modeled as graphs. A graph $G(V, E)$ is a set V of vertices, which can represent objects and a set of edges $E \subseteq \{V * V\}$, which represents relations between the vertices.

The miRNA Regulatory Network. The miRNA network consists of miRNAs and protein coding genes. We also know that miRNAs regulate certain genes. We can construct a graph that represents this information: The set of vertices V is the set of miRNAs and genes. Then, for every regulation, we insert an edge that joins the two respective vertices. These edges have a weight, the score that is associated to the respective prediction. Since no direct relations exist between two miRNAs or two genes, this is a bipartite graph where the two partitions $V1$ and $V2$ are: $V1$: the set of miRNAs, $V2$: the set of genes.

Gene Ontology. The GO stores terms that describe information related to the biological role of genes. It has three main branches with different aspects of genes functionality. The first branch, "biological process" stores information about the various broad biological goals that are served by genes. The second branch, "molecular function" stores terms that describe various tasks that are performed by individual gene products. Typically, series of molecular functions form biological processes. Finally, the third branch, the "cellular component" stores various sub-cellular locations where gene products operate. The GO is organized in a hierarchical manner, so the terms are placed in layers that go from general to specific. The first layers are quite general and are used only to create main categories.

Since the GO consortium is an active project, new terms are inserted and others are updated or removed. In order to keep information about the removed terms three additional sets are used, one for each main branch and obsolete terms are placed in them. This information can be modeled with a graph, where each term is represented by a vertex. At this point the existence of synonyms must be mentioned. Synonyms that describe the same term are used in the GO. For all the synonyms of a term only one vertex is used in order to keep the size of the graph small. Edges are used to declare the relations. They have to be directed to show which is the general and which the specific term. The GO hierarchical construction guarantees the absence of cycles in the graph. Thus, the GO hierarchy is a Directed Acyclic Graph (DAG). Since every term appears in exactly one branch and has no edge to the other two, the GO consists of three *DAGs* merged under a common root. The three obsolete sets are also placed under this root.

2.2 Graph Drawing Techniques

Graph drawing has applications in many different fields such as computer and social networks, E-R models or any other network. In [12] a large number of graph drawing algorithms are described. Here we review appropriate drawing techniques for the visualization of miRNA regulatory networks and the GO.

Visualization of miRNA Networks. Bipartite graphs are typically drawn based on the idea of the placement of each partition in a column. The main problem is to compute the ordering of vertices in the partitions that gives the lowest number of crossings. This is an NP-complete problem [13] even if the ordering of one partition is fixed [14] so various heuristics have been proposed to determine an ordering which gives a small number of edge crossings [15,16,17,18]. The size of a miRNA regulatory network (some thousands of genes and some hundreds of miRNAs) makes the adoption of such a technique inefficient. Efforts are focused on the prediction of regulations and not the network itself, so no special techniques seem to be available for miRNA regulatory networks. Here we introduce an interactive technique using circular drawings that allows the user to see only the portion of the network that interests him and not the full network.

Visualization of the GO. DAGs (like the GO) are typically drawn using algorithms that try to display their hierarchical structure. The polyline drawing convention is the basis for such algorithms, [19,20] and the main idea is to create hierarchical layers, [21,22,23]. Another approach to the visualization of hierarchies are treemaps [24,25,26], where vertices that are low in the hierarchy are placed over those that are in the first layers in the hierarchy. This method works for trees, but can handle DAGs as well, since they can be transformed into trees. Tools like GOfish [27], Amigo[28], CGAP[29] and dagEdit [30], visualize the GO with expanding lists or trees, which can manipulate small hierarchies, but face difficulties for large hierarchies. Treemaps which are a space-efficient manner for displaying hierarchies and also support fast and easy navigation have more recently been used for the visualization of the GO [31,32]. In both of them additional information e.g. about the number of genes related to the GO terms is used and weights are assigned to the terms based on this, offering a visualization of the part of GO that is related to the available information. We also use treemaps but choose to display the GO without additional information. This way we have a general-purpose browser for the whole GO. One has still access to the relations between genes and GO terms through interactive methods.

