EdgeLap: Identifying and discovering features from overlapping sets in networks

Jessica Wong*

Abstract—Microbes are single celled organisms that can be found everywhere in the world from the air to the soil on the ground. Generally speaking, microbes will be found in extremely close proximity to other species of microbes due to mutual beneficial relationships where one species produces something that another species requires for survival; these groups of microbes are called a community. A widespread question in the microbiological community is trying to identify what microbial interactions are commonly found or not found throughout different communities. To address this question, EdgeLap was developed. EdgeLap is a tool that creates visualizations to help its end users, microbiologists, easily identify and discover common microbial interactions. Although EdgeLap is specifically designed with microbial communities in mind, it can be applied to many other biological networks or even social networks where examining the edges in a network is a question of interest.

1 Introduction

Microbes are single celled organisms that can be found on virtually every surface of the earth. Generally speaking, microbes are found in groups made up of different kinds of single celled organisms and this group of microbes is called a community. The concept of microbial species is difficult to assess experimentally so microbes are grouped into operational taxonomic units (OTUs) based on their DNA. In the rest of this paper, microbial species will be referred to as an OTU.

Mediating the flow of matter and energy throughout the soil environment is a plethora of microbial diversity often known only through their molecular signatures. Owing to both the complexity of the soil environment, and the abundance of uncultivated soil microorganisms, the ecological rules guiding microbial community dynamics are difficult to determine. Uncovering potential relationships between microbial community members requires identification of microbial interactions. Co-occurrence network analyses can be used to identify potential microbial relationships; however, comparing these networks is a difficult task.

EdgeLap is a tool designed to help microbiologists identify and discover common OTU interactions that are found across many different communities. It not only helps visualize OTU interactions that are common to all of the different communities—it helps visualize OTU interactions that commonly appear in a subset of the communities being examined. Whereas most set comparison tools and techniques focus on visualizing the common items between sets, EdgeLap focuses on the relationships between items in a set rather than the items themselves. EdgeLap also allows its users to differentiate between the types of edges that are displayed. Users can choose to examine just positively or negatively co-occurring OTUs or they can loosen the restrictions to examine OTUs that interact in different communities but may not necessarily have the same type of interaction within those communities. In this paper, EdgeLap works with the Long-term Soil profile (LTSP) dataset, one of the world's largest coordinated research networks that addresses basic and applied scientific questions related to forest management [12]. However, there is no reason that EdgeLap could not be applied to other data domains and problems where the relationship between items is to be investigated.

2 Related Work

Studying microbial co-occurrence networks is not a new problem and there have been many previous attempts to come up with tools to solve this problems [8, 7]. However, the focus in these efforts seem to be on based on computing the co-occurrence numbers and then finding an appropriate graph or scatterplot or Euler diagram to show the information rather than come up with a tool to help identify and discover co-occurrence networks. There have been both commercial and non-commercial tools produced for investigating sets and networks but these tools do not necessarily deal with the problem of identifying common relationships between items that occur across multiple sets. While the relationships between items could be considered as items themselves thus reducing EdgeLap's problem to be another set comparison problem, EdgeLap's ability to show common edges between network sets at various granularities is an extension of the set comparison problem.

2.1 Visualizing networks

Current tools used in the biological data domain focus on displaying a network [1, 2, 3] but does not have any set comparison functionality. These tools excel at displaying an individual network as they were not designed to show set intersections, they cannot solve the problem EdgeLap is trying to tackle. It is due to the existence of these tools that EdgeLap does not focus on displaying individual networks; rather, it is focused on more creating a visualization that will allow its users to make comparisons between different sets.

2.2 Set Intersection

Visualizing sets and set intersections is a well studied problem in the information visualization field [6]. Some tools in this area focus on comparing or displaying similar elements between sets [10, 13, 4] while others focus on showing different types of information in a set [9]. The tools closest to EdgeLap are Radial Sets [5] since Radial Sets also focus on displaying common items in overlapping sets.EdgeLap is actually adapted from Radial Sets but due to the need to show extra information about each network, Radial Sets does not completely satisfy EdgeLap's requirements. Therefore, EdgeLap extended and adapted the Radial Sets idiom.

The other set visualization techniques have more emphasis on showing a variety of information from the different sets in a more spread out format [9, 10] which did not suit EdgeLap as it really only had two large pieces of information to show. Arguably, EdgeLap could be adapted to use a visualization encoding closer to Domino [9] and UpSet [10] but the end users preferred the look of Radial Sets as opposed to Domino or UpSet. OnSet [13] does deal with visualization relationships in a network but it deals with binary data and the dataset used in this paper was seen as non-binary due to the different possible types of interaction between element types hence OnSets idioms and encodings were not adapted into EdgeLap.

*email: jhmwong@cs.ubc.ca

3 Task and Data Abstraction

3.1 Dataset

EdgeLap uses a dataset that was derived from over 700 DNA samples of microbial communities harvested from soil samples obtained six different geographical locations across Canada and the United States. Each site was divided into four forest plots where microbial DNA was extracted from soil using four different DNA harvesting methods: unmanaged, mild, moderate, and heavy. The extracted DNA was then sequenced and used to identify the OTUs present within each sample. In order to identify potential microbial interactions, co-occurrence networks were constructed for: all samples, each geographic location, each site within each geographic location, and each forest treatment within each geographic location resulting in a total of 52 networks. The network visualization and calculation of network properties has been completed outside of the scope of this project.

