


SOFTWARE ARTICLE

Open Access



Mango: combining and analyzing heterogeneous biological networks

Jennifer Chang^{1†}, Hyejin Cho¹ and Hui-Hsien Chou^{1,2*†} 

*Correspondence:

hhchou@iastate.edu

†Equal contributors

¹Department of Genetics,
Development and Cell Biology,
Iowa State University, 50011 Ames,
Iowa, USA

²Department of Computer Science,
Iowa State University, 50011 Ames,
Iowa, USA

Abstract

Background: Heterogeneous biological data such as sequence matches, gene expression correlations, protein-protein interactions, and biochemical pathways can be merged and analyzed via graphs, or networks. Existing software for network analysis has limited scalability to large data sets or is only accessible to software developers as libraries. In addition, the polymorphic nature of the data sets requires a more standardized method for integration and exploration.

Results: Mango facilitates large network analyses with its Graph Exploration Language, automatic graph attribute handling, and real-time 3-dimensional visualization. On a personal computer Mango can load, merge, and analyze networks with millions of links and can connect to online databases to fetch and merge biological pathways.

Conclusions: Mango is written in C++ and runs on Mac OS, Windows, and Linux. The stand-alone distributions, including the Graph Exploration Language integrated development environment, are freely available for download from <http://www.complex.iastate.edu/download/Mango>. The Mango User Guide listing all features can be found at <http://www.gitbook.com/book/j23414/mango-user-guide>.

Keywords: Systems biology, Heterogeneous data integration, Biological pathway analysis, 3D visualization, Graph mathematics

Background

In the present Big Data era, one of the great challenges is to be able to compare or integrate diverse data types. Modern biological research produces large and heterogeneous data sets, and there are many ways to categorize or display each type of data. The 2014 Nucleic Acids Research Database Special Issue counted 1552 online biological databases [1]. It is often illuminating, even essential, to examine important biological problems using different types of data. For example, new discoveries often emerge when a biologist is able to interrogate gene expressions in the context of biological pathways [2]. A common method to analyze related data relies on graphs, or networks, where data of various types are linked and key network features or subsets are identified [3–5].

Many graph analysis solutions have been written in Java, most notably Cytoscape [6]. Started in 2002, Cytoscape has an impressive array of features. However, like other Java programs, the software slows to non-operational levels when handling large (>1 M link) biological networks due to Java Virtual Machine limitations [7]. Non-Java graph tools either do not provide analysis functions, or provide only libraries which users must incorporate into their own software solutions. Overall, many graph tools focus

solely on one functionality, i.e., either analysis or visualization, and require users to integrate two or more tools for one project. Multi-graph comparison and integration are further complicated by differing graph attributes from heterogeneous data sets. Many tools ignore or limit the number of attributes associated with a graph. A comparison of currently available graph analysis and visualization software [6, 8–10] is given in Table 1.

To address these limitations, we have developed a stand-alone graph analysis and visualization software environment called Mango to aid biologists and other researchers efficiently integrate and explore heterogeneous networks larger than previously possible. A 4 million link network can be loaded into Mango in 30 seconds on a Mid 2010 Mac mini computer with a 2.4 GHz (Gigahertz) Intel Core 2 Duo processor and 8 GB RAM (random access memory). As a comparison, Cytoscape took 6 minutes to load that same network file on the same computer using its default configurations. Mango possesses the scalability to handle larger networks, the expressive power of a new Graph Exploration Language (Gel) and the convenience of unlimited graph attributes with automatic graph attribute merging and promotion. Within the integrated development environment, Gel commands can be edited, run line-by-line, or saved as scripts to reproduce results. Script files enhance the speed and reproducibility of analysis [11]. Mango provides both comprehensive graph analyses and real-time 3-dimensional (3D) visualization. Mango is a cross-platform C++ program that runs on Mac OS X 10.9 or later, Windows 7 or later, and many Linux variants. It is freely available from our website (<http://www.complex.iastate.edu/download/Mango>) and the Mango User Guide is hosted at GitBook (<http://www.gitbook.com/book/j23414/mango-user-guide>).

Implementation

The Mango user interface

Mango updates its display in real-time at each stage of analysis to facilitate the integration and modification of multiple large networks. Mango contains a primary window divided into four areas (Fig. 1). The graph canvas area is fully interactive, responding to mouse and keyboard actions to zoom, move, rotate, and auto-layout the displayed graphs. By dragging and rearranging tabs, multiple graphs can be viewed simultaneously, easing multi-network comparison. Mango functions are mostly carried out through its command console or Gel code editor. The Gel code editor allows commands to be run line-by-line, edited, and saved as Gel script files. Gel script files can then be shared among researchers, reproducing a 3D layout or network analysis pipeline. Finally, the data area lists currently loaded graphs, their sizes and attributes. Interactive real-time network visualization in Mango helps hone and refine each step of analyses. Mango is built on multiple layers of implementation that are seamlessly combined to form an integrated solution for graph analysis (Fig. 2).

The Graph Exploration Language (Gel)

A graph is defined as a set of nodes (V) and links (E) where a node represents some entity and a link represents a relationship between a pair of entities. In practice, graphs also have added annotations called attributes. Currently, Gel provides four basic data primitives *string*, *int*, *float* and *double* as well as aggregate data types *node* (V_{attr}), *link* (E_{attr}) and *graph*.

