An Interpretation of Biological Metabolites and their Reactions Based on Relation Degree of Compound Pairs in KEGG XML Files

Myungha Jang
Department of Computer Science and Engineering, Ewha Womans University, Seoul, South Korea
Department of Natural Sciences and Mathematics, Wesleyan College, Macon, GA, United States.
Email: myunghajang@gmail.com

Jiyoung Whang
Department of Computer Science and Engineering, Ewha Womans University, Seoul, South Korea
Email: jiyoung88@ewhain.net

Coleen S. Lewis
Department of Natural Sciences and Mathematics, Wesleyan College, Macon, GA, United States.
cslewis@wesleyancollege.edu

and Hyun S. Park(*),
Institute of Bioinformatics, Ewha Womans University, Seoul, Korea
& Institute of Bioinformatics, Macrogen Inc., Seoul, Korea
Email: neo@ewha.ac.kr

Abstract*—Biological pathways can be characterized as networks and grouped as metabolic pathways, gene regulatory networks, gene interaction networks and signal transduction pathways. It is important that edge crossings in biological pathway diagrams are kept to a minimum by strategic placement of vertices for simplification. The basic graph layout technique deals with the problem of positioning the vertices in a way to maximize understandability and usability in a graph. However, when dealing with a very large number of nodes in a global metabolic pathway, strategically positioning vertices is not enough. Understanding the properties of the metabolites and the biological reactions is crucial to pave the way for the formulation of new strategies for further development of automatic layout for global metabolic pathway. In this paper, we provide a statistical analysis of metabolic reactions based on the parsing result of publicly available XML files in KEGG. The analysis leads to a new node-abstracting scheme according to the newly defined concept, 'relation degree of compound pairs'. The concept would suggest valuable information to software developers for graph-based visualization tools for analyzing networks in cell biology.

***Index Terms*—metabolic pathway, parsing, XML, relation degree, drawing algorithm, edge crossing, statistics of metabolites**

I. INTRODUCTION

Recent advances in large-scale experimental technologies have resulted in an accumulation of unmanageable biological pathway data. Due to the large number of possible interactions, there is a great need for computer-assisted tools to manage the experimental data.

Biological pathways can be characterized as networks and grouped as metabolic pathways, gene regulatory networks, gene interaction networks and signal transduction pathways. Especially, graphical diagrams are intuitively helpful in understanding metabolic pathways. The Kyoto Encylocpedia of Genes and Genomes (KEGG, http://www.genome.ad.jp/kegg) [1], a valuable resource in this research, provides a metabolic pathway database that contains graphical diagrams in the form of wiring diagrams. Original KEGG Pathway diagrams are manually drawn and stored as bitmap image files. These single pathways are displayed as interactive image maps with links to additional information on enzymes and to adjacent pathways.

However, metabolic studies, used to be dedicated to a single pathway, recently began to focus on the multiple pathways [2]. KEGG now provides the KEGG Metabolism Atlas by manually combining about 120 existing metabolic pathway maps as shown in Figure 1 [3]. However, static approach to represent metabolic pathway diagrams still offers no flexibility.

While the computer-assisted automatic layout scheme

Based on "A Statistical Analysis of Relation Degree of Compound Pair on Online Biological Pathway Databases", by Myungha Jang, Arang Rhie, Jiyoung Whang, Sanduk Yang and Hyun S. Park, which appeared at the Third International Conference on Ubiquitous Information Management and Communication, ACM Jan 15-16, 2009, Suwon, Korea.