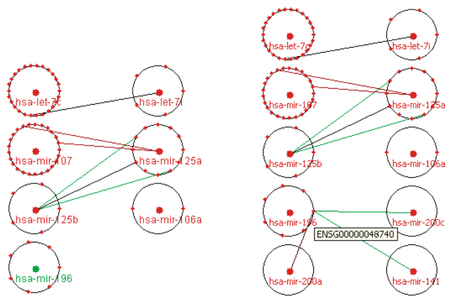
3 Drawing the miRNA Network

For this network, the main requirement is that the bipartite structure must be clear. Also since a prediction-score is assigned to each regulation we also want to see this information. Small bipartite graphs are typically drawn using two columns, but the size of the network and the difference in the cardinalities of the two partitions (miRNAs are expected to be 1-10% of the number of genes) inhibit the adoption of such a technique. Furthermore, due to the large degree of many miRNA-vertices, there are many edge crossings and the result is very cluttered. In addition, most of our available drawing area is occupied by the GO. as we discuss later, thus the space that is available for the miRNA regulatory network is limited. While ideally one would like to have a visualization of the whole regulatory network where the bipartite structure and other properties are clear, due to the aforementioned problems this can not be achieved. But we can circumvent these difficulties with an interactive visualization technique, where the user specifies what he wants to see.

In order to draw the miRNA regulatory network interactively we use two lists, and allow the user to select genes and miRNAs. If some miRNA is selected it is drawn using a small circle and all genes that are regulated by it are placed on the periphery of a circle whose center is the miRNA. This way, we avoid drawing lines for many edges and obtain a much clearer result. An edge has to be drawn only if some gene has been previously drawn. This happens when the user selects to display some miRNA which regulates a gene that is also regulated by some other previously selected miRNA (and is drawn on its periphery). We also suggest that dots for miRNAs are a bit larger than genes, to further clarify the bipartite structure. The miRNAs are placed in two columns, in the order they are selected. The first miRNA in the left column, the second in the right, the third in the left under the first and so on. The names of the miRNAs are printed in the interior, while genes names are shown when the mouse is over them. An example is shown in figure 1(a).

The user can also select some gene from the respective list. If the gene has already been drawn on the periphery of a cycle, all miRNAs that regulate it and have not been yet drawn are placed in the lists and the respective lines are drawn to indicate the relationship. If the gene has not been already drawn, all regulating miRNAs are drawn and the gene is placed on the periphery of the first of them. The newly added miRNAs are not expanded (no genes are placed on their periphery and no lines are added) in order to keep the image as clear as possible. This is shown in figure 1. the network of fig. 1 by selecting gene ENSG00000048740 leads to the drawing in fig. 1(b).

For genes with many related miRNAs a large number of new cycles can be produced. Thus, an option to remove some miRNA from the network is available on right click. This drawing method offers many advantages. First of all, the user sees only the portion of the network that is of interest to him. Secondly, since this image is being created in a step-by-step interactive process, it is easy to remember where each term is and thus, maintain a mental map of the network. The bipartite structure of the network is also obvious: miRNAs appear in the



(a) Initial display. (b) Expansion.

Fig. 1. The sub-network, after selecting hsa-let-7c, hsa-let-7i, hsa-mir-107, hsa-mir-125a, hsa-mir-125b, hsa-mir-106a, hsa-mir-196 and the expansion after selecting gene ENSG00000048740

middle of the cycles and are a bit larger, while genes appear on the periphery. Due to the convention that genes on the periphery are regulated by the miRNA in the center, many edges are implied and do not have to be drawn. With some other drawing technique which displays all edges for the part of the network in Figure 1, 82 more edges would have to be drawn. Finally, this method is very fast, since every selection simply adds a few elements to the already constructed image. An equivalent gene-oriented visualization is available as an option with the roles of miRNAs and genes interchanged.

Weighted edges
Every prediction has an associated score, which represents its "strength", so it is important to show this information. To this end, the median value of the scores for the edges is computed. All edges that have score below the median are colored in red and the brightness reflects the value. The lower the score, the brighter the color. For edges that have score equal to the median black is used, while edges with higher score are colored using green where again, the brightness reflects the value. Since viewing edges is important, we have added some features. By clicking on a vertex, all the associated edges which are initially visible are hidden. So even if a complex image has been created, the user has the ability to unclutter it. By re-clicking, the edges are revealed. Similarly, for the edges that are implied by the placement of vertices on the periphery of a cycle, an option to show them is also available(Fig 2).