Each soil microbial community network is represented in the form of two files. The first file lists what OTUs are present in a sample and these OTUs would be considered as the data items or nodes in a network. The second file lists which OTU is correlated with another OTU (there would be the links between nodes in a network); these co-occurrences can be positive or negative. Essentially, a community can be thought of as a network with the OTUs as nodes and the OTU interactions as edges. The networks are undirected and asymmetric and there is no guarantee on the size of the networks in relation to each other. It is also possible for networks to have no edges in common with each other. EdgeLap works with the second type of file.

In this dataset, networks have thousands of nodes. On average, a network has around 60 500 edges but can range from approximately 30 edges to 291 000 edges. It is not expected that the networks share nodes; instead, it is expected that the networks will share edges. The total number of edges in all of the networks of this dataset is 605479.

3.2 Task Abstraction

The basic task EdgeLap is designed to accomplish is to identify and locate common links across any combination of networks in a dataset. EdgeLap can visualize not just edges that appear in all the networks the user wants to compare; it can also visualize sets of links that are common to a subset of the networks currently being examined. For example, if the user wants to compare four networks, EdgeLap will help visualize the common edges found between networks: 1 and 2; 1 and 3; 1 and 4; 2 and 3; 2 and 4; 3 and 4; 1, 2, and 3; 1, 2 and 4; 2, 3, and 4; and 1, 2, 3, and 4. Basically, for n networks being examined, EdgeLap can provide a way to visualize 2^n combination of networks. The goal is to generate a hypothesis about the types, and number of co-occurrences found between OTUs in different communities. In summary, the main tasks of EdgeLap are:

1. Discover features, and similarities/differences between networks
2. Explore the distribution of shared links between networks
3. Identify features in the grouping of links, and similarities between networks
4. Compare networks with each other

4 Solution and Design Justification

4.1 Network Glyph

To get a general sense of how many edges in a network are actually in common with the other networks it is currently being compared to in the visualization, a histogram is used inside the network mark. The top bar in the histogram represents the total number of edges present in the network; the yellow portion displays edges that are not shared with any other networks while the white portion displays the total number of edges in that network that are shared. The next bar after shows the number of edges that are shared with exactly one network. The bar after that shows the number of edges that are shared with exactly two networks. This pattern continues until the last histogram bar.

Table 1. Summary of encoding idioms used in EdgeLap

Data Attribute	Mark
Network	Glyph
Quantity/distribution of links in a network	• Length (histogram inside network glyph describes quantity/distribution of links using length) • Position on a common scale • Colour
Links in overlap	• Line mark • Connection
Link Attributes	• Saturation • Glyph • Colour

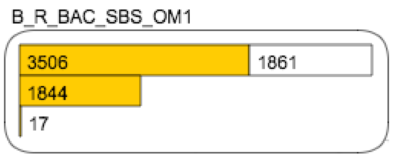


Fig. 1. The network glyph helps provide information to the user regarding how similar a community is to the other communities being currently compared. The yellow portion of the histogram bars denote how many edges that particular network shares with the other networks it is currently being compared with. The top of the histogram starts by showing how many edges that network shares with 0 other networks. The next bar down shows how many edges that network shares with exactly 1 other network. This pattern repeats for the rest of the histogram bars.

4.2 Circle Glyph

In order to help users differentiate between which sets are participating in multi-network relationships (any relationships involving more than two networks), lines connect a circle glyph to the networks it represents an overlapping set for. This encoding was chosen to help users pick out which networks were involved in a specific multi-network relationship as well as to help users gain a preliminary intuition of which networks might share many interactions without having to create another radial set to investigate further. The circle encoding can be one of six possibilities shown in figure 2

4.3 Varying saturation of the colours of lines connecting networks

For m networks, there are 2^m relationships each network can participate in. Since lines denote relationships between networks, the natural consequence is that the number of lines present inside the visualization is very large and hence, hard to decipher. Colour saturation was used to help users differentiate between lines that depicted edges that were shared between different numbers of networks. Since the users were more interested in cases where more networks share a common

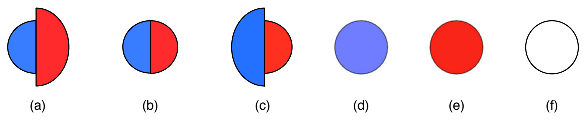


Fig. 2. The circle glyphs will represent the relationship between the cardinality of the set of positive co-occurrences and the cardinality of the set of negative co-occurrences. (a) There are less positive co-occurrences than negative co-occurrences, (b) there are an equal number of positive and negative co-occurrences, (c) there are more positive co-occurrences than negative co-occurrences, (d) there are only positive co-occurrences, (e) there are only negative co-occurrences, and (f) when the co-occurrences do not need to be of the same type and the more important question is if there even is any co-occurrences between OTUs.

edge, lines that denote these relationships use more saturated colours. Likewise, sets that depict edges shared between a smaller number of networks are represented by lines that use less saturated colours as shown in figure 3. Thickness was not considered as an encoding due to the sheer number of lines that could result from the radial set; thickness would quickly become indiscernible given enough lines. Colour saturation was used rather than using different hues in order to abide by the "get it right in black and white principle" [11]. The idea of using different colour hues that were of different saturations was discarded as I felt that it would put too much emphasis on the different sets of lines as opposed to the circle glyphs. As the circle glyphs are of more importance as compared to the lines that are connecting to it, we wanted the extra emphasis of colour placed on the circle glyphs. Also, using different colour hues for the lines along with the coloured circle glyphs seemed too overwhelming.