Table 1 Comparison of graph visualization software

Software	Code	Graph analysis features	Visualization	Limitations
Cytoscape (v. 3.2.1)	Java	<ul style="list-style-type: none"> · Many algorithms for systems biology · Can add GO or KEGG attributes · Plug-ins available 	<ul style="list-style-type: none"> · 2D predetermined layout · 3D predetermined layout (via plug-in) · 	<ul style="list-style-type: none"> · Can only merge 2 graphs at a time · 6 min to load a network with 4 M links but no visual afterward
Gephi (v. 0.8.2)	Java	<ul style="list-style-type: none"> · Intuitive graph statistics · Automated graph algorithm citation · Generalized for all types of graphs · Plug-ins available 	<ul style="list-style-type: none"> · 2D and 3D layouts but graphs cannot be rotated in 3D · Graph layout animation helps maintain mental map 	<ul style="list-style-type: none"> · Cannot display multiple graphs on one screen · Limited by JVM constraints; cannot load a network with 4 M links
GUESS	Java	<ul style="list-style-type: none"> · GYTHON, a language for graph analysis · Can map information attributes to visual attributes 	<ul style="list-style-type: none"> · 2D layout only · Update with user commands 	<ul style="list-style-type: none"> · Cannot be run on MacOS 10.9, Windows 7, or Redhat Linux 6.0
Graphviz	C	<ul style="list-style-type: none"> · No graph analysis capabilities 	<ul style="list-style-type: none"> · Rich set of predetermined 2D layouts · Streamlined command line interface 	<ul style="list-style-type: none"> · Not an interactive system · Cannot efficiently handle graphs over 100 nodes
Neo4j (v. 2.1.7)	Java	<ul style="list-style-type: none"> · Graph database system · Cypher graph query language · Queries are based on a combination of topology and attributes 	<ul style="list-style-type: none"> · Relies on JSON for visualization · 2D layouts only · Have to click a node or link to see its attributes on a separate panel 	<ul style="list-style-type: none"> · Designed as a backend to database support rather than for visualization · Nodes are only labeled by numbers · The whole database is one huge graph
Tulip (v. 4.6.1)	C++	<ul style="list-style-type: none"> · A set of C++ libraries for graph analysis · Can also be run as stand-alone program · Plug-ins can be created in Python 	<ul style="list-style-type: none"> · 2D visualization · 3D is available through plug in · Had some 3D layout algorithms 	<ul style="list-style-type: none"> · More useful to users who program C++ or python directly · More analysis than visualization features
NetworkX (v. 1.6.1)	Python	<ul style="list-style-type: none"> · Python module for graph analysis · Rich set of network algorithms 	<ul style="list-style-type: none"> · Must export to other software or modules for visualization 	<ul style="list-style-type: none"> · Useful only as an analysis tool
Mango (v. 1.10)	C++	<ul style="list-style-type: none"> · Provides general graph mathematics · Heterogeneous graph analysis with ease · Takes ~30 s to load a 4M link network 	<ul style="list-style-type: none"> · Interactive 3D layouts and controls · Real-time large graph visualization · User customizable visual attributes 	<ul style="list-style-type: none"> · Does not yet have plug-in feature · Does not yet use GPU speedup · Limited set of preset layouts

Benchmarks were performed on a 2010 Mac mini that has 8 Gb RAM and runs 64-bit MacOS X 10.9 with a 2.4 GHz Intel Core 2 Duo processor. All software were run using their default configurations

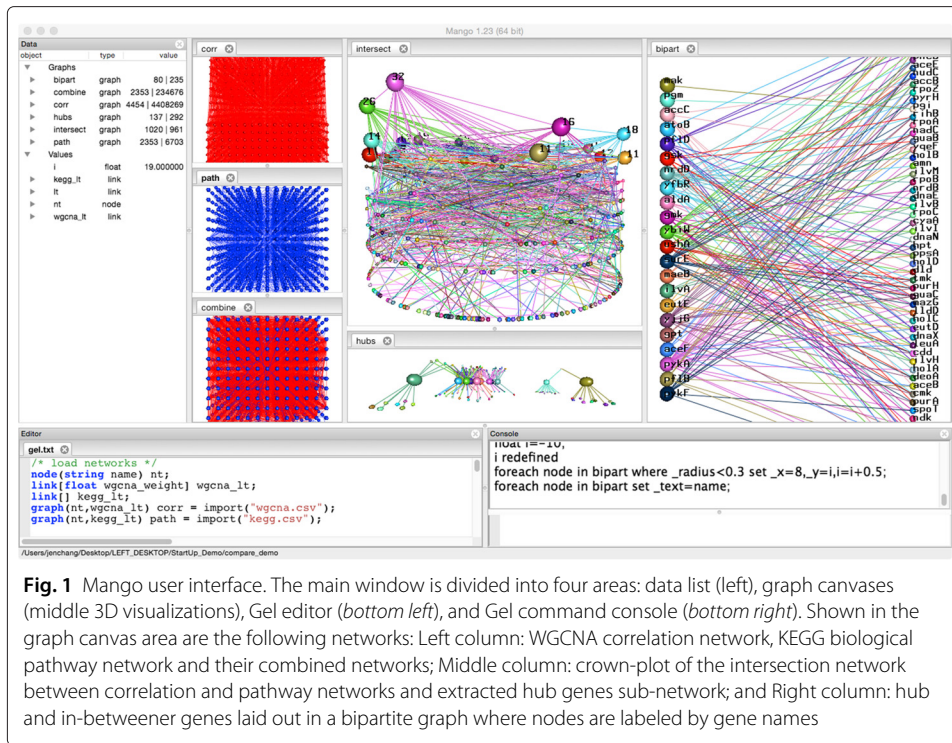


Fig. 1 Mango user interface. The main window is divided into four areas: data list (left), graph canvases (middle 3D visualizations), Gel editor (bottom left), and Gel command console (bottom right). Shown in the graph canvas area are the following networks: Left column: WGCNA correlation network, KEGG biological pathway network and their combined networks; Middle column: crown-plot of the intersection network between correlation and pathway networks and extracted hub genes sub-network; and Right column: hub and in-betweener genes laid out in a bipartite graph where nodes are labeled by gene names

$$G = \{V, E\} \quad \text{where} \quad V = \{v_1, v_2, v_3, \dots, v_n\}$$

$$E \subseteq \{(v_i, v_j) | v_i, v_j \in V\}$$

$$V_{attr} = \{a_1, a_2, a_3, \dots, a_{m_1} | type(a_i) \in \{int, float, double, string\}\}$$

$$E_{attr} = \{a_1, a_2, a_3, \dots, a_{m_2} | type(a_i) \in \{int, float, double, string\}\}$$