4 Drawing the Gene Ontology DAG

In order to display the GO DAG efficiently three aspects must be considered: a) display the whole graph, b) maintain the hierarchy, c) traverse through the graph easily. The GO DAG currently contains about 20000 terms. Any traditional graph drawing technique would fail to draw such a large graph efficiently within

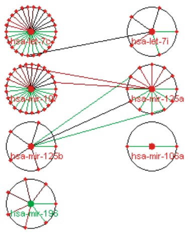
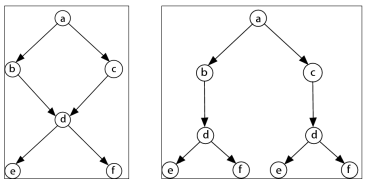


Fig. 2. Figure 1 including in-cycle edges

the limited area of a computer monitor. The hierarchical structure though, allows for the decomposition to a hierarchical tree. This process creates a tree with even more vertices but enables us to use treemaps which is among the most space-efficient techniques to draw large trees.

4.1 Converting a DAG into a Tree

The method that decomposes a DAG to a tree is based on the placement of multiple copies for vertices with many in-coming neighbors, since these are the vertices that destroy the tree structure. Fig. 3 shows an example.



(a) DAG (b) "equivalent" tree

Fig. 3. DAG to tree conversion

First we create the root of the tree which is the (unique) source of the DAG. Then for every vertex v in the DAG, a recursive method *insertNode* is called. This method inserts the necessary copies of a vertex in the tree after ensuring that all the ancestors have been inserted. If v has been processed *insertNode* simply returns, otherwise it adds all the incoming neighbors of v in the tree one by one with recursive calls. Once a recursive call for an incoming neighbor u terminates, the necessary copies of u have been inserted in the tree. Now we can insert copies of v to be children of u as many times as the copies of u are. Repeating this for every parent of v in the DAG adds the necessary copies in the tree. For fast access to the many copies of a vertex we use a hashtable where the key is a vertex and the value is a list with all of its copies.

```
DAG2TREE( Directed acyclic graph $G$)
 1  Hashtable $vertexCopies$
 2  $treeRoot = G$.vertexWithInDegree0
 3  $vertexCopies$.put($treeRoot, treeRoot$)
 4  for each vertex $v$ in $G$
 5     do INSERTNODE($v$)
```

```
INSERTNODE($v$)
 1  if not $vertexCopies$.get($v$).isEmpty() //if inserted
 2     then
 3          return
 4  for each incoming neighbor $w$ of $v$
 5     do
 6         INSERTNODE($w$)
 7         for each $w\_copy$ in $vertexCopies$.get($w$)
 8            do
 9                make a new copy $v\_copy\ of\ v$
10                insert $v\_copy$ to be a child of $w\_copy$
11                append $v\_copy$ in $vertexCopies$.get($v$)
```

The hashtable performs the search operation in constant time ($O(1)$). Hence, the cost for calling $insertNode$ for one vertex in the DAG, is linear to the number of ancestors of the respective copies in the resulting tree, or equivalently to the number of incoming edges for all copies of the vertex, plus the cost of potential recursive calls. Since $insertNode$ is called for every vertex creating different edges, the overall complexity is proportional to the sum of the incoming degrees of all vertices, or the number of edges in the tree. So the overall complexity is $O(E_{tree})$ where E_{tree} is the number of edges in the tree. This process leads to a tree with $\sim$ 100.000 vertices. The careful management of synonyms which was discussed in Section 2.1 pays back here: If all synonyms were represented with different vertices, the resulting tree would have $\sim$ 150.000 nodes.

4.2 Treemaps

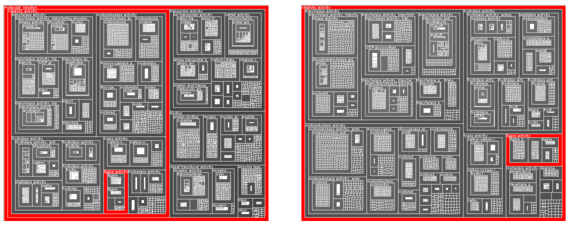
Treemaps were introduced by Johnson and Shneiderman in 1991 [24,25]. In treemaps, every node of the tree is represented by some rectangular area. The main idea is to place each vertex on top of its parent's rectangle starting from the root and proceeding in a depth-first-search way. This way, layers of hierarchy are created. On the upper, visible layer only the leaves appear.

```
TREEMAP( vertex v, rectangular area area)
 1  v.area=area;
 2  divide $area$ to the children of $v$
 3  for each child $c$ of $v$
 4     do TREEMAP(c,c$'s$ computed area )
```

```
CALLTREEMAP(treeRoot,drawingArea)
 1  TREEMAP(treeRoot,drawingArea)
```

