GraphCrowd: Harnessing the Crowd to Lay Out Graphs with Applications to Signaling Networks

Divit P. Singh

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in fulfillment of the requirements for the degree of

Master of Science
in
Computer Science and Applications

T.M. Murali, Co-Advisor
Kurt Luther, Co-Advisor
Eli Tilevich

May 04, 2016
Blacksburg, Virginia

Keywords: Crowdsourcing, Graphs, Layout, Algorithms, Signaling networks

GraphCrowd: Harnessing the Crowd to Lay Out Graphs with Applications to Signaling Networks

Divit P. Singh

(ABSTRACT)

Automated analysis of networks of interactions between proteins has become pervasive in molecular biology. Each node in such a network represents a protein and each edge an interaction between two proteins. Nearly every publication that uses network analysis includes a visualization of a graph in which the nodes and edges are laid out in two dimensions. Several systems implement multiple types of graph layout algorithms and make them easily accessible to scientists. Despite the existence of these systems, interdisciplinary research teams in computational biology face several challenges in sharing computed networks and interpreting them.

This thesis presents two systemsGraphSpace and GraphCrowdthat together enhance network-based collaboration. GraphSpace users can automatically and rapidly share richly- annotated networks, irrespective of the algorithms or software used to generate them. A user may search for networks that contain a specific node or edge, or a collection of nodes and edges. Users can manually modify a layout, save it, and share it with other users. Users can create private groups, invite other users to join groups, and share networks with group members. Upon publication, researchers may make networks public and provide a URL in the paper.

GraphCrowd addresses the challenging posed by automated layout algorithms, which incorporate almost no knowledge of the biological information underlying the networks. These algorithms compel researchers to use their knowledge and intuition to modify the node and edge positions manually to bring out salient features. GraphCrowd focuses on signaling networks, which connect proteins that represent a cells response to external signals. Treating network layout as a design problem, GraphCrowd explores the feasibility of leveraging human computation via crowdsourcing to create simplified and meaningful visualizations. GraphCrowd provides a streamlined interface that enables crowd workers to easily manipulate networks to create layouts that follow a specific set of guidelines. GraphCrowd also implements an interface to allow a user (e.g., an expert or a crowd worker) to evaluate how well a layout conforms to the guidelines.

We use GraphCrowd to address two research questions: (i) Can we harness the power of crowdsourcing to create simplified, biologically meaningful visualizations of signaling networks?(ii) Can crowd workers rate layouts similarly to how an expert with domain knowledge would rate them? We design two systematic experiments that enable us to answer both questions in the affirmative. This thesis establishes crowdsourcing as a powerful methodology for laying out complex signaling networks. Moreover, by developing appropriate domain-specific guidelines for crowd workers, GraphCrowd can be generalized to a variety of applications.

This work was supported by the National Science Foundation (DBI-1062380) and the National Institute of General Medical Sciences of the National Institutes of Health (R01GM095955).

GraphCrowd: Harnessing the Crowd to Lay Out Graphs with Applications to Signaling Networks

Divit P. Singh

(GENERAL AUDIENCE ABSTRACT)

Biologists often perform experiments which generate complex data and results. After gathering these results, they must analyze them in order to make sense of them. In order to represent complex data such as interactions between molecules in a cell, they use networks (graphs). The flexibility and the expressability that networks provide allow for analysis to be done on the complex data that biolgoists gather. This is also known as network analysis.

Nearly every publication that uses network analysis includes a visualization of a graph in which the nodes and edges are laid out in two dimensions. Several systems exist that allow for researchers to upload their networks and make them easilty accessible to fellow collaborators. Despite the existence of these systems, interdisciplinary research teams in computational biology face several challenges in sharing computed graphs and interpreting them.

This thesis presents two systems-GraphSpace and GraphCrowd-that together enhance network-based collaboration. GraphSpace users can automatically and rapidly share complex networks generated from their experiments. Users of this application may re-arrange the graph as they see fit as well as share their graph with other users. Groups may also be created to allow for a collection of users to share information among each other. A user may also make their graphs public and allow anyone with internet access to view them.

When users upload a network to any application, the network incorporates no knowledge of how to lay out its elements. Depending on the application, the elements of the network may be randomly placed or have a pre-defined method of how to arrange all elements of networks. However, none of these methods incorporate any biological information of the networks themselves. In order to circumvent this, we developed GraphCrowd. GraphCrowd explores the feasibility of leveraging human computation via crowdsourcing to create simplified and meaningful visualizations. GraphCrowd provides a streamlined interface that enables crowd workers to easily manipulate networks to create layouts that follow a specific set of guidelines. GraphCrowd also implements an interface to allow a user (e.g., an expert or a crowd worker) to evaluate how well a layout conforms to the guidelines.

We use GraphCrowd to address two research questions: (i) Can we harness the power of crowdsourcing to create simplified, biologically meaningful visualizations of signaling networks?(ii) Can crowd workers rate layouts similarly to how an expert with domain knowledge would rate them? We design two systematic experiments that enable us to answer both questions in the affirmative. This thesis establishes crowdsourcing as a powerful methodology for laying out complex signaling networks. Moreover, by developing appropriate domain-specific guidelines for crowd workers, GraphCrowd can be generalized to a variety of applications.

Dedication

I dedicate my thesis to my father: Hari Pal Singh. You are not only my motivation, but my inspiration.

Acknowledgments

Committing to studying my higher education at Virginia Tech was one of the best decisions I have made. The sense of community, the support from professors, and the love of life-long friends have left an impressionable mark on me. Without the people that I have had the absolute pleasure to have met at this university, I would never have gotten this far. Without the love, support, and encouragement from my family, friends, and advisors, this thesis would not have been possible.

A special thanks to the people who molded me into the person I am today, my parents: Hari Pal Singh and Kulwant Kaur. You have always been there to support and encourage me to pursue my passions. To this day, you have never let me feel that I was missing out on anything in my life. Thank you for always encouraging my curiosity and always willing to put my needs before yours. It is only because of your sacrifices and hard-work, that I was able to get an opportunity at higher education. Mom, thank you for always showering me with love and for putting up with my sarcastic responses. Dad, you have always been my best friend and mentor and will always continue to be. Thank you for leading by example and for being my one true confidante. You have always been my role model. I only hope that one day I can become the leader you are.

I would like to thank my sister, Ceenil Kaur. You were always there to protect me when I couldn't protect myself. Thank you for always looking out for me and for believing in me when I didn't believe in myself. It has been amazing experience seeing you transform from an overprotective, older sister to an amazing mother of two. You have always been and will continue to be my second mom. Robby Singh, you have become the brother than I never had. Thank you for always providing me with advice on how to tackle situations and always being available for help whenever I needed someone to talk to. You are an amazing friend and an excellent father.

I would especially like to thank both of my advisors: Dr. T.M. Murali and Dr. Kurt Luther. Dr. Murali, thank you for having the patience and the resolve to work with me through these last two years. Your work ethic and attention to detail have last a life-long impression on me. You have always amazed me with your impressive memory. You are by-far the best Sporcle player I have met. I truly appreciate your understanding and guidance in explaining concepts. It has been an absolute pleasure to have worked with you and am

extremely grateful to have had the experience to be a part of your research group. Dr. Luther, thank you for being one of the nicest, most understanding professors I have had the pleasure of knowing at Virginia Tech. With every conversation we have, I not only become more knowledgeable, but also ecstatic on the topic we discuss. Before even becoming my advisor, you always encouraged and were willing to listen to my ridiculous app ideas. Your classes have been some of the my favorite classes I have ever taken. Thank you for being patient with me and for working through my mistakes throughout my time as your student. I would like to thank my committee member: Dr. Eli Tilevich. I am extremely grateful for your guidance throughout my Undergraduate studies. You have a special talent in coming up with ideas for applications that draw recognition and it has been an absolute pleasure working with you.

I would like to thank Sanchit Chadha, my roommate and best friend since the first day of college. Thank you for putting up with my constant sarcasm. You always raised the bar and constantly pushed me. When I look back at all the fond memories at Virginia Tech, you're always associated with them. Your passion for hip-hop has left a life-long impression on me. From playing table tennis in your garage to spending a summer interning together in San Diego, we have had one heck of a ride. I would also like to thank your parents, who have always showered me with love and support.

I would like to thank one of my closest friend and someone I consider to be like a brother to me: Devin Bist. Thank you for always looking out for me and for putting up with my idiosyncrasies. It has been an absolute thrill to watch you grow into the man you are today. You always believed in me and your constant nagging always pushed me to pursue other interests. Your encouragement and words of wisdom provided me with confidence that I would've never had otherwise.

I would like to thank my two favorite post-docs: Anna Ritz and Allison Tegge. I can't thank you enough for all the guidance you provided me throughout my Graduate studies. Both of you always went far out of your ways to answer my questions. Without your patience and resolve, I don't think I would ever have been able to complete GraphSpace. Thank you to my closest friends since freshman year: Josh Orrick and Ben Tronrud. Josh, you always challenged me both academically and physically. I will always remember our Basketball and Football games together. Ben, without you, I don't think I would've had half the fun as I did at Virginia Tech. Thank you Antuan Byalik, your quick-wit and humor has made for some of my most memorable experiences. To Ishita Ganotra and Hassan Almas, although we may not have known each other long, you guys have quickly become some of my closest friends as I finish up Graduate School. Thank you so much for being excellent friends and for re-instilling my passion for coding.

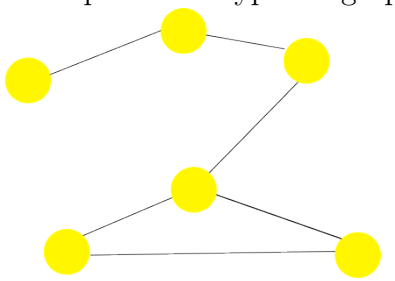
Contents

Chapter 1

Introduction

Graphs are a common method to illustrate information. There are many types of graphs such as bar graphs, circle graphs, and line graphs. The type of graphs that we consider contain a set of nodes and edges where an edge may connect two nodes together. These graphs may also be referred as networks. Figure 1.1 shows an example of such a graph.

Figure 1.1: An example of the types of graphs we explore



In these graphs, edges can either be directed or undirected. An undirected edge for a graph signifies a two-way relationship, whereas a directed edge signifies a one-way relationship. For example, consider Facebook's friendship model. Suppose all users of Facebook were represented as nodes in a graph. An edge between any two users signifies that they are friends. A Facebook user P1 is friends with a user P2 if and only if user P2 is friends with user P1. An undirected edge would be the appropriate edge to represent this mutual relationship between the two users. Conversely, if we consider Twitter's model, directed edges would be the appropriate type of edge to represent the relationship between two users. In Twitter's model, if a user P3 decides to follow person P4, it is not necessary for user P4 to follow user P3. This one-way relationship would best be represented by a directed edge. Figure 1.2 shows an example of the differences between the interactions of the two social networks.

Figure 1.2: Undirected edge represents Facebook friendship model and directed edge represents Twitter's follower model

Systems Biology is an area of research where graphs are heavily used. This area aims to understand complex systems in Biology [1]. A complex system is a system that is made up of many, interconnecting parts. Each of these parts have their own set of properties. When these parts interact, additional properties, known as emergent properties are created [2]. An example of a complex system is cell signaling.

There are many proteins within a cell. These proteins communicate with each other through chemical signals which then communicate with the cell's nucleus to perform functions. Communication also often happens between cells through chemical signals. Chemical signals are released from a cell in the form of proteins known as ligands. For other cells to detect these signals, they must have corresponding receptors for the signal. Receptors are proteins which exist on the membrane of the cell. Once a ligand binds to an appropriate receptor, it relays signals to the cell on which the receptor exists. These signals may, in turn, alter the cell and lead to downstream signaling events. Here, we use the term downstream to describe any subsequent changes to the function of the cell that are produced as a result of a signal. A possible event that may occur as a result of a signal within a cell is the activation of transcription factors. A transcription factor is a protein that controls which genes are on or off(up-regulated and down-regulated) when transcribing DNA. The entire set of communications (chemical signals) that that are produced by a receptor as a result of binding to a ligand is known as a signaling pathway [3]. Figure 1.3 shows an example of modeling this process. This is considered a complex system since it has many proteins interacting together. There are many signals that may be produced. These signals result in different events happening in the cell as well as between cells.

Figure 1.3: Interactions inside of a cell

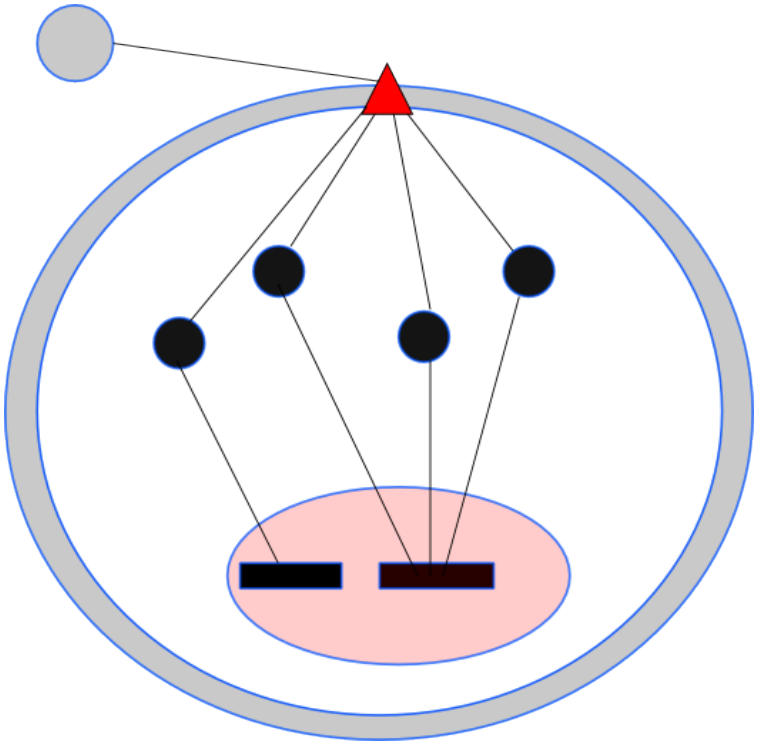


Figure 1.4: Legend for figure 1.3

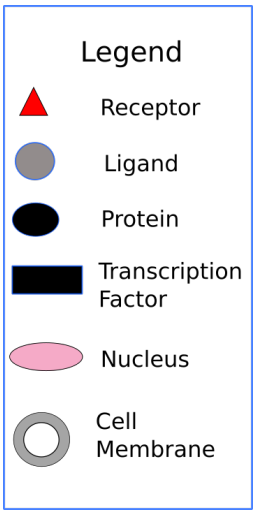


Figure 1.5 shows a reconstruction of a part of a signaling pathway that was produced by data presented in ToxCast [4]. ToxCast is a research program whose aim is to expose cells to chemical compounds and screen changes in their biological activity that may suggest potential toxic effects.

In this graph, triangles represent receptors, ellipses represent proteins, and rectangles represent transcription factors. This pathway shows a small part of the signaling pathway produced when the receptor BMPR2 is exposed to Bisphenol-A(BPA), a chemical compound. The figure shows that the receptor has interactions with four other proteins. Notice how some edges are directed, while some are directed. The undirected edges represent a physical interaction, in other words, both proteins physically bind to each other. The directed edge represents a regulatory interaction, meaning that one protein activated another protein. This is a directed edge since it is a one-way relationship. The BMPR2 receptor activated the SMAD1. The SMAD1 protein shown does not activate the BMPR2 receptor. The color of the proteins show represent that the specific protein is part of another known pathway. Each protein that interacts with a transcription factor relays a signal which, in turn, controls the regulation of genes. This simple graph of eight nodes represents a plethora of information regarding the signaling pathway. It is because of the expressiveness graphs provide that they have become a ubiquitous part of Systems Biology. Almost all publications that use graph analysis include visualizations that contain nodes and edges to model information.

Figure 1.5: An example of a signaling pathway

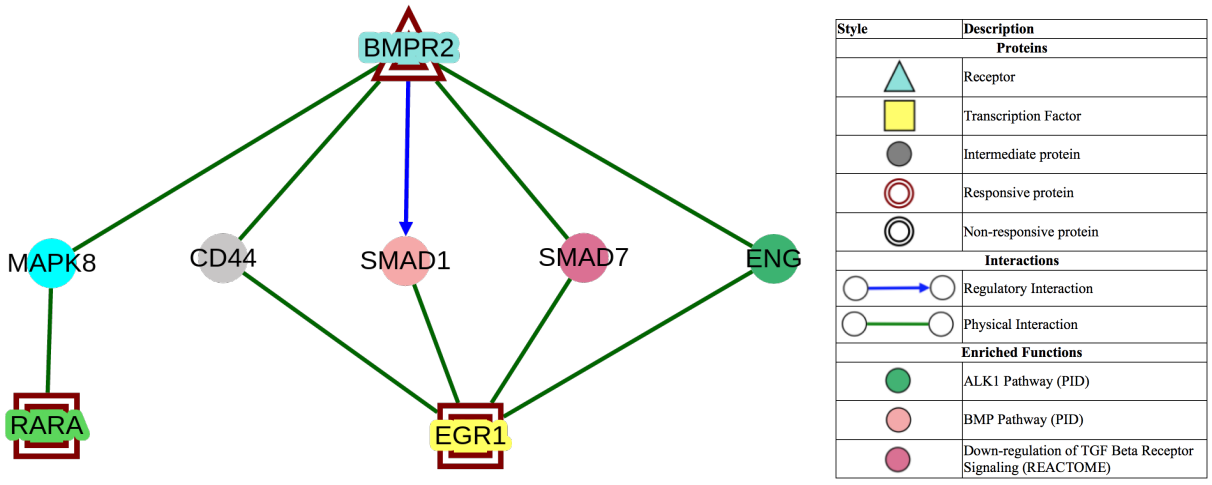
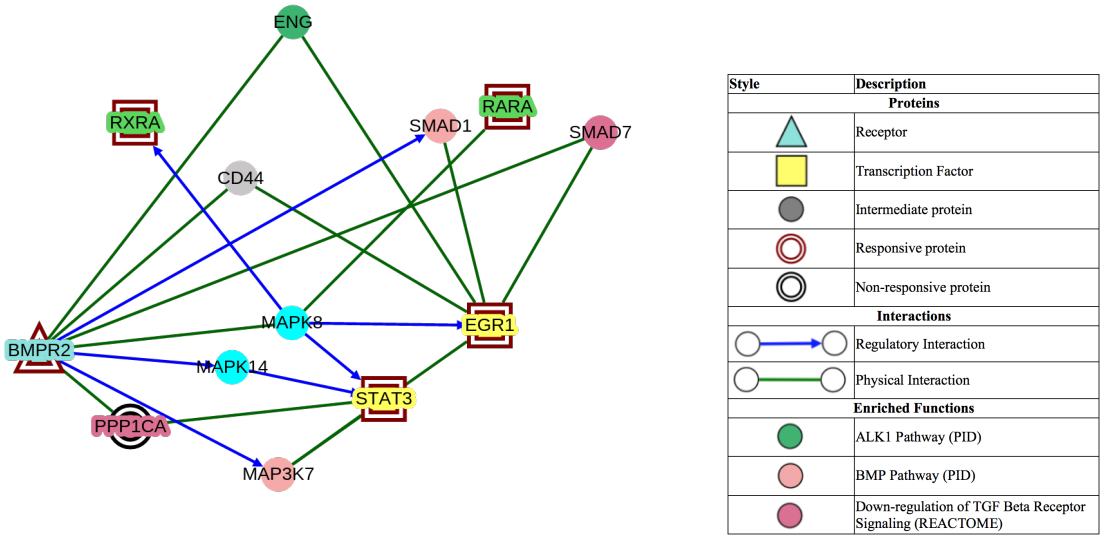


Figure 1.5 is manually layed out in a manner that emulates interactions with proteins throughout its pathway as shown in figure 1.3. This layout helps anyone who is interpreting the graph. Figure 1.6 shows the same graph; however, nodes in this graph are randomly placed. By comparing the two figures, it is easy to see that visualization is a crucial component in the interpretation of graphs.

Figure 1.6: Poorly placed nodes make it harder to decipher graphs



1.1 A Collaborative Platform for Viewing, Storing, and Sharing Graphs

Computational Biologists often generate data that is best represented in graphs. Their workflow often includes generating dozens to hundreds of graphs to model their data. After the graphs have been generated, these researchers may spend additional overhead by going into their favorite graph visualization software to re-arrange the graphs as they seem fit. Many systems exist for visualizing and laying out these graphs. However, they tend to be specific to the algorithms used to lay out graphs. If a researcher wishes to lay out a graph using a different algorithm, it often involves importing their graphs into a system which supports their desired layout algorithms. After these graphs are layed out properly, it is not uncommon that researchers share them among collaborators. To do this, researchers may create images of their graphs and send them via email or a platform where multiple view have access to view the graphs.

These steps incur overhead to all researchers involved in the process. GraphSpace was created to alleviate this overhead. GraphSpace is a web-based platform that collaborating researchers may use for storing, visualizing, and sharing graphs. After acquiring the information that researchers need to model, they may upload their graphs and store them on GraphSpace. They may view the graph and as well re-arrange elements of the graph by either manually moving the elements or by using the various provided automatic layout algorithms. Collaboration is built into GraphSpace, allowing researchers to share graphs. Collaborators may view these graphs as well as help lay them out.

1.2 A Crowd-based System for Generating Layouts

Researchers in the Murali lab who use GraphSpace tend to programmatically create many graphs. They often spend time manually laying out each graph to be biologically meaningful. This task consumes considerable amounts of time. Algorithms to automatically lay out these graphs do exist. However, these algorithms don't retain the biological context of the graphs. These algorithms lay out graphs using the properties of the graphs rather than the context of the graphs, producing layouts that are not biologically meaningful. A possible approach to fix this would be to see a pattern of the types of graphs being created by the researchers and create an algorithm which emulates this pattern. However, this approach would be specific to certain types of graphs and would be inflexible and domain specific.

When researchers first create their graphs, there is no sense of placement of nodes and edges. It is up to the creativity of the researcher to lay out their graphs in a manner which best illustrates the information. There is no true right answer for laying out the graphs:there may be many different layouts for one graph which may all illustrate information in a meaningful manner. Rather than having researchers lay out their own graphs, we explored methods to create a flexible system that would generate meaningful layouts for the graphs created by researchers. In order to create a flexible system for generating meaningful layouts as well as allow for creativity to be incorporated into these layouts, we explored the area of crowdsourcing.

Crowdsourcing is the process of obtaining needed services from a large group of people [5]. The crowd in the crowdsourcing can be thought of as anyone who is willing to do a task or a job. For example, assume that a web designer has created a website and needed users to interact with their website in order to determine any bugs and/or any suggestions for the website. Instead of the designer personally reaching out to individual people or a group of people that they may know, imagine if they had the ability to attract significant amounts of people to their website and quickly acquire the data that she needs. In this example, anyone that is willing to look through her website and provide suggestions are part of the crowd.

Signaling pathway graphs represent a plethora of information. To construct a biologically meaningful layout, a user must have domain knowledge of the pathway under construction. However, obtaining a crowd which has domain knowledge and is willing to work on congnitive tasks such as laying out these graphs is difficult. Assuming that a crowdworker(a person who is part of the crowd) has no domain knowledge on the type of graphs we use allows us to recruit more workers to lay out the graphs. In our approach, we assume that the crowdworkers have no domain knowledge. Stemming from this assumption, we arrived at our first **research question**: Can we harness the power of crowdsourcing to create simplified, biologically meaningful, mechanistic visualizations of signaling pathway graphs? Evaluating these layouts would incur overhead on the researcher. It would take a researcher a considerable amount of time to manually sift through the layouts searching for the most biologically meaningful layouts. Thus, exploring alternate methods to evaluate layouts without relying

on researchers became an area of interest. This lead to our second **research question**: Can the crowd rate these layouts similarly to how an expert with domain knowledge would rate them?

GraphCrowd: an extension to GraphSpace, was developed to answer these questions. Using Amazon Mechanical Turk [6] as its platform on which to publish crowdsourcing tasks, we conducted experiments where we asked the crowd to lay out signaling pathway graphs following a set of guidelines. Next, we asked collaborators in the Murali research group to lay out the same graphs given the same guidelines. For all the layouts that were produced, we asked both crowdworkers and an expert in the area of signaling pathways to rate them on a Likert scale [7] on how well the guidelines were followed. We compared these ratings in order to decipher if crowds truly can create biologically meaningful visualizations of signaling pathway graphs without any domain knowledge. Our comparisons show that crowdworkers produced layouts that are more biologically meaningful than current computational layout algorithms and sometimes as good as layouts that researchers would create themselves.

1.3 Contributions

The first contribution of this thesis is the GraphSpace platform. There are an abundance of graph visualization applications available to researchers. Although these applications allow for the creation and manipulating of graphs, researchers must incur the overhead of manually sharing their work with collaborators. In order to reduce this overhead, we developed an open-source application that supports the creation, storage, and collaboration of graphs between collaborators.

Another contribution is the crowdsourcing extension to GraphSpace: GraphCrowd. For researchers who collect and generate many graphs, the task of laying out the produced graphs to be biologically meaningful can become daunting. In order to address this challenge, we developed GraphCrowd: a crowdsourcing framework built on GraphSpace. GraphCrowd leverages crowdsourcing by asking crowdworkers to lay out graphs uploaded by researchers. However, most of the graphs that researchers upload tend to be very complex and contain information that require domain knowledge to comprehend. In GraphCrowd, we attempt to leverage non-expert crowdworkers to create biologically meaningful layouts of signaling pathway graphs with out the need for domain knowledge.

Finally, the third contribution of this thesis is the evaluation of how well crowdworkers create biologically meaningful layouts. We evaluate if these layouts beat current automatic layout algorithms and if they are comparable to layouts that are produced by researchers themselves.