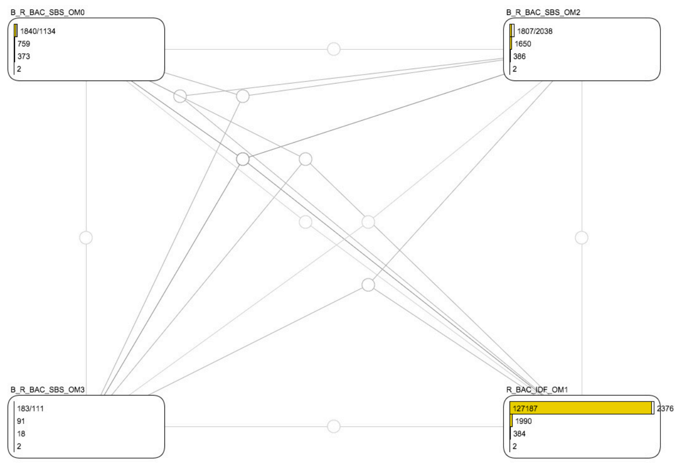


Fig. 3. The lines that connect from a network to the same circle glyph are considered as one set of lines. Depending on the number of lines in the set, lines are of different colour saturations.

4.4 Changes from the proposal

4.4.1 Not displaying the network level view

The network level view was dropped from EdgeLap due to the underestimation of the number of common OTU interactions that would be found in each overlapping set. Originally it was thought that there would be around 6-10 common OTU interactions found but it was found that there were some test cases where there were around 1500 common OTU interactions. Placing 1500 nodes in a network graph where each node is in a spot that would make the edges of the network not cross over it too many times is a non-trivial task.

The network level view feature was replaced with displaying the textual information of the items in the overlapping set in a new table. While having a long list of items without any ability to filter or sort the list is not very ideal, it was the best compromise as the end users of EdgeLap already had tools to visualize a single network and the network level view was not an essential requirement. The"Export to CSV" feature in section 6.1.3 can be used by the end users to export a list of items in that overlapping set which can then be input in the end users' own tool to visualize a network. As the end user can still obtain a visualization of the network, dropping this feature was not seen as decreasing the usefulness of EdgeLap.

4.4.2 Orientation of network glyphs

Initially, the network glyphs were designed to be rotated differently depending on where they were located. However, after some consideration about the difficulty of comparing the histograms inside the network glyph, it was decided that the network glyphs should be aligned in the same direction.

4.4.3 Circles glyphs are all the same size

The proposal originally planned to use different circle sizes to indicate the size of the overlapping sets—overlapping sets with higher cardinality would have larger circles. However, generally speaking, the more networks that were involved with an overlapping set, the smaller the set became. However, with the number of lines present in the middle of the visualization already partially obscuring circles, it was decided that having a variety of circle sizes would just add extra time for the user when it came to trying to search for what was needed. As the main function of the circle glyphs is to provide a quick tidbit of information to the user regarding the specific overlapping set, it was decided that the glyph itself served that purpose well enough without having to include a size encoding as well.

4.4.4 Dropping the total edges histogram bar from the network mark

Originally, EdgeLap's design had an extra bar in each of the histograms located in the network marks. The extra bar was meant to represent the total number of edges that were present in the visualization and was initially designed in hopes of being able to show how much or how little a network contributed to the overall edge sharing; it was also a quick way for the user to tell relative sizes of networks as compared to each other. However, this approach became impractical due to the histogram quickly becoming unreadable once three or more networks were compared. The histogram bar that represented the total number of edges was big enough to dwarf over the bars that represented the individual network and those bars became a single line essentially rendering the histogram useless. Multiplying each of the histogram bars by a constant helped increase the size of the bars but the ability to compare the size of a network to the size of the sum of all the networks disappeared as the difference in bar length no longer gave a good visual of the difference in network sizes. Therefore, the histogram bar representing the total number of edges was taken out as it was not serving its intended purpose.

4.4.5 All lines connecting to circle glyphs

The proposal initially designed the line marks to only have a circle glyph if the lines connected more than two networks. However, for uniformity issues and to avoid having the lines be unnoticed due to a high number of lines in the visualization, it was decided that all line marks connecting networks should have a circle glyph.

4.4.6 Loading screen

The idea of a loading screen was never initially discussed as the performance concerns were not significant at that time. However, after the visualization was created, it was noticed that it was not evident when a new visualization was generated if there was already a preexisting proposal on the screen. In order to make EdgeLap more usable for the user, a loading screen was added to ensure that it was clear when a new visualization had been drawn.

4.4.7 Larger histogram showing up on a new tab instead of overlay

The larger histogram view as described in section 6.1.4 was originally created as an overlay view that would appear over the visualization in EdgeLap's main screen. However, having the large histogram as an overlay made it hard for users to refer back to the histogram as they would have to close the histogram before being able to click another network glyph to investigate another histogram. Having the large histogram appear in another tab is easier as the user can open up many different histogram tabs to flip back and forth histograms.

5 Medium-level implementation discussion

EdgeLap is comprised of Java, PHP, HTML, CSS, JavaScript, and Processing code. It is a web based tool that works with Amazon RDS (or any relational database). The code is broken up into two sections: 1) Java, and 2) PHP/HTML/JavaScript/Processing. Aside from the libraries (section 5.1), all other code files are specifically written for EdgeLap.