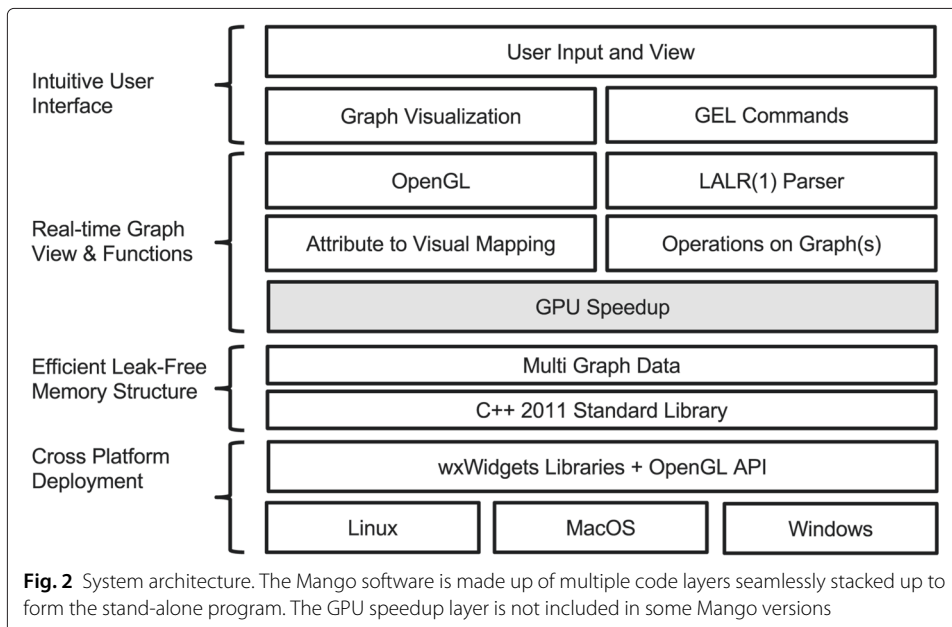


Fig. 2 System architecture. The Mango software is made up of multiple code layers seamlessly stacked up to form the stand-alone program. The GPU speedup layer is not included in some Mango versions

Each nodes and link type can have any number of attributes of the four primitive types in any order, and each of the attributes has a distinct name and specified data type (e.g. *string*, *int*, *float*, and *double*). The first attribute in a node type must be a *string* to denote the node name, and a link is identified by a pair of node names. All node and link attributes have default values, which are usually zero for numeric types or the empty string, but users can define other default values during node and link type declarations. Graphs are defined based on a pair of *node* and *link* types. For example, the following Gel code defines and initializes two graphs G_A and G_B , also shown in Fig. 3a. Node type and link type are defined with the given attributes inside parentheses and brackets; the brackets denote non-directional link types (whereas arrows $\langle \rangle$ denote directional link types). For example, G_A is declared with ntA and ltA , and is also initialized by the graph literals enclosed within the braces.

```
node(string id, int count) ntA;
link[float weight] ltA;
graph(ntA,ltA) A = {("a",1)[0.4] ("b",2)[0.4] ("d",4), a[0.8] ("c",3)};

node(string id, string tag) ntB;
link[float weight] ltB;
graph(ntB,ltB) B = {("b","g")[0.3] ("d","m")[0.3] ("e","c"), b[0.2] ("c","g")[0.2]e};
```

Other than defining a graph in the native graph exploration language, Mango can read graph data in tabular or CSV (comma separated values) format using the **import** command. A properly formatted graph file lists nodes with their attributes and then links with their attributes. A single line containing a hyphen separates the node list from the link list. The full description of the **import** command is in the Mango User Guide.

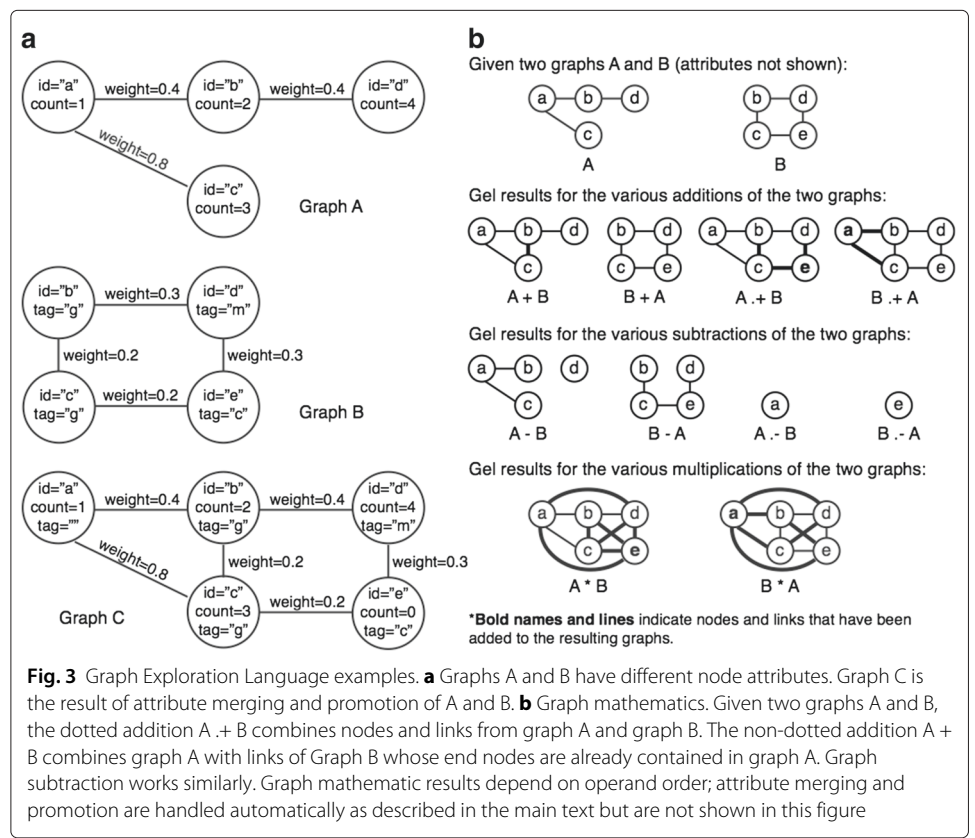


Fig. 3 Graph Exploration Language examples. **a** Graphs A and B have different node attributes. Graph C is the result of attribute merging and promotion of A and B. **b** Graph mathematics. Given two graphs A and B, the dotted addition A .+ B combines nodes and links from graph A and graph B. The non-dotted addition A + B combines graph A with links of Graph B whose end nodes are already contained in graph A. Graph subtraction works similarly. Graph mathematics results depend on operand order; attribute merging and promotion are handled automatically as described in the main text but are not shown in this figure