1.4 Thesis Roadmap

Chapter 2 presents a survey of existing applications that contain similar functionalities to the applications discussed in our thesis. It presents prior work that has been done in the area of both graph visualization as well as crowdsourcing frameworks. In Chapter 3, we discuss the design, structure, and implementation of GraphSpace. In Chapter 4, we discuss GraphCrowd and its crowdsourcing functionalities. In Chapter 5, we will take a deep dive into our experiments in evaluating if crowsourcing is a viable option when creating simplified, meaningful layouts for complex graphs. Chapter 6 discusses the potential future directions of both applications and my conclusions for my thesis.

Chapter 2

Literature Review

My work not only focuses on creating a crowdsourcing system for generating layouts, but also creating a system for viewing, manipulating, an sharing graphs. In this chapter I discuss prior work in both areas. I present existing applications that allow users to generate graphs and how their functionalities differ from GraphSpace. Next, I discuss existing crowdsourcing framework and applications that played a role in the design and implementation of GraphCrowd.

2.1 Graph Visualization Applications

While searching for systems that could satisfy the needs for creating GraphSpace, we discovered that there were many alternatives for graph visualizations [8] [9] [10]. These systems provided excellent interfaces for visualizing graphs as well as creating them. They also provide algorithms to automatically lay out these graphs.

Cytoscape [11] is an open-source software project that provides biologists a conceptual framework in which to layout and query networks. GraphSpace supports the importing of networks created by Cytoscape GraphSpace is a web application that allows for the visualization of any type of graph. It allows for the manual as well as automatic laying out of all elements of networks. It primarily differs from Cytoscape in the sense that it is used as a platform for collaborators to share graphs with each other. Figures 2.1 and 2.2 show a graph generating in Cytoscape and how it looks like in GraphSpace.

Figure 2.1: Graph generated in Cytoscape desktop application

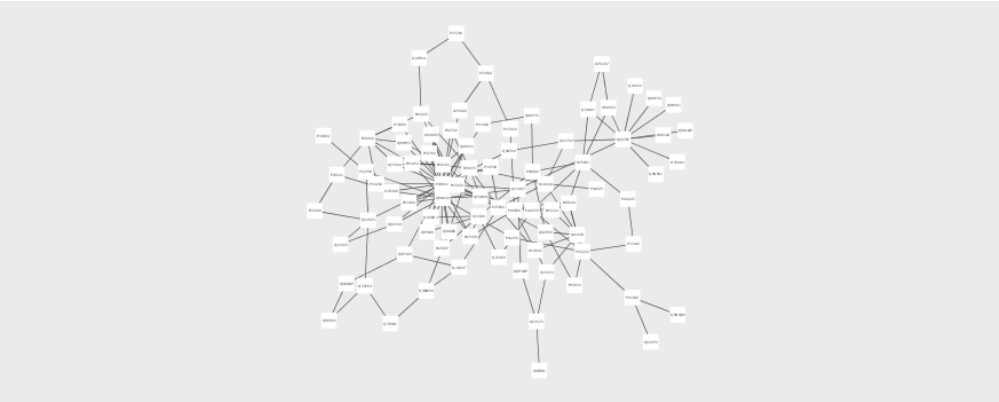
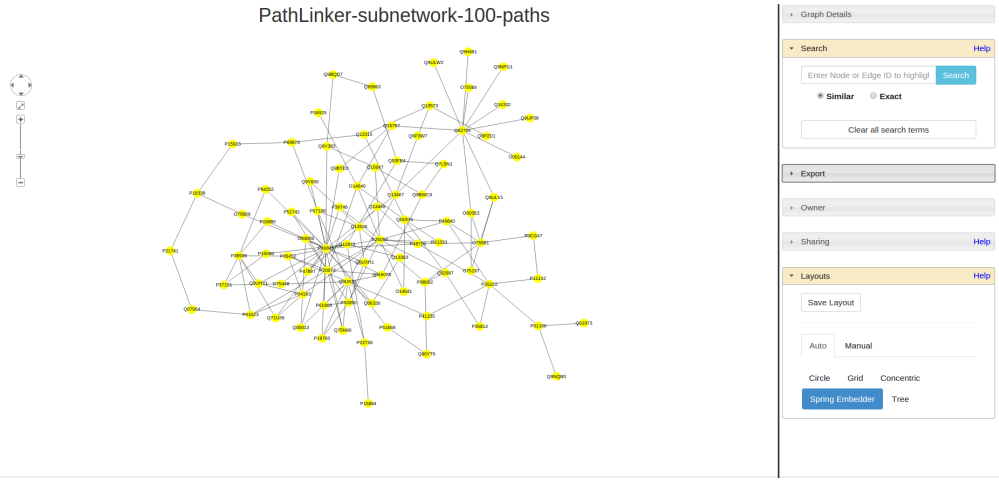
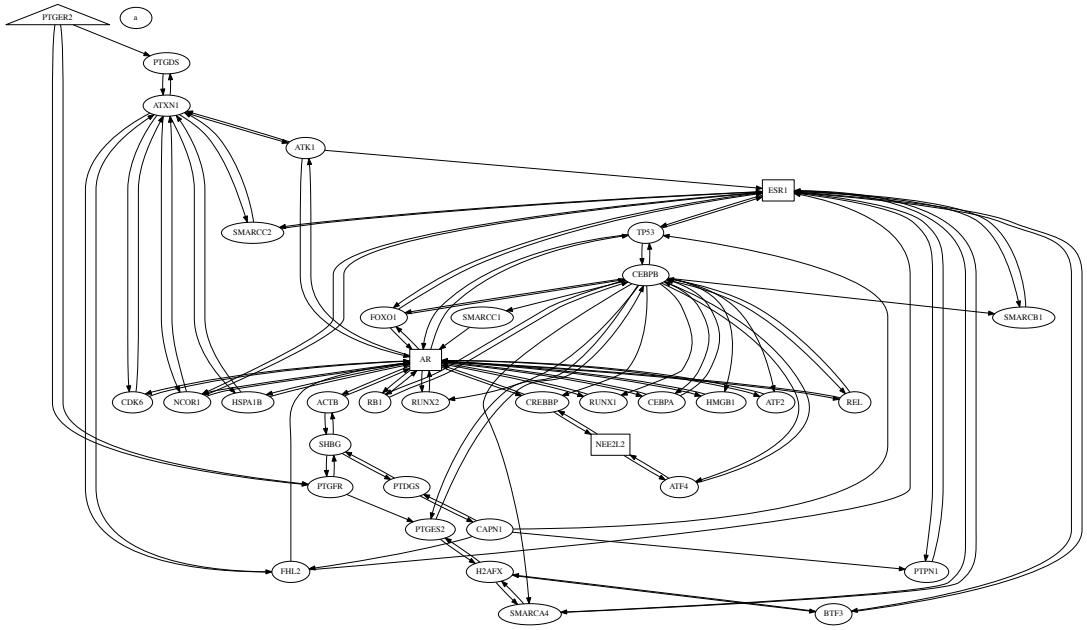


Figure 2.2: Cytoscape-generated graph imported into GraphSpace



GraphViz [12] is also an open-source visualization software that represents the structural information of graphs and networks as diagrams. In addition to creating useful diagrams by taking descriptions of graphs using its DOT language, it also provides automatic algorithms to layout the provided graphs and networks. Similar to Cytoscape, this library is another option to generate a graph given a set of data from a researcher. This library provides distinct set of layout algorithms for researchers to apply on their graphs. GraphSpace has a different set of layout algorithms as well as collaboration features for all of its graphs. Figure 2.3 shows how a signaling pathway graph looks like in GraphViz.

Figure 2.3: Diagram produced by GraphViz for signaling pathway graph



Gephi [13] is an open-source software for graph and network analysis. It is capable of displaying large networks in real-time by utilizing a 3D render engine for its graphs. It supports highly configurable layout algorithms. Examples of configurations include speed, gravity, repulsion of nodes in the graph. It provides the user with an abundant source of filters and tools to aid in analyzing and manipulating the graph. In addition, it also provides tools to analyze statistics of the graph itself. Gephi is an excellent library for exploring the area of graph theory. It provides users with an ample set of tools to analyze the graphs. Although it supports the analysis of graphs, there is no sense of built-in collaboration which GraphSpace supports. Gephi is strictly tool for exploring the relationships a complex graph contains.

NetworkX [9] is a Python language package for exploring and analyzing graphs and graph algorithms. It contains data structures to represent various types of graphs. In addition to having a way to programmatically represent graphs, it also has various graph algorithms for calculating graph properties. Some properties supported by NetworkX include clustering, shortest paths, and degree distribution. NetworkX is unique in the sense that it allows a user to construct a network programmatically. NetworkX is similar to GraphSpace in the sense that it allows for the construction and visualization of graphs. NetworkX provides various tools to compute statistics on the graphs it produces which GraphSpace does not currently support. However, similar to other graph visualization libraries, it does not support collaboration of these networks.

Igraph [10] is an open-source portable library used to create, manipulate and visualize net-

works. Igraph has the ability to handle large graphs (containing millions of nodes) efficiently. Igraph also provides many automatic layout algorithms to apply on its graphs. Its portability allows for rapid prototyping of systems to utilize its features. It is similar to NetworkX in the sense that it supports the building of networks through high-level programming languages and allows for analysis on these networks. Similar to NetworkX, it is a tool for graph creation, visualization, and analysis. There is no sense of collaboration built into the system.

Mathematica [14] is a very well known platform. Its software allows for the computation of graphs and network analysis. It supports hundreds of functions and standard graph algorithms. Mathematica is well-suited for graph-theory related networks. For example, analyzing social networks or statistics on a given graph are among the many use-cases where Mathematica shines. It not only supports graphs, but provides computation library. GraphSpace offers a platform on which users may store their graphs and collaborate on them. GraphSpace provides layout algorithms as well as functionalities to manipulate the graph.

GraphCommons [15] is a collaborative "network mapping" platform. It supports the creation, visualization, as well as the collaboration of the graphs. It is a web application that employs a freemium business model. Any person can create a free account with this service. They may also create, visualize, and collaborate on the graphs. However, if a user wants to privately store a graph, GraphCommons charges a monthly fee. GraphCommons contains many of the features that GraphSpace supports. However, GraphSpace is an open-source standalone package. Any user can install it on their Unix based machine and create their own instance. In addition, GraphSpace is free-to-use, imposing no limits for users to uploading or sharing graphs. GraphCommons provides one automatic layout algorithm, whereas GraphSpace provides multiple. GraphSpace also supports a REST API which allows for users to programmatically communicate with GraphSpace.

2.2 Crowdsourcing

There were many systems that served as inspiration for GraphCrowd. These system provided excellent information to help design and implement our system. They also served as an excellent guide for how to conduct experiments for evaluations.

CrowdCrit [16] is a system which leverages paid crowsourcing to generate and visualize high-quality visual design critique. In order to aid the crowdworker in giving valuable feedback, it first offers them a list of critique statements. Crowdworkers choose one that applies to the creative work under review. From there, they can add more granular feedback about the work. Workers can also see reviews by previous workers. CrowdCrit provides tools to aid the crowdworker in generating valuable feedback on a piece of creative work. GraphCrowd differs from CrowdCrit because it focuses on laying out graphs, not tasks to improve graphic desing. GraphCrowd not only allows for generating layouts of graphs, but also allows for the evaluation of generated layouts. Its main purpose is to offer a platform to produce creative

work. It uses feedback to determine if the produced work is satisfactory. In a similar work, Voyant [17] also provides structured feedback by using micro tasks to gather feedback from crowdworkers to judge how well visual guidelines have been followed. Instead of using rubrics to improve quality of crowd feedback, its approach is more free-form. It follows CrowdCrit approach by providing a rubrics to rate creative work on.

Acquiring high-quality results in an uncontrolled, online setting is difficult [18]. Mitra [19] conducted experiments to evaluate to see if person-oriented strategies are advantageous over process-oriented strategies. They found that person-oriented strategies were more effective in obtaining high-quality results for varying difficulty of tasks where subjective interpretation was required. When considering developing GraphCrowd, a huge consideration was how do we evaluate if a layout produced by a crowdworker is satisfactory? Is it enough to simply give the crowdworkers a graph along with guidelines and expect them to produce valid layouts? In an effort to answer this question, we looked at CrowdScape [20]. CrowdScape is a system that allows for the evaluation of complex crowd work. It uses mixed initiative [21] machine learning along with interactive visualizations to present a user of the system information about worker behavior and worker logs [22]. It aims to try to evaluate crowd work for which there is no right answer. In this system, a user is presented with several filters. Each filter controls a characteristic about the work that crowdworker produces. Examples of filters include total time spent on the task, logs of user interaction while doing work, and even the mouse movements of crowdworkers. Using this system, a user picks a few points that they classify as "good" and "bad". These points are used as training data. This training data is fed into the Machine Learning algorithm which analyzes similar characteristics between each point. From here, it presents the user of points that contain similar characteristics as the points they considered "good". Using this system, a user does not have to evaluate every piece of work that is produced by the crowd. This system requires some user input in order to train the data. For each type of task, this must be done. Any subsequent task then uses prior data to recommend acceptable work.

GraphCrowd's entire premise is to alleviate the owner of a graph from having to do any additional work other than generating the graph. It does track user activity, but only for the purpose of avoiding crowdworkers submitting barely modified layouts. GraphCrowd did not employ mixed initiative [21] machine learning due to the flexible nature of the graph it supports. Graphs can be of various sizes and serve different purposes. Valid work between graphs may contain radically different characteristics. We did not want to penalize crowdworkers if a Machine Learning algorithm does not properly evaluate work. Instead, we resort to the crowd to once again provide feedback for their work. Simply put, CrowdScape is a system to recommend work that it believes a user will find useful based on characteristics of the work the user already likes. GraphCrowd uses the crowd to produce and evaluate its own work.

Work has been done to examine whether a crowdworker can provide feedback that is just as good as an expert's feedback [23]. In order to test this, Yuan et al. conducted asked students from a university design course to submit drafts of their work. Next, they recruited

workers from Amazon Mechanical Turk [6] and Upwork [24] to provide feedback. Specifically two workflows were used. The first workflow asked to provide feedback given a rubric of design principles. The second workflow simply asked for open-ended responses. Students were then given both types of feedback and were asked which type of feedback was more helpful. Their studies revealed that open-ended expert responses were generally more useful, whereas responses based on a rubric revealed crowdworker feedback was just as helpful as expert feedback. This paper mainly focused on the quality of feedback for a creative work. Creative work was provided in this paper, whereas in GraphCrowd, we not only ask the crowd to produce creative layouts following guidelines, but also ask them to evaluate the quality of the produced layouts. Its focus is to completely eliminate the need for researchers to spend any time laying out graphs that they generate.

Crowdsourcing allows one to recruit humans to complete tasks that would otherwise be algorithmically challenging. Jagadeesan et al. [25] utilize crowdsourcing to see if this is a viable approach when trying to identify economic ways of solving spatial dissimilarities of 3D objects. In this study, they uploaded multiple views to Amazon Mechanical Turk and asked subjective questions such as "Which objects are most representative" as well as asking them to put similar shapes together. They also asked researchers from Purdue University to conduct similar tasks. After comparing the results between the crowd and the researchers, they found that both groups classified shapes very similarly. This suggests that crowds can be a useful approach in subjective reasoning tasks. GraphCrowd also used researchers and crowdworkers to conduct subjective tasks. However, instead of asking them to rate shapes, we asked them to create entire layouts of graphs as well as evaluate them.

CritViz [26] is an online framework for obtaining real-time critiques for students in creative classes. For students in these creative classes, obtaining feedback is often a hard process. Design is a subjective task require the cognitive abilities of the creator as well as for the person critiquing the work. The subjective nature of these tasks make it hard for students to obtain feedback for large settings where having single discussions becomes impractical. Instead, they created a system where students themselves uploaded their assignments. Next, from all the assignments in the pool, they rated them. This allowed for students to receive feedback from many students as well as implicitly enforced a prerequisite of the reviewers having domain knowledge. This was an interesting read because it used as a similar approach as GraphCrowd. The same population of creating work were also rating work. Although GraphCrowd uses this approach, it relates it to creating layouts of graphs. In addition, GraphCrowd assumes no domain knowledge of any individuals laying out a graph and/or rating it.

There are multiple crowdsourcing sites other than Amazon Mechanical Turk. The worker growth of Amazon Mechanical Turk is starting to stagnate [27]. This means that the workers are becoming extremely efficient in completing tasks on the platform. Similar tasks and familiarity with doing tasks reduce effect sizes. In order to mitigate this issue, Peera et al. evaluated six other platforms. Of the three they tried to publish surveys for workers to complete, they found the half of them didn't even allow them to publish their tasks due to

administrative or time-constraint issues. They discovered that the only viable alternative to Mechanical Turk was CrowdFlower [28]. They conducted large scale tasks on this system and discovered that its large user-base helped them obtain results and helped mitigate the amount of workers that may be familiar with a particular system in order to game it. More specifically, this paper provided information for researchers on different crowdsourcing platforms they could use to target specific types of audience.

2.3 Citizen Science Platforms

In most of the crowsourcing systems discussed so far, crowdworkers were paid to do their work. However, it is not uncommon that a person wants to do the work simply because their interests self-align with the task being asked to perform. People who are willing to do tasks in this manner are known as citizen scientists. Citizen science [29] is an important concept as it allows scientists and researchers to gather data and manpower that previously would be unattainable. Pathfinder [30] is a system which gives these citizen scientists a platform on which to not only collect data, but also collaborate on it. It allows for users to open up discussions to analyze and even question their findings. It challenges the original concept of citizen science being one where amateurs collect data and experts analyze it. GraphCrowd differs from Pathfinder because it currently does not support citizen scientist support. Its crowdsourcing functionality is primarily utilized to generate layouts. It is not a platform that expects crowdworkers to analyze the meaning of graphs. The purpose being GraphCrowd is to see if non-expert crowdworkers can lay out a graph as well as an experienced researcher, provided some guidelines. The purpose of Pathfinder is to provide a system to promote collaboration among citizen scientists as well as a central place for all collected data to be deposited. It aims to try to involve citizen scientists in more than just data collection.

Foldit [31] is a multi-player online game in which players collaborate and compete to create accurate protein structure models. This project used the concept of citizen science [29] and gamification [32] to recruit players into designing algorithms or "cookbooks" to efficiently fold proteins. Players were able to collaborate and iterate on algorithms, producing one such algorithm that was almost identical to an algorithm independently created by researchers. This game emphasized the power of using crowdsourcing. This differs from GraphCrowd because it presents a game for protein folding, whereas GraphCrowd is a platform for laying out graphs. Foldit used citizen scientists for its player base. GraphCrowd makes no such prior assumption, recruiting any worker who is willing to do the task.

Common Sense [33] is an initiative to design hardware and software sensing devices that will allow users to collect data about the environment. Their vision is to integrate these common devices into common mobile devices so users may collect data for places that would otherwise be unreachable. As an initial trial, they attached air quality systems on street sweepers to collect measurements for each street. By doing this, they were able to acquire data on the streets of San Francisco. They also implemented web-based visualization tools

for anyone to view this data as well as online community features to discuss phenomena discovered when analyzing the data. This approach shows great promise as it allows for the acquiring of valuable data with little to no overhead. It is our hope to eventually incorporate GraphCrowd into using citizen science platforms so that volunteers may also spot patterns in identifying criterion for creating and improving visualizations of graphs.

Sensr [34] is a framework that was created to aid non-technical people without any programming background to create mobile applications. These mobile applications would consist of tools to help collect data. In this system, they launched a website with basic template of things an *author* or person who creates a mobile citizen science application would need to start collecting data. In this system, a user who wants to start collecting their own data would simply fill out a form and use drag-and-drop features to build a simple user interface that other volunteers can access and use as the main point of discussion and collection. When these applications are created, they are launched as *campaigns*. Volunteers can subscribe, report, and review data for these campaigns. Although a more thorough user interface with support of various platforms is still needed, they were able to lay the groundwork for promoting citizen science.

Chapter 3

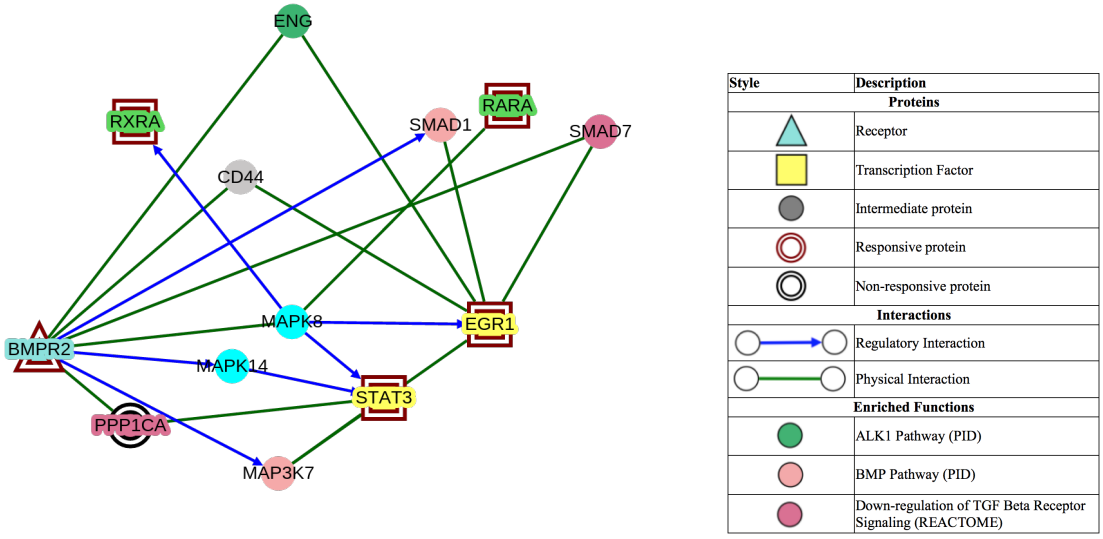
GraphSpace

Graphs have become ubiquitous in Systems Biology. Many applications exist for visualizing results through graphs. However, these applications tend to be specific to the computational algorithm used. In addition, standalone packages have limited features when it comes to sharing these generated graphs between collaborators. In order to address these issues, we developed GraphSpace: an easy-to-use web-based platform that collaborating research groups can use for storing, interacting with, and sharing networks.

3.1 System Walkthrough

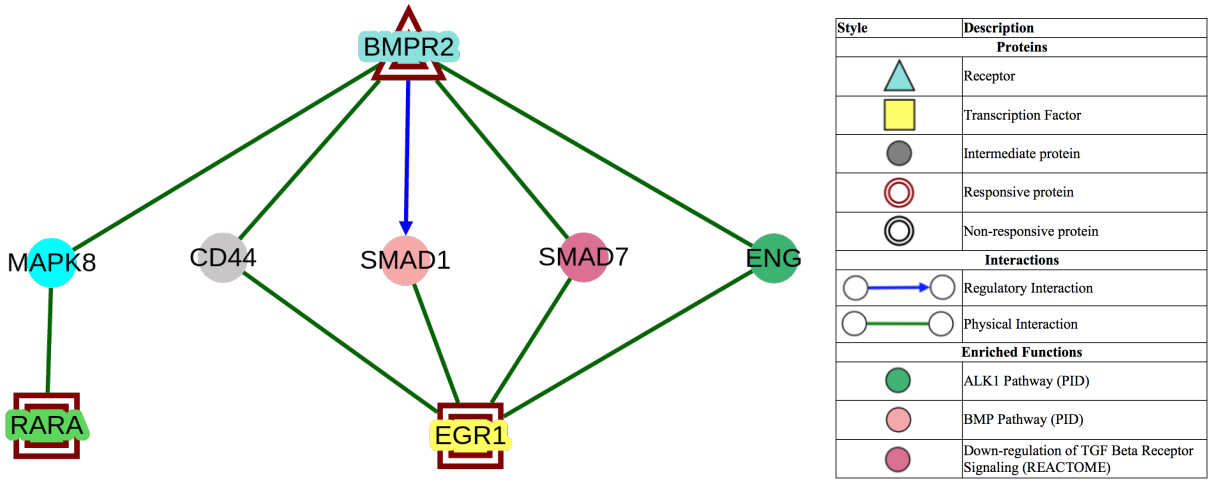
Mary is a computational biologist. She often generates hundreds of graphs which she uses to help model the results of running her experiments. Whenever she gathers data from an experiment, she generates a graph using an existing graph visualization application [15] [11]. Next, she interprets the graph and spends time laying out the graph in a manner which biologically makes sense. The graph she is currently generated is one to represent a a part of a cell signaling pathway [2]. Figure 3.1 one shows the graph that Mary originally generates from her experiments.

Figure 3.1: Represents part of cell signaling pathway that results when protein BMPR2 is exposed to Bisphenol-A.



Mary's goal is to share her results from this experiment with her collaborators. However, before she does, she wants to re-arrange the graph so that the proteins appear in the order the interact with each other in the pathway. She decides that she will arrange these proteins in top-down fashion where nodes that occur earlier in the pathway appear towards the top. She manually spends her time laying out this graph. Since it is a small graph, this does not take her too long. Finally, she takes a screen shot of the model and emails it to her collaborators. Figure 4.3 shows the graph after she re-arranges the nodes.