*Corresponding author: Hyun S. Park

© 2010 ACADEMY PUBLISHER
doi:10.4304/jsw.5.2.187-194

is preferred, NP-hard problems are still left to find solutions for the automatic layout of biological pathways with respect to aesthetics. A current state-of-the-art survey in this field reveals that research and development in automatic layout of biological pathway maps are still in infancy, at least from an aesthetic perspective. [4-8]

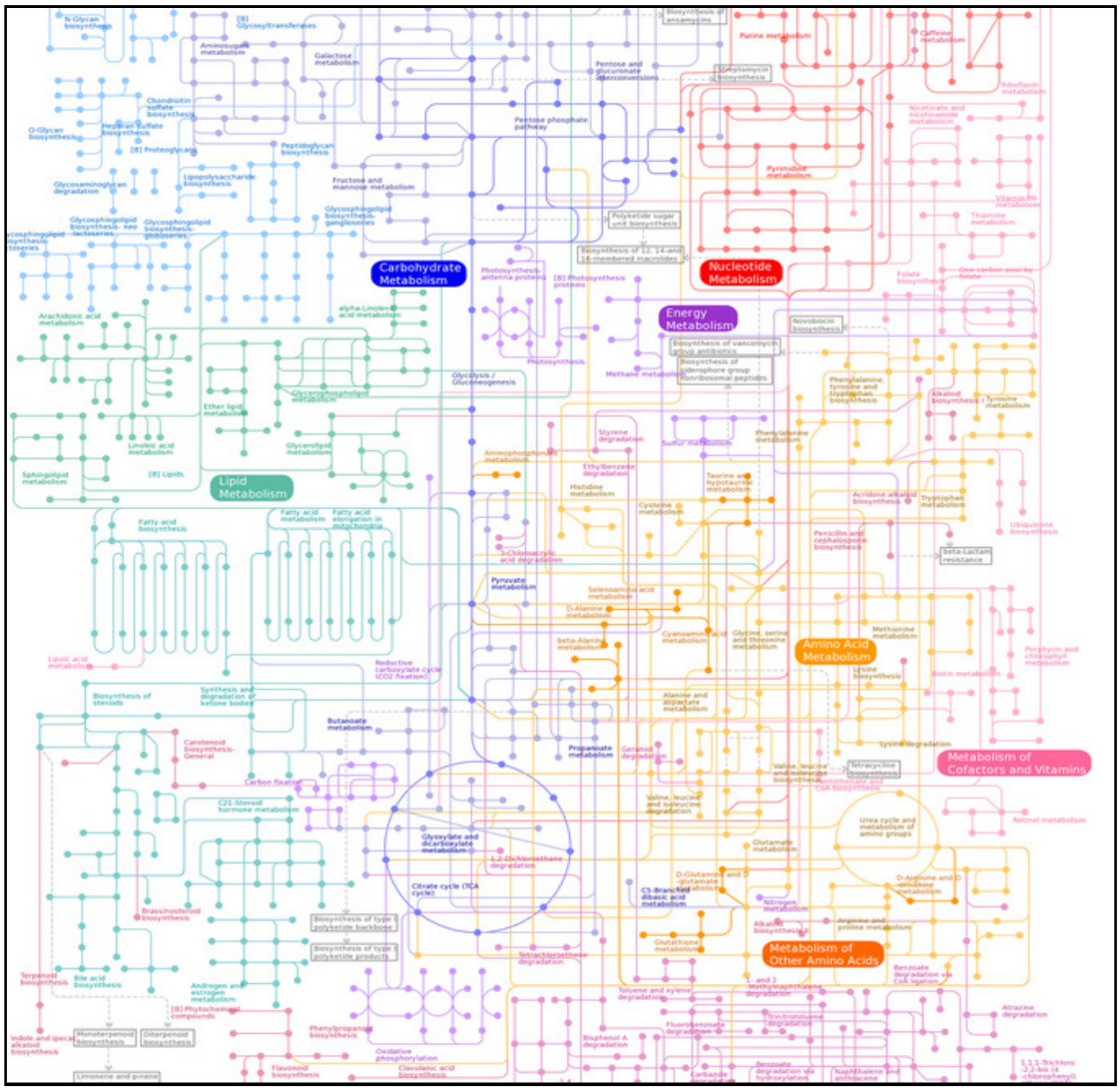

Figure 1. KEGG metabolism Atlas (http://www.genome.jp/kegg/atlas/metabolism/) is a new graphical interface to the KEGG suite of databases.