If instead of the root some other vertex is used for calling the algorithm, the subtree rooted under this vertex is shown. This means that the algorithm itself can be used for zooming-in and out. For the space division in step 2 various approaches have been proposed. The original idea, *"slice and dice"* [24,25]was to divide the area to slices with size proportional to some metric defined for the vertices. This metric typically is the weight in case of weighted trees and the number of leaves in the subtree for non-weighted trees. For better visualization results the orientation of the slices alters from horizontal to vertical between subsequent layers. A drawback is that *slice and dice* tends to create long and thin rectangles. In an attempt to avoid this, Bruls et. al. proposed *squarified treemaps* [26]. They divide the area using a heuristic which aims to create rectangles as close to squares as possible. This additional constraint usually alters the placement of vertices while zooming the tree in and out. In [24] the authors also propose nesting to show internal vertices and the hierarchical structure. An additional advantage of nesting is that the unoccupied space can be used for placing labels. The thin slices of *slice & dice* create an obstacle in this concept. We support nesting in the following manner: Whenever some area is assigned to a vertex, a small border is placed on its boundary and only its interior is given as drawing area for the children. The user has the option to set the size of the border. We have chosen to use the *squarified* approach since the results of *slice & dice* for large trees are relatively poor. We also manage to keep track of an interesting vertex after zooming in and out by highlighting the path from the root of the tree to it (Fig 4). Larger figures are available at*www.ics.forth.gr/~reczko/isbmda06*

We also provide an option to show only a number of hierarchical layers in order to hide small, hardly visible rectangles that are produced for vertices in the deeper layers. Zoom-in allows the user to access deeper layers. Zoom-in is straightforward and is performed when some term is left-clicked. For zooming out we provide three options, which become available on right-click. The user can move one or two layers higher. Alternatively, he can move up to the root of the tree and display the whole GO.



(a) *"ligase activity"* in the *"molecular function"* branch

(b) *"ligase activity"* in the *"catalytic activity"* branch

Fig. 4. Treemaps of subtrees of the GO. The term *"ligase activity"* is highlighted with a red border.

4.3 Integrated Visualization of the miRNA Regulatory Network and the Gene Ontology

We have discussed how to visualize an interesting part of the miRNA regulatory network and how to explore the GO with treemaps. Here, we discuss how one can combine them using information about GO terms that are related to genes. miRNAs are also related to GO terms through their regulating genes.

Gene Ontology Based Analysis of the miRNA Regulatory Network. Every gene in the network is related to at least one term in the GO. So the first requirement is to see the genes that are related to some specific term. In order to ask this question, the user has to select some term and choose the "Show Genes" option. All genes that are related to the selected GO term and belong to the visible portion of the network on the left are highlighted (colored in green). An instance is illustrated in Fig. 5.

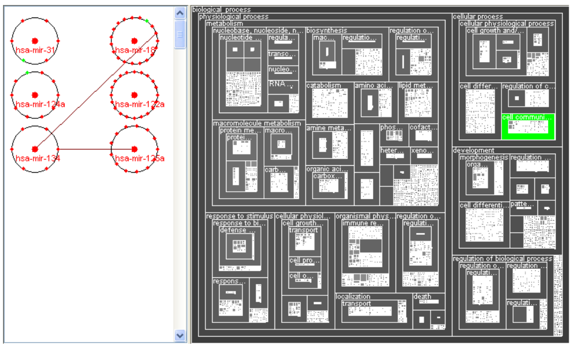


Fig. 5. Highlighted genes on the cycles on the left(87470, 68383, 101384) are related to "cell communication"

With the drawing method used for the miRNA regulatory network, only a portion of the regulatory network is drawn. This way the user sees only the part of interest and not the whole complex network, but some of the genes that are related to some GO term may not be visible. These additional genes can be shown with the option "Expand & show Genes".