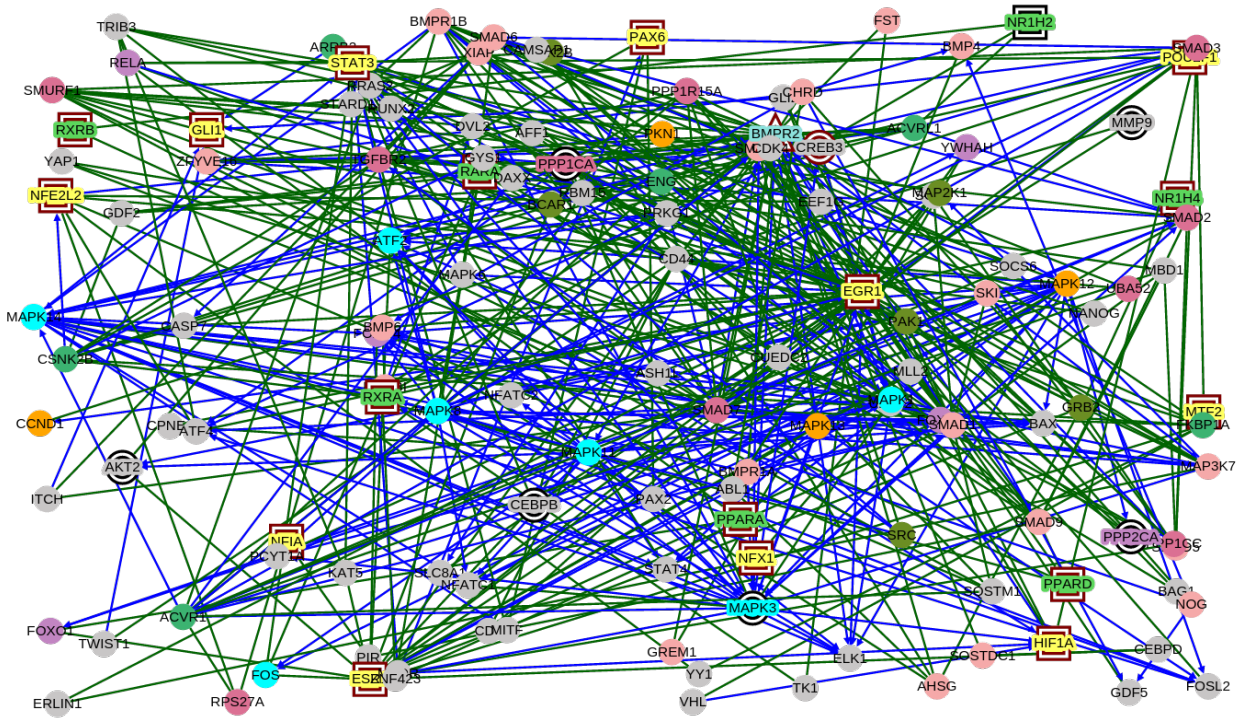
Figure 3.2: Graph after Mary manually lays out the elements.



One of her collaborators is very intrigued by this pathway and requests that she conduct a

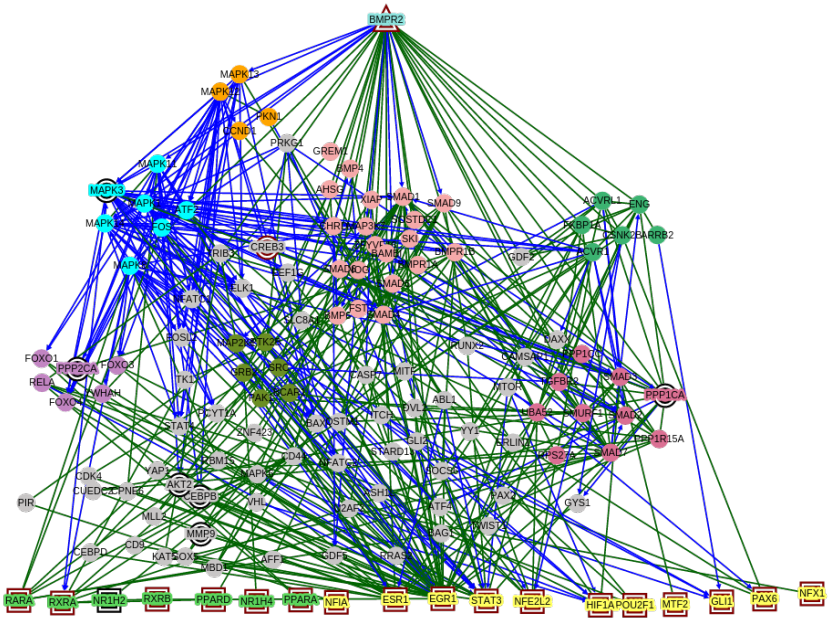
much larger experiment for this cell signaling pathway. Accommodating to their requests, Mary conducts a much larger experiment, figure 3.3 shows the graph Mary sees after completing her experiment.

Figure 3.3: Signaling Pathway graph when it is first uploaded to GraphSpace



Having no other tools to help her quickly lay this graph out in a manner which biologically makes sense, she spends a considerable amount of time laying out the graph. Figure 3.4 shows this graph after Mary is finishing re-arranging the graph.

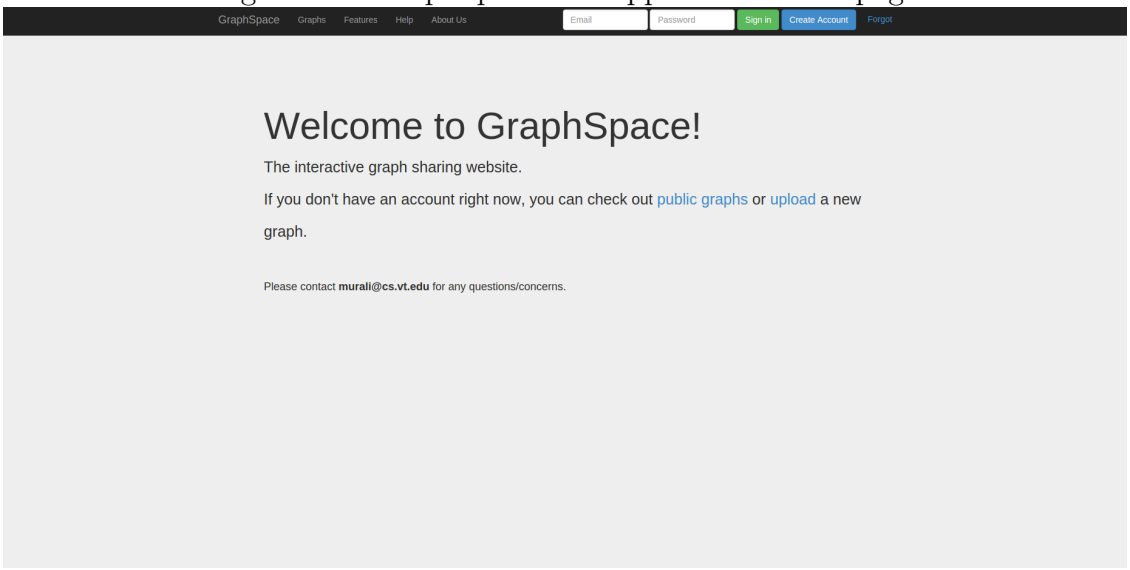
Figure 3.4: Signaling Pathway graph manually layed out by a researcher



3.1.1 Public vs Private Graphs

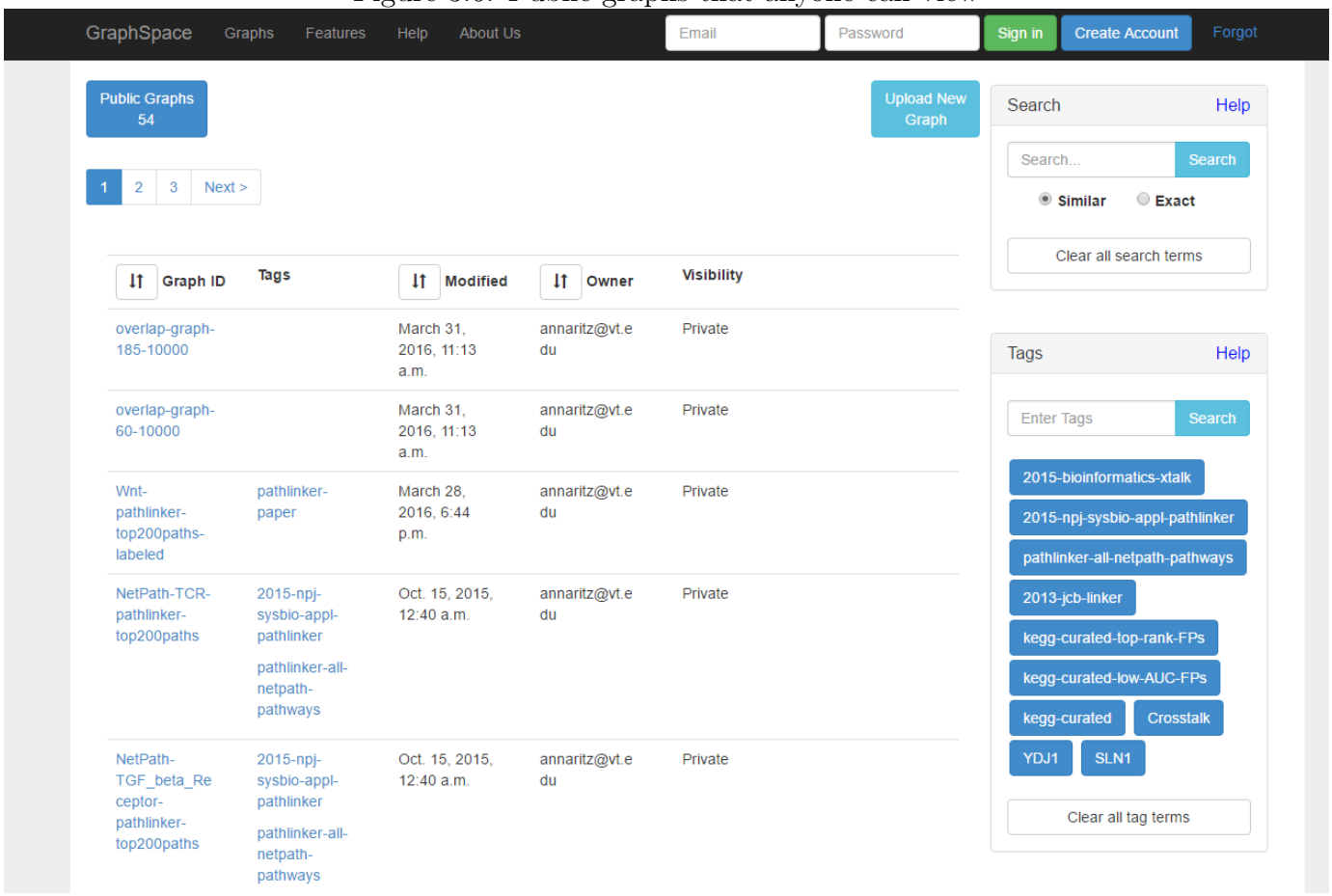
Her collaborators are very impressed and ask her to experiment with many other pathways and provide the results. Frustrated with the amount of overhead that she must incur to accommodate these requests, she researches if there is a system which can make her life easy. From researching on the web, she lands upon GraphSpace. Figure 3.5 shows what she sees. She notices that there are graphs nested under the “Public Graphs” section. She is able to view these graphs without creating an account.

Figure 3.5: GraphSpace web application home page



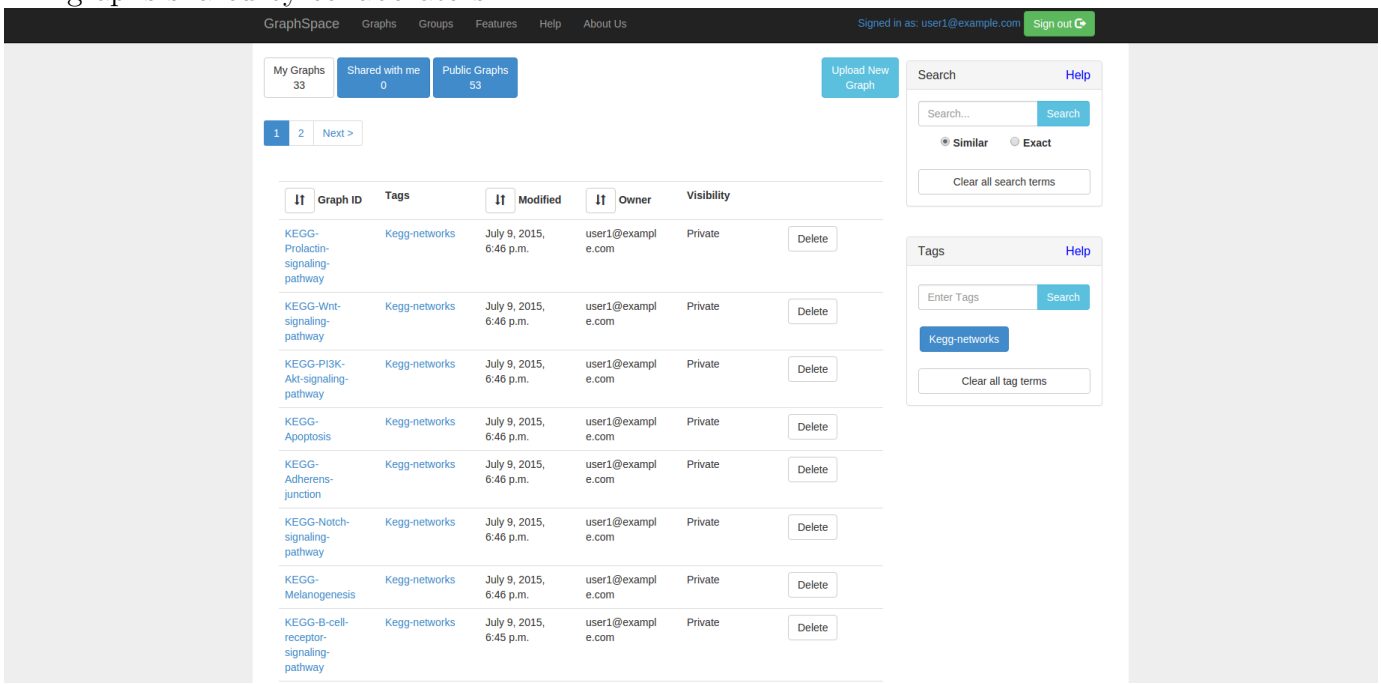
She clicks on "Graphs" link and sees that there are graphs uploaded by other people which she can see without logging in.

Figure 3.6: Public graphs that anyone can view



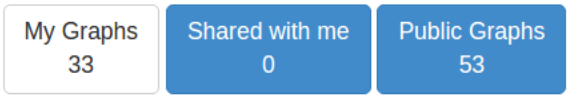
Excited at the amount of overhead she could potentially reduce using this website, she creates an account and starts using the application to quickly upload her graphs. After a week of using GraphSpace, she has uploaded many graphs and is very familiar with its interface. She has even invited her collaborators to this website who in turn have also uploaded some graphs. Figure 3.7 shows a picture of what Mary sees when she logs into GraphSpace.

Figure 3.7: Logging into GraphSpace allows for viewing graphs owned by a user as well as graphs shared by collaborators.



Her graphs are organized into three categories: My Graphs, Shared with Me, and Public Graphs. When she first uploads her graphs into GraphSpace, they only appear under My Graphs section. Only she is able to see them. The Shared with me category contains all graphs that her collaborators shared with her. Public graphs category contain all the graphs which the creators wish to share with everyone. She notices that she can only see the My Graphs and Shared with me categories after signing into her account.

Figure 3.8: The logged in user of GraphSpace has 33 graphs which they uploaded. There are no graphs that are shared with the user. There are 53 public graphs that the user may view.

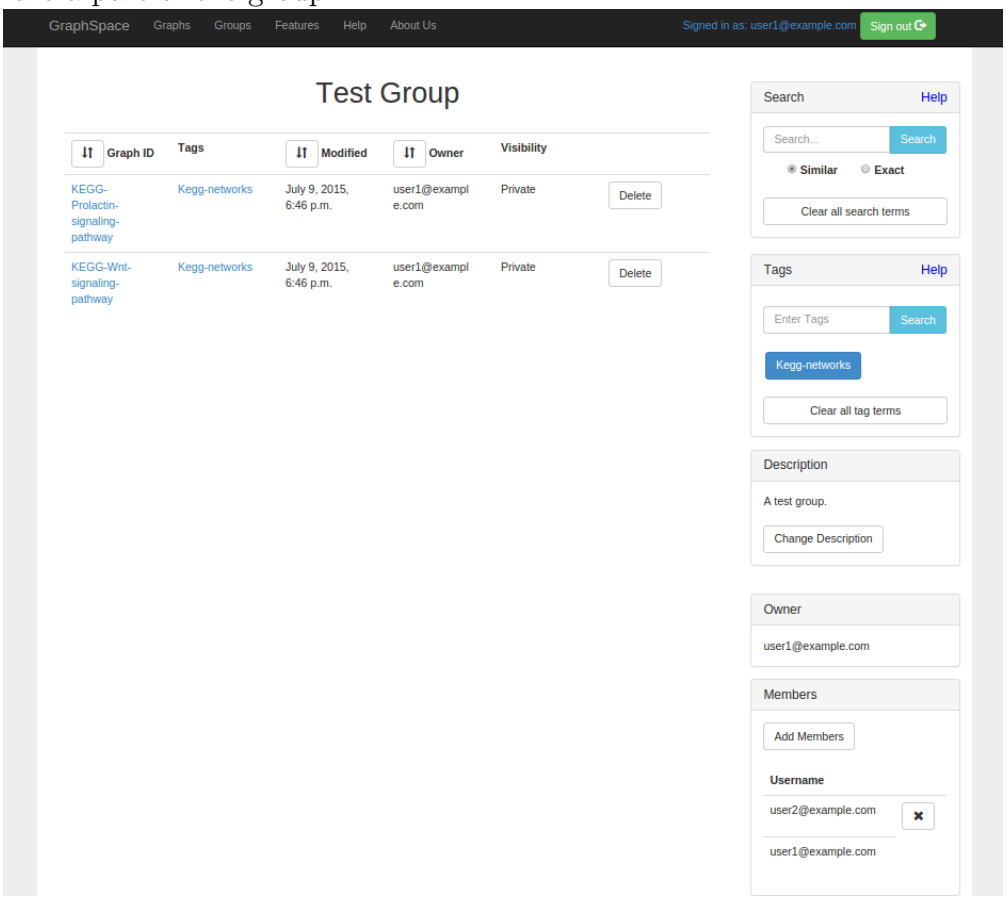


3.1.2 Groups

Mary's wants to share share her graphs with her collaborators. She notices that there is an option called "Groups". She realizes that this allows her to create a group and invite all of her collaborators on GraphSpace to become members of the group. Since she is the

group *owner*, she has the option has the option to share any graphs which she owns with this created group in addition to having the power to invite/remove any user of GraphSpace to/from her group. Figure 3.9 shows information about the group she created.

Figure 3.9: A group in GraphSpace contains graphs shared with the group along with members that are a part of the group.



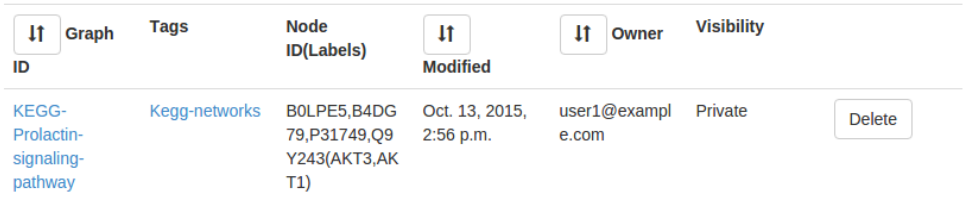
Her collaborators, who are members of this group also decide to start using GraphSpace. They upload their graphs and share them with the group as well. They notice that they can't invite other users to the group and have to ask Mary to. They may however, share their graphs and view graphs that other members of the group shared with them.

3.1.3 Search for Graphs

Being an avid user of GraphSpace, Mary owns many graphs. To find the graphs she wants to view, she uses the search functionality which allows her to search for all graphs that she is permitted to see. All graphs that return from her search queries are placed into their

respective categories as shown in Figure 3.8. Suitable search terms may consist of the name of a graph, a node that may be inside of a graph, or an edge that may reside inside of a graph. Multiple terms may be used as part of a search query provided they are separated by commas. She decides to search for all graphs that 1) contains a node called "AKT1" and 2) named "KEGG-Prolactin-signaling-pathway". To communicate what she is searching for, she enters the following search query into GraphSpace: "AKT1, KEGG-Prolactin-signaling-pathway". GraphSpace provides her with all graphs that are named "KEGG-Prolactin-signaling-pathway" *and* contain a node called "AKT1". Figure 3.10 shows the result of running her query.

Figure 3.10: A graph returned from a search query which both 1) contains a node called AKT1 and 2) named "KEGG-Prolactin-signaling-pathway".



Mary accesses GraphSpace on a new computer. She types in the search queries to find the graphs she is looking for. She realizes that her query only returned graphs that are public. She soon realizes that this is because she forgot to sign into her GraphSpace account. After signing in, her search queries also return all graphs that she owns and are shared with her that match the query.

Mary's search queries include looking for all graphs that have a certain edge. To find such graphs, she includes the names of the nodes from which the edge is coming from and is pointing to separated by a colon. She wants to search for all graphs which contain an edge between nodes AKT1 and AKT3, so she enters the following search query: "AKT1:AKT3". As a result, GraphSpace returns all graphs that contain this edge. Clicking on the graph takes her to the graph with the element of interest, an edge in this case, highlighted in order to draw attention to it.

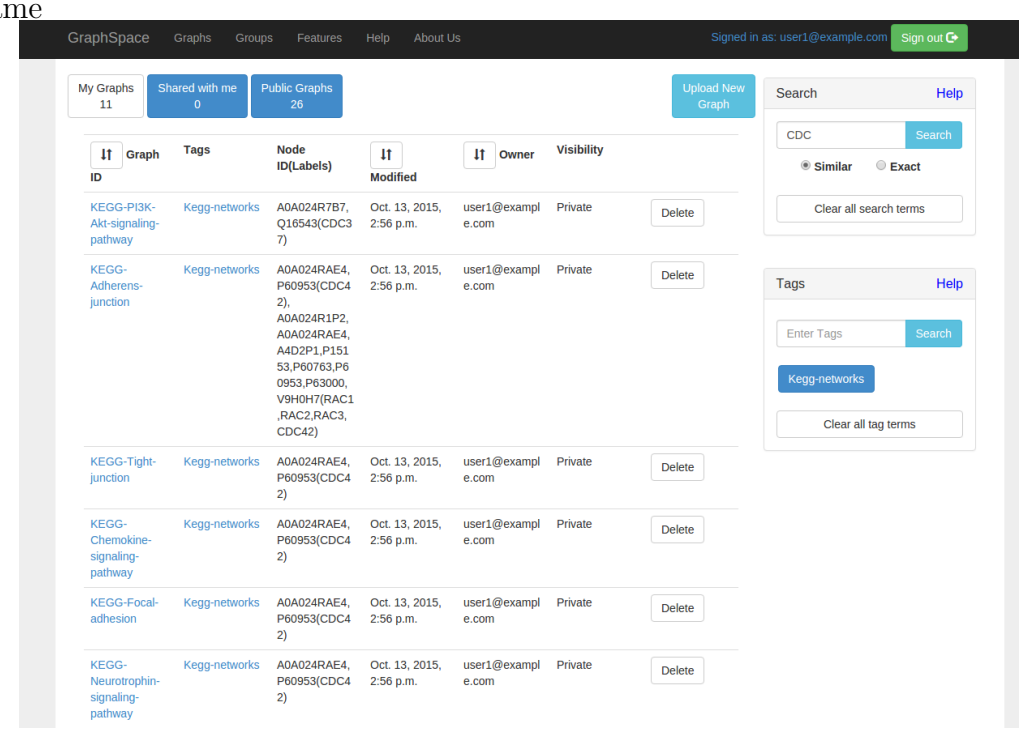
Mary notices that there are two types of options for search: Similar and Exact searching which she can use to search for graphs.

Similar Searching

Mary is unsure about exactly what she is searching for. She knows that she is searching for any graphs which contain proteins starting with "CDC". Using the search functionality, she uses similar search since doesn't know the exact proteins she wants to see. GraphSpace

returns all graphs that contain "CDC" as *part* of its name and all graphs which contain nodes which have "CDC" as part of their name regardless. Mary notices that this search is case insensitive. Figure 3.11 shows an example of running Mary's query.

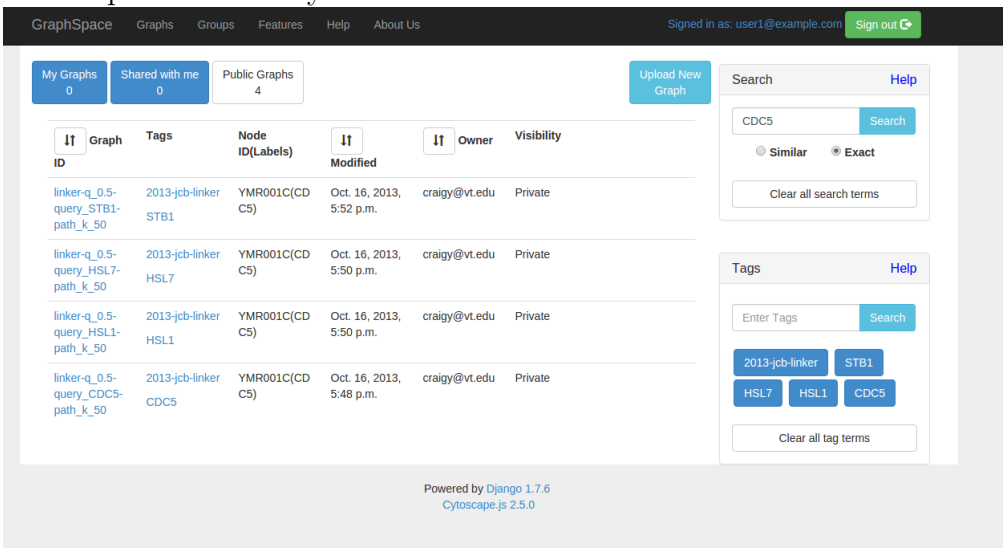
Figure 3.11: Graphs returned by similar search have nodes that contain "CDC" as part of their name



Exact Searching

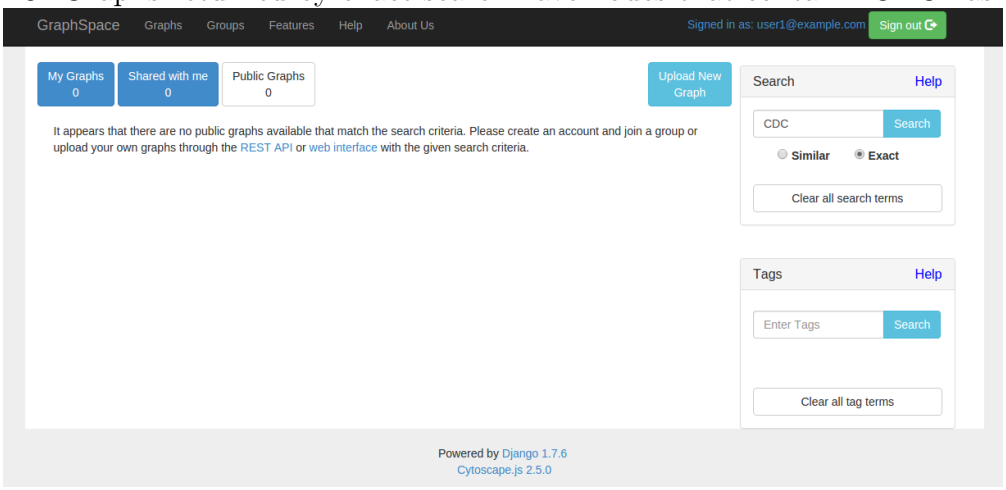
Mary knows exactly what she is searching for: all graphs which have the protein "CDC5". She wants to be as specific as possible, so she wants the search to be case sensitive as well. She uses exact search option to accomplish this. All graphs returned have either nodes that are named "CDC5" or the name of the graph is "CDC5". Figure 3.12 shows the result of running Mary's query.