Aesthetic criteria have been developed to increase the comprehensibility of automatically produced graphs. Such criteria include: minimizing the total number of edge crossings, minimizing the area of the drawing, displaying the symmetries of the graph, and minimizing the smallest angle between two edges adjacent to the same vertex. Through several experiments however, it has been proven that reducing the number of edge crossings is the primary aesthetic to consider [9]. Thus, it is crucial for software developers to understand the properties of the metabolites in the reactions in order to strategize approaches to produce a better layout.

When parsing and analyzing documents of KGML, the KEGG Markup Language is a good method of understanding the properties of metabolites and their reactions. In this paper, we provide a parsing result of a publicly available database with its statistic analysis, KEGG, using our KGML parser. Additionally, we suggest a node-abstracting scheme according to the concept of 'relation degree of compound pairs' we defined in this paper and investigated a group of compounds which are exclusively related to each other.

II. IMPLEMENTATION OF KGML PARSING MODULE

The Extensible Markup Language (XML) is a general-purpose specification for creating custom markup languages by allowing users to define the mark-up elements. More specifically it is to help KEGG use an XML format for the metabolic pathways discussed in this paper. The molecular reaction network is the most unique data object in KEGG, which is stored as a collection of pathway maps in the PATHWAY database.

© 2010 ACADEMY PUBLISHER

As of KEGG Release 47.0+/07-01 of July 08, 94,068, KEGG pathways are generated from 372 reference pathways. Figure 3 is an example of well-formed XML document of the KEGG Map00630, corresponding to Figure 1. Each object is identified by the KEGG object identifier, consisting of a five-digit number prefixed by an upper-case alphabet, such as C00101 (Chemical compound: 00101), and R01394 (Reaction: 01394), or prefixed by a 2-4 letter code for PATHWAY such as map00630.

In dealing programmatically with an XML document, it must first be parsed to enable access to the data in the document by the application using the parser. The KGML Parser module is dedicated for parsing KEGG Markup Language (KGML), an XML representation of KEGG pathway maps. The parsing module of KEGG XML files has been implemented on Eclipse platform and has been deployed with Sun's J2SE Reference Implementation. Three classes, KGMLParser, Handler, and DBConnector represent the domain model of the system, based on MVC (Model-View-Controller) model. A great challenge encountered with this is to develop automated and tractable techniques to ensure static-type safety and ultimately optimize the program. This includes basic reasoning tasks involving complex constructions [10].

III. PARSING RESULTS AND ANLYSIS OF METABOLIC PATHWAY XML DOCUMENTS

A. In/Outgoing Edges of Compounds

In a global pathway, positioning vertices impact the number of edge crossings in a layout even more than in a single pathway. Especially, compounds with a higher number of in/outgoing edges have greater influence on automatic graph layout schemes. Therefore, perceptive strategies in dealing with those compounds are required in order to achieve a better readable visualization of the graph from a graph-theoretic perspective.

In metabolic pathways, each compound could be a substrate or a product in several reactions of pathways in metabolism. An edge serves as a connection between a substrate and a product in a reaction. The numbers of in/outgoing edges of each compound were counted by searching reaction data that we obtained from parsing KGML files.

Implementation

Metabolite reaction information obtained from KGML documents are saved in a table named 'reaction' in database. Table 1 shows its table schema. The composite primary key consists of 'mapID' and 'rID(reaction ID)'. *'Reversible'* field states if a reaction is reversible. While sub1 and prod1 are required to have values, the others could be null if a reaction only has one substrate and one product.

reaction						
mapID	**rID**	sub1	sub2	prod1	prod2	reversible
Char(9)	Char(10)					Boolean

Table 1. The schema of table 'reaction', which contains metabolite reaction information parsed from KGML documents.