Mango system-defined graph attributes are appended to user defined attributes. The system-defined attributes are related to the 3D visualization of a network and define such attributes like node position, node color, or link width. Therefore, generating any 3D visualization is a matter of mapping user defined information attributes to system defined visualization attributes [12]. By dynamically changing these mappings, animations and simulations can be accomplished in Mango. A full listing of the visualization attributes is in the Mango User Guide.

Standards for combining heterogeneous graphs

When combining two or more graphs, much of the confusion stems from what will happen to the nodes and links. Since a graph contains both node and link sets, our formally defined dotted and non-dotted graph mathematic operators allow users to specify node-centric or link-centric operations precisely. Recall the two graphs G_A and G_B .

$$G_A = \{V_A, E_A\} \qquad G_B = \{V_B, E_B\}$$

Merging nodes and links is represented by the dotted addition.

$$G_A \cdot + G_B = \{V_A \cup V_B, E_A \cup E_B\}$$

However, suppose that the user is only concerned with the nodes in G_A , such as a set of important genes, and merely wants to combine the new links between those genes from G_B . The non-dotted addition merges links from G_B only between nodes already in G_A .

$$G_A + G_B = \{V_A, E_A \cup \{(v_i, v_j) | v_i, v_j \in V_A, (v_i, v_j) \in E_B\}\}$$

In a similar fashion, dotted and non-dotted subtraction between two graphs are defined as follows.

$$G_A \cdot - G_B = \{V_A \setminus V_B, (v_i, v_j) | (v_i, v_j) \in \{E_A \setminus E_B\}, v_i, v_j \in \{V_A \setminus V_B\}\}$$

$$G_A - G_B = \{V_A, E_A \setminus E_B\}$$

Other operations such as producing intersections and bipartite graphs are defined as follows.

$$G_A \cdot \& G_B = \{V_A \cap V_B, E_A \cap E_B\}$$

$$G_A \& G_B = \{V_A, E_A \cap E_B\}$$

$$G_A * G_B = \{V_A \cup V_B, E_A \cup E_B \cup \{(v_i, v_j) | v_i \in V_A, v_j \in V_B, v_i \neq v_j\}\}$$

$$G_A ** G_B = \{V_A \cup V_B, E_A \cup E_B \cup \{(v_i, v_j) | v_i \in V_A, v_j \in V_B\}\}$$

The above mathematics can be extended across multiple graphs to create unions ($G_A \cdot + G_B \cdot + G_C$), differences ($G_A \cdot - G_B \cdot - G_C$ or $G_A - G_B - G_C$), intersections ($G_A \cdot \& G_B \cdot \& G_C$) and inverse graphs ($G_A * G_A - G_A$). The graph operations can be mixed and matched to produce more complex results. Figure 3b demonstrates a few of the graph mathematics visually.

When graphs are combined in mathematical operations, attributes from two graphs might conflict. For example, the link between b and d nodes in G_A may have a *weight* attribute of 0.4 while the link between b and d nodes in G_B may have a *weight* attribute of 0.3. Gel handles attribute conflicts by giving preference to the left operand. During the

operation $G_A . + G_B$, the left operand G_A takes precedence and the resulting graph will have *weight* value 0.4. An exception to this rule is when the conflicting attributes in G_A happen to be at their default values (default values can be defined by users). In those cases, the attributes of graph G_B will be copied. This automatically merges useful non-default information from G_B into the resulting graph.

When heterogeneous graphs are combined, their unique attributes can be selectively preserved. Recall that the nodes in G_A have attributes *id* and *count* while nodes in G_B have attributes *id* and *tag*.

$$\begin{array}{ll} V_{A,attr} = \{id, count\} & E_{A,attr} = \{weight\} \\ V_{B,attr} = \{id, tag\} & E_{B,attr} = \{weight\} \end{array}$$

Because nodes in G_B only share the *id* attribute with G_A , when G_B is added to G_A as in $G_A . + G_B$, the *count* attribute of nodes copied from G_B is automatically set to the default value 0 but their *tag* attribute is ignored. To preserve both G_A and G_B attributes, users can define a new node type that includes all attributes. This is called attribute promotion. In our example, a new node type containing *id*, *count* and *tag* attributes is defined and used by the new G_C to receive all attributes from G_A and G_B .

$$\begin{array}{ll} V_{C,attr} = \{id, count, tag\} & E_{C,attr} = \{weight\} \end{array}$$

However, simply writing $G_C = G_A . + G_B$ will not work as the *tag* attribute from G_B is already lost after the addition of G_B to G_A but before the result is assigned to G_C . The correct steps to preserve graph attributes during heterogeneous graph mathematics are demonstrated below (Fig. 3a):

```
node(string id, int count, string tag) ntC;
link[float weight] ltC;
graph(ntC, ltC) C=A; // copy id and count attributes from graph A
C.+=B; // then merge with tag attributes from graph B
```

Flexible node and link type definition coupled with an intuitive set of attribute promotion and merging rules ease the combination of heterogeneous graphs in Gel. Thus users can focus on graph level operations instead of attribute level selection, sorting, and merging.