5.1 Libraries used

Four libraries were used in this project:

- jQuery
- Processing
- MySQLConnector
- c3p0-0.9.5-pre10

jQuery is used to handle event callbacks with information from the datastore and interfacing between EdgeLap's HTML/JavaScript/Processing code and the PHP code working with the datastore. Processing is a library needed for the visualization portion of EdgeLap. The MySQLConnector, and c3p0-0.9.5-pre10 libraries are used in the Java code so that the code can work with the datastore.

5.2 Data Management

MySQL was the original preferred datastore option as opposed to Amazon RDS. However, using MySQL was a struggle as there were a series of installation problems that eventually cumulated in the reformatting of the primary development machine. Even after MySQL was setup, it was discovered that the primary development machine could not handle the level of database insertion needed for EdgeLap's dataset. Amazon RDS was chosen at this point due to the ease of setting up the database and because it could easily handle a large dataset without crashing. Thread management was implemented in order to avoid the MySQL crash from database insertion that was experienced earlier. Queries were batched in groups of 100 and a pool of worker threads was responsible for inserting the batches into the database. The database connection pool was also managed in order to help speed up interactions with the database in a multi-threaded environment. Essentially all this was done to make the inserting into the remote AWS database possible, since without it, the latency would in the order of several hours.

5.3 Visualization Creation

HTML, CSS, JavaScript, Processing, and PHP was used to create the actual visualization. The main screen of EdgeLap consists of a Processing sketch embedded into the HTML page. PHP was used to query the datastore and the results of that query would be passed back through PHP to JavaScript. The JavaScript portion would then communicate with the Processing code in order for the visualization to be displayed.

A large part of the struggle with creating EdgeLap came from unfamiliarity of the languages used for the project, the lack of proper debugging functionality in the Processing IDE, and the general lack of information on the Internet about the integration of Processing into HTML. A lot of the functions that would normally work in a Processing sketch did not work properly once it was embedded in a HTML page. It seems that Processing sketches embedded in a webpage are exported as a Java applet which meant a lot of the Processing specific functions did not work as intended.

A semi-difficult part of the project was constructing the SQL queries and checking the validity of the answers. Part of the problem stemmed from the fact that the LTSP dataset provided had an error in it when it was generated. A corrected version of the dataset was unable to be obtained for the project as the generation of the corrected version of the dataset took three months. Checking the results of the SQL queries often took a long time since there were times when the dataset had to be checked for consistency before the validity of the answer could be confirmed.

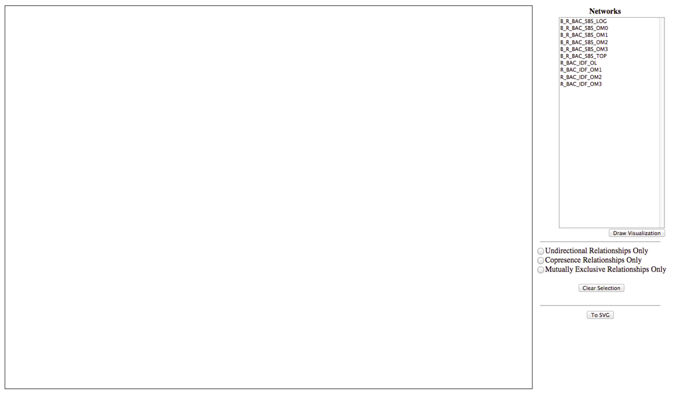
Another difficult part of the project was exporting to SVG. Since Processing by itself does not have a default export canvas to SVG feature. I had to import a library from Gliffy Inc which enabled the creation of a separate canvas context. After creating a mock canvas context, I then had to recreate the same image in the mock canvas as the one that is shown on screen.

6 Results

6.1 Scenarios of Use

6.1.1 To Begin

To begin, the user would select the files that he/she wanted to examine from the list of networks on the right (figure 5). The list of networks populated would be determined from what files have been loaded in the database. After making the selections, the user would have to click the"Draw Visualization" button located on the bottom of the list of files in order to draw the visualization. Between the time that ”Draw Visualization” was clicked and the visualization appears, there will be a loading screen that appears while the information is being fetched from the database (figure 6); the loading screen will disappear once the visualization has rendered. The user can select what types of edges he/she wishes to include in the visualization from the radio buttons underneath the list of networks.



Networks

B_R_BAC_SBS_LOG
B_R_BAC_SBS_OM0
B_R_BAC_SBS_OM1
B_R_BAC_SBS_OM2
B_R_BAC_SBS_OM3
B_R_BAC_SBS_TOP
R_BAC_IDF_OL
R_BAC_IDF_OM1
R_BAC_IDF_OM2
R_BAC_IDF_OM3

Draw Visualization
Undirectional Relationships Only
Copresence Relationships Only
Mutually Exclusive Relationships Only
Clear Selection
To SVG

Fig. 4. This is what EdgeLap looks like when it is first launched in a web browser. The list of networks available to use for a visualization appears on the right.

If none of the three options are selected, then the visualization that is generated will use the white circle glyphs (figure 7). It indicates that the sets displayed have OTU interactions where two OTUs have some sort of relationship in all of the networks that the glyph is connected to, but the OTUs may not have the same type of interactions throughout the different networks.

If the ”Unidirectional Relationships Only” option is selected, the blue/red circle glyphs (option a, b, and c in figure 2) will be used in the visualization (figure 8). This visualization would show OTUs that have the same type of relationship in all the networks connected to the circle glyph.