A snapshot of the program code for counting in/outgoing edges of each compound is shown in Figure 2. If the compound engages in a reaction as a substrate, the number of product compounds in the reaction becomes the number of outgoing edges, and vice versa.

```
/**
 *  @param compoundID
 *  @return the number of in/outgoing edges
 */
public int countEdgeNum(String cpID)
{
    String query = "select distinct * from reaction where"
      + "where (subs1 = '"+cpID+"') or (subs2 = '"+cpID+"')'
      + " or (prod1 = '"+cpID+"') or (prod2 = '"+cpID+"')";
    ResultSet rs;
    int count = 0;
    try {
        rs = db.getQueryResult(query);
        while(rs.next())
        {
            if((rs.getString("subs1")!=null
                && rs.getString("subs1").equals(cpID)) ||
                (rs.getString("subs2")!=null
                && rs.getString("subs2").equals(cpID))) {
                if(rs.getString("prod1")!=null) count++;
                if(rs.getString("prod2")!=null) count++;
            }
            if((rs.getString("prod1")!=null
                && rs.getString("prod1").equals(cpID)) ||
                (rs.getString("prod2")!=null
                && rs.getString("prod2").equals(cpID))) {
                if(rs.getString("subs1")!=null) count++;
                if(rs.getString("subs2")!=null) count++;
            }
        }
    }catch(Exception ie){
        ie.printStackTrace(); }
    return count;
}
```

Figure 2. A code snapshot of the method for finding the number of in/outgoing edges of a compound

Analysis

In total, 4382 compounds that appear at least once in a single pathway were observed in this analysis. Figure 3 shows a graph of a number of compounds according to the number of in/outgoing edges. 2862 compounds, which take up 65.3% of the total compounds, take part in less than two reactions. The compound with the greatest number of edges is C00024, with 76 edges connected.

© 2010 ACADEMY PUBLISHER

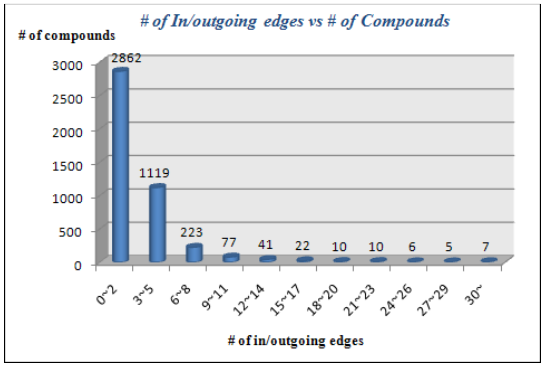

Figure 3. The number of compounds versus the range of the number of edges

B. Relation Degrees of Compound Pairs

Definition

As was mentioned in Section III.A, approximately 70% of all compounds have less than two edges. To narrow down the analysis within those compounds, we define the term *relation degree of a compound pair* as the number of compounds that appear through existing reactions between two compounds. Figure 4 shows an example with a reference pathway in KEGG. There are 5 compounds connected between two compounds C11522 and C05442. Those are C11523, C15782, C15783, C08821, C01753, and they have two in/outgoing edges, which means that they appear only once in one pathway. In this case, we state that the relation degree of the compound pair (C11522, C05422) is 5 and it is denoted as 'RD(C11522, C05422) = 5'.

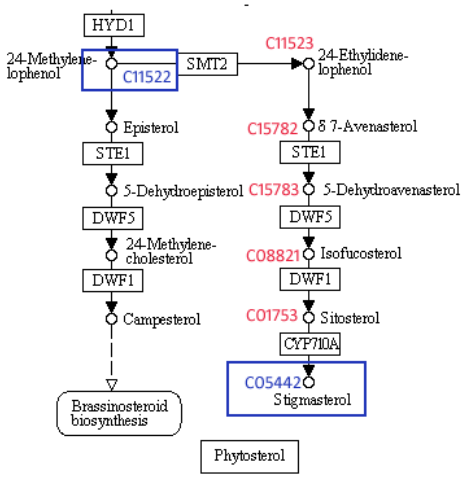

Figure 4. KEGG reference pathway of 'Steroid biosynthesis' shows an example of the relation degree of the compound pair (C11522 ,C05442).