As an example for the miRNAs known to be related to the category "neurogenesis" the subset targeting genes in the category "axonogenesis" is highlighted in figure 6.

Search for GO Related Properties of the miRNA Regulatory Network. Complementary, the user can search for GO properties based on genes or miRNAs. If some gene is selected, the GO terms that are related to it are colored in bright green. Because of the option to show only some layers some terms may not be visible . In this case, the closest visible ancestor is highlighted

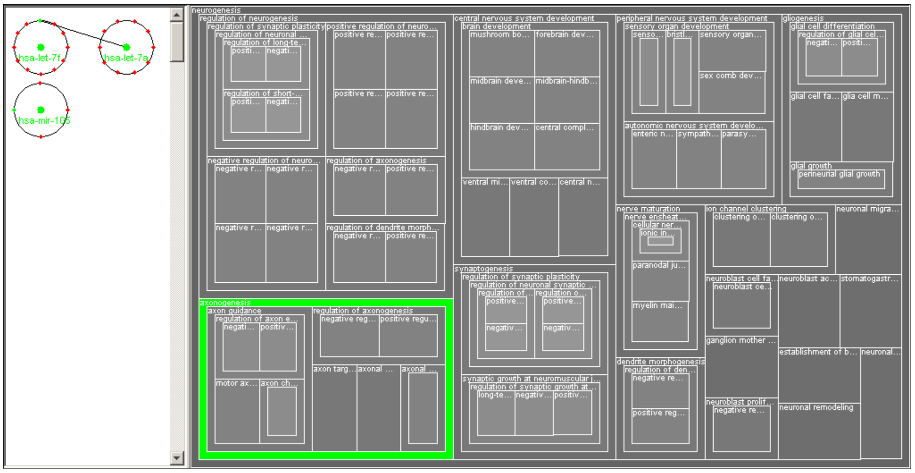


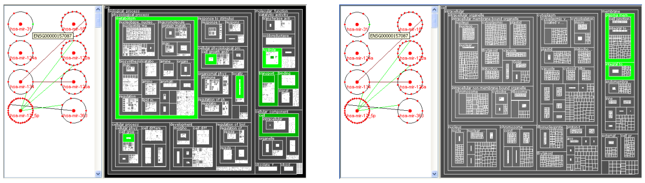
Fig. 6. Highlight visible genes and miRNAs related to *"axonogenesis"*

using dark green (Fig. 7(a)). If the branch is of interest, one can zoom in and identify the related term, which is in bright green(Fig. 7(b)).

An interesting relation between a miRNA and the GO is implied by the GO terms that are related to the genes it regulates.

Selecting a miRNA, the set of GO terms that are related to at least one of the genes it regulates are colored. The coloring is the same as in the previous case. For *hsa-mir-17_5p*, the miRNA with the maximum number of predicted target genes, the result appears in Figure 8.

For miRNAs with many target genes, a large portion of the GO is highlighted. The user can see the terms that are related to the genes, but does not know the number of genes that are related to these terms. So, we support an option to iterate through the genes that are regulated by some miRNA and display for every gene the related GO terms with animated highlighting. Now, the user can see how often some term is highlighted and is able to observe emerging patterns of reoccurring terms. With this feature it became obvious that 17 out of the 21 genes that are regulated by *hsa-mir-98* are related to the category *"binding"* or a even more specific term, emphasizing the likelihood of those predictions.



(a) (b)

Fig. 7. Left: All terms related to gene *ENSG00000157087* (PLASMA MEMBRANE CALCIUM-TRANSPORTING ATPASE 2). Right: After zooming-in *"cell"*.

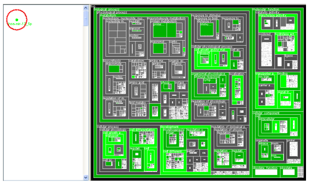


Fig. 8. All GO terms that are related to *hsa-mir-17_5p*

5 Discussion

We have introduced a novel combination of visualization methods to explore the relation of microRNAs and their target genes. The interactive construction of miRNAs and functionally related targeted genes will be of great use in discovering the functional role of many miRNAs. The treemap based browser might establish a standard 'mental map' of the GO and can independently be used to visualize any GO related information. Currently we support data in the format presented in [2] but the support for other prediction methods is planned.