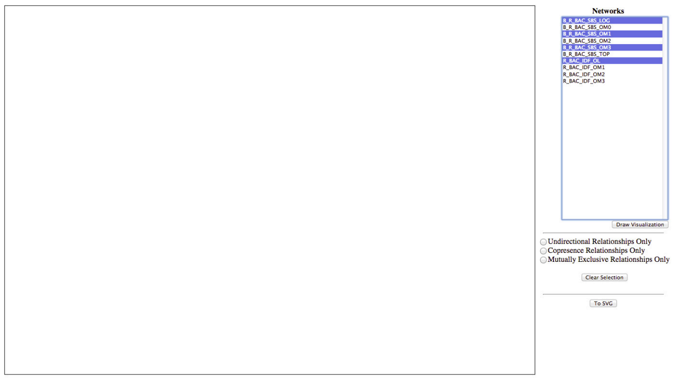


Fig. 5. A user can select 2-7 networks to generate a visualization for. In this case, the user has selected four networks.

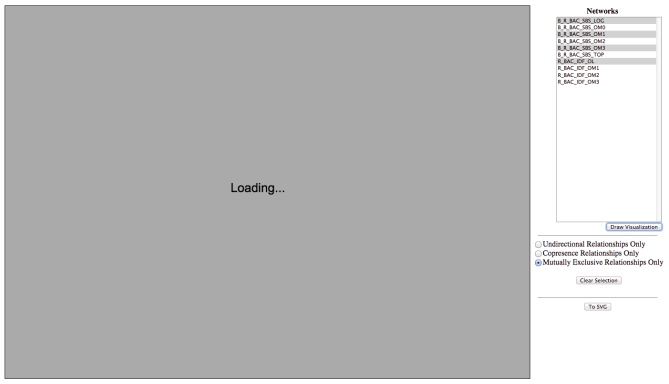


Fig. 6. While the visualization is rendering, a loading screen will appear.

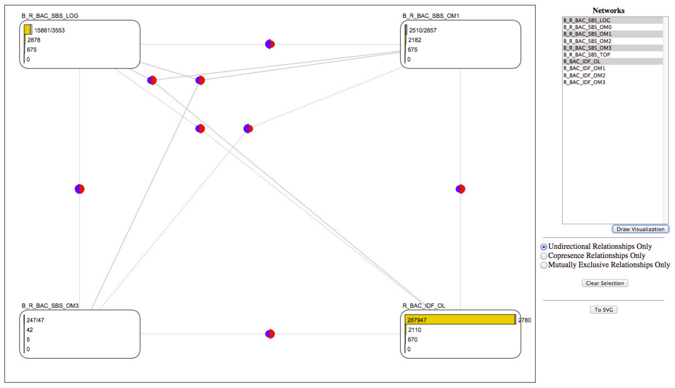


Fig. 8. If the user has selected the "Unidirectional relationships only" option when generating the visualization, the circle glyphs that appear in the visualization will be varied depending on the types of edges found in each intersecting set.

If the "Copresence Relationships Only" option is selected, the visualization would use the blue circle glyphs to display sets with copresence edges (i.e., positively co-occurrence edges) (figure 9).

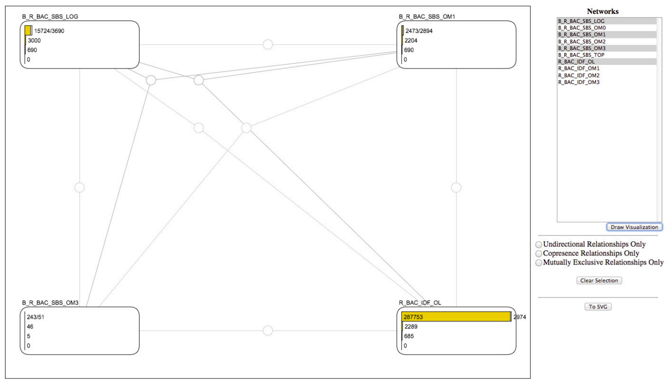


Fig. 7. If the user has not selected a specific type of edge that he/she is looking for, the visualization will display all overlapping sets that have two OTUs with some relationship in all the related networks. For example, if there are four networks, this visualization would show OTUs that have some relationship to each other even though they might have not a positive correlation through all the networks being examined.

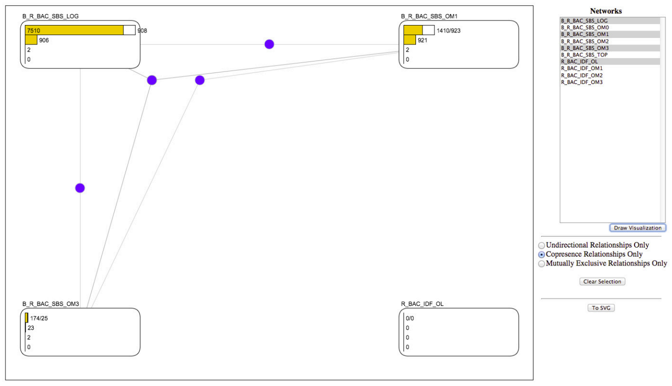


Fig. 9. If the user has specifically selected to look at only copresence relationships, the circle glyphs that appear on the visualization will reflect that.

If the "Mutually Exclusive Relationships Only" option is selected, the visualization would use the red circle glyphs to display sets with mutually exclusive edges (i.e., negatively correlated edges) (figure 10).