Implementation

To express a compound's link to other compounds in a reaction, a data structure called *CompoundNode* was defined. An object of CompoundNode has two linked objects. The compound to the left is termed 'leftCompound' and the one to the right, 'rightCompound'. Since we are only dealing with compounds with less than two edges here, the terms leftCompound and rightCompound are adopted simply to describe two connected compounds. Note that those terms do not imply any direction of the actual reaction. A chain of CompoundNode objects is contained in a *CompoundGroup*. CompoundGroup object takes a pair of two *CompoundNode* objects as a constructor parameter and consists the group with connecting elements between the pair. Figure 5 and Figure 6 show the snapshots of a part of each code.

```
public class CompoundNode {
    private CompoundNode leftCompound;
    private CompoundNode rightCompound;
    private String cpdName;
    private int linkedNum;
    private boolean hidden;
    private boolean visited = false;

    public CompoundNode(String cpdName) {
        this.cpdName = cpdName;
        linkedNum = 0;
    }
    public void setVisited() {
        visited = true;
    }
    public boolean insert(CompoundNode compound){
        if(leftCompound == null)
        {
            linkedNum++;
            leftCompound = setToRightCompound(compound);
            return true;
        }
        else if(rightCompound == null)
        {
            linkedNum++;
            rightCompound = setToLeftCompound(compound);
            return true;
        }
        return false;
    }
    public boolean hasMoreCompounds() {
        if(this.hasRightCompound())
            return true;
        else return false;
    }
```

Figure 5. A code snapshot for CompoundNode.java

© 2010 ACADEMY PUBLISHER

```
public class CompoundGroup {

    private CompoundNode cpdStart;
    private CompoundNode cpdEnd;
    private int relationDegree, groupNum, size, index;
    private CompoundNode[] cpdGroupSet;
    public CompoundGroup(CompoundNode cpd1, CompoundNode cpd2)
    {
        if(cpd1.hasRightCompound()) {
            cpdStart = cpd1;
            cpdEnd = cpd2;
        }
        else {
            cpdStart = cpd2;
            cpdEnd = cpd1;
        }
        index = -1;
        consistGroupSet();
    }
    public boolean hasMoreCompounds() {
        if(index < size)
            return true;
        else
            return false;
    }
    public CompoundNode nextCompound() {..
    public CompoundNode getCompound(int i) {..
    public CompoundNode getCompoundPairA() {..
    public CompoundNode getCompoundPairB() {..
    public int getRelationDegree() {..
    public int getGroupNum()..
    private void consistGroupSet()..
}
```

Figure 6. A code snapshot for CompoundGroup.java

Algorithm

Compounds with less than two edges form a candidate group for pairing that is eligible to have a relation degree. Starting with one compound in the candidate group, it links itself with compounds to relate in a reaction. Those connected compounds recursively search a link until it meets a threshold; one of those have more than two compounds to relate (i.e. too many edges), which is the maximum number of connections allowed according to its data structure.