References

1. R.C. Lee et. al. C. elegans heterochronic gene lin-4 encodes small rnas with antisense complementarity to lin-14. *Cell*, 75:843–854, 1993.
2. B. John et. al. Human microrna targets. *PLoS Biology*, 2:e363, 2004.
3. A. J. Enright et. al. microrna targets in drosophila. *Genome Biol.*, 5:R1, 2003.
4. A. Stark et. al. Identification of drosophila microrna targets. *PLoS Biology*, 1:E60, 2003.
5. B.P. Lewis et. al. Prediction of mammalian microrna targets. *Cell*, 115:787–798, 2003.
6. S. Pfeffer et. al. Identification of virus encoded micrornas. *Science*, 304:734–736, 2004.
7. M. Kiriakidou et. al. A combined computational-experimental approach predicts human microrna targets. *Genes dev.*, 18:1165–1178, 2004.
8. D. Grün et. al. microrna target predictions across seven drosophial species and comparison to mammalian targets. *PLoS Comp. Biol.*, 1:e13, 2005.
9. A. Krek et. al. Combinatorial microrna target predictions. *Nature Genetics*, 37:495–500, 2005.
10. M. Rehmsmeier et. al. Fast and effective prediction of microrna/target duplexes. *RNA*, 10:1507–1517, 2004.
11. `http://www.geneontology.org`.
12. G. Di Battista et. al. *Graph Drawing: Algorithms for the Visualization of Graphs.* Prentice Hall, 1999.
13. M.R. Garey and D.S. Johnson. Crossing number is np-complete. *SIAM J. Algebraic Discrete Methods*, 4:312–316, 1983.
14. P. Eades et. al. On an edge crossing problem. In *Proc. of the Ninth Australian Computer Science Conference*, pages 327–334. Australian National University, 1986.

15. P. Eades and D. Kelly. Heuristics for drawing 2-layered networks. *Ars Combin.*, 21-A:89–98, 1986.
16. P. Eades and N. Wormald. Edge crossings in drawings of bipartite graphs. *Algorithmica*, 11(4):379 – 403, 1994.
17. E. Măkinen. Experiments on drawing 2-level hierarchical graphs. *Inter. Journ. Comput. Math.*, 36:175–181, 1990.
18. M. May and K. Szkatula. On the bipartite crossing number. *Control Cybernet*, 17:85–98, 1988.
19. G. Di Battista and R. Tamassia. Algorithms for plane representations of acyclic graphs. *Theoret. Computation Scien.*, 61:175–198, 1988.
20. G. Di Battista et. al. Constrained visibility representations of graphs. *Informat. Process. Letters*, 41:1–7, 1992.
21. K. Sugiyama et. al. Methods for visual understanding of hierarchical systems. *IEEE, Trans. Syst. Man. Cybern*, 11(2):109–125, 1981.
22. D. J. Gschwind and T. P. Murtagh. *A Recursive Algorithm for Drawing Hierarchical Directed Graphs.* Department of Computer Science, Williams College, 1989.
23. M.J. Carpano. Automatic display of hierarchized graphs for computer aided decision analysis. *IEEE, Transact. Syst. Man. Cybern*, 10(11):705–715, 1980.
24. B. Johnson and B. Shneiderman. Treemaps: a space-filling approach to the visualization of hierarchical information structures. In *Proc. of the 2nd Intern. IEEE Visualization Conference*, pages 284–291, Oct. 1991.
25. B. Shneiderman. Tree visualization with tree-maps: a 2d space-filling approach. *ACM Transactions on Graphics*, 11(1):73–78, Sept. 1992.
26. D. M. Bruls et. al. Squarified treemaps. In *Proceedings of the joint Eurographics and IEEE TVCG Symposium on Visualization*, pages 33–42, October 2000.
27. `llama.med.harvard.edu/~berriz/GoFishWelcome.html`.
28. `www.godatabase.org/cgi-bin/amigo/go.cgi`.
29. `cgap.nci.nih.gov/Genes/GOBrowser`.
30. `sourceforge.net/project/showfiles.php?group_id=36855`.
31. E. H. Baehrecke et. al. Visualization and analysis of microarray and gene ontology data with treemaps. *BMC Bioinformatics*, 5, 2004.
32. P McConnell et. al. Applications of tree-maps to hierarchical biological data. *Bioinformatics*, 18(9):1278–1279, 2002.