Many graph analyses require traversing all nodes and links to perform a calculation based on graph attributes or topology. Gel provides the *select* command to pull out a subgraph based on user-defined conditions. These conditions can be related to stored attribute values or topology properties. Gel also allows mapping or computing new attribute values across a graph on a per-node or per-link basis with the *foreach* command, which efficiently applies a set of user-defined calculations across all nodes or links that optionally meet certain conditions. The same command can also be used to tally attribute values across all nodes and links. The following demonstrates the two types of Gel commands:

```
graph(nt,lt) hubs = select node from A where in+out>3;
graph(nt,lt) thresh = select link from A where weight>0.2;
foreach link in thresh where weight>1.0 set weight=1.0;
foreach link in thresh set _r=weight, _g=weight, _b=weight;
foreach node in hubs where type=="gene" set _radius=0.2+(in+out)/2.0, count++;
```

In addition to the data types, graph mathematics, automatic attribute handling and traversal commands; Gel also provides commands for object modification, data examination, input and output, code execution, graph construction, and simulation. A growing set of built-in functions for mathematics, visualization control, graph layouts, and statistical reporting are also provided. To explore all Gel commands and functions, type the help command in Mango or consult the online User Guide.

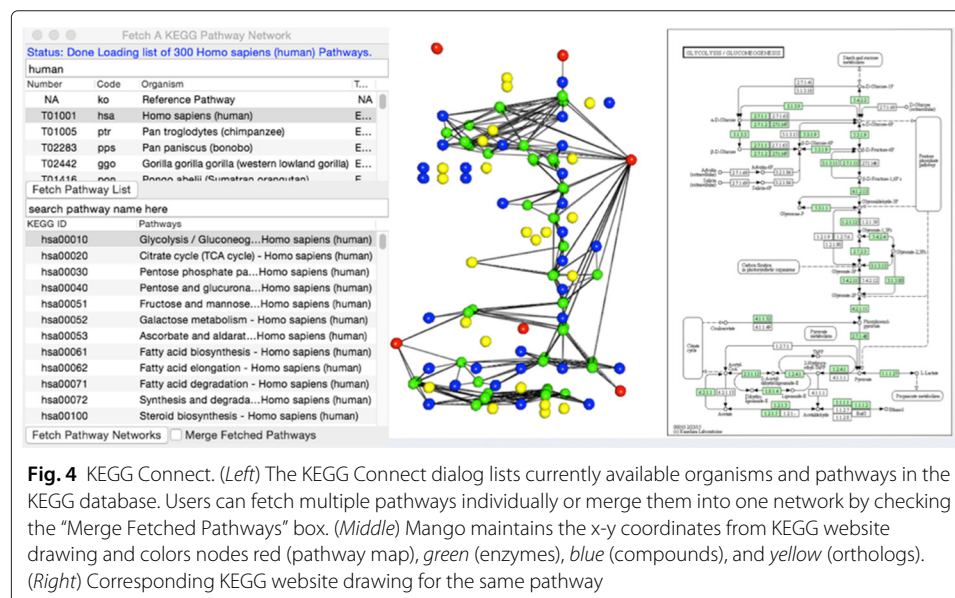
The Mango system and its Graph Exploration Language are data agnostic, meaning that any type of network can be loaded and analyzed – users have total control of node and link attribute definitions and their associations within Mango. Our goal is to make this software widely available to all researchers and promote its use in solving ever more complex biological research problems.

KEGG connect

The KEGG Connect dialog demonstrates how Mango can fetch network data directly from online biological databases. KEGG Connect queries the KEGG (Kyoto Encyclopedia of Genes and Genomes) database (<http://www.genome.jp/kegg>) and selectively downloads pathways grouped by organisms. Within the downloaded pathway, nodes maintain their 2-dimensional (2D) coordinates from the KEGG visualization. The nodes are colored red, blue, green and yellow representing pathway maps, compounds, genes, and orthologs respectively (Fig. 4). Multiple pathways can be downloaded either as individual networks or as one merged network. If multiple networks are merged, each pathway will be given a different z coordinate value, so the pathways are layered in 3D space. We intend to connect Mango to more biological databases soon.

Results and discussion

We present a few network analysis examples to illustrate the use of Mango in this section. Examples of comparing different types of biological networks and the scalability of Mango to large networks are provided.



Network data collection

Four large *E. coli* network data sets were collected. The *corr* 4 M link network was computed using the WGCNA (weighted gene coexpression network analysis) package in R [13] on microarray data measuring the expression of 4454 *E. coli* genes in cells grown under 10 different conditions (GSE61736, [14]). The *path* biological pathways of *E. coli* were downloaded from the KEGG database (<http://www.genome.jp/kegg>) and combined into a single pathway network. The *go* network was constructed using *E. coli* GO (gene ontology) information retrieved from the gene ontology website (<http://geneontology.org/page/download-annotations>); *E. coli* genes that share at least one GO term are linked. Finally, the protein-protein interaction (*ppi*) network was retrieved from the supplementary materials of a 2014 paper [15]. Sizes and attributes for the 4 large networks are summarized in Table 2.

Large heterogeneous network comparison

For all networks, nodes are identified by gene names with no additional attributes, thus the following node type declaration can be shared among the networks:

```
node(string name) nt;
```

All networks have undirected links but differ in their link attributes (the *path* network does not contain any link attributes), thus the following 4 link type declarations are used to load the different networks:

```
link[float corr_weight] corr_lt;
link[] path_lt;
link[int count, string go_terms] go_lt;
link[string source] ppi_lt;
```

After the node and link type declarations, the *corr* network, *path* network, *go* network, and *ppi* network can be imported into Mango for all-to-all network comparisons:

```
graph(nt,corr_lt) corr = import("wgcna.csv");
graph(nt,path_lt) path = import("kegg.csv");
graph(nt,go_lt) go = import("go.csv");
graph(nt,ppi_lt) ppi = import("ppi.csv");
```

For the integration of the networks, a common link type including all available link attributes is declared:

```
link[float corr_weight, int count, string go_terms, string source] c_lt;
```