When both leftCompound and rightCompound meet threshold compounds, the program defines them as a pair and the number of compounds connecting between the pair becomes the relation degree of the pair. Below is a code snapshot used in the program searching for relation degree of compound pairs.

```
public void searchGroupElements(CompoundNode cpd)
{
        if(cpd == null) return;
        if(cpd.isVisited()) return;

        ResultSet rs = null;
        LinkedList<CompoundNode> cpdQueue
                    = new LinkedList<CompoundNode>();
        int count=0, cnt=0;
        cpd.setVisited();
        rs = db.getQueryResult("select * from reaction where " +
            "(subs1 = '"+cpd.getCpdName() +"') or (subs2 = '"+
            cpd.getCpdName()+"') or (prod1 = '"+cpd.getCpdName()+"') "
            + "or (prod2 = '" +cpd.getCpdName()+"')");
```

```
try {
    rs.last();
    count = rs.getRow();
if(count <= 2)  {
    rs.beforeFirst();
    while(rs.next())
    {
    if((rs.getString("subs1")!=null &&
        rs.getString("subs1").equals(cpd.getCpdName())) ||
        (rs.getString("subs2")!=null &&
        rs.getString("subs2").equals(cpd.getCpdName()))) {
 else if((rs.getString("prod1")!=null) &&
            rs.getString("prod1").equals(cpd.getCpdName()) ||
            (rs.getString("prod2")!=null &&
            rs.getString("prod2").equals(cpd.getCpdName()))) {
    //...
    }
    }

    cnt = cpdQueue.size();
    if(cpd.getLinkedCompoundsNum() == 0 && cnt == 2) {
        // case A
        cpd.insert(cpdQueue.poll());
        cpd.insert(cpdQueue.poll());
    }
    else if(cnt == 1) {
        if(cpd.getLinkedCompoundsNum() == 0) {
            // case B
            setCpdPair(cpd);        }
        cpd.insert(cpdQueue.poll());
    }
    else {      // case C
        setCpdPair(cpd);
    }
}
else {     // case C
    setCpdPair(cpd);            }
searchGroupElements(cpd.getLeftCompound());
searchGroupElements(cpd.getRightCompound());
}catch(Exception e) {
    e.printStackTrace();  }

   return;
   }
}
```

Figure 7. A pseudo code of searching for a relation degree of a pair.

In this logic, to determine whether the compound is one of a pair or one of those that is a part of a relation degree of the pair, three cases should be considered; case A, case B and case C. The cases are indicated in the comment of the pseudo code in Figure 7. In case A, when a compound does not yet have a link and two compounds are to be connected, the two compounds are linked as its leftCompound and rightCompound, and it proceeds recursively with these two compounds. In case B, a compound does not yet have a link and there is only one compound to be designated as its leftCompound, which becomes one end of the link chain. In case C, when a compound already has one link and it has more than one compound to be connected, because it does not have any available link, it is set as a part of pair. Figure 9 illustrates each case.

Analysis

While applying the algorithm explained above, we obtained a result of relation degrees for 839 pairs of compounds with an average relation degree of 2.14 as

© 2010 ACADEMY PUBLISHER

shown in figure 8. When grouped with a compound pair and compounds that consist relation degree of the pair, a total of 2503 compounds are included in searched groups, where 1366 compounds belong to compounds in relation degree and 1137 compounds are represented in pairs.

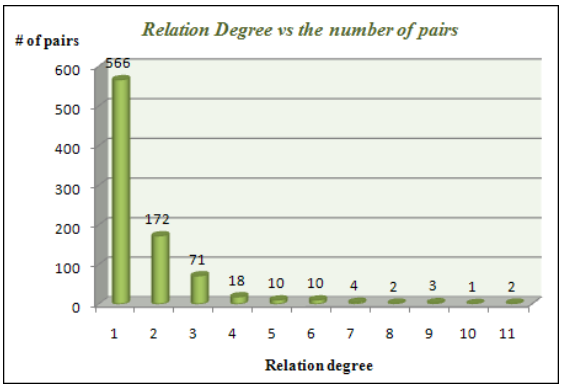

Figure 8. The number of searched groups per each Relation Degree.

IV. DISCUSSION AND FUTURE WORK

Our attempts to visualize several metabolic pathways in a single map with an automatic graph layout have not been quite successful; although it has reduced a significant number of edge crossings in the graph layout, it still resulted in a perplexing diagram. Compound pairs that have a relation degree of more than 1 are expressed with blue icons and blue edges in Figures 9. Several compound nodes that make a relation degree between a compound pair are abstracted in a blue edge connected between two compounds of a pair.

Through this work, we learned the necessity of analyzing the characteristics of metabolites and reactions to strategize approaches for our goals of accomplishing desirable layouts for automatic graphs. Therefore, using our KGML parser and algorithm, we analyzed parsing results of KEGG XML documents and obtained statistical data regarding the number of in/outgoing edges of compounds and relation degrees of compound pairs as we defined in this paper.