Figure 3.12: Graphs returned by exact search have nodes that contain "CDC5" as its name



She decides to search for "CDC" term using exact search knowing that there were no graphs that were exactly named "CDC5" and no proteins with that exact value as well. Figure 3.13 shows the result of running an exact search with "CDC" search term. Notice that 3.11 and 3.12 returned produced different results.

Figure 3.13: Graphs returned by exact search have nodes that contain "CDC" as its name

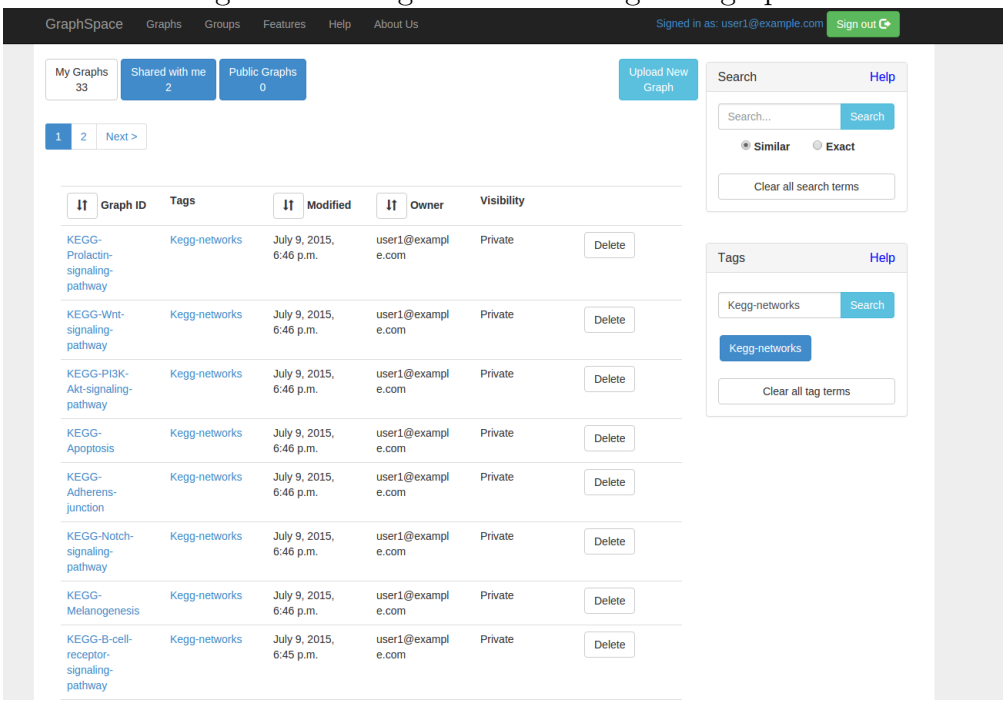


3.1.4 Tags

Another avid GraphSpace user, Bob has been asked to create many graphs that represent different pathways from KEGG [35]. After creating the graphs, he wants to organize them

so that he can see all of the graphs that he created for these pathways. Bob knows he can simply search for all graphs if he includes the term "KEGG" as the name of the graphs or if an element in the graph has "KEGG" as part of its name. However, this may produce other graphs that he didn't upload. In order to further organize his graphs, he uses tags. For all the graphs he uploads, Bob adds a special attribute "Tags" which allow him to organize all of his graphs under this attribute. Bob attaches a tag called "Kegg-networks" to his graphs. After they are uploaded, he searches for his tag term. GraphSpace returns all graphs that he has permission to see that exactly match the particular tag. Figure 3.14 shows the graphs returned from graphs using tags. If Bob wanted, he could have attached multiple tag terms separated by commas for his graphs. He could then search for any of the tag terms to retrieve her graphs. Bob finds this functionality quite useful and decides to use this organization for future papers by attaching his graphs with tags which corresponding to the respective papers they are created for.

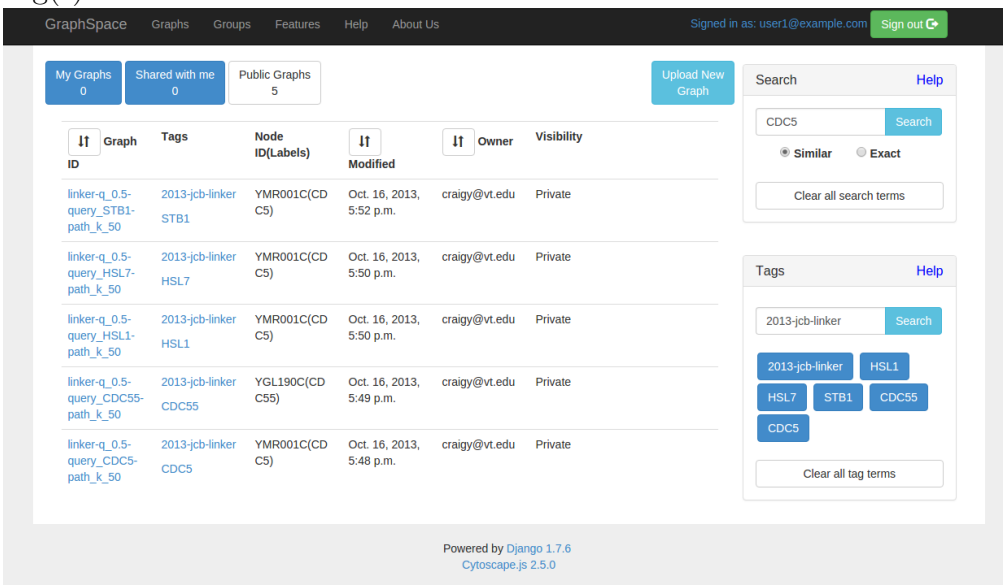
Figure 3.14: Tags are used to organize graphs.



3.1.5 Combining Searches and Tags

Bob wants to find all graphs that have the tag "Kegg-networks" and also have the term CDC5 associated with them. In order to specify this, he uses both search and tags. By doing so, GraphSpace returns all graphs that contain the search term *and* the tag term. Figure 3.15 shows the result of running Bob's query.

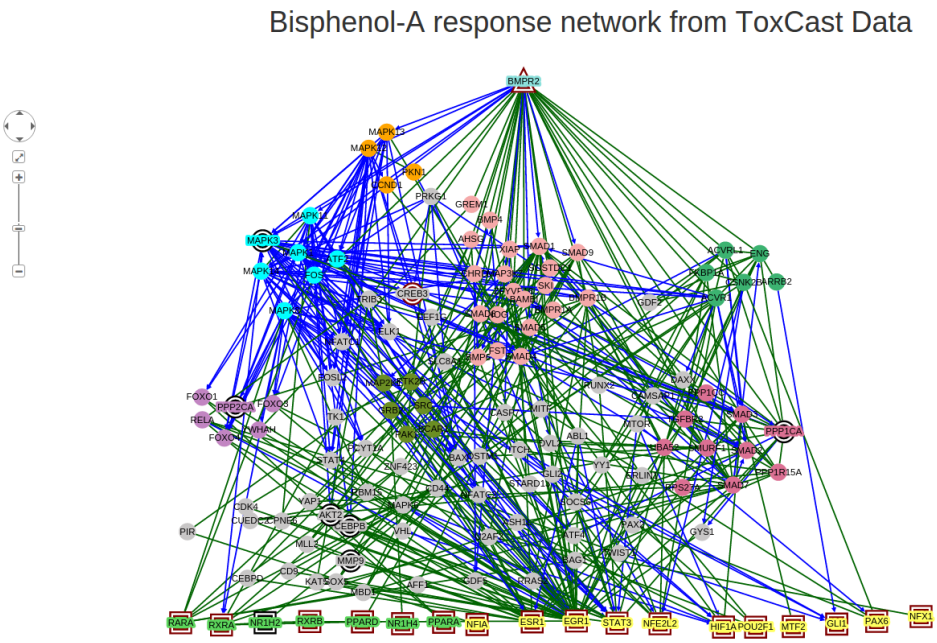
Figure 3.15: Graphs returned both contain CDC5 as part of the graph and contain the specified tag(s).



3.1.6 Visualization of Graphs

GraphSpace utilizes Cytoscape.js: an open-source graph theory library [36] to render graphs submitted via JSON objects as well as the ability to re-arrange the placement of elements when viewing the graph. Figure 3.16 shows how a graph that Bob has uploaded is rendered in GraphSpace. Bob may move any element in the graph. The arrow keys to the left of the graph allow him to move the view port of the graph. He may also zoom in to the graph and zoom out of the graph via the slider provided below the arrow keys. The particular signaling-pathway graph may be viewed at http://graphspace.org/graphs/ategge@vt.edu/27360-89-0:Bisphenol-A-NCIPID-edges.

Figure 3.16: An example of how a graph is rendered in GraphSpace.



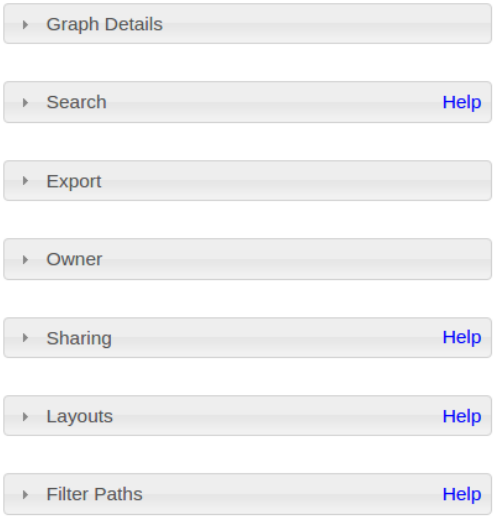
Both a node and an edge have required properties that they must satisfy in addition to optional CSS properties to customize the appearance of each element in the graph. Most of these properties are supported by Cytoscape.js. However, there are a few additional GraphSpace specific properties that are available for users to take advantage of: the "popup" and "k" properties. Both an edge object and a node object may contain these optional properties. The "popup" property may contain an HTML-parsable string that will be displayed in a popup window when a user clicks the element. If this property is not included, then no popup will be displayed when she clicks an element. Figure 3.17 shows how a popup would look like in GraphSpace. Since a popup is HTML-parsable, it may contain links to external sources as well as tables and figures.

Figure 3.17: Clicking on an element with the "popup" property reveals embedded information for that element.

3.1.7 GraphSpace Panels

When viewing a graph, Mary notices there are multiple panels each with their own functionalities. Features for the graph are allowed depending on if a user has proper write-permission for a graph and if a user is even allowed to see a graph. Figure 3.18 shows all the panels that Mary (owner of the graph) will see while viewing a graph.

Figure 3.18: Each panel provided with the graph contains its own features



Graph Details Panel

This panel contains all information associated with the graph itself. Mary would put information such as a graph legend, an explanation for the the significance of the graph, as well as any other information that would give a researcher or collaborator background information about the graph being viewed. Mary may upload HTML-formatted content in addition to plain text content.

Search Panel

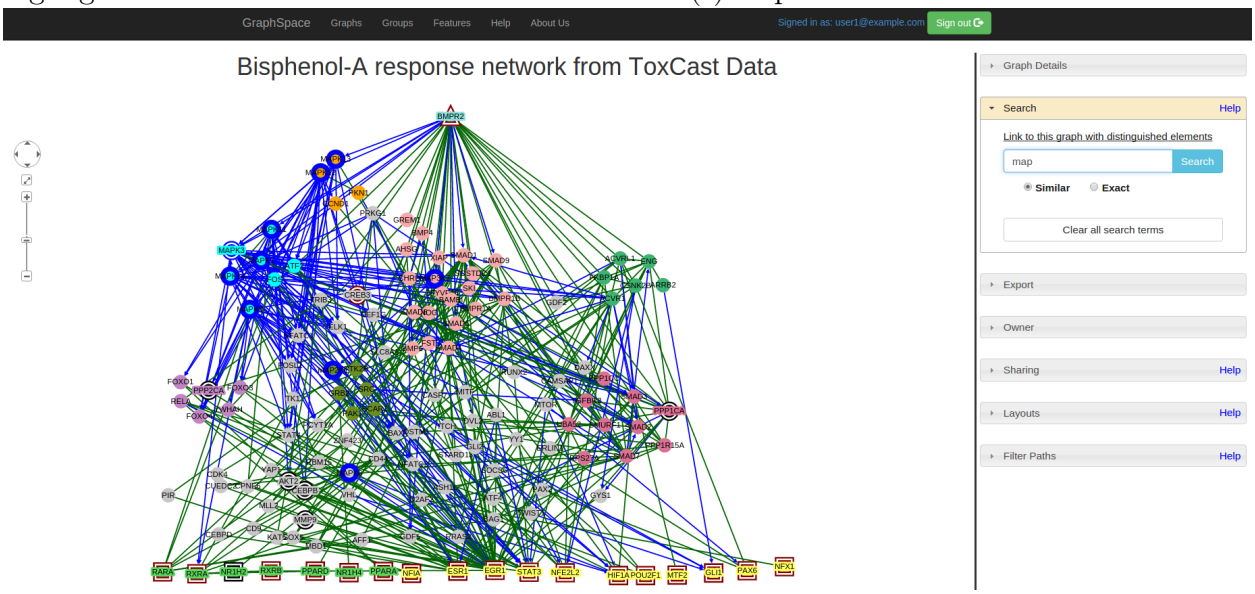
A graph may contain many nodes and edges. To help Mary or her collaborators find elements that are points of interest, GraphSpace provides this panel that allows a user to quickly find nodes and edges that match the search term(s) within the viewed graph. Multiple terms may be searched for at the same time provided they are separated by a comma (“,”). There are two types of searches that GraphSpace provides for the user when viewing a graph: Similar and Exact searching. If any elements in the graph match the search term(s), a blue outline will surround the matching element to help the user find the element of interest. Additionally,

a link will also be generated if any graphs match the search term(s). This link contains the current graph as well as the search term(s) entered by the user. This link is provided as an aid to the researcher so they may have a quick and easy method to share a specific graph with elements of interest already highlighted. If there are no elements that match the search term(s), then no link is displayed and the user will be notified appropriately.

Similar Search

Mary wants to search for all elements within the graph being viewed which have values that partially match the search term(s). Figure 3.19 shows how the graph shown in figure 3.16 would look like if Mary used similar searching to search for all nodes which contain "MAP" as part of its value. All nodes now contain a blue border around them in addition to a link that is generated in the Search panel.

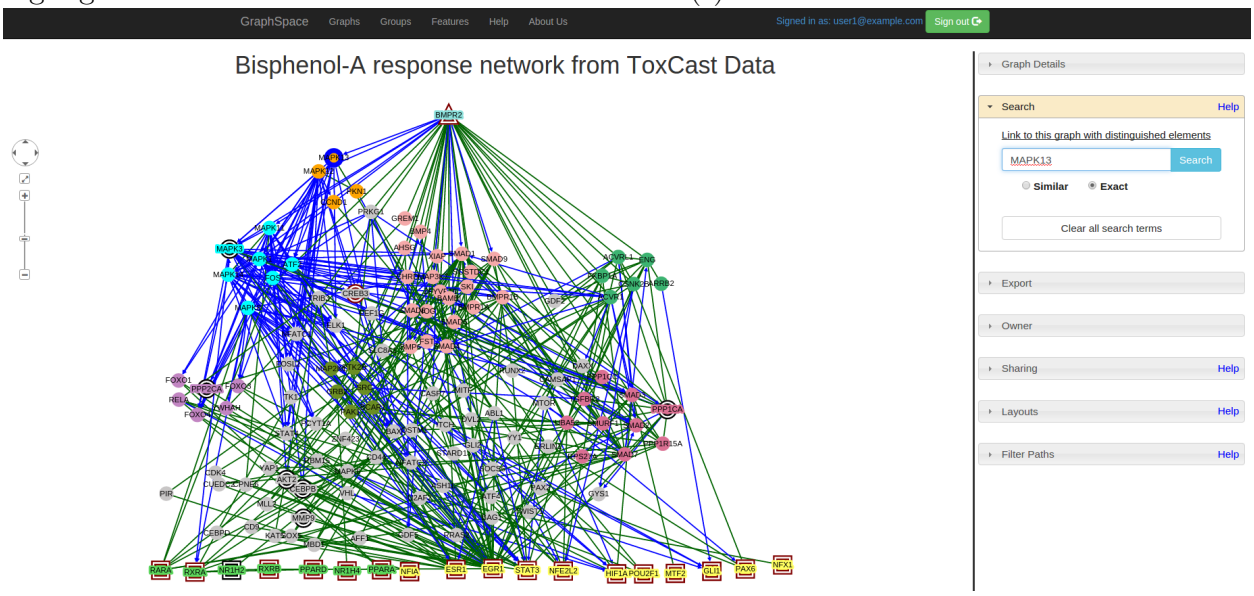
Figure 3.19: A graph which contains nodes that are being searched for. Similar searches highlights all nodes which contain the search term(s) as part of its value.



Exact Search

The *Exact* search option under the search bar allows Mary to search for all elements within a graph that are identical to the search term. Figure 3.20 shows what would be returned if Mary used exact search for "MAPK13". Only that specific node is highlighted.

Figure 3.20: A graph which contains nodes that are being searched for. Exact searches highlights all nodes which contain the search term(s) as its value.



Node vs Edge searching

All nodes and edges have an "id" property associated with them. This property may be accessed by viewing the JSON of the graph. Both a node and an edge may be searched for using this property. In addition to searching for nodes and edges via their id's, GraphSpace supports searching for nodes by their values (using either similar or exact searches). If a user doesn't know the id for an edge, they must search for an edge using the following syntax [tail: head], where tail represents either the value or the id of the node an edge is coming from and head represents the value or id of the node an edge is pointing to. For example, assume that there are two nodes: YEF3 and SIR2 and they are connected by a directed edge that comes from YEF3 to SIR2. To search for either one of the nodes, simply type the value for that node. To highlight the edge that connects both of these nodes, the search term would be "YEF3:SIR2".

Export Panel

This panel allows a Mary to view the JSON for a graph as well as export the image of the graph she is viewing as a Portable Network Graphics(PNG) image. This panel is useful when Mary wants to use the image of the graph she is viewing for a paper as it gets the exact dimensions of the graph that is currently being viewed (including any zooming modifications the user makes). If there are any terms that are highlighted when she clicks the PNG button under this panel, the resulting downloaded image will be the graph with those elements

highlighted as well.

Owner Panel

If Mary is a member or an owner of a group, she is allowed to see graph that are shared by other members of the group. In order to let her know who the owner of a specific graph is, this panel was created to display this information. This allows her to contact a graph owner directly if there are any questions raised for a specific graph.

Sharing Panel

Collaboration is at the heart of GraphSpace. This panel was created for researchers such as Mary to share their graphs with any group that they are a member of or own. All groups that a graph is shared with is displayed in this panel. If Mary no longer wishes to share a graph with a specific group, she may also unshare a graph through this panel. If Bob was a member of a group which Mary unshares her graph from, then he can no longer view her graph.

Layouts Panel

Mary may save any layout she has created for her graph or any graph that is shared with her here. Layouts that are shared by other users of GraphSpace for this graph also appear here. This panel also contains automatic layout algorithms to apply on the graph. There are 5 automatic layout algorithms provided. Figure 3.21 shows the Circle layout. It arranges all nodes such that they form a circle.

Figure 3.21: Signaling Pathway graph using Circle layout algorithm

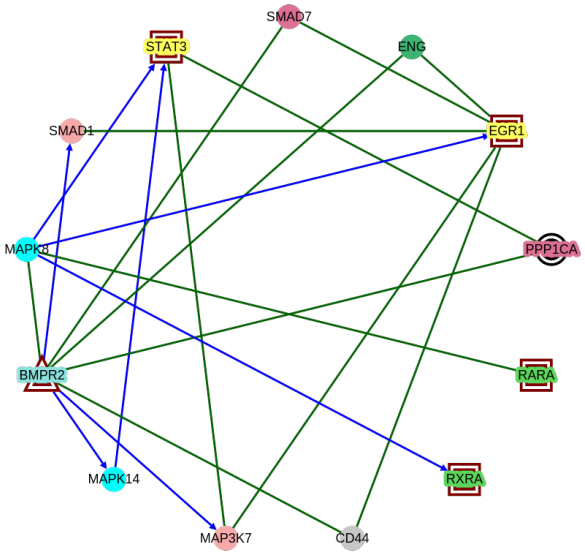
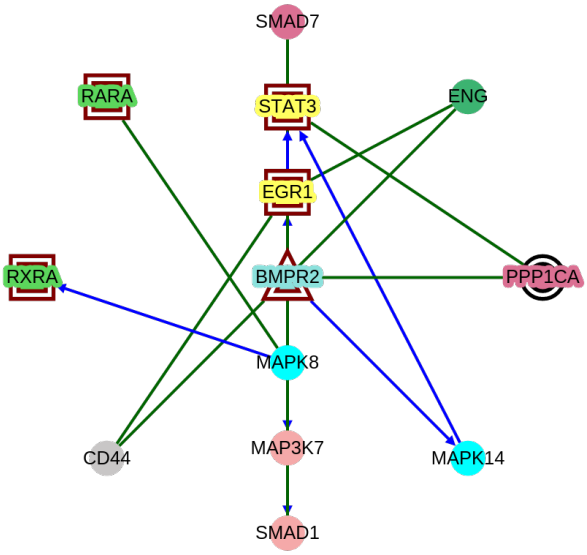


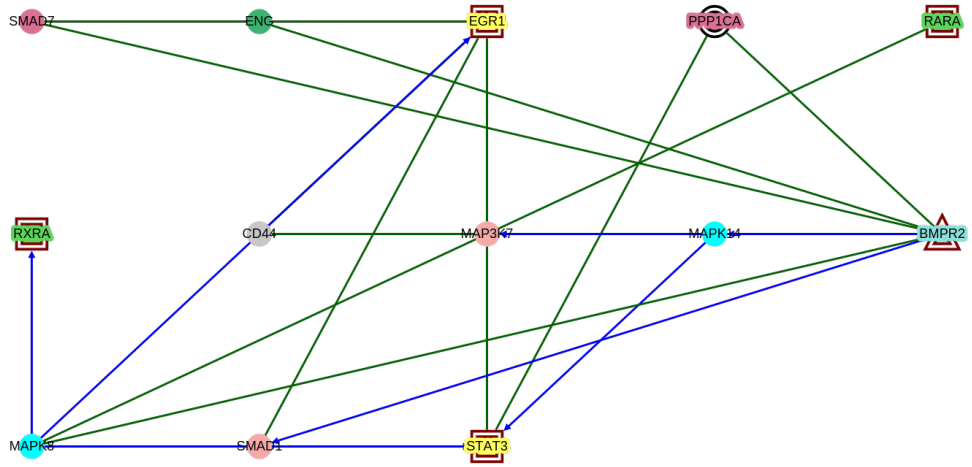
Figure 3.22 shows the Concentric layout. The Concentric layout positions nodes in circles based on levels. The smaller the level of a specific node, the smaller the circle. The levels that each node belongs to are randomly assigned.

Figure 3.22: Signaling Pathway graph using Concentric layout algorithm



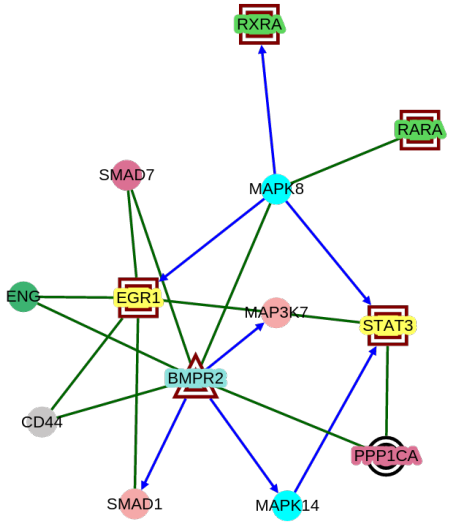
The Grid layout arranges all the nodes into a rectangular grid. Figure 3.23 shows the Grid layout.

Figure 3.23: Signaling Pathway graph using Grid [36] layout algorithm



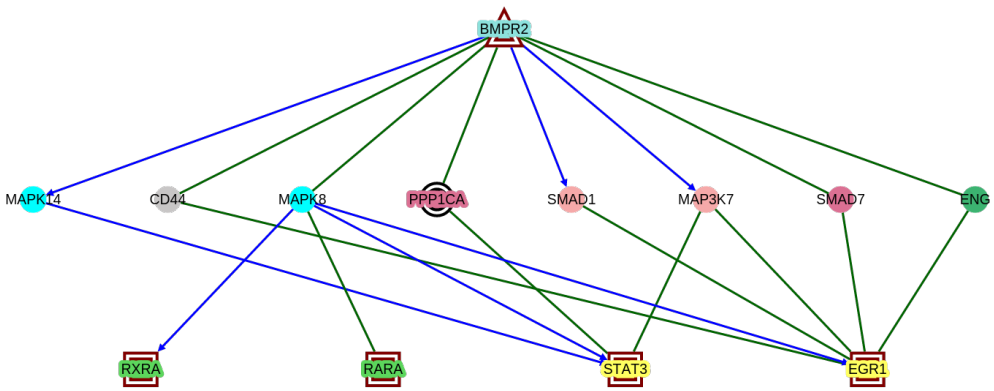
The Cosine Spring Embedder(Cose) layout relies on a physics simulation to layout out graphs [37]. Figure 3.24 shows an example of Spring Embedder layout.