Table 2 Summary of 4 large heterogeneous biological networks for *E. coli*

Network	Nodes	Links	Node attribute(s)	Link attribute(s)
corr	4,454	4,408,269	gene name	WGCNA correlation weight
path	2,353	6,703	gene name	none
go	3,764	2,208,090	gene name	count and string of shared GO terms
ppi	2,042	3,888	gene name	source of evidence (Y2H, LIT or both)

Unconnected nodes and duplicate links have been removed from some of the networks. In all 4 networks, nodes are identified by gene names and differ in their link attributes

Once the networks are loaded into Mango, Gel mathematics allow network integration and comparisons. For example, the comparison of the *corr* and *path* networks are visualized in the top two panels in the left column of Fig. 1. The top middle panel in Fig. 1 is the result of the following Gel intersect operation.

```
// intersection of path and corr networks
graph(nt,c_lt) intersect = path .& corr;
```

The *corr-path* intersection network contains 961 links with 1020 nodes. The all to all comparisons of these four networks were completed in Mango and the common links among the networks were summarized in Fig. 5. All possible intersections among the four *E. coli* networks can be worked out with a few lines of Gel code each. Bench-marked time for different types of Gel mathematics between the large *corr* and *path* networks are listed in Table 3.

Flexible real-time network exploration and visualization

Over-plotting of nodes and links becomes more of a challenge as network sizes get bigger. For example, the *corr* and *path* networks and their combination can be visualized in Mango but provide limited biological interpretation (the left column of panels in Fig. 1). In this example, we continue to explore the intersection of the two networks by querying certain node and link attributes, imposing thresholds to reveal important features, and map these features to network visualization.

First we arrange all nodes in the intersection network along a circle in the x-y plane and map the node connectivity to their z-axis coordinates. Nodes are assigned random colors and higher z-axis node colors are bled down the links to emphasize hubs. Nodes above a threshold are emphasized by increasing their radius and labeling them with gene names and connectivity.

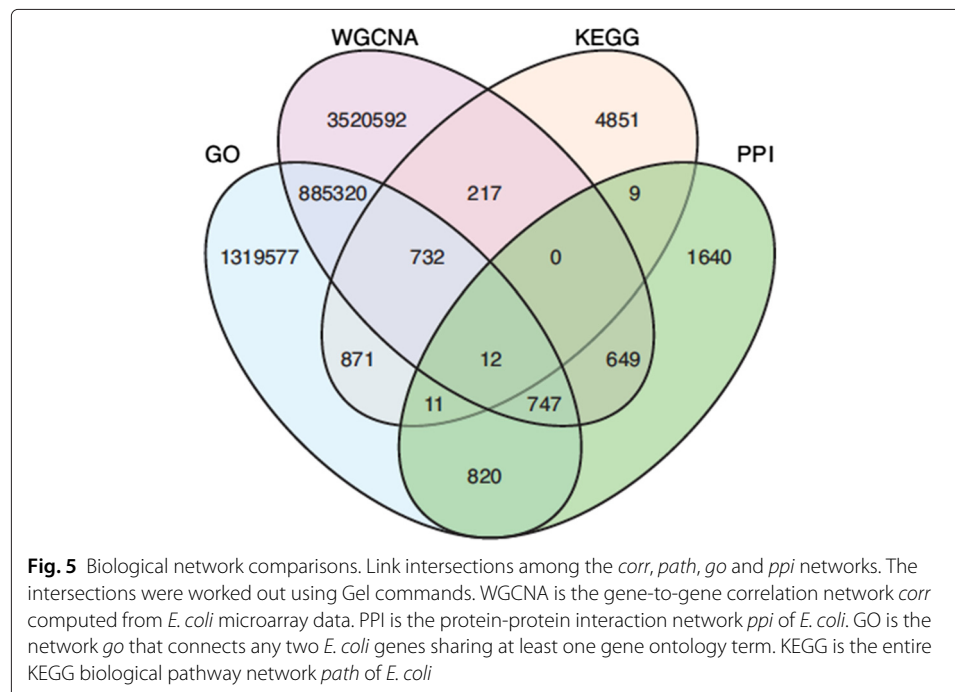


Table 3 Benchmarking the speed of Gel mathematics on massive graphs

Gel operation	Time (in seconds)	Average
4 M+ = 8 K	0.92, 0.35, 0.27, 0.60, 0.56	0.54
8 K+ = 4 M	1.25, 1.15, 1.03, 1.02, 1.02	1.09
4 M- = 8 K	0.52, 0.33, 0.62, 0.33, 0.25	0.41
8 K- = 4 M	1.09, 1.28, 1.09, 1.16, 1.19	1.16
4 M.+ = 8 K	0.69, 0.60, 0.57, 0.31, 0.40	0.51
8 K.+ = 4 M	12.06, 12.09, 12.05, 12.23, 12.32	12.15
4 M.- = 8 K	0.55, 0.41, 0.25, 0.26, 0.32	0.36
8 K.- = 4 M	0.90, 0.85, 0.83, 0.98, 0.74	0.86
4 M* = 8 K	22.94, 23.74, 23.35, 22.98, 23.03	23.21
8 K* = 4 M	36.75, 35.33, 35.23, 35.38	35.67
copy = 4 M	7.90, 7.76, 7.85, 7.73, 7.87	7.82
copy = 8 K	0.30, 0.52, 0.45, 0.34, 0.29	0.38

The 4 M link network is the gene correlation network generated by WGCNA. The 8 K link network is the combined KEGG pathway network. Benchmarks were performed consecutively on a 2010 Mac mini that has 8 Gb and runs 64-bit MacOS X 10.10 with a 2.4 GHz Intel Core 2 Duo processor. The time to copy the networks is also listed. All operations, including the copy operation, were performed using single thread in RAM

```

layout(intersect,"circle");
foreach node in intersect set _z=(in+out);
foreach node in intersect set _r=rand(),_g=rand(),_b=rand();
foreach link in intersect where in._z>=out._z set _r=in._r,_g=in._g,_b=in._b;
foreach link in intersect where in._z<out._z set _r=out._r,_g=out._g,_b=out._b;

// Label nodes by connectivity to choose a threshold
foreach node in intersect set _text=(in+out);

// Emphasize hubs
foreach node in intersect where (in+out)>10 set _radius=0.8;
foreach node in intersect where (in+out)<=10 set _text="";