In this paper, biological pathway analysis has been proposed based on a public metabolic database in an XML document parsing result. This establishes a preliminary step to provide automatic layout algorithms for multiple pathways, in the future. Without the analysis of the edges of compounds in the pathway map, visualizing all the pathways globally in a single atlas map would only yield a confusing diagram. A solid algorithm of graph layout based on this statistic analysis for automatic layout in biological networks will be left for future work.

© 2010 ACADEMY PUBLISHER

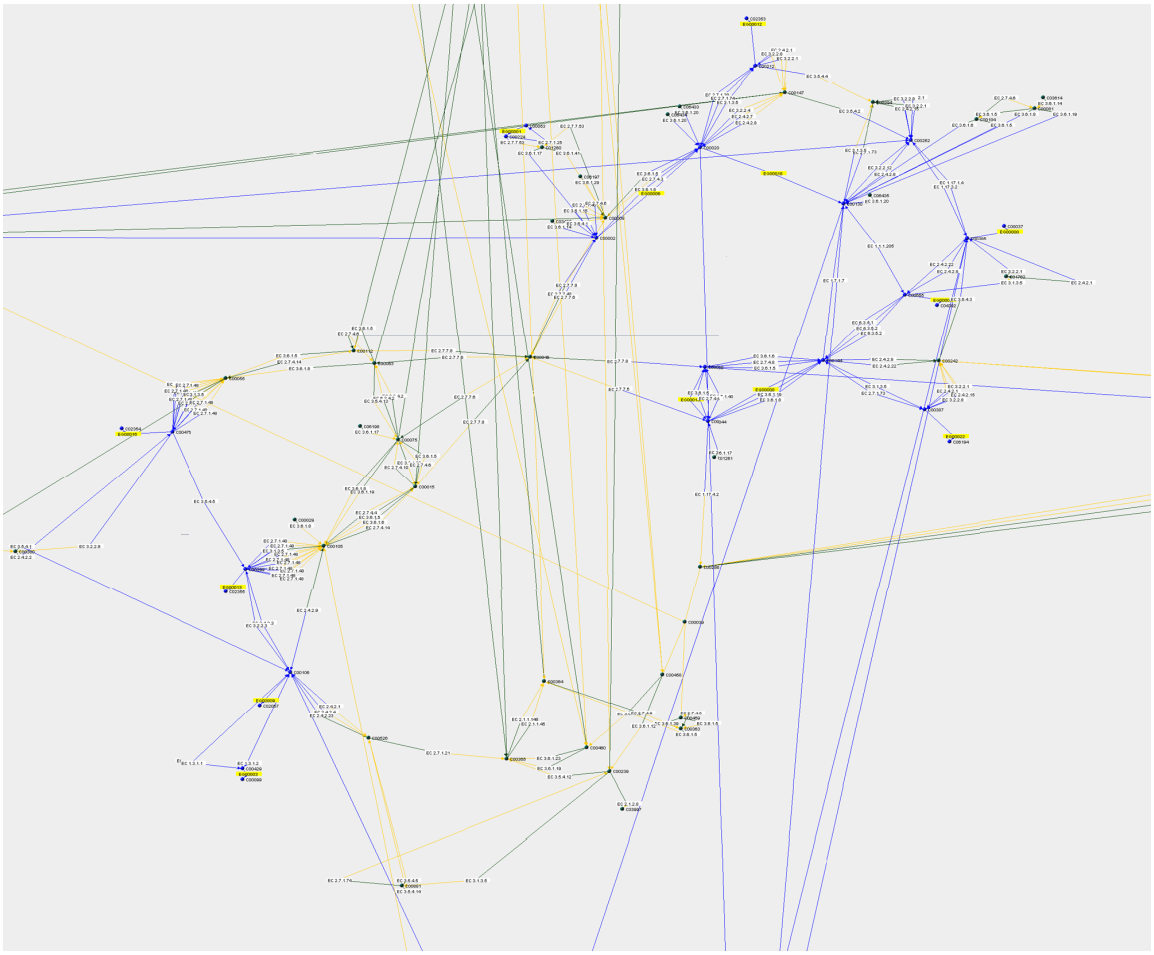

Figure 9. A Snapshot of Nucleotide Metabolism(Purine Metabolism + Pyrimidine Metabolism) Automatic Layout, nodes abstracted with relation degree of compound pairs