Figure 3.24: Signaling Pathway graph using Compound Spring Embedder [37] layout algorithm



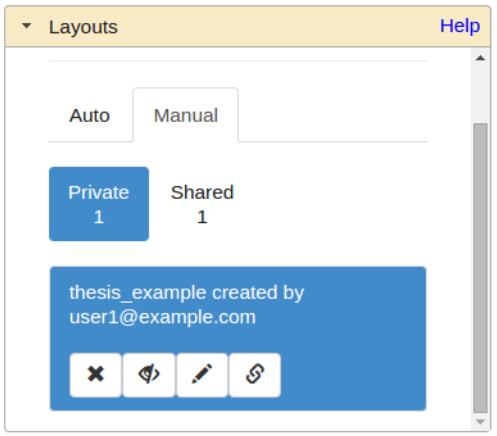
The tree layout uses a Breadthfirst algorithm to construct a tree structure for a graph with one node nested above all other nodes in the graph. Next, it breaks all other nodes into horizontal levels. Figure 3.25 shows an example of the Tree layout.

Figure 3.25: Signaling Pathway graph using Breadthfirst algorithm



GraphSpace automatically adjusts the zoom filter in order to fit all elements for a graph when using automatic layouts. The manual layout option includes two subsections: Private and Shared. The Private section under Manual layouts contains all layouts that Mary may have created and saved for a specific graph. Only Mary is allowed to see layouts under the Private section. For each layout private layout that is created, there are 4 options provided. Figure 3.26 shows an example of a private layout.

Figure 3.26: An example of all the options that are available to the owner of a layout for a graph.



Since Mary is the creator of the layout, she may delete the specific layout, share the layout, change the name of a layout, and obtain the direct link to the layout. In order to share

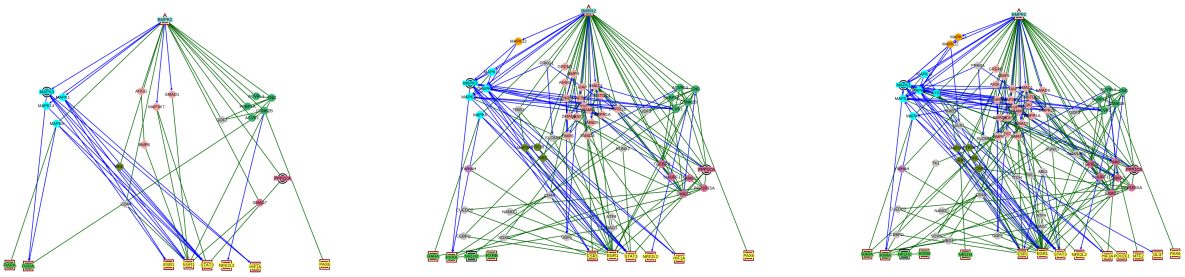
layout, the graph must be shared with at least 1 group. Starting from the leftmost button and moving to the right, each button shown in figure 3.26 accomplishes these tasks.

The Shared section under Manual layouts contains all layouts that are shared by either the graph owner or members of a group that the graph is shared with. All layouts under this section are accessible to anyone who is allowed to view the graph. If a graph is public, then all layouts under this section can be seen by anyone that uses GraphSpace. If Mary shares a layout that she created for her graph, she has the option of making a layout the default layout for a graph. When a layout is set as the default layout for a graph, then whenever a user of GraphSpace visits that graph, it will automatically use that layout. If Mary decides to unshare the default layout, then the graph will resort back to randomly laying out elements of the graph.

Filter Paths Panel

Mary uploads a graph which contain nodes and edges which have the "k" property associated with them. When she goes to view the graph, she notices that there is an additional panel available to her: Filter Paths Panel. This panel has two properties: Number of highly probable paths and Max highly probable paths. Number of highly probable paths reveal interactions that occur at a certain value. She uses the "Max highly probable paths" option to tell GraphSpace to adjust the view port to only fit elements that contain "k" values up to Mary's specified value. This value helps GraphSpace spread out elements in the graph. Figure 3.27 shows an example of this feature. Mary may use this feature to step through pathways as they sequentially occur. At different values of "k", elements of the entire graph are revealed. Mary uses this feature to incrementally step through the signals of a pathway as they occur. The "k" can be thought of as a unit of time. All elements that are revealed show that the interactions have occurred at "k" unit of time.

Figure 3.27: Signaling pathway graph incrementally being shown using filtering.



3.1.8 REST API

For registered GraphSpace users such as Mary and Bob, an application program interface (API) is provided. This allows anyone to programmatically communicate with GraphSpace

by writing scripts that send requests to GraphSpace. This API follows REST [38] architecture. It allows access to almost all of the functionality that is provided through the web application. An example of when it would be advantageous to use the REST API would be for tasks that involve multiple iterative tasks such as uploading many graphs and/or sharing graphs with groups. This also enables more advanced users of GraphSpace to create scripts that programmatically interact with functionalities.

3.2 Implementation and Availability

GraphSpace is an open-source project. GraphSpace is written in Python [39] and uses Django [40] as its web framework. In order to render and manipulate graphs, it leverages Cytoscape.js [36]. GraphSpace uses SQLite as its database.

GraphSpace is publicly available on http://graphspace.org. Its code is hosted on GitHub [41]. Currently, it has over 500 commits and 2 releases. In addition, GraphSpace development also takes advantage of GitHub issues: where anyone may post an issue regarding a bug, enhancement, or modification request. Throughout the development of GraphSpace over 100+ issues have been generated by the Murali group and almost all issues have been worked on. The code hosted on GitHub include instructions on how to set up a local version of GraphSpace on any computer running a Unix-based operating system. An easy-to-install guide was included in order to encourage open-source contribution to the codebase in addition to aid the adoption of GraphSpace for other institutions who wish to run a closed-version of GraphSpace. GraphSpace is currently hosted on an Amazon Elastic Compute Cloud t2.medium instance [42]. This instance contains 4 GB RAM as well as 100GB of storage.

3.3 Chapter Summary

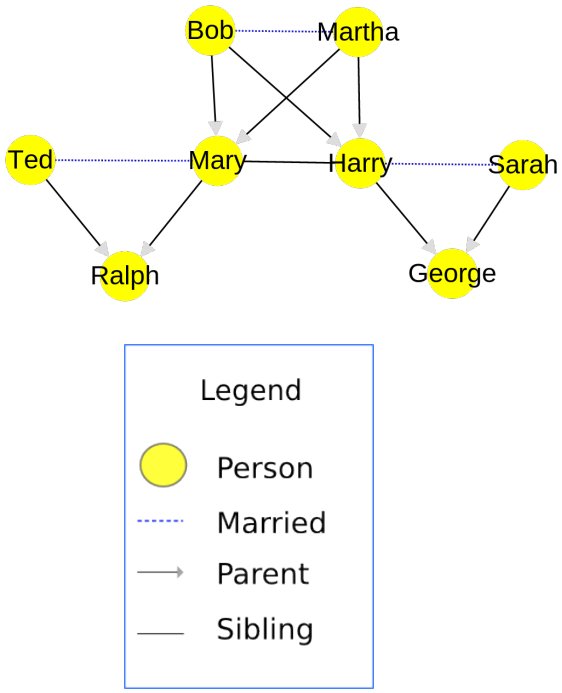
GraphSpace is a web application for storing, sharing, and visualizing graphs. GraphSpace provides a plethora of features for users to interact with while viewing a graph. GraphSpace supports the re-arranging the structure of any graph by allowing users to save their custom arrangement or layouts of these graphs. These layouts may be created by using one of the provided automatic layout algorithms or created by manually moving elements to their desired locations. GraphSpace presents a platform where users may share their graphs with their collaborators through the use of *groups*. Users may search for graphs as well as elements inside graphs. In addition, GraphSpace also provides a method of organization through the user of *tags*. By providing all of these features, GraphSpace alleviates the overhead researchers may incur from trying to create graphs to trying to view and share them.

Chapter 4

GraphCrowd

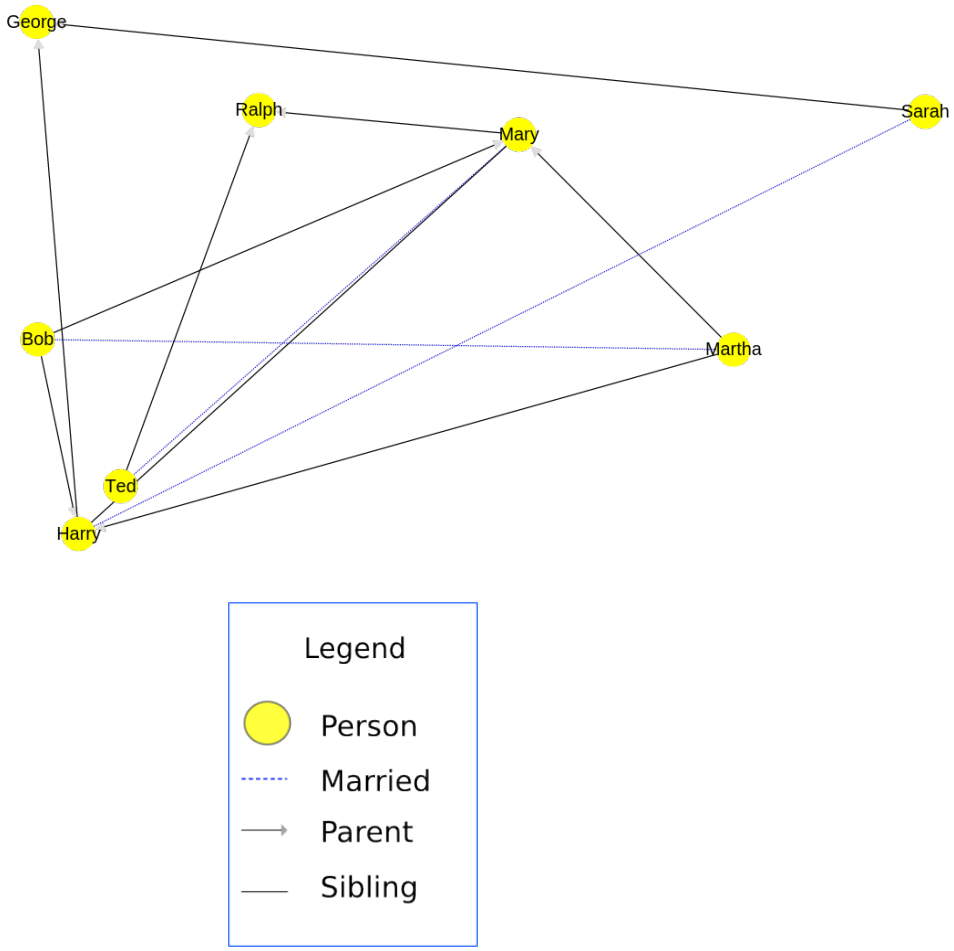
A picture is only worth a thousand words if it's meaningful. Graphs are often used to convey information that would otherwise be explained with a large amount of text. Researchers spend time to gather their data, interpret them, and create graphs/networks conveying their information. The visualization of these graphs plays a significant role in conveying the information. If a graph is not visualized properly, then very little information will be conveyed. For example, figure 4.1 shows a picture of a family tree. In this graph, a directed edge means that the node that the target of the edge is the descendant of the node that is the source of the directed edge. A black undirected edge means that they are siblings. A blue, dotted edge means that they are married to each other.

Figure 4.1: An organized family tree



The placement of the nodes reveals an inherent hierarchy. By arranging the nodes of the graph in such a manner, it conveys a sense of structure to the interpreter of the family tree. For example, if they wanted to know who the grandfather of "George" is, they can easily navigate their way up the tree. Figure 4.2 shows the same family tree with no apparent methodology of organization. The task of finding George's grandfather would be more difficult to find in figure 4.2 than it would be for 4.1.

Figure 4.2: A disorganized family tree



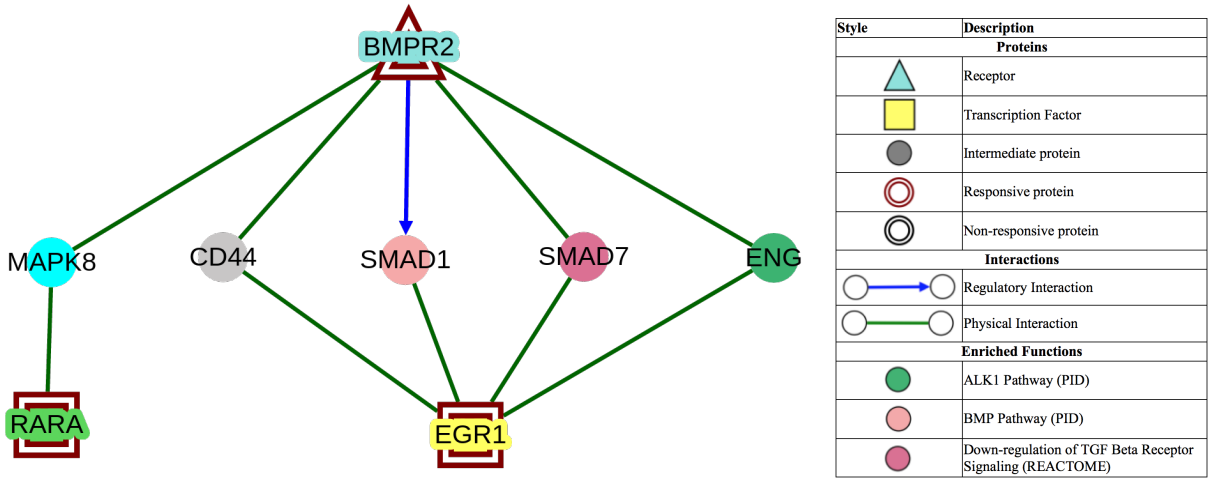
Researchers often generate many graphs to represent the data they collect. The size of the graphs range from very few to very many nodes. Laying out these graphs in a coherent manner takes time. Automatic layout algorithms do exist to aid the researcher in laying out the graphs without any additional user input. However, these algorithms tend to operate on the properties of the elements of graph itself rather than the context of the graph. An example of this includes equidistantly placing all nodes in a graph. The distance between each node is dependent on the number of the nodes in the graph. Thus, if the researcher wanted to convey information where distance between nodes was important, algorithms such as these would be useless to the researcher. In addition, automatic layout algorithms tend to be domain specific. Our intent was to find an approach that was flexible enough to support any type of layouts of graphs.

Laying out these graphs in a coherent manner takes time. Researchers must spend valuable

amount of time arranging each graph they upload. Laying out these graphs requires creativity from the researcher who use their cognitive abilities to decipher if a particular layout is acceptable. There may be multiple acceptable layouts for one graph which easily convey information in a meaningful manner. To create a system that would provide both a flexibility and support creativity, we decided to utilize human computation. Human computation leverages humans to solve problems that computers cannot. Humans have conceptual intelligence and perceptual capabilities that computers currently don't possess [43]. To recruit many humans to lay out graphs uploaded by researchers, we leveraged crowdsourcing. Instead of the researchers laying out their graphs, we leverage crowdworkers to do this for them. In order to do this, we developed a crowdsourcing extension to GraphSpace: GraphCrowd.

GraphCrowd provides functionalities to recruit the crowd to layout graphs. Currently, GraphCrowd supports graphs that fall under the rubric of "Graph Biology". For the purposes of GraphCrowd, we only focused on modeling signaling pathways graphs: directed or undirected connections between pairs of molecules. In these graphs, a protein is a node. If there is any physical interaction between two proteins, they are connected via an (un)directed edge. If there is a directed edge between two proteins, it represents a regulatory interaction. Figure 4.3 shows an example of one such graph.

Figure 4.3: Signaling pathway graph



Almost every publication in these fields contain graphs or graphs which contain nodes and edges laid out in two dimensions. In these graphs, the placement of the nodes and edges are determined by assigning x and y coordinates to all the node such that no two nodes overlap and that two nodes that are connected by an edge are physically placed near each other. An example of a method which utilizes these concepts is the Compound Spring Embedder layout [37]. However, one major drawback of constructing such a layout is that it incorporates almost no knowledge of the biological information underlying the graphs. Researchers often use their biological knowledge of the graphs to manually alter the positions of the nodes

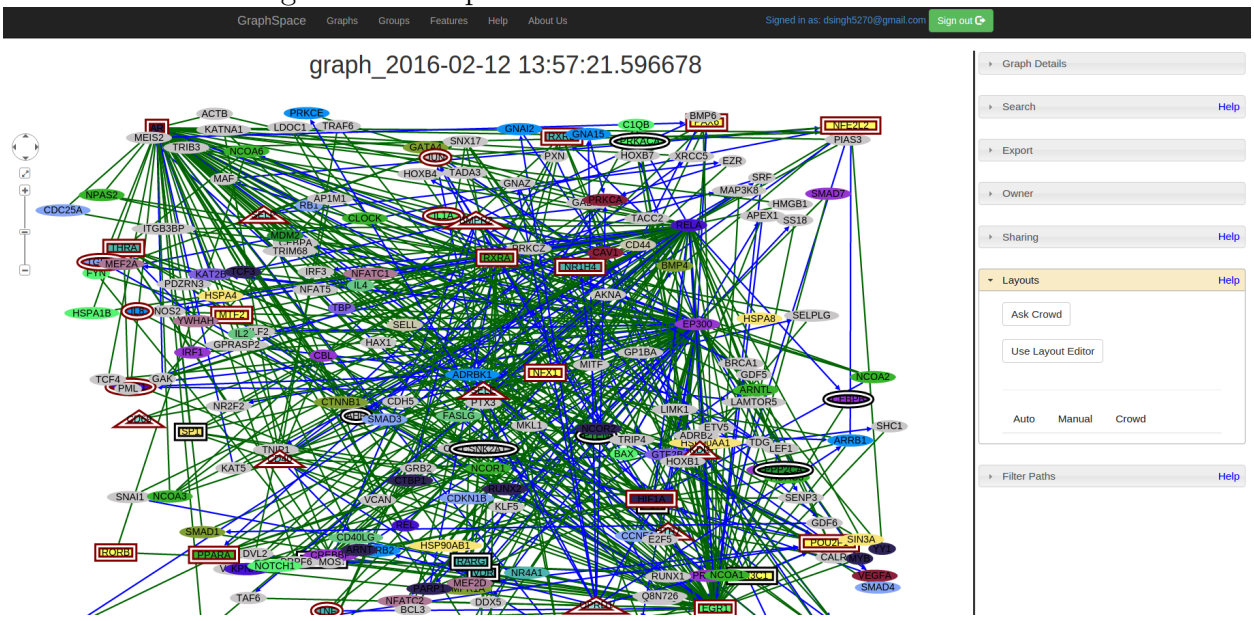
and edges to bring out salient features. GraphCrowd presents a unique approach for graph visualization that combines the ease of automated layouts with the ability for humans to observe patterns in order to produce information layouts for graphs. GraphCrowd uses mixed-initiative [21] crowdsourcing to create meaningful and intuitive layouts of chemical response graphs.

4.1 Researcher vs. Designer

GraphCrowd is composed of two types of users: *researchers* and *designers*. *Researchers* may be biologists, scientists, or anyone that wishes to upload their graphs directly into GraphCrowd and/or collaborate with other users by joining or creating groups. Alice is a researcher who uses GraphCrowd. Alice uses GraphCrowd to upload, manipulate, and share her graphs with her collaborators. Notice that Alice uses GraphCrowd similar to how researchers would use GraphSpace. The only difference now is that Alice doesn't need to manually lay out graphs she uploads to GraphCrowd. Instead, she can use crowdworkers to do these tasks for her. *Designers* are people who lay out graphs that are uploaded by *researchers*. John is a designer. He does not upload his own graphs. Instead, he lays out graphs uploaded by researchers such as Alice. John uses provided guidelines to aid him in creating meaningful layouts.

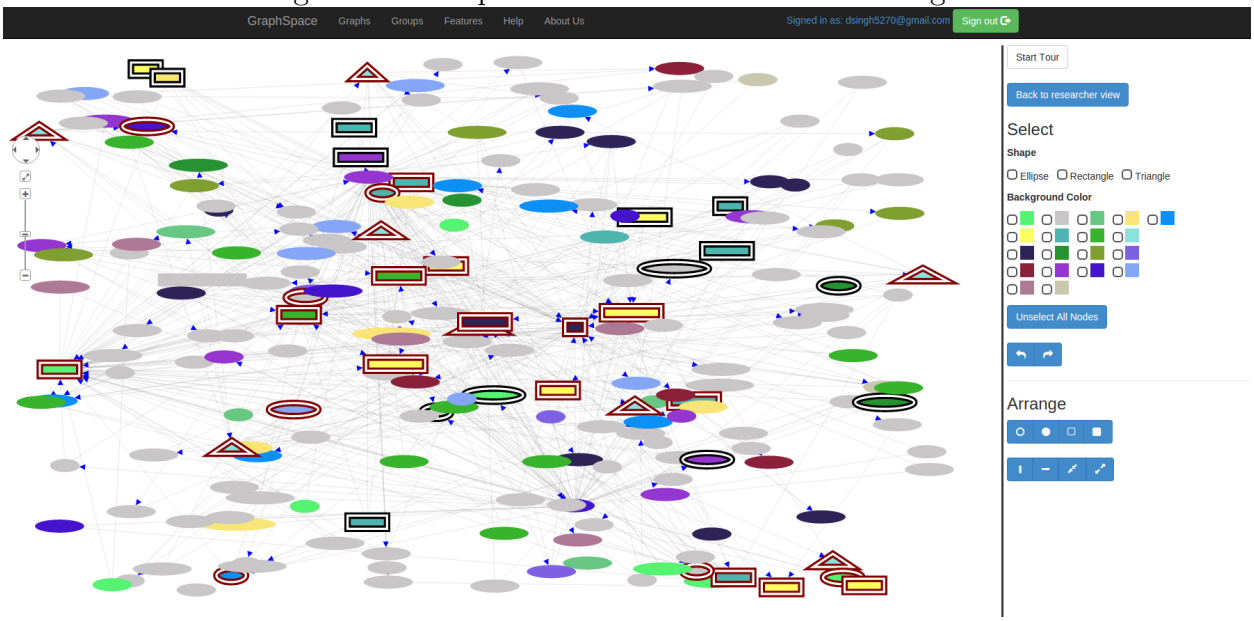
Alice and John are presented with user interfaces to aid them in their respective tasks. When Alice uploads a graph into GraphCrowd, she is presented with an interface that is similar to one that users of GraphSpace would see when viewing a graph. Figure 4.4 shows the interface that Alice would see while viewing a graph she uploaded to GraphCrowd. The only difference between GraphSpace's user interface and GraphCrowd's for Alice is that she now has the option to recruit crowdworkers to lay out her graphs by clicking the "Ask Crowd" button.

Figure 4.4: GraphCrowd user interface for a researcher



In GraphCrowd, crowdworkers may be thought of as designers. In other words, John is a crowdworker who was recruited to lay out Alice's graphs. Figure 4.5 shows the view John would see when viewing Alice's graph. Notice that information such as the labels on the nodes and thickness of the edges are hidden from John's view. This simplification of Alice's graph is implemented so John may solely focus on laying out the graph and not be distracted from any information that does not pertain to him. The interface provided to John aids him in quickly laying out the graph according to the specified guidelines. If Alice wanted to quickly lay out her graphs without recruiting crowdworkers, she may switch her role to being that of a designer. This presents her with the designer interface, allowing her to use the same tool palette that John is providing with.

Figure 4.5: GraphCrowd user interface for a designer



4.2 GraphCrowd Workflow

Now that we have discussed the different roles that GraphCrowd provides, we will now discuss the workflow that GraphCrowd operates under to harness the crowd to produce layouts. Alice uploads a graph into GraphCrowd. Since this sytem is extension of GraphSpace, she can still view, manipulate, and share her graphs with collaborators. In addition to these features, she can also recruit crowdworkers to lay out her graphs for her, saving her both time and effort. Alice wants to recruit crowdworkers to work on her graphs, so she clicks the "Ask Crowd" button shown in figure 4.4. When this happens, GraphCrowd launches a *task*. A *task* can be thought of as an activity that a crowdworker such as John has the ability to work on and complete. The system that GraphCrowd uses to publish these tasks is called Amazon Mechanical Turk [6].

4.2.1 Amazon Mechanical Turk

Amazon Mechanical Turk [6] is crowdsourcing platform where users can publish tasks and have anyone who is registered as a crowdworker in this system to work on those tasks for a monetary amount. More specifically, a requester is the person who publishes a task and wants crowdworkers to work on it. When Alice launches a task for her graphs, she is the requester. The type of task as well as the monetary amount for working on a task is decided by the her. A crowdworker is someone who works on these tasks for the monetary amount

assigned to each task. In this case, John would be the crowdworker. When he sees a task, he may accept the task or HIT (Human Intelligence Task). Once the task is complete, John notifies Amazon Mechanical Turk that the task assigned to him is complete by submitting the task for review on Amazon Mechanical Turk. After the work is submitted, Alice determines if the work conducted is satisfactory. If the work is satisfactory, then John receives the money associated with the task. If the work is not satisfactory, then John is not be paid for his work. GraphCrowd uses this platform to recruit crowdworkers such as John to work on layouts for graphs that the researchers such as Alice upload.