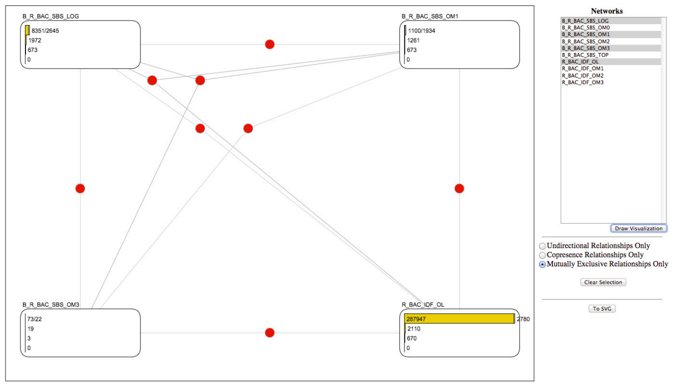
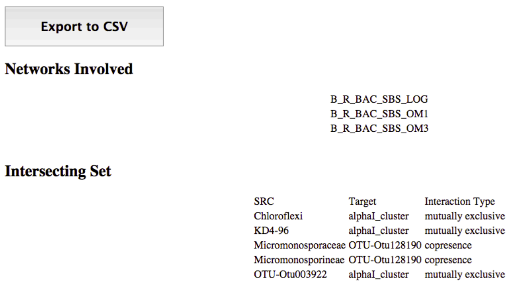


Fig. 10. If the user has specifically selected to look at only mutually exclusive relationships, the circle glyphs that appear on the visualization will reflect that.

6.1.2 Examining the Overlapping Set

The main window (the one of the left) will display the visualization based on the networks the user has specified. If the user wishes to examine the elements that are present in a specific overlapping set further, he/she can click on the circle glyph representing that overlapping set to have a list of the elements in set appear in another tab.


Export to CSV

Networks Involved

B_R_BAC_SBS_LOG
B_R_BAC_SBS_OM1
B_R_BAC_SBS_OM3

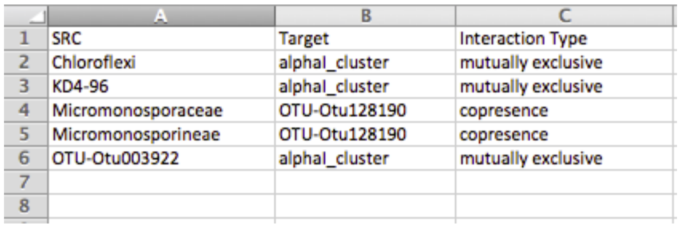
Intersecting Set

SRC	Target	Interaction Type
Chloroflexi	alphal_cluster	mutually exclusive
KD4-96	alphal_cluster	mutually exclusive
Micromonosporaceae	OTU-Otu128190	copresence
Micromonosporineae	OTU-Otu128190	copresence
OTU-Otu003922	alphal_cluster	mutually exclusive

Fig. 11. By clicking on a circle glyph, a new tab will appear with the list of all the edges that are in that set.

6.1.3 Exporting to CSV

If the user has performed the usage scenario described in 6.1.2, then the user will have the option of exporting that information to a CSV file for future use. Simply click the "Export to CSV" button located at the top of the list of elements (see figure 11) to download the file.



	A	B	C
1	SRC	Target	Interaction Type
2	Chloroflexi	alphal_cluster	mutually exclusive
3	KD4-96	alphal_cluster	mutually exclusive
4	Micromonosporaceae	OTU-Otu128190	copresence
5	Micromonosporineae	OTU-Otu128190	copresence
6	OTU-Otu003922	alphal_cluster	mutually exclusive
7			
8			

Fig. 12. The CSV file exported from the "Export to CSV" button shown in figure 11 will contain the same information as what is displayed on the screen in figure 11.

6.1.4 Examining Network Attributes

A secondary piece of information available to the user are general numbers about how many edges are shared with the different networks involved in current visualization (e.g., how many links does the network share with other networks in the visualization, how many of those links are only shared with exactly one network, exactly two networks, so on and so forth). If the user wishes to see a larger version of the histogram with labels for each bar, he/she can click on the network mark to have a larger labeled version of the histogram appear in a new tab.

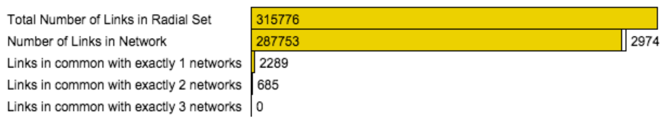


Fig. 13. A larger and labelled version of the histogram that the user can see if he/she clicks on the network mark.

6.1.5 Exporting to SVG

If the user wishes to export the visualization show on the left hand side of EdgeLap's main screen (see figure 4), he/she can click on the "To SVG" button. An SVG of the visualization will then be downloaded (figure 14).

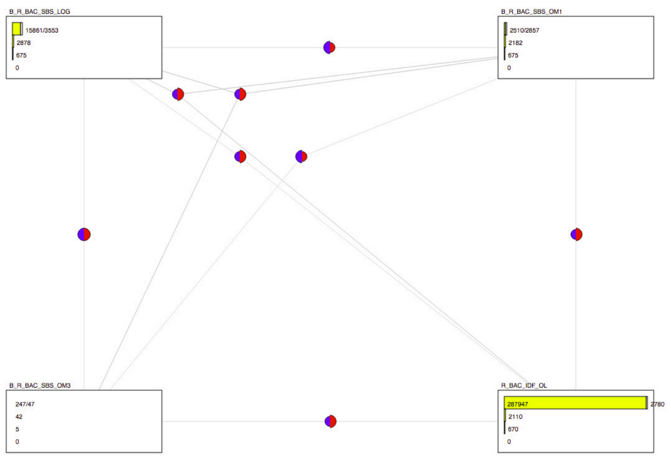


Fig. 14. The user has the option of downloading the visualization on screen as an SVG file. The information captured in the SVG file is the same as what is in the visualization but it looks slightly different from what is shown in the main screen of EdgeLap. This figure depicts what the SVG file looks like.