```

The resulting network layout, called a **crown-plot**, is shown on the top pane in the middle column of Fig. 1. The hub genes and their links can be pulled into a new sub-network. The sub-network called hubs is then flattened and spread out using a force-directed layout built into the graph panel by right-clicking on the panel. The hub genes are raised one level. Genes that are not themselves hubs but connect two or more hubs are raised to a third level. The following Gel code accomplishes all these except the force-directed layout, which is performed by right-clicking on the panel:

```

auto hubs = select link from intersect where in._radius>0.3 || out._radius>0.3;
foreach node in hubs set _x=rand(-5,5),_y=rand(-5,5),_z=0;
/* right click on graph to start and stop force-directed algorithm */
foreach node in hubs where _radius>0.3 set _z=3;
foreach node in hubs where _radius<0.3 && (in+out)>1 set _z=6;

```

The 3-layer hubs network is shown in the lower panel in the middle column of Fig. 1, which contains other genes on the bottom layer, hub genes on the middle layer and in-betweeners on the top layer. It is worth mentioning that the in-betweeners genes on layer 3 would have been obscured by other genes in a simple list of genes ordered by connectivity. We can further pull out the hubs and in-betweeners into another sub-network for closer inspection with the following Gel code:

```

auto bipartite=select node from hubs where (in+out)>1;
int i=-20; foreach node in bipartite where _radius>0.3 set _x=-10,_y=i, i++;
i=-50; foreach node in bipartite where _radius<=0.3 set _x=10,_y=i, i++;
foreach node in bipartite set _text=name;

```

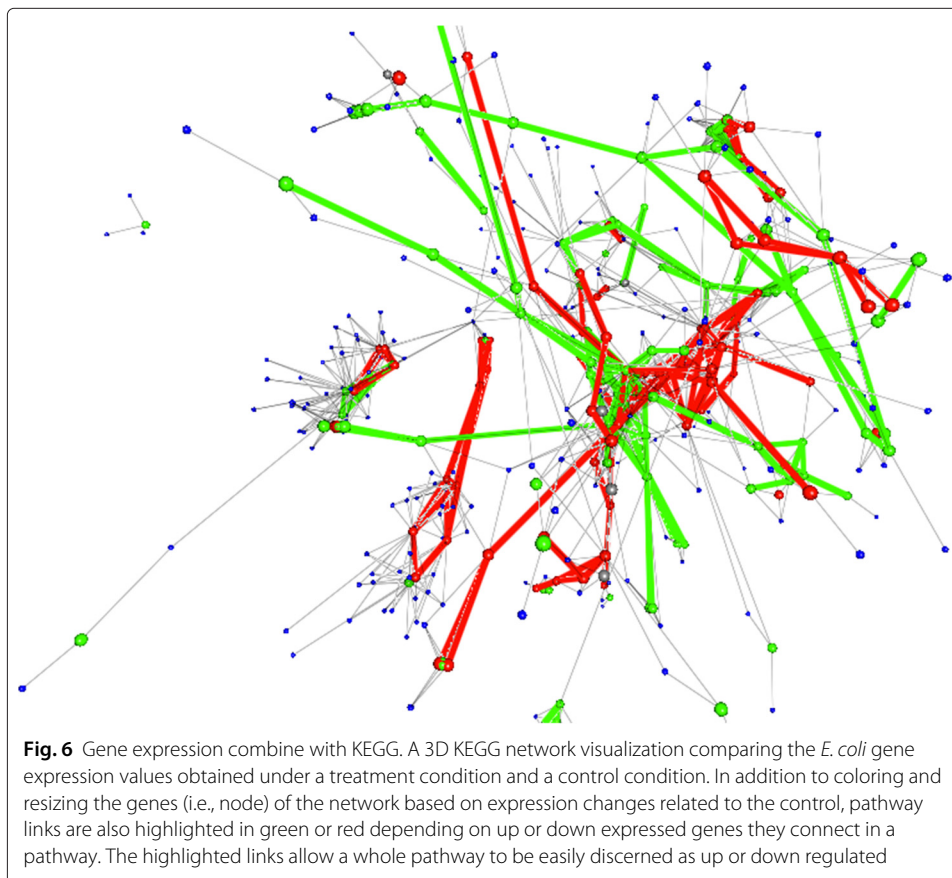
This sub-network is laid out as a bipartite graph shown on the right panel in Fig. 1, with hubs on the left and the in-betweeners on the right. This example shows how to map informational attributes of a graph to its visual attributes using Mango. The resulting visual displays help the user decide threshold values, extract sub-networks of interest, and further explore the data.

Microarray expression combined with KEGG biological pathways

E. coli gene expression under control and multiple treatment conditions were measured by microarrays (GSE61736, [14]). A subset of the data containing one control and one treatment expression values was loaded into Mango and overlaid onto downloaded *E. coli* KEGG biological pathways. The expression data, *E. coli* KEGG pathways, and Gel script are available for download from https://github.com/j23414/Mango_Workshop.

The results of the visualization can be seen in Fig. 6. Genes are colored green or red where their expression levels are up or down relative to the control condition. KEGG pathway components that do not have mapped gene expression values are colored gray. Compounds are colored blue and are largely ignored although they could be used to incorporate metabolomic concentration values. The Gel commands to color gene nodes are given below:

```
foreach node in sum where tr2==control && type=="gene" set _r=0.2,_g=0.2,_b=0.2;  
foreach node in sum where tr2>control && type=="gene" set _r=0,_g=1,_b=0;  
foreach node in sum where tr2<control && type=="gene" set _r=1,_g=0,_b=0;
```



More than coloring nodes in a network, we are able to color the links and thereby highlight entire pathways that are up or down-regulated. This is possible because KEGG pathways also contain gene to gene links, not just gene to compound links.

```
foreach link in sum where in._r==out._r && in._r>0.5 set _r=1,_width=4;
foreach link in sum where in._g==out._g && in._g>0.5 set _g=1,_width=4;
```

The final network can be saved and reloaded to regenerate the same 3D visualization.

```
save "sum.txt",sum;
clear;           // clears all data objects
run "sum.txt";  // reloads the sum network
```

Mango networks are saved natively into Gel commands, thus running the saved code recreates the original graphs in Mango. In addition, the networks can be exported to tabular data using the **export** command. The tabular data can then be read by many other software programs, e.g., Excel, R, Matlab, Cytoscape, and other graph software or databases. Full descriptions of the interoperability and other features of Mango are available in the User Guide.