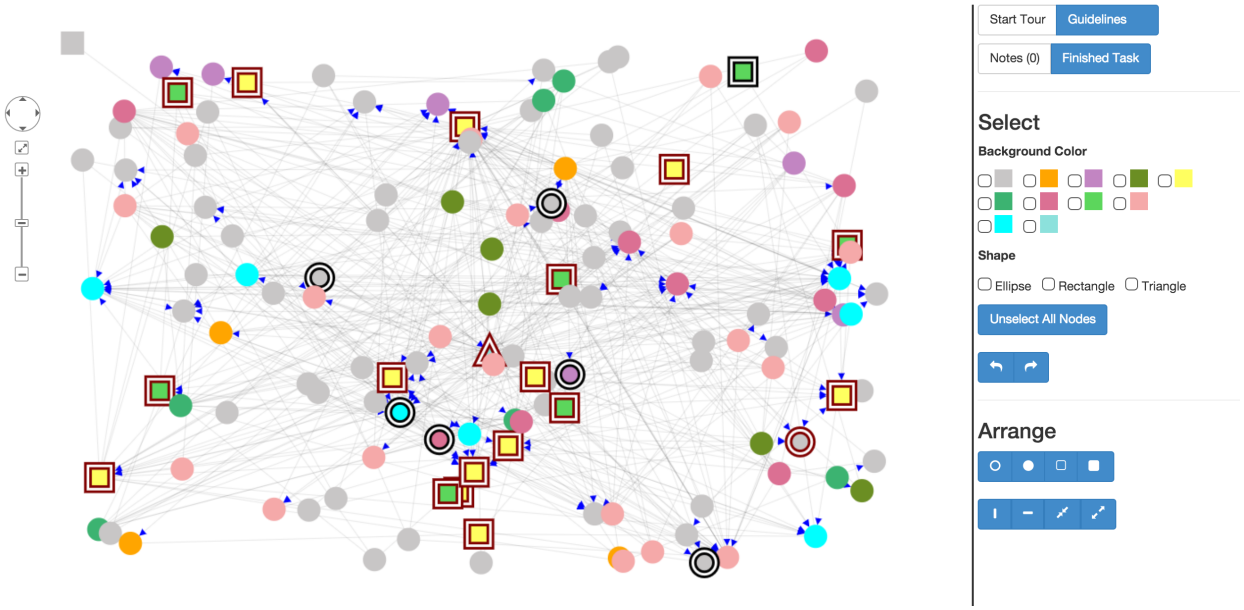
4.2.2 GraphCrowd Tasks

There are two types of tasks that GraphCrowd launches when Alice launches a task for her graph: Graph Layout Task and a Graph Approval Task. Figure 4.4 shows the "Ask Crowd" button that Alice clicks in order to launch tasks to recruit crowdworkers to lay out her graph. Once she clicks this button, five layouts are generated for the current graph. Each of these generated layouts contains nodes and edges that are randomly placed. For each of these randomly generated layouts, a Graph Layout Task is launched on Amazon Mechanical Turk.

Graph Layout Task

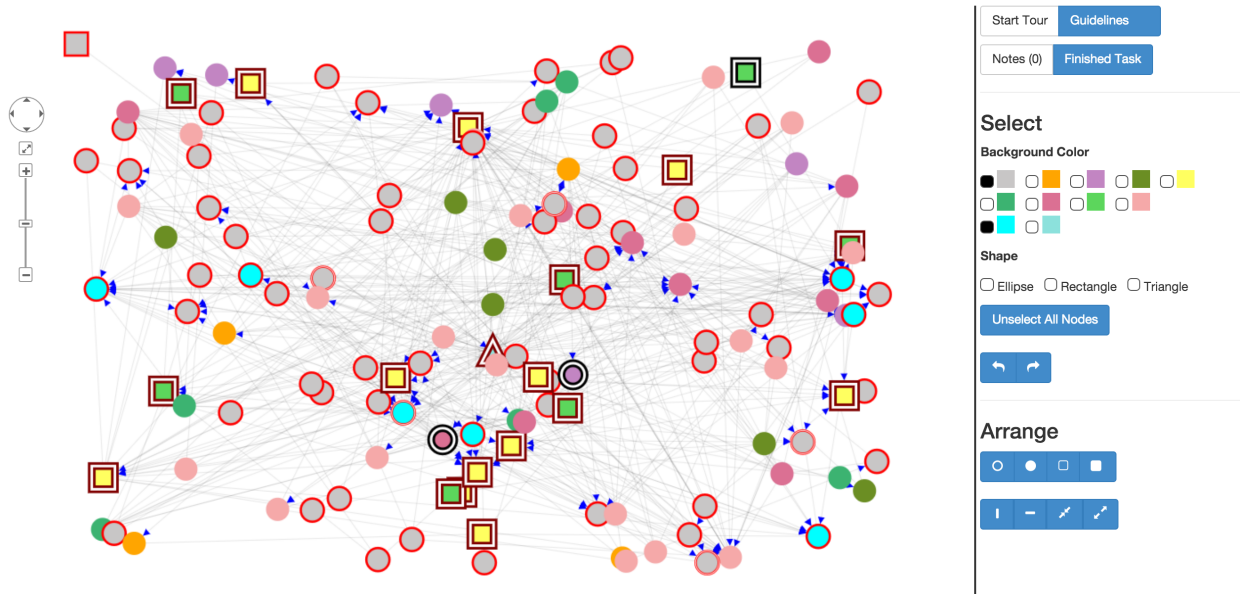
John, who is currently on Amazon Mechanical Turk looking for tasks to do, notices Alice's lay out tasks. John accepts these tasks and goes to GraphCrowd to view her graphs. GraphCrowd presents John with the *designer* interface since he only wants to lay out Alice's graphs. John is presented with a simplified graph along with guidelines to aid him in creating a meaningful layout for Alice's graph. Figure 4.6 shows the interface that John sees when working on Alice's graph. John clicks the "Start Tour" button which guides him through all the tools at his disposal by giving a brief introduction of what each tool allows him to do.

Figure 4.6: GraphCrowd Layout Task



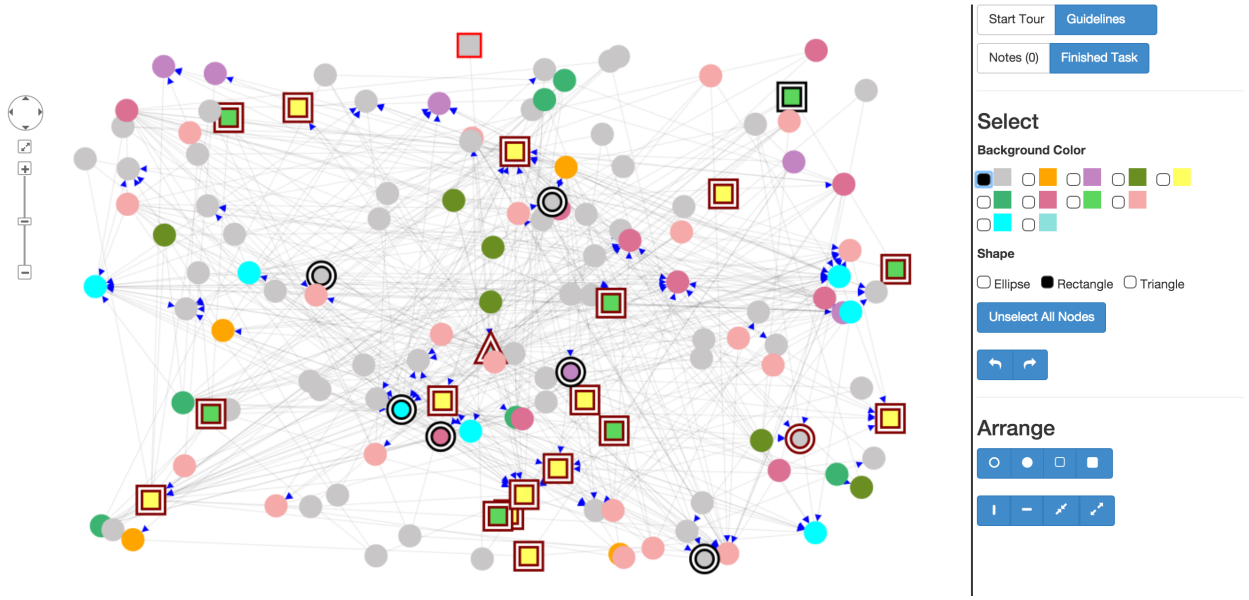
The tool palette that is provided to designers such as John is divided into 3 sections. The first section contains information regarding the task itself. This section is where John can view the guidelines again as well as any previous notes for the specific layout. He may add his own notes. When John finishes laying out the graph, he tells GraphCrowd by clicking the "Finished Task" button. The "Select" section provides John with a method to select multiple nodes that match the chosen filters. He can select nodes that contain a specific background color and/or a specific shape. All values in the background color and shape sections are dynamically created by reading through the contents of the graph, therefore all options that are provided will always have the complete set of all background colors and shapes that are in a graph. All boxes that are checked inside each subsection are treated as an "or" search of elements. In figure 4.7, two background colors are checked. All nodes that have either background color are highlighted. Once nodes are highlighted, they are are treated as a temporary collection that may be moved together. If John checks two shapes instead of two background colors, then all nodes that contain either shape would be highlighted.

Figure 4.7: All nodes that have either background color are selected



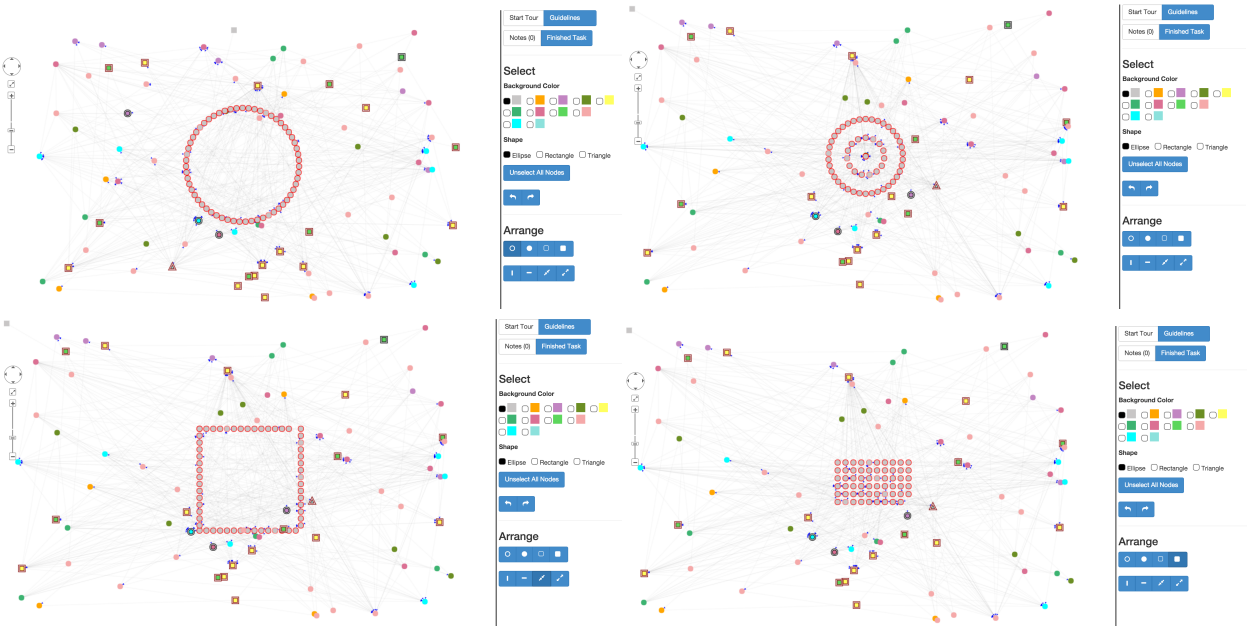
Designers also have the option to further narrow down their selection of nodes by using both filters. When both filters are used, it is treated as an "and" search of elements. Figure 4.8 shows an example of John using both a background color and a shape in order to narrow down their selection of elements. In this figure, the only node(s) that will be selected will be node(s) that have a silver background color *and* rectangle as its shape. For this particular graph, there is only one node (at the top of the graph) that matches all filters.

Figure 4.8: Only nodes that have the checked background color(s) *and* shape(s) are selected



The Arrange section allows all selected nodes(s) to be arranged into specific shapes/formations. The first row of buttons provided are shown in figure 4.9.

Figure 4.9: Different arrangements selected nodes the tool palette provides.



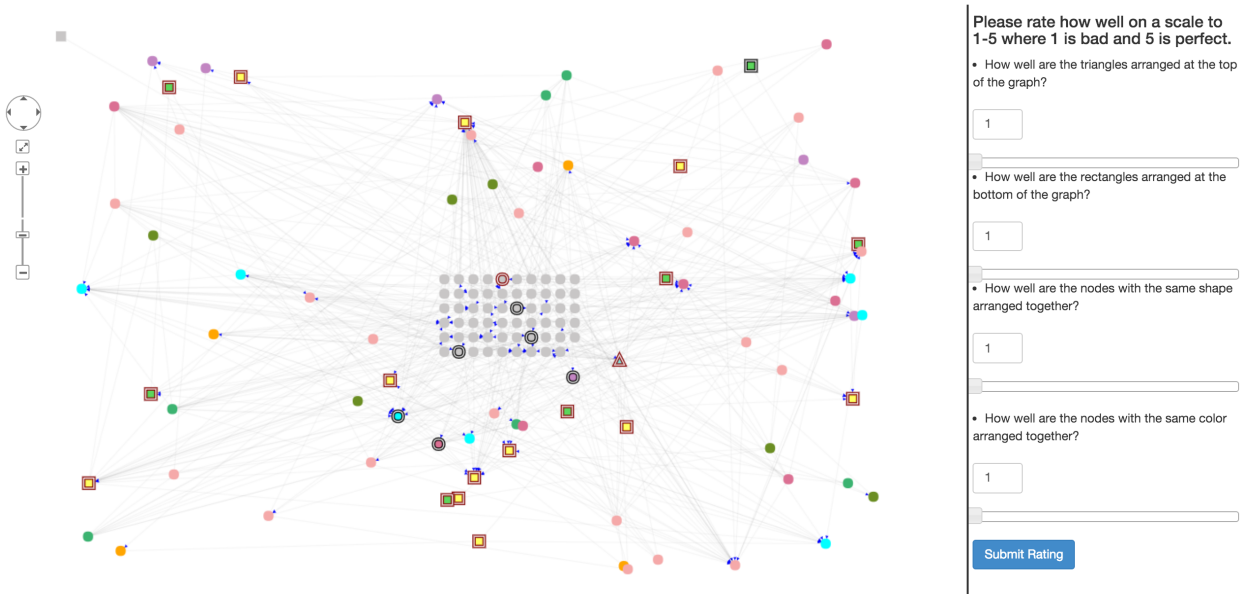
When John is done with laying out the graph, he clicks "Finished Task". As a result, a code

is generated and shown to him. John copies this code and pastes it in Amazon Mechanical Turk's input box provided to him when he first accepted the task. If John submits an invalid code, he is not compensated for his work. The only instance where GraphCrowd would not give John a valid code would be if he barely puts in any effort when laying out the graph. When John arrives at GraphCrowd from Amazon Mechanical Turk, GraphCrowd tracks to see if he puts in any effort into the task through the use of task fingerprinting [22]. Once he is finished with the task, and GraphCrowd determines if he has done sufficient work by presenting him with a code, five subsequent Graph Approval Tasks are launched on Amazon Mechanical Turk.

Graph Approval Task

Dave is another crowdworker on Amazon Mechanical Turk who is looking for tasks to do. After John is finished laying out Alice's graphs, GraphCrowd launches five Graph Approval Tasks on Amazon Mechanical Turk. Crowdworkers such as Dave may work on these tasks. Dave clicks the link and goes to GraphCrowd. He is presented with Dave's layout along with the guidelines Dave used to lay out the graph. For each guideline, Dave may choose a value from 1 to 5, where 1 is the worst and 5 is the best possible value for depicting how well a specific guideline is met. These ratings are then used to determine how the quality of John's layout. Figure 4.10 shows the interface Dave is presented with. After Dave finishes rating the graph, he clicks the "Submit Rating" button. Similar to the Graph Layout Task, a code is generated which Dave submits as his answer on Amazon Mechanical Turk to get paid for his work.

Figure 4.10: Graph Approval Task that crowdworker would see for a given layout.



Finally, after these tasks are concluded, Alice can go back to her graph and see that there are five layouts under the "Crowd" option in the Layouts Panel. She can click on any of the layouts to reveal the crowdsourced work produced from Amazon Mechanical Turk crowdworkers such as John who work on the Graph Layout Tasks.

4.3 Chapter Summary

GraphCrowd is a crowdsourcing extension to GraphSpace. It leverages the cognitive abilities of humans to work on creative tasks. Instead of researchers relying on automatic layout algorithms or manually laying out graphs themselves, GraphCrowd leverages crowdworkers to work on layouts for graphs. When a researcher first uploads a graph onto GraphCrowd, they have the option of recruiting crowdworkers to lay out their graphs. When a researcher chooses this option, five random layouts are generated for that specific graph. Each of these random layouts are launched as tasks for crowdworkers to lay out following provided guidelines. These tasks are known as Graph Layout Tasks. These task ask crowdworkers to simply lay out the graphs following guidelines. After a Graph Layout Task is finished, five Graph Approval Tasks are launched for the specific layout. These tasks simply ask crowdworkers to rate how well the layout being reviewed follows guidelines. These guidelines help determine the quality of the layout produced by crowdworkers. After this process has concluded, researchers will have meaningful layouts for their graphs.

Chapter 5

GraphCrowd Experiments

When developing GraphCrowd, we had the following research questions in mind: Can the crowd lay out graphs in a simplified, biologically meaningful manner as good as researchers and automatic layout algorithms? As a follow-up, can the crowd also rate these layouts as good as experts? To answer these questions, we conducted two experiments. The results of these experiments show promise that crowds can both create biologically meaningful layouts of graphs as well as rate them similarly to how an expert would rate the layouts.

5.1 Can the crowd lay out graphs?

Our first experiment was to determine if the crowd can create biologically meaningful, mechanistic visualizations of signaling pathway graphs. Our hypothesis was that if given simple guidelines, crowdworkers can produce better layouts than automatic layout algorithms and produce as good as layouts created by researchers themselves.

5.1.1 Method

To evaluate our first research question, we uploaded six signaling pathway graphs to GraphCrowd. These graphs varied both in the number of elements in the graph as well as the visual properties of the element such as background color. For each graph, ten layouts were created. These layouts randomly placed elements in the graph. We asked both collaborators with domain knowledge and crowdworkers to lay out these graphs adhering to the following guidelines: 1) How well are the triangles arranged at the top of the graph? 2) How well are the rectangles arranged at the bottom of the graph? 3) How well are the nodes with the same shape arranged together? 4) How well are the nodes with the same color arranged together?

Obtaining Layouts from Collaborators

An internal study was conducted among ten collaborators from the Murali researcher group. Collaborators in this research group had domain knowledge of cell signaling pathways. Each collaborator in this study was assigned three Graph Layout Tasks for different graphs. In this study, no layout was worked on more than once. For each of the six signaling pathway graphs, five layouts were worked on by collaborators in the Murali research group.

Obtaining Layouts from Crowdworkers

After the internal study among collaborators, there were five random layouts for each of the six signaling pathway graphs. For each of these random layouts, we launched Graph Layout Tasks on Amazon Mechanical Turk. These crowdworkers were provided the exact interface that was shown to collaborators in our internal study. Crowdworkers were asked to lay out the graphs using the same guidelines used during our internal study.

5.1.2 Results

In total, ten layouts were obtained for each of the six signaling pathway graphs. Five of the ten layouts for each graph were layouts produced by collaborators in our internal study. The remaining five layouts for each graph were layed out by crowdworker from Amazon Mechanical Turk. Figure 5.1 shows one random layout of a graph that was shown to both collaborators and crowdworkers. Figure 5.2 show the layouts that were produced by collaborators for one of the signaling pathway graphs. Figure 5.3 show the layouts that were produced by crowdworkers for the same signaling pathway graph.

Figure 5.1: Original graph shown to 5 collaborators.

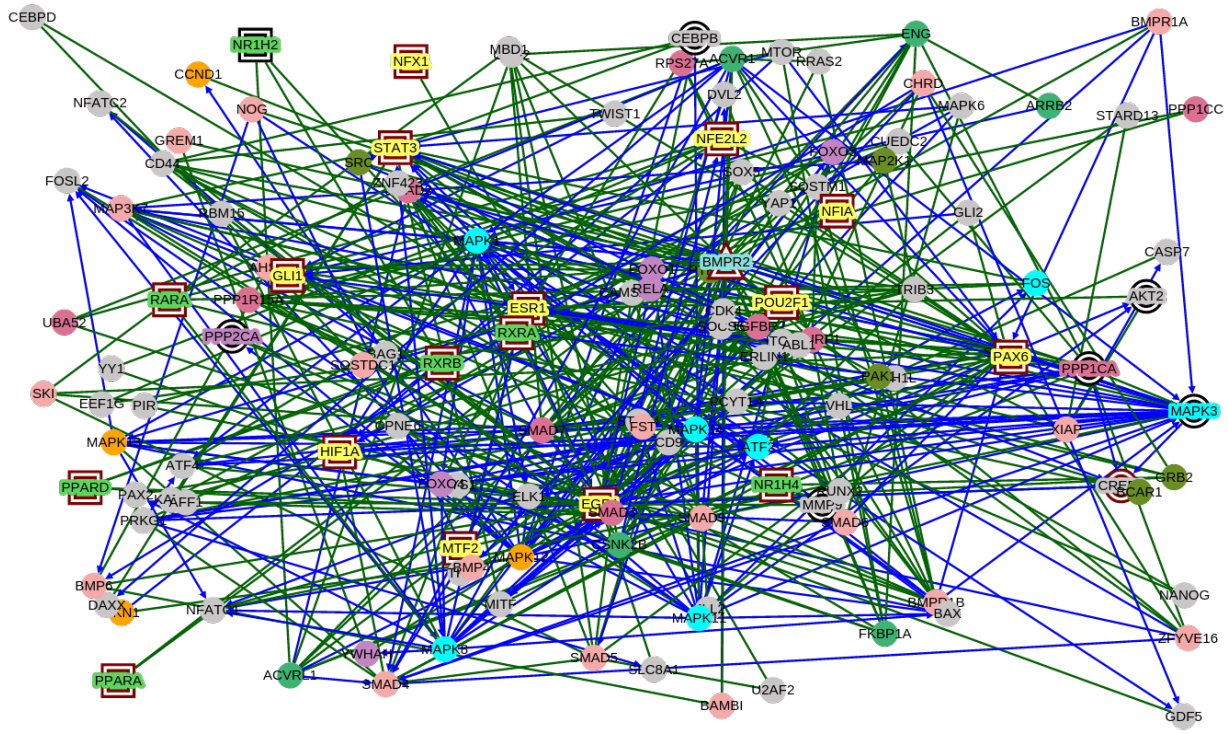


Figure 5.2: Layouts that were produced by five different collaborators for the same graph.