6.1.6 Does the interface solve the problem?

EdgeLap's interface allows the end users to solve the problem they were interested in, namely, what common edges are found in a set and all of its vary subsets of networks. Particularly, I think the loading screen was a good idea as it helped indicate to the user when the visualization was done loading. Without the loading screen, it was often unclear when a visualization had finished loading. A user would have to watch the screen for a quick flicker on the visualization to realize that a new one had been drawn in place. The flicker was often easy to miss and resulted in the user having to either a) wait for an extended amount of time to ensure that a new visualization had been drawn or b) click on the "Draw Visualization" button again and carefully watch for the flicker. With the loading screen, it is obvious when the visualization is being generated verses not knowing if the flicker had already come and gone.

I think the interface could be improved by either considering another idiom to replace the circle glyphs currently representing overlapping sets or finding a better method to draw all the circle glyphs and its corresponding lines. With an increased number of networks compared, the middle of the visualization can get cluttered with various lines and circle glyphs. The lines often are very close or slightly overlap each other which can cause readability issues. It would be worth either further investigating an idiom like LineSets [4] or implementing a filtering option to try and reduce the number of lines shown at any one time. Idioms like LineSets [4] could help due to the way it shows intersections between sets without the use of connecting lines but due to the sheer number of combinations that is required to be examined, using the LineSet [4] idiom would not be very practical.

7 Discussion

7.1 Strengths and weaknesses

A definite strength of EdgeLap is the scale of data it can show. Showing 2^n overlapping sets for n networks is not an easy task and although the current solution is not perfect, it does manage to show all the overlapping sets to the user. However, the tradeoff for being able to show a large amount of data is performance. While it is understandable that the performance is slow due to the number of searches it has to perform for each overlapping set and the size of the dataset, it does not change the fact that it can take two to four seconds for the visualization to be generated. Given that were is a loading screen to help the user's experience with waiting for a visualization to be generated, performance is a weakness but there does not seem to be a good way around the performance issue without having the visualization itself changed.

Another strength of EdgeLap is that it uses fairly similar visual encodings and idioms for the different polarity of edges. The user does not have to specifically readapt his/her expectations or method of reading the visualization even though he/she might be visualizing different things. Once the general idioms and encodings of EdgeLap has been explained, any visualization generated by it can be easily read by users.

As mentioned in section 6.1.6, the amount of lines in the middle of EdgeLap's visualizations can be highly cluttered. Also, the colour saturation scales for the lines connecting to the circle glyphs could be slightly altered to produce more of a obvious difference. Lines connecting a different number of networks are of different saturations; the more networks a set of lines is connecting to, the higher the line saturation. The colour saturation for the different types of lines was chosen to be numerically equidistance from each other. However, upon closer examination, it seems that it is harder to distinguish between the lighter saturations as opposed to the darker saturations. It would be a good idea to play around with the values some more to see if a better saturation balance could be reached.

7.2 Lessons learned

I think spending more time upfront with the design was a very good idea as it allowed for a relatively smoother path when it came to implementing. I spent a lot of time in the proposal stage getting to know the domain and the application's specific purpose before designing the visualization which resulted in EdgeLap's design not really significantly differing from the proposal stage to the final finished product. This definitely was an advantage towards implementing this project as I spent less time on communicating back and forth with the end users on various changes that were needed and more time on actual implementation. Compared to the other teams, I also started the implementation stage later as I had spent some extra time after the proposal finalizing final design questions and ideas with EdgeLap's end clients. However, I found this to my definite advantage as once I started the implementation stage, I did not have many design changes such that I had to erase and redo portions of the project repeatedly.

Something else I learned was to investigate the problems with the integration of various languages prior to choosing the development language. When I first researched about the basic languages needed for the project, I only looked at whether or not a language could do what I needed it to do. For example, I looked to see if it was possible to export a Processing sketch into a SVG file without looking at much specifics about the whole process. However, when it came time to actually implement the "Export to SVG" functionality, I realized it was much harder than I thought. The same idea was also true for the idea of using Processing in a web page—I knew it was possible but I did not know how much more trouble it caused. In the future, I will definitely look more closely at the mechanics and ease of how to do something and research more into the various known integration issues before further pursuing that path.

I also learned a lot about working with medium sized data. Truthfully speaking, in terms of big datasets, a 605 000 line dataset is not big at all but it is definitely of a much bigger scale than what I have worked with previously. Getting around the barrier of what it takes to insert and work with a dataset of this size was fairly rewarding to me. I got to look at different ways to design and improve the underlying database in order to try and speed up performance and while I did not end up using these designs because of their marginal gain to the project, I did definitely learn a lot about different things a DBA could consider when designing a database.