REFERENCES

[1] M. Kanehisa and S. Goto, "KEGG: Kyoto Encyclopedia of Genes and Genomes", *Nucleic Acids Res*. 28, 27-30, 2000

[2] H. Jeong, B. Tombor, R. Albert, Z. N. Oltvai and A.-L. Barabasi, "The Large-scale Organization of Metabolic Networks", *NATURE*, vol. 407, pp.651-654, 2000

[3] S. Okuda, T. Yamada, M. Hamajima, M. Itoh, T. Katayama, P. Bork, S. Goto, and M. Kanehisa, "KEGG Atlas mapping for global analysis of metabolic pathways", *Nucleic Acids Res*. May 13. 2008.

[4] M.Y. Becker and I. Rosas, "A graph layout algorithm for drawing metabolic pathways", *BIOINFORMATICS*, vol. 17, no. 5, pp461-467, 2001

[5] Y. Wang, "*Familiar Layouts Generation for Metabolic Pathway Graph Visualization"*, MS Thesis, Case Western Reserve University, 2008

[6] P.D. Karp, S.Paley. "Automated drawing of metabolic pathways", *Proc. of the 3rd Intl. Conference on Bioinformatics and Genome Res*., 225 – 238.

[7] E.H. Song, M.K. Kim, and S.H. Lee, "A Metabolic Pathway Drawing Algorithm for Reducing the Number of Edge"*, Genomics & Informatics,* vol. 4, pp 118-124, 2006

[8] M. Kato et al, "Automatic Drawing of Biological Networks Using Cross Cost and Subcomponent Data", *Genome Informatics*, vol.16, no 2, pp 22-31, 2005

[9] H.C. Purchase, J. Allder, D. Carrington, "Graph Layout Aesthetics in UML Diagrams", *User Preference, Journal of Graph Algorithms and Applications*, vol.6, no.3, pp. 255-279, 2002

[10] S.H. Kang, M.H. Jang, J.Y. Whang, and H.S. Park, "Parsing KEGG XML Files to Find Shared and Duplicate Compounds Contained in Metabolic Pathway Maps: A Graph-Theoretical Perspective", *Genomics & Informatics*, vol. 6, no. 3, pp 147-152, 2008

Myungha Jang is a student at the school of Computer Science and Engineering, Ewha Womans University. She is currently studying at Wesleyan College in United States as an exchange student. Her research interests encompass bioinformatics, natural language processing, and programming language theory.

© 2010 ACADEMY PUBLISHER

Jiyoung Whang is currently a student at the school of Computer Science and Engineering, Ewha Womans University, Seoul, South Korea. Her research interests include bioinformatics, robotics, embedded software and operating systems. She is a member of Young Engineers Honor Society (YEHS) sponsored by the National Academy of Engineering of Korea(NAEK).

Coleen S. Lewis is a student at Wesleyan College, Macon, GA in United States. She is a biology major and neuroscience minor. Her research interests lie in bioinformatics as it relates to molecular mechanisms that control human metabolic processes, microorganism interactions and immunologic responses.

Dr. Hyun S. Park received his bachelor's degree in electronic engineering at Seoul National University, his master's in computer engineering at the University of Pennsylvania in 1994, and his Ph.D. in computer science at the University of Cambridge, England, in 1997. Dr. Park's research interests are primarily in the area of bioinformatics, natural language processing, and cognitive science.

© 2010 ACADEMY PUBLISHER