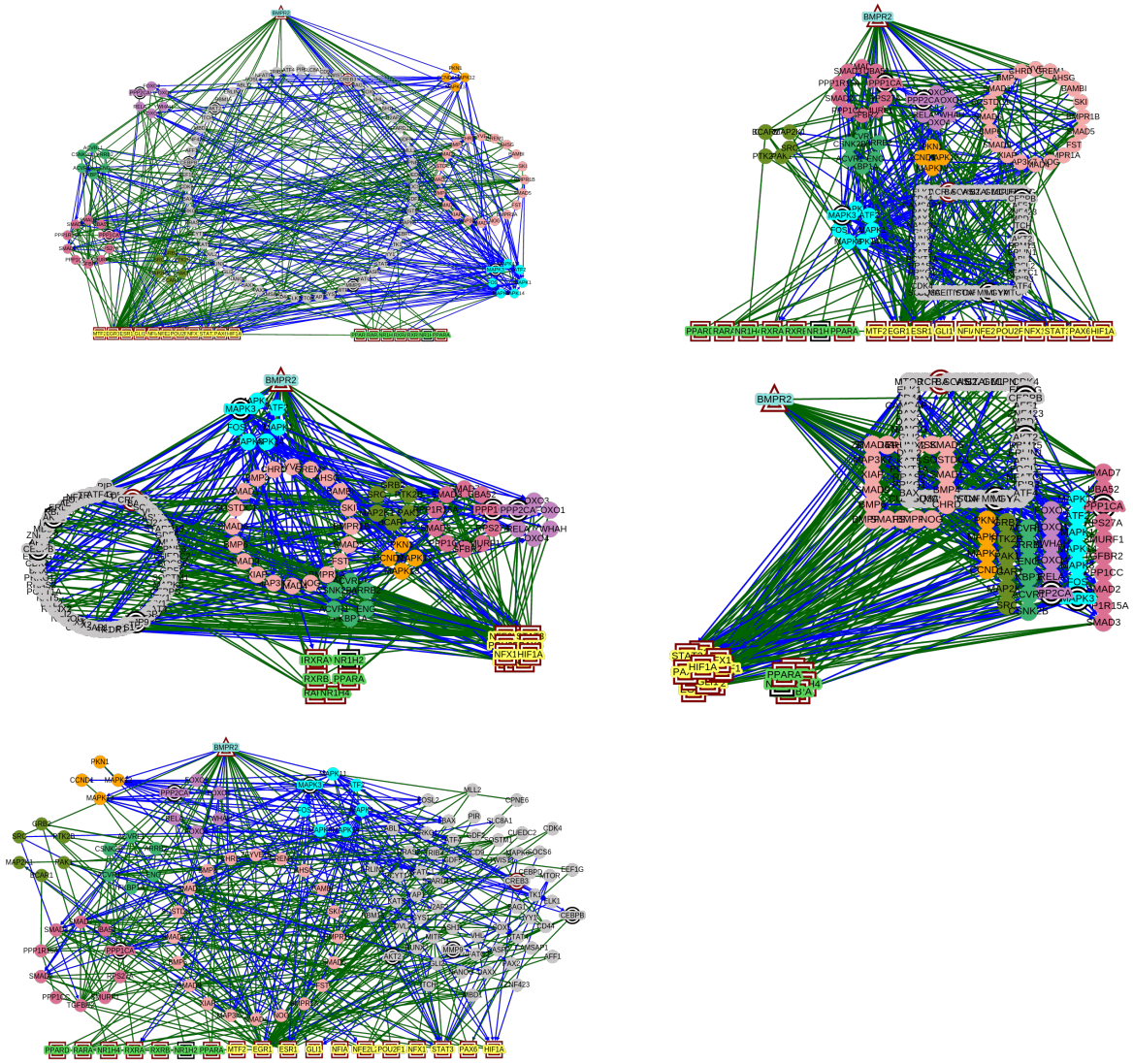
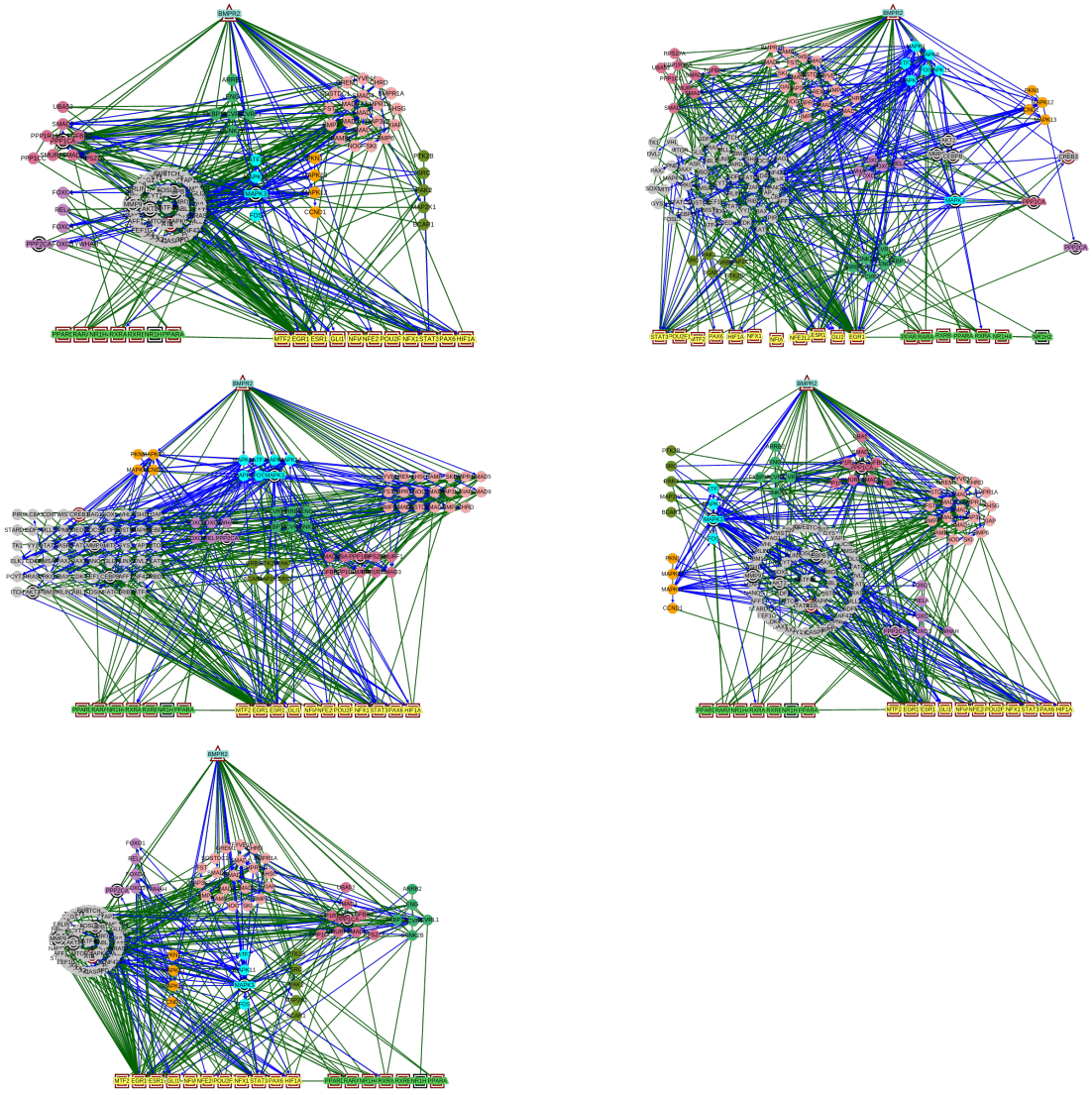


Figure 5.3: Layouts that were produced by crowdworkers for the same graph shown in figure 5.2.



To evaluate the quality of the produced layouts, we asked an expert to evaluate all of the produced layouts for the six signaling pathway graphs. The expert had no knowledge of who created the layouts. In addition to providing the expert with layouts produced both by collaborators and crowdworkers, we also asked the expert to evaluate layouts produced by GraphViz using their Dot algorithm [12]. In total, the expert was asked to look at 66 layouts, eleven layouts for each of the six signaling pathway graphs. To obtain how well a layout followed the guidelines, the expert was asked to do Graph Approval Tasks for all of the layouts. We compared how the expert worker rated these three groups of layouts: collaboratoror-produced, crowd-produced, layouts produced by automatic layout algorithm.

For all four guidelines, we compared how the expert rating varied for these types of layouts. Figures 5.4-5.9 show how the expert rated the layouts across all four guidelines.

Figure 5.4: Expert ratings between collaborator-produced layouts and automatic layouts for guidelines 1 and 2.

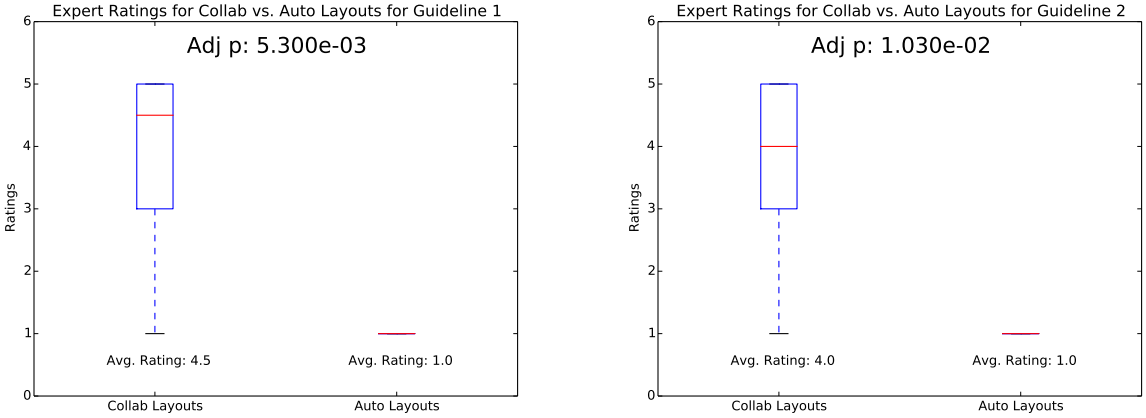


Figure 5.5: Expert ratings between collaborator-produced layouts and automatic layouts for guidelines 3 and 4.

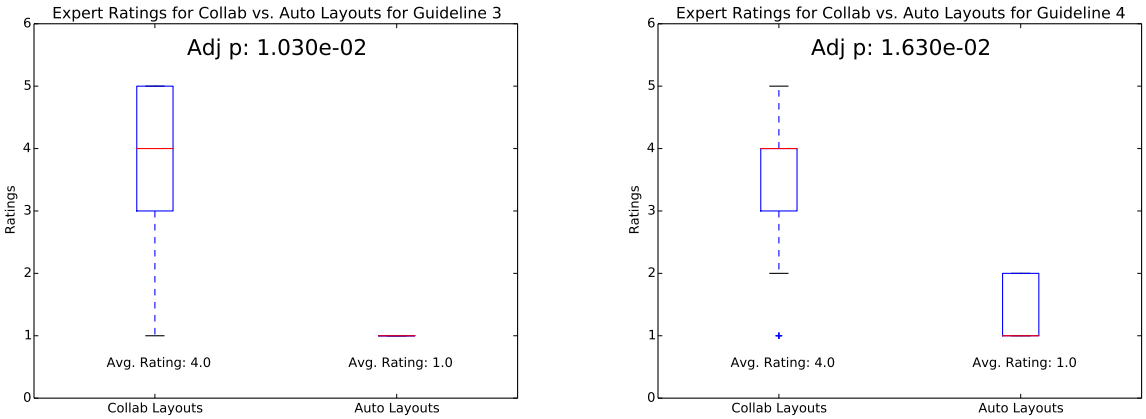


Figure 5.6: Expert ratings between crowd-produced layouts and automatic layous for guidelines 1 and 2.

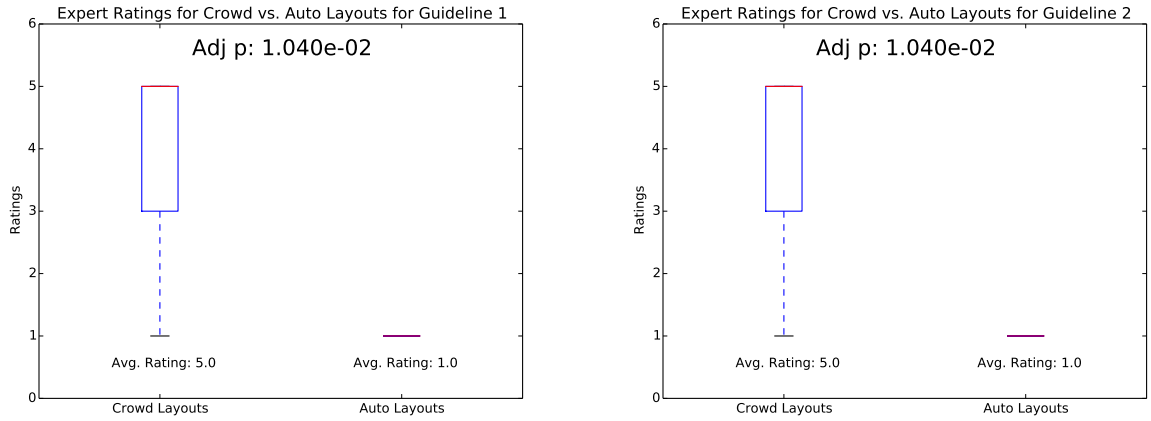


Figure 5.7: Expert ratings between crowd-produced layouts and automatic layous for guidelines 3 and 4.

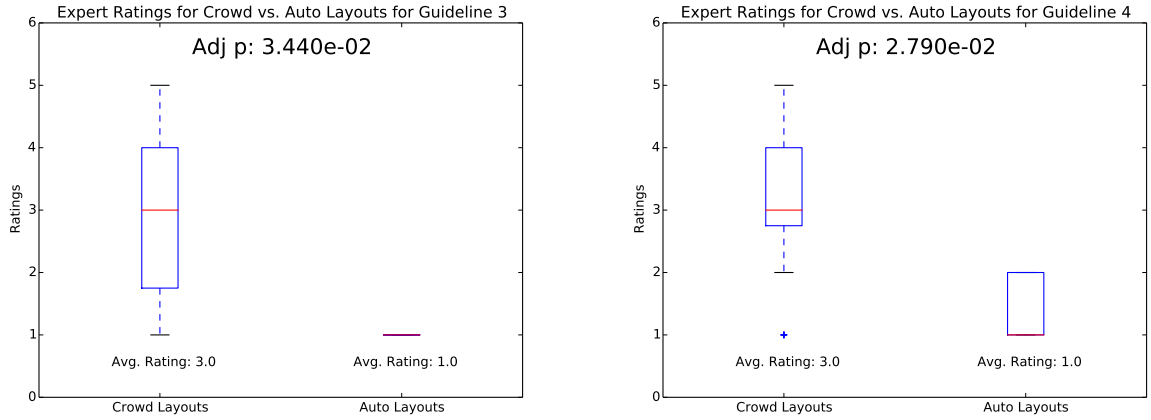


Figure 5.8: Expert ratings between collaborator-produced layouts and crowd layouts for guidelines 1 and 2.

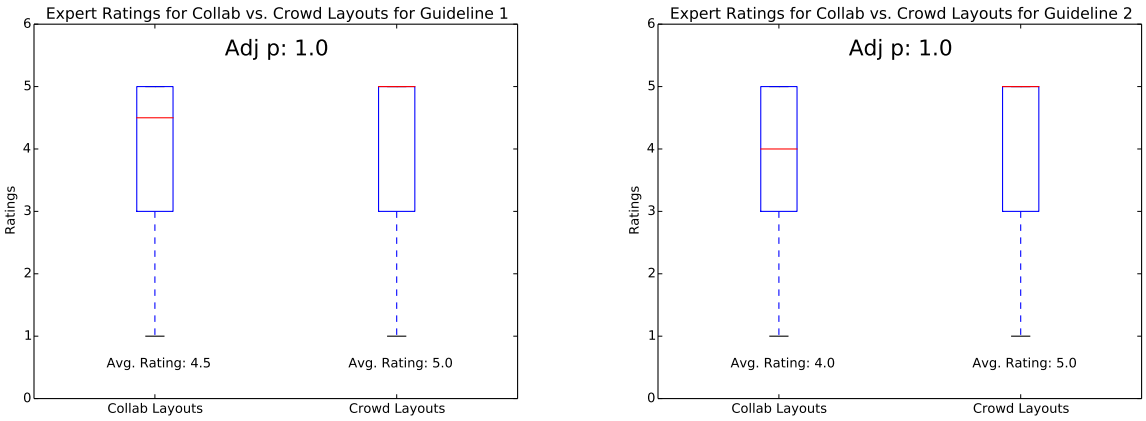
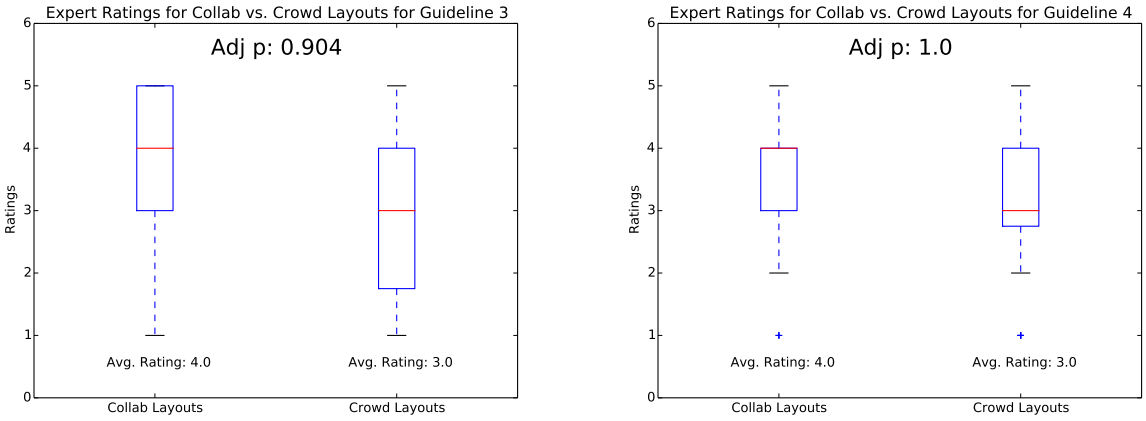


Figure 5.9: Expert ratings between collaborator-produced layouts and crowd layouts for guidelines 3 and 4.



For these plots, the red line in each box represents the median (50th percentile) value of the distribution of ratings. The bottom edge of the box signifies the 25th percentile whereas the top edge of the box represents the 75th percentile of the ratings. The highest and lowest values are represented by the horizontal lines connected at the end of the dotted lines. These horizontal lines represent the extremes of our ratings for the layouts. Any crosses that lie outside of these horizontal lines can be seen as outliers. The average rating across all the layouts for the specific guideline is also provided to show the difference in ratings between the types of layouts being compared. For comparing whether there is a significant difference between how the expert rated the types of layouts being compared, we also conducted Mann-Whitney U tests [44]. For these tests, a p-value < 0.05 means that there

is a significant difference between the ratings of the layouts being compared. Conversely, a p-value of $>= 0.05$ means that the ratings between the two layouts being compared are indistinguishable. The only layouts whose ratings were indistinguishable across any and all guidelines were when comparing how the expert rated collaborator-produced layouts and the crowd-produced layouts.

Discussion

Figures 5.4 and 5.5 show the difference between expert ratings for layouts produced by collaborators and layouts produced by an automatic layout algorithm. By comparing the distributions, it is apparent that the expert rated them very differently. In addition, the average rating across all guidelines between the two types of layouts show that the expert rated the collaborator produced layouts much higher than the layouts produced by the automatic layout algorithms. The p-values shown by comparing the distributions show further solidify our finding that experts rated collaborator produced layouts much differently than layouts produced by the automatic layout algorithm. These results were not surprising: we expect that our collaborators would produce layouts that follow guidelines since they have previous domain knowledge of the types of layouts we are looking for our signaling pathway graphs.

Figures 5.6 and 5.7 show the difference between the expert ratings for layouts produced by crowdworkers and layouts produced by an automatic layout algorithm. These figures were of particular interest because they clearly show that the experts rated crowdworker layouts considerably higher than layouts produced by automatic layout algorithms. The average rating across all guidelines between the two types of layouts show that crowdworker layouts resulted in much higher average rating for each guideline. This was a result that we were hoping to see as this shows that crowdworker layouts are of higher quality than layouts produced by automatic layout algorithms. The p-value also further shows that there is a stark difference between the ratings of the two types of layouts.

Figures 5.8 and 5.9 show the difference between the expert ratings for layouts produced by collaborators and layouts produced by crowdworkers. These distributions seem to be fairly similar. Moreover, the average ratings between the two types of layouts are also very similar to each other across all of the guidelines. This shows that the expert rating the two types of layouts very similarly. In addition, our p-value shows that the difference between how the expert rated the two types of layouts are indistinguishable. This confirms our hypothesis that crowdworkers can produce biologically meaningful layouts.

Another interesting feature of our results is that even our collaborators produced layouts which our expert did not believe to be perfect. If they did, then all of layouts produced by our collaborators would have a 5 as an average rating for all of the guidelines. This suggests that our guidelines may need to be improvised to better communicate the characteristics that we are looking for.

5.2 Can the crowd rate layouts as good as experts?

As a follow-up study to determine if crowds can create biologically meaningful layouts, we explored whether it was neceassar to have an expert determine if layouts met all of the guidelines. We explored possibilities of incorporating Machine Learning to extract characteristics of a good layout to predict if subsequent layouts were of comparable quality [22]. However, our aim was to make GraphCrowd a flexible system. For the various types of graphs that may be used in GraphCrowd, the spectrum of features may be too diverse for Machine Learning techniques to accurately evaluate. As an alternative, we considered using crowdworkers to rate the quality of layouts through Graph Approval Tasks. Our findings reveal that in addition to laying out graphs, the crowd also shows potential for rating these layouts similarly to how an expert would.

5.2.1 Method

As mentioned in our previous experiment, we obtained three types of layouts for each of the six signaling pathway graphs. Five layouts for each graph were produced by both collaborators and crowdworkers. For each graph we also included an automatically generated layout [12]. For each of the 66 layouts, we launched five Graph Approval Tasks. We decided to launch five Graph Approval Tasks(each performed by different crowdworkers) for each layout instead of one in order to mitigate lazy crowdworkers who don't actually perform the task given to them with integrity.

5.2.2 Results

Similar to our first experiment, we compared how crowdworker rated the three groups of layouts: collaboratoror-produced, crowd-produced, layouts produced by automatic layout algorithm. Figures 5.10-5.15 show how crowdworkers rated the layouts across all four guidelines.

Figure 5.10: Crowd ratings between collaborator-produced layouts and automatic layouts for guidelines 1 and 2.

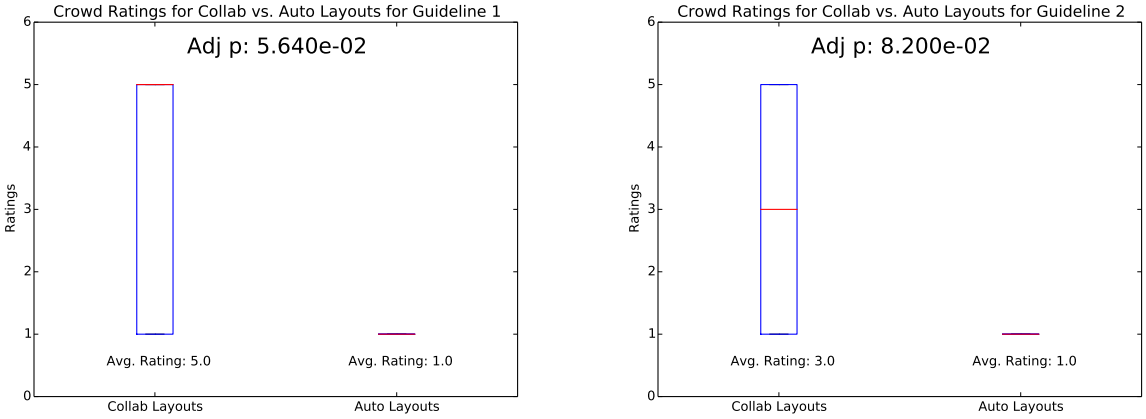


Figure 5.11: Crowd ratings between collaborator-produced layouts and automatic layouts for guidelines 3 and 4.

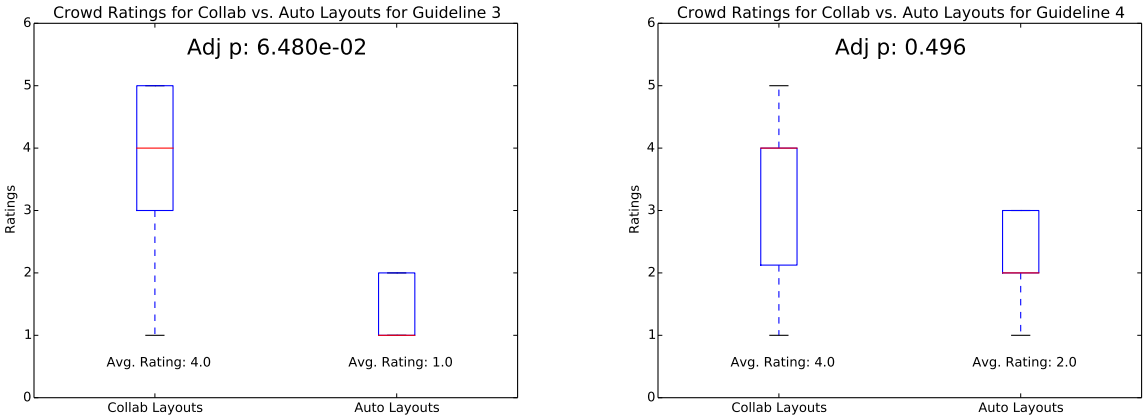


Figure 5.12: Crowd ratings between crowd-produced layouts and automatic layous for guidelines 1 and 2.

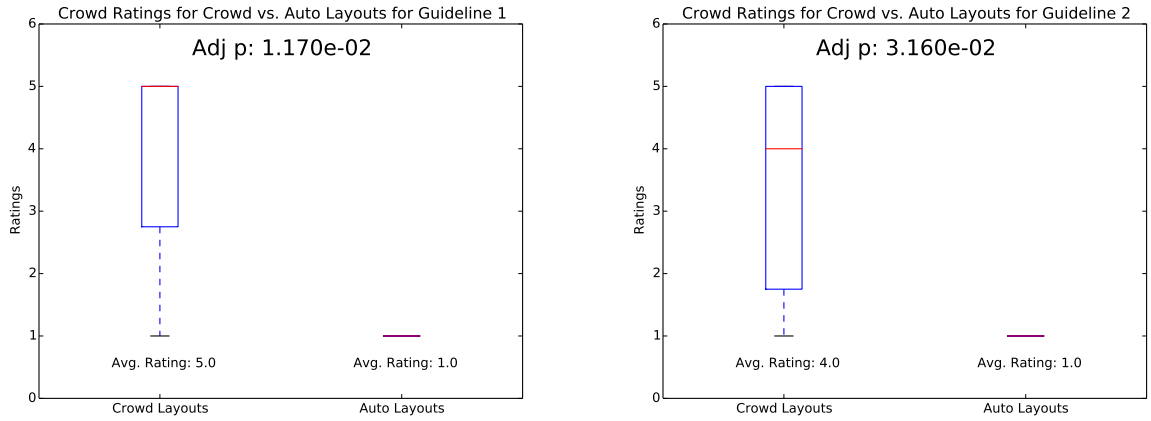


Figure 5.13: Crowd ratings between crowd-produced layouts and automatic layous for guidelines 3 and 4.

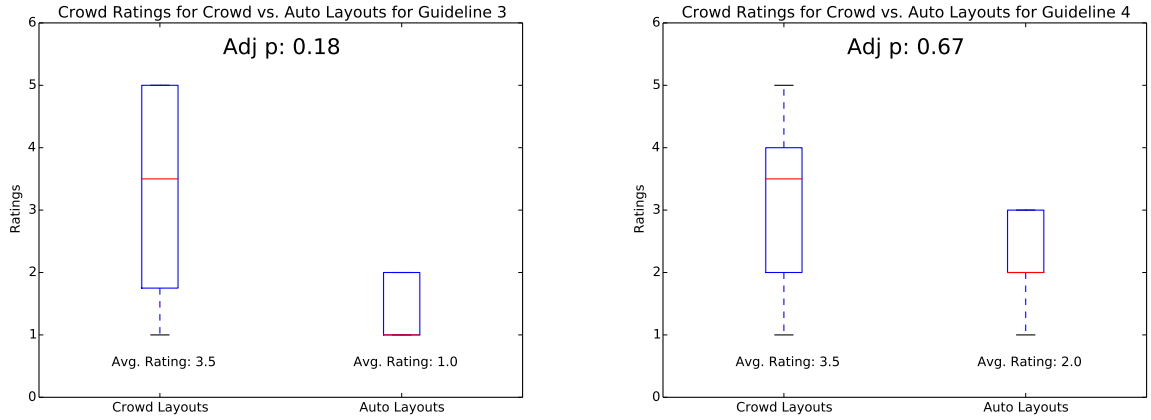


Figure 5.14: Crowd ratings between collaborator-produced layouts and crowd layouts for guidelines 1 and 2.