8 Future Work

If there was more time, I would definitely implement a filtering option for the user so that the user could choose how many sets appear on the visualization. As the number of networks to compare increases, the number of sets examined quickly increases. Even with comparing 2 to 7 networks, the number of intersecting sets that need to be considered quickly explode from 4 (2^2) to 128 (2^7). This leads to many lines and glyphs being draw in the middle of the networks and definitely make it much harder to tell which lines are connecting to which glyphs. Lines are also sometimes drawn over glyphs which causes it be obscured from the user. While the glyphs are not totally covered by lines and invisible, it definitely adds another layer of complexity for the user when trying to investigate different overlapping sets. It would be useful to have a filtering option so that the user can choose what types of overlapping sets he/she wants to look at in that point in time. For example, the user could choose to only examine overlapping sets for any three of the n networks being currently examined or the user could examine overlapping sets for any two, three or four of the n networks being examined. I feel that this would help declutter the space in the middle of the networks as well as help the user quickly pick out which circles are of relevance to them.

Another thing I would implement would be hovering over a circle glyph would cause all the lines that the glyph is associated from to be highlighted. The increase in the number of lines often makes it hard to tell which lines are associated to which glyphs and having the associated lines highlighted while hovering over a circle glyph would give the user context when looking at a certain intersecting set. Also, having some summary information appear (e.g., the number of items in the set) while hovering over a circle glyph would be ideal as it can provide the user with some preliminary information about the intersecting set without having the user click and switch tabs.

The algorithm to draw lines connecting networks to circle glyphs could also be improved by trying to implement some logic to avoid having lines drawn too close to each other. The way the lines are currently being drawn essentially depend on what networks they are trying to connect. However, this leads to a lot of lines being drawn closely together. For example, if there were lines connecting networks 1, 2, and 3 and lines connecting networks 1, 2, 3, and 4, the lines can be very close together—especially the parts of the lines that are closer to the network marks. This also can make it the visualization hard to read. EdgeLap could try to take some lessons LineSets [4] as it deals with a very similar problem.

I would also like to improve performance. Part of the current performance issue is due to the number of overlapping sets it has to check (2^n for n networks examined). I think this problem could be solved by either implementing some sort of caching mechanism or by modifying the visualization in order to cut down on the number of queries needed to generate one mechanism. With the current implementation, every time a visualization needs to be generated, EdgeLap issues a whole new set of queries. If we had a situation where the user wants to generate a visualization by adding in the edge filtering option (figure 8, 9, 10), we should be able to reuse the current data that has already been fetched to further filter it. The visualization could also be modified in order to cut down on that number to improve performance. If the filtering option was implemented, the performance of EdgeLap would probably increase along with it as there would be less overlapping sets to calculate data for.

Another interesting addition to EdgeLap could be to find some way to visualize or display how many edges in a network are positive in one network and negative in another. I think the circle size could be an interesting idiom to try and use with this kind of question.

9 Conclusion

EdgeLap is a visualization tool used to identify and discover features in overlapping networks. A variety of idioms and encodings including glyphs, colour, colour saturation, aligned positioning, and length were used in the tool. There is functionality to export both the visualization and the underlying information in textual form, which helps make EdgeLap a standalone application able to be easily setup and used. Although further work could be done to enhance EdgeLap, EdgeLap is successful in its task to visualize common edges that appear in a set and subsets of networks and has accomplished the four tasks in 3.2 that it aimed to achieve.

References

[1] Cytoscape. `http://www.cytoscape.org/what_is_cytoscape.html`. Accessed: 2014-10-30.

[2] Gephi. `http://gephi.github.io/`. Accessed: 2014-10-30.

[3] Hive plotter. `http://www.hiveplot.net/`. Accessed: 2014-10-30.

[4] B. Alper, N. H. Riche, G. Ramos, and M. Czerwinski. Design study of linesets, a novel set visualization technique. *Visualization and Computer Graphics, IEEE Transactions on*, 17(12):2259–2267, 2011.

[5] B. Alsallakh, W. Aigner, S. Miksch, and H. Hauser. Radial sets: Interactive visual analysis of large overlapping sets. *Visualization and Computer Graphics, IEEE Transactions on*, 19(12):2496–2505, 2013.

[6] B. Alsallakh, L. Micallef, W. Aigner, H. Hauser, S. Miksch, and P. Rodgers. Visualizing sets and set-typed data: State-of-the-art and future challenges. 2014.

[7] A. Barberán, S. T. Bates, E. O. Casamayor, and N. Fierer. Using network analysis to explore co-occurrence patterns in soil microbial communities. *The ISME journal*, 6(2):343–351, 2011.

[8] K. Faust and J. Raes. Microbial interactions: from networks to models. *Nature Reviews Microbiology*, 10(8):538–550, 2012.

[9] S. Gratzl, N. Gehlenborg, A. Lex, H. Pfister, and M. Streit. Domino: Extracting, comparing, and manipulating subsets across multiple tabular datasets. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '14)*, 2014. to appear.

[10] A. Lex, N. Gehlenborg, H. Strobelt, R. Vuillemot, and H. Pfister. Upset: Visualization of intersecting sets. 2014.

[11] T. Munzner. A K Peters Visualization Series. CRC Press, 2014.

[12] R. F. Powers, D. A. Scott, F. G. Sanchez, R. A. Voldseth, D. Page-Dumroese, J. D. Elioff, and D. M. Stone. The north american long-term soil productivity experiment: Findings from the first decade of research. *Forest Ecology and Management*, 220(13):31 – 50, 2005.

[13] R. Sadana, T. Major, A. Dove, and J. Stasko. Onset: A visualization technique for large-scale binary set data. 2014.