Conclusion

We have developed a powerful new program Mango for multi-network analysis and visualization. Mango enables scientists to test hypotheses on large heterogeneous networks, identify crucial features, and extract analysis results all within its integrated environment. Compared with existing programs, Mango extends the capability and convenience of large heterogeneous data analysis on a personal computer.

The Mango system was designed to be data agnostic, meaning that any type of network data can be loaded and analyzed – users have total control on node and link attribute definitions and their associations within Mango. Mango can load networks with millions of links, integrate and explore large amounts of data following Gel commands, and help users deduce predictions or outcomes that can be validated in labs. It is our goal to make this software widely available to all researchers to promote its use in solving ever more complex biological research problems. As Mango developers, we will continue to provide support and further develop the software according to user needs.

Availability and requirements

- **Project name:** Mango 1.24.
- **Project home page:** <http://www.complex.iastate.edu/download/Mango/>
- **Operating system(s):** Mac OS X 10.9 or later, Windows 7 or later, and Linux variants. Both 32- and 64-bit operating systems are supported.
- **Programming language:** C++
- **Other requirements:** An Internet connection for online database access.
- **License:** Free versions available; specific license agreement included with each distribution.
- **Any restriction to use by non-academics:** Specific restrictions included with each distribution and license agreement.

Abbreviations

2D, 2-dimensional; 3D, 3-dimensional; CSV, comma separated values; Gel, graph exploration language; GO, gene ontology; GHz, Gigahertz; KEGG, Kyoto Encyclopedia of Genes and Genomes; PPI, protein-protein interaction; RAM, random access memory; WGCNA, weighted gene correlation network analysis

Acknowledgements

We thank Dr. Jo Anne Powell-Coffman, Zebulun Arendsee, and Kannan Sankar for proof-reading the draft manuscript and offering valuable suggestions.

Funding

This work is partially supported by the National Science Foundation grant DBI-0850195 and the Iowa State University Plant Sciences Institute Scholar grant to HC. JC is partially supported by the James Cornette Research Fellowship. None of these funding agencies had any role in the design of the study, data collection, analysis and interpretation, or in writing the manuscript.

Availability of data and materials

https://github.com/j23414/Mango_Workshop.

Authors' contributions

JC developed the Mango system and drafted the manuscript. HJC carried out the *E. coli* studies and collected the microarray data. HHC developed the Gel language and revised the manuscript. All authors read and approved the final manuscript.

Competing interests

JC and HHC have founded a software company and have licensed Mango from Iowa State University for further development. A free and functional Mango will always be made available to the public which can be downloaded and used by anyone including commercial entities.

Consent for publication

Not applicable.

Ethics approval and consent to participate

Not applicable.

Received: 18 March 2016 Accepted: 20 June 2016

Published online: 02 August 2016

References

1. Fernández-Suárez XM, Rigden DJ, Galperin MY. The 2014 nucleic acids research database issue and an updated nar online molecular biology database collection. *Nucleic Acids Res.* 2014;42(D1):1–6.
2. Khatri P, Sirota M, Butte AJ. Ten years of pathway analysis: current approaches and outstanding challenges. *PLoS Comput Biol.* 2012;8(2):1002375.
3. Jeong H, Mason SP, Barabási AL, Oltvai ZN. Lethality and centrality in protein networks. *Nature.* 2001;411(6833):41–2.
4. Albert R, Barabási AL. Statistical mechanics of complex networks. *Rev Modern Phys.* 2002;74(1):47.
5. Pavlopoulos GA, Secrier M, Moschopoulos CN, Soldatos TG, Kossida S, Aerts J, Schneider R, Bagos PG, et al. Using graph theory to analyze biological networks. *BioData mining.* 2011;4(1):10.
6. Smoot ME, Ono K, Ruscheinski J, Wang PL, Ideker T. Cytoscape 2.8: new features for data integration and network visualization. *Bioinformatics.* 2011;27(3):431–2.
7. Jarukasemratana S, Murata T. Recent large graph visualization tools: a review. *Inf Media Technol.* 2013;8(4):944–60.
8. Adar E. Guess: a language and interface for graph exploration. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.* Montreal, Quebec, Canada: ACM; 2006. p. 791–800.
9. Bastian M, Heymann S, Jacomy M, et al. Gephi: an open source software for exploring and manipulating networks. *ICWSM.* 2009;8:361–2.
10. Auber D. Tulip—a huge graph visualization framework. In: *Graph Drawing Software.* Berlin Heidelberg: Springer; 2004. p. 105–26.
11. Sandve GK, Nekrutenko A, Taylor J, Hovig E. Ten simple rules for reproducible computational research. *PLoS Comput Biol.* 2013;9(10):1003285.
12. Wilkinson L. *The grammar of graphics.* New York: Springer; 2006.
13. Langfelder P, Horvath S. Wgcna: an r package for weighted correlation network analysis. *BMC Bioinforma.* 2008;9(1):559.
14. Cho H, Chou HH. Thermodynamically optimal whole-genome tiling microarray design and validation. *BMC Res Notes.* 2016;9(1):305.
15. Rajagopala SV, Sikorski P, Kumar A, Mosca R, Vlasblom J, Arnold R, Franca-Koh J, Pakala SB, Phanse S, Ceol A, et al. The binary protein-protein interaction landscape of escherichia coli. *Nat Biotechnol.* 2014;32(3):285–90.