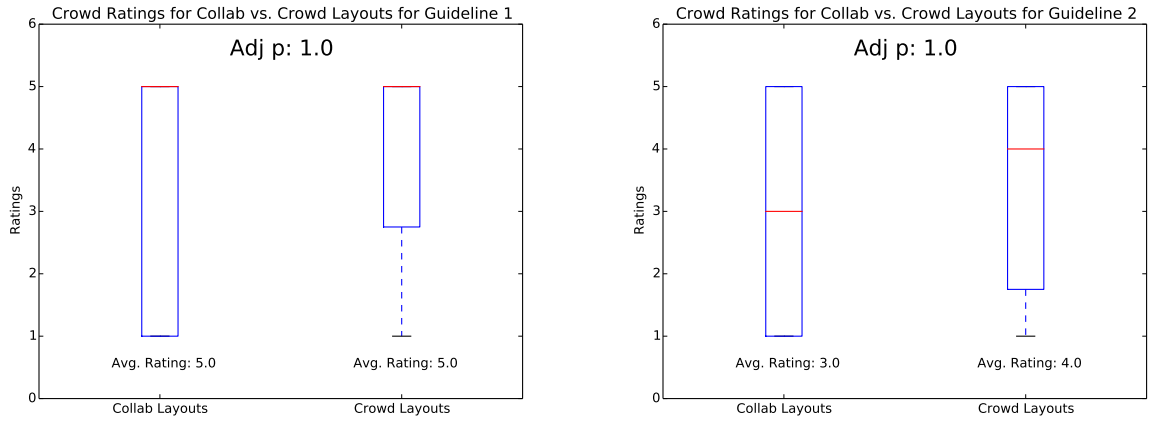
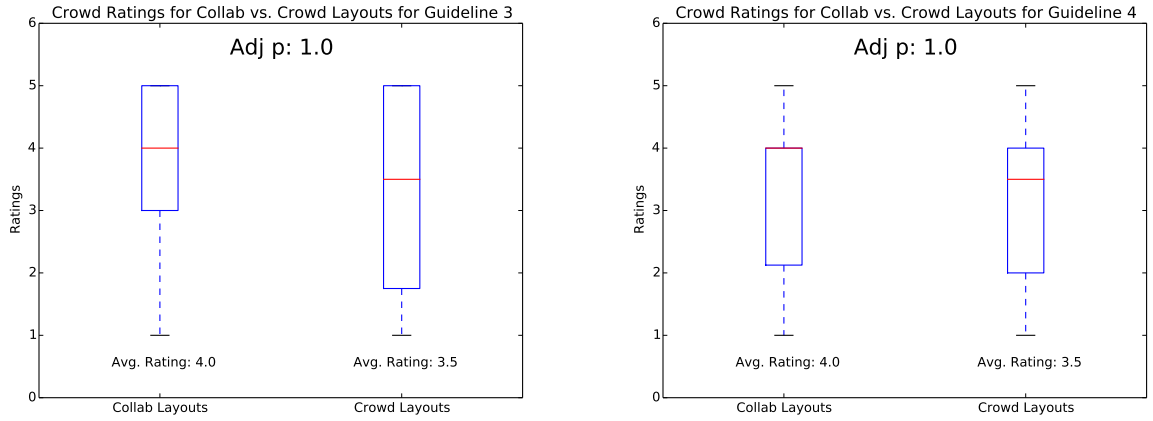


Figure 5.15: Crowd ratings between collaborator-produced layouts and crowd layouts for guidelines 3 and 4.



Since there were five crowdworkers who rated the same layouts, it is worth analyzing the extent to which the crowdworkers agreed on the rating of guidelines across all layouts. Table 5.1 shows the results of conducted a weighted Fleiss' Kappa [45]. The reason we conducted a weighted Fleiss' Kappa is so that we can measure how similar or different ratings are. In this weighted kappa, the farther apart the ratings are for a particular layout, the lesser the amount of weight is given to that rating. This effectively allows us to also keep track similar results vs. radically different results.

Table 5.1: Weighted Fleiss' Kappa Values for each of the guidelines across all layouts

Guidelines	Fleiss' Kappa Value
1	0.397
2	0.311
3	0.133
4	0.251

Next, we compare how crowdworkers rated all layouts across all guidelines. This comparison is done to see if crowdworkers produce similar ratings across all instances. Figures 5.16 and 5.17 show the comparison of crowdworker vs. expert ratings for all of the guidelines.

Figure 5.16: Average Crowdworker vs. Expert ratings for Guidelines 1 and 2

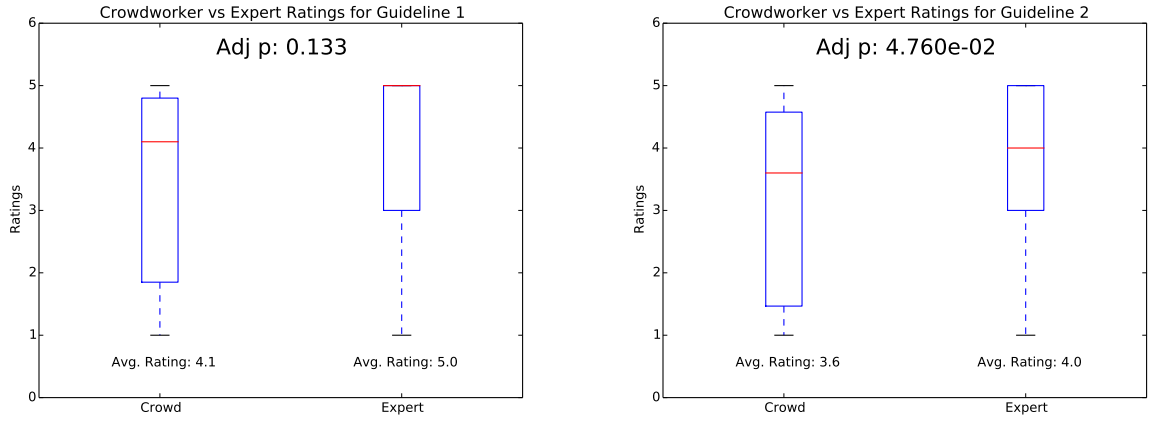
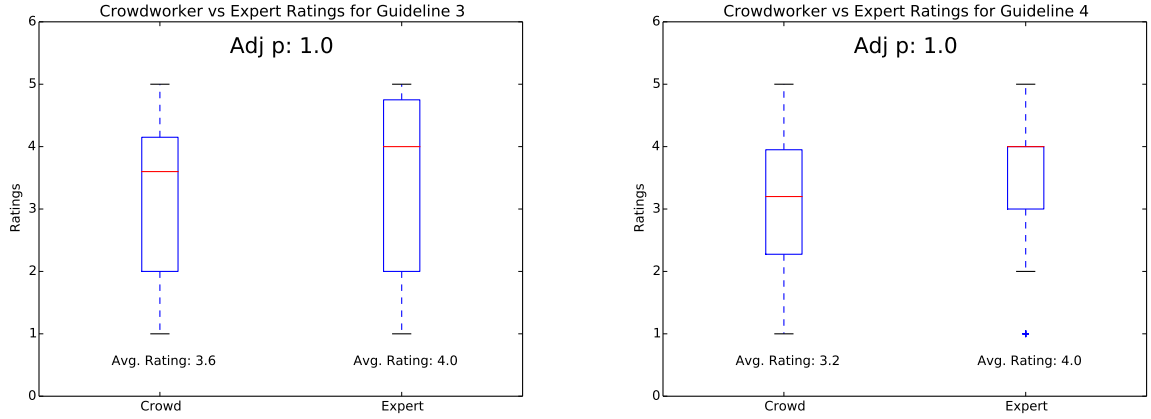


Figure 5.17: Average Crowdworker vs. Expert ratings for Guidelines 3 and 4



Discussion

This experiment was conducted as a follow-up to our original experiment. The purpose of this experiment was to determine if crowds can also rate layouts similarly to how an expert would rate them. First, we measured how crowdworkers rated the different types of layouts. Similar to ratings from our expert in our first experiment, the crowd also evaluated collab and crowd layouts very similarly by producing a very large p-value. These similarities show that experts and crowds seemed to reach the same consensus suggesting that crowds can rate layouts as good as experts.

As stated before, there were multiple crowdworkers used to rate the same layouts. In oder to measure the inter-rater reliability of the crowdworkers for each of the guidelines, we applied Fleiss' kappa [45] for each of the guidelines. Table 5.1 shows the results of running this test for each of the guidelines. Values from this statistic that are between 0.01 - 0.20 are said to be in slight agreement. Values that are between 0.21 - 0.40 are said to be in fair agreement [46]. For all guidelines except guideline 3, the crowdworkers seem to have fair agreement among themselves. Guideline 3, however, has slight agreement. A possible reason for this may stem from the ambiguity of the guideline, causing crowdworkers to choose different values for this rating.

Finally, in order to see how similar crowd ratings were compared to expert ratings we compared expert and crowd ratings across all layouts for each of the guidelines. Across all the guidelines, the average rating assigned for all of the layouts between crowdworkers and expert workers differed by less than 1.0 point. However, for guidelines 1 and 2, the difference between crowdworker and expert worker ratings seemed to be significantly different. An interesting pattern emerged when analyzing the differences between crowdworker and expert worker ratings: crowdworkers were often more critical of their ratings than our expert was. Possible reasons for this may be that crowdworkers were unsure of certain guidelines, thus were more critical of them. Another possible reason may be that there could possibly be crowdworkers who submitted invalid ratings of the layouts. This may result in the difference shown between the two types of worker ratings. As mentioned before, a probable source of the difference between the two type of ratings may be ambitiousness introduced by our guidelines. However, although some of the ratings were different between crowdworkers and expert workers: a few observations can be made. First, both expert and crowdworkers both rated automatically generated layouts as the lowest between the three groups of layouts. Next, both crowdworkers and expert workers seemed to produce similar ratings for layouts produced by both collaborators and crowdworkers. Finally, crowdworkers tend to be more critical of rating layouts produced than our expert. Although the comparison of the distribution of ratings between crowdworkers and expert workers were different, we believe that our results show that crowdworkers can, in fact, rate layouts similar to how an expert would rate them.

5.3 Chapter Summary

We conducted two experiments to evaluate GraphCrowd and answer our research questions. Our first research question: "Can we harness the power of crowdsourcing to create simplified, biologically meaningful, mechanistic visualization of signaling pathway graphs?" was answered by our first experiment. We asked both collaborators and crowdworkers to lay out signaling pathway graphs provided guidelines. We also used a popular algorithm to lay out these graphs. We collected all of these layouts and asked an expert to rate them. After comparing expert ratings, we discovered that there is no distinguishable difference between how experts viewed layouts produced by crowdworkers and collaborators. In addition, we also saw that layouts produced by crowdworkers were rated higher than layouts produced by the automatic layout algorithm. This showed that we can, in fact, harness crowdsourcing to create biologically meaningful layouts of graphs that are better than automatic layout algorithms. It was not uncommon that our expert rated layouts produced by crowdworkers as highly as they rated layouts produced by collaborators. This showed that crowds may produce layouts as good as layouts produced by collaborators.

As a follow-up, we conducted our second experiment which tried to answer: "Can the crowd rate layouts similarly to how an expert with domain knowledge would rate them?". We asked crowdworkers to rate the layouts obtained in our first experiment. Our results showed that crowdworkers also found that layouts produced by crowdworkers were better than layouts produced by automatic layout algorithms. Similarly to how an expert rated the layouts, there was no distinguishable difference in the way crowdworkers rated layouts produced by other crowdworkers and layouts produced by collaborators. These similarities between crowdworkers and our expert shows promise that crowds do rate layouts similarly to how experts would rate them. The significance of this is that crowdworkers can be used as a source for both creation and evaluation of layouts, further alleviating the burden researchers incur when laying out their graphs.

Chapter 6

Conclusions and Future Work

Researchers often use graphs to describe information that would otherwise be described in long texts. There exist many applications to aid researchers in generating and viewing these graphs. There are also applications that allow for the manipulation of these graphs by enabling the researcher to manually modify the structure of the graph or by providing automatic layout algorithms. These applications tend to be specific to the computational algorithm used to lay out the graphs. They may spend additional time manually laying out a graph or use different systems which support a desired computational layout algorithm. It is very common that researchers collaborate on projects. After they have taken the time to generate and re-arrange their graphs, they must incur additional overhead to share them among collaborators.

One contribution of my thesis is the creation of a collaborative, open-source platform: GraphSpace. This contribution is significant because it presents researchers with an option to not only create, visualize, and manipulate graphs, but also allows them to collaborate on them as well. We provide a tool for re-arranging the structure of any graph uploaded to GraphSpace by allowing users to save their custom arrangement or layouts of these graphs. They may either manually lay out a graph or use the various algorithms supported by GraphSpace to re-arrange the graph. GraphSpace users have may join groups to share their graphs among collaborators as well as see graphs uploaded by other members of a group. GraphSpace also provides organization of all graphs uploaded by a user through the use of tags. A researcher who has multiple graphs for a particular paper or conference can upload their graphs into GraphSpace and save their layouts for these graphs. Next, they can make their graphs public so that anyone with a valid link may access the graphs. By attaching tag(s) to the graphs they uploaded, all they have to do is provide a link to GraphSpace which contains their specified tag without incurring any additional hosting overhead. To extend the user of GraphSpace, it also supports a REST API which a researcher can communicate with rather than having to physically interact with the application to take advantage of its features.

A picture is only worth a thousand words if its meaningful. Currently there exists no automatic layout algorithm to create biologically meaningful, mechanistic visualizations of graphs. Rather than trying to implement our own automatic algorithm to solve this issue, we explored more flexible alternatives which would allow for multiple types of graphs to be layed out as specified by the uploaders of graphs. Another contribution of my thesis was the creation of a crowdsourcing enhancement to GraphSpace: GraphCrowd. This contribution aimed to alleviate the task of a researcher manually laying out a graph. Although existing automatic layout algorithms exist, they often are limited in their functionality. There is no one size fits all. In order to alleviate the researcher from manually laying out a graph, we explored the use of crowdsourcing to lay out complex graphs. GraphCrowd exemplifies how the crowd can be utilized to do complex, creative tasks given a simple set of guidelines. In addition, it also highlights how the crowd can evaluate its own work, filtering out low quality work. For researchers who tend to upload many graphs at once, this may seem like an ideal alternative to invest valuable time on layouts.

We published the tasks of laying out graphs as well as evaluating them on Amazon Mechanical Turk, where crowdworkers would layout these graphs as well as rate them to determine how well they follow the provided guidelines. Next, we conducted an internal study of 10 participants and asked them to layout the same graphs using the same guidelines provided to the crowdworkers. From these layouts, we asked crowdworkers to evaluate them based on the same guidelines by launching Graph Approval Tasks for these layouts. Finally, we asked an expert worker to rate the layouts produced by both crowdworkers and our participants from our internal study. After comparing the results collected from the approval tasks of crowdworkers and the ratings obtained by the approval tasks by our expert worker(s), we discovered that it is possible to harness the crowd to produce biologically meaningful layouts.

In conclusion, my thesis explores if crowds have the ability to generate simplified, meaningful visualizations of otherwise complex graphs. I implemented not only a collaborative, opens-source platform which researchers can create, visualize, and manipulate graphs, but also implemented a novel approach to laying out graphs: by recruiting crowdworkers. I conducted experiments in which I determined that crowdworkers are able to lay out graphs better than current automatic algorithms and sometimes as good as layous produced by researchers themselves. In addition, I also conducted experiments to determine if crowdworkers can also evaluate the quality of the crowdworker produced layouts. The results of this experiment reveal that in addition to laying out graphs, crowdworkers also rate layouts similarly to how experts rate them.

6.1 Future Work

To further aid in creating biologically meaningful layouts, we plan on building features into GraphCrowd to enable users to discover and highlight mechanistic paths in graphs that lead from receptors to downstream transcription factors. Users will be able to highlight paths with

mechanistic information drawn from databases. Next, we will implement an algorithm for counting the number of downward-pointing paths in a layout. When a user lays out a graph, for each node that contains a directed edge, we will implement a dynamic programming algorithm that will run in linear time to determine this value. While users are working on a layout, this path counting algorithm will be set as a measure of progress for completing a layout. This will become the scoring function on which workers will use to determine how biologically acceptable their layout is. Our method is similar to the scoring functionality used by Foldit [31]. Whenever a worker changes the layout, this scoring function is re-computed, allowing for a worker to analyze and modify their layouts accordingly.

GraphCrowd currently only recruits paid crowdworkers from Amazon Mechanical Turk. Our future plans are to incorporate citizen scientists for GraphCrowd tasks. We will launch GraphCrowd on Zooniverse [47]: the world's largest online science portal. This portal has over one million registered members. We will also launch our tasks on university students who are studying Systems Biology. We will then investigate the key differences, such as motivation and performance between these three types of users: paid crowdworkers from Amazon Mechanical Turk, Undergraduate students enrolled in Systems Biology classes, and volunteer citizen scientists from Zooniverse. Questions which we plan to tackle include: "Which crowd types perform best and worst in each role?" "Which have the best and worst experiences while participating?" "What strategies to high-performing users employ that set them apart from low performers?" After all evaluations have concluded, we will finally incorporate GraphCrowd into the standard release of GraphSpace. We will launch this merged application online as well as make it available as standalone open-source module.

Bibliography

[1] H. Kitano, "Computational systems biology," *Nature*, vol. 420, no. 6912, pp. 206–210, 2002.

[2] U. S. Bhalla and R. Iyengar, "Emergent properties of networks of biological signaling pathways," *Science*, vol. 283, no. 5400, pp. 381–387, 1999.

[3] "Nci dictionary of cancer terms," apr 2016.

[4] D. J. Dix, K. A. Houck, M. T. Martin, A. M. Richard, R. W. Setzer, and R. J. Kavlock, "The toxcast program for prioritizing toxicity testing of environmental chemicals," *Toxicological Sciences*, vol. 95, no. 1, pp. 5–12, 2007.

[5] D. C. Brabham, *Crowdsourcing*. Mit Press, 2013.

[6] A. M. Turk, "Amazon mechanical turk," *Retrieved August*, vol. 17, p. 2012, 2012.

[7] J. Brooke *et al.*, "Sus-a quick and dirty usability scale," *Usability evaluation in industry*, vol. 189, no. 194, pp. 4–7, 1996.

[8] C. T. Lopes, M. Franz, F. Kazi, S. L. Donaldson, Q. Morris, and G. D. Bader, "Cytoscape web: an interactive web-based network browser," *Bioinformatics*, vol. 26, no. 18, pp. 2347–2348, 2010.

[9] D. A. Schult and P. Swart, "Exploring network structure, dynamics, and function using networkx," in *Proceedings of the 7th Python in Science Conferences (SciPy 2008)*, vol. 2008, pp. 11–16, 2008.

[10] G. Csardi and T. Nepusz, "The igraph software package for complex network research," *InterJournal, Complex Systems*, vol. 1695, no. 5, pp. 1–9, 2006.

[11] P. Shannon, A. Markiel, O. Ozier, N. S. Baliga, J. T. Wang, D. Ramage, N. Amin, B. Schwikowski, and T. Ideker, "Cytoscape: a software environment for integrated models of biomolecular interaction networks," *Genome research*, vol. 13, no. 11, pp. 2498–2504, 2003.

[12] J. Ellson, E. Gansner, L. Koutsofios, S. C. North, and G. Woodhull, "Graphvizopen source graph drawing tools," in *Graph Drawing*, pp. 483–484, Springer, 2001.

[13] M. Bastian, S. Heymann, M. Jacomy, *et al.*, "Gephi: an open source software for exploring and manipulating networks.," *ICWSM*, vol. 8, pp. 361–362, 2009.

[14] S. Wolfram, *The mathematica book*. Wolfram Media, Incorporated, 1996.

[15] B. Arıkan, "Network intelligence for all," *Leonardo*, vol. 46, no. 3, pp. 268–269, 2013.

[16] K. Luther, A. Pavel, W. Wu, J.-l. Tolentino, M. Agrawala, B. Hartmann, and S. P. Dow, "Crowdcrit: Crowdsourcing and aggregating visual design critique," in *Proceedings of the Companion Publication of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing*, CSCW Companion '14, (New York, NY, USA), pp. 21–24, ACM, 2014.

[17] A. Xu, H. Rao, S. P. Dow, and B. P. Bailey, "A classroom study of using crowd feedback in the iterative design process," in *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*, CSCW '15, (New York, NY, USA), pp. 1637–1648, ACM, 2015.

[18] J. Heer and M. Bostock, "Crowdsourcing graphical perception: Using mechanical turk to assess visualization design," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, (New York, NY, USA), pp. 203–212, ACM, 2010.

[19] T. Mitra, C. J. Hutto, and E. Gilbert, "Comparing person-and process-centric strategies for obtaining quality data on amazon mechanical turk," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pp. 1345–1354, ACM, 2015.

[20] J. Rzeszotarski and A. Kittur, "Crowdscape: interactively visualizing user behavior and output," in *Proceedings of the 25th annual ACM symposium on User interface software and technology*, pp. 55–62, ACM, 2012.

[21] M. Walker and S. Whittaker, "Mixed initiative in dialogue: An investigation into discourse segmentation," in *Proceedings of the 28th annual meeting on Association for Computational Linguistics*, pp. 70–78, Association for Computational Linguistics, 1990.

[22] J. M. Rzeszotarski and A. Kittur, "Instrumenting the crowd: using implicit behavioral measures to predict task performance," in *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pp. 13–22, ACM, 2011.

[23] A. Yuan, K. Luther, M. Krause, U. ICSI, S. Vennix, S. P. Dow, and B. Hartmann, "Almost an expert: The effects of rubrics and expertise on perceived value of crowdsourced design critiques," *CSCW'16 Computer Supported Cooperative Work and Social Computin*, 2016.

[24] "Upwork," apr 2016.

[25] A. P. Jagadeesan, A. Lynn, J. R. Corney, X. Yan, J. Wenzel, A. Sherlock, and W. Regli, "Geometric reasoning via internet crowdsourcing," in *2009 SIAM/ACM Joint Conference on Geometric and Physical Modeling*, pp. 313–318, ACM, 2009.

[26] D. Tinapple, L. Olson, and J. Sadauskas, "Critviz: Web-based software supporting peer critique in large creative classrooms," *Bulletin of the IEEE Technical Committee on Learning Technology*, vol. 15, no. 1, p. 29, 2013.

[27] E. Peer, S. Samat, L. Brandimarte, and A. Acquisti, "Beyond the turk: An empirical comparison of alternative platforms for online behavioral research," *Available at SSRN 2594183*, 2015.

[28] J. De Winter, M. Kyriakidis, D. Dodou, and R. Happee, "Using crowdflower to study the relationship between self-reported violations and traffic accidents," *Procedia Manufacturing*, vol. 3, pp. 2518–2525, 2015.

[29] J. P. Cohn, "Citizen science: Can volunteers do real research?," *BioScience*, vol. 58, no. 3, pp. 192–197, 2008.

[30] K. Luther, S. Counts, K. B. Stecher, A. Hoff, and P. Johns, "Pathfinder: an online collaboration environment for citizen scientists," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 239–248, ACM, 2009.

[31] F. Khatib, S. Cooper, M. D. Tyka, K. Xu, I. Makedon, Z. Popović, D. Baker, and F. Players, "Algorithm discovery by protein folding game players," *Proceedings of the National Academy of Sciences*, vol. 108, no. 47, pp. 18949–18953, 2011.

[32] S. Deterding, M. Sicart, L. Nacke, K. O'Hara, and D. Dixon, "Gamification. using game-design elements in non-gaming contexts," in *CHI'11 Extended Abstracts on Human Factors in Computing Systems*, pp. 2425–2428, ACM, 2011.

[33] P. M. Aoki, R. Honicky, A. Mainwaring, C. Myers, E. Paulos, S. Subramanian, and A. Woodruff, "Common sense: Mobile environmental sensing platforms to support community action and citizen science," 2008.

[34] S. Kim, J. Mankoff, and E. Paulos, "Sensr: evaluating a flexible framework for authoring mobile data-collection tools for citizen science," in *Proceedings of the 2013 conference on Computer supported cooperative work*, pp. 1453–1462, ACM, 2013.

[35] M. Kanehisa and S. Goto, "Kegg: kyoto encyclopedia of genes and genomes," *Nucleic acids research*, vol. 28, no. 1, pp. 27–30, 2000.

[36] M. Franz, C. T. Lopes, G. Huck, Y. Dong, O. Sumer, and G. D. Bader, "Cytoscape. js: a graph theory library for visualisation and analysis," *Bioinformatics*, vol. 32, no. 2, pp. 309–311, 2016.

[37] U. Dogrusoz, E. Giral, A. Cetintas, A. Civril, and E. Demir, "A layout algorithm for undirected compound graphs," *Information Sciences*, vol. 179, no. 7, pp. 980–994, 2009.

[38] M. Jakl, "Representational state transfer," 2005.

[39] G. VanRossum and F. L. Drake, *The Python Language Reference*. Python software foundation Amsterdam, Netherlands, 2010.

[40] A. Holovaty and J. Kaplan-Moss, *The definitive guide to Django: Web development done right*. Apress, 2009.

[41] D. Singh, "Graphspace," 2016.

[42] E. Amazon, "Amazon elastic compute cloud (amazon ec2)," *Amazon Elastic Compute Cloud (Amazon EC2)*, 2010.

[43] L. Von Ahn, "Human computation," in *Design Automation Conference, 2009. DAC'09. 46th ACM/IEEE*, pp. 418–419, IEEE, 2009.

[44] P. E. McKnight and J. Najab, "Mann-whitney u test," *Corsini Encyclopedia of Psychology*, 2010.

[45] J. L. Fleiss, "Measuring nominal scale agreement among many raters.," *Psychological bulletin*, vol. 76, no. 5, p. 378, 1971.

[46] J. R. Landis and G. G. Koch, "The measurement of observer agreement for categorical data," *biometrics*, pp. 159–174, 1977.

[47] K. Borne and Z. Team, "The zooniverse: A framework for knowledge discovery from citizen science data," in *AGU Fall Meeting Abstracts*, vol. 1, p. 0650, 2011.