POLITECNICO DI MILANO
Corso di Laurea Magistrale in Ingegneria Informatica
Dipartimento di Elettronica, Informazione e Bioingegneria

BRANCHINGSETS: A VISUALIZATION TECHNIQUE FOR THE INTERACTIVE INSPECTION OF BIOLOGICAL PATHWAYS

Necst Lab
Novel, Emerging Computing
System Technologies Laboratory

Relatore: Prof. Santambrogio Marco Domenico

Tesi di Laurea di:
Francesco Paduano, matricola 818226

Anno Accademico 2014-2015

Curiosity is gluttony.

To see is to devour.

ACKNOWLEDGMENTS

I would never make a comprehensive list all the people I am grateful to. Otherwise, the preamble of this thesis would be even thicker than the thesis itself.

I would like to thank my advisors **Marco Domenico Santambrogio** and **Angus Graeme Forbes**, they both guide me during this year with their persistent motivation, energy and enthusiasm.

I am grateful to all the other professors that taught me during the last years. My deepest gratitude goes to all of them.

A sincere thank to **mum**, **dad** and my uncle Stefano. They removed every barrier between me and my dreams.

Thanks to my dear uncle Ugo, if he hadn't given me the first floppy disk with a compiler when i was 12, I wouldn't be here now.

Domenico, thanks for being my big brother. Without any doubt, the person who had the biggest influence in my life.

And **Chiara**, **Serena**, **Donata** for being elder sisters. Chiara, thanks for the premature math lessons, Serena for being my life teacher, Donata for being my music advisor.

Viola, thank for being on my side for the last year. For being my constant inspiration.

Thanks to my friend **Federica**. Thank for teaching me to aim high.

ACKNOWLEDGMENTS (continued)

There are no adequate words for describing my gratitude for **Bruno** and **Raffaele**. They are the most outstanding and entertaining friends. Your absence in the last months has been the biggest hurdle of my every-day happiness.

Andrea and **Giada** for their friendship during these university years.

Yasmine and **Valentina** for being the most lackluster, loafer and tacky friends. Keep taking care of Bruno! Thanks **Daniele** for your exaggerated and unresting spirit.

Marianna, **Antonino**, **Michele**, **Simone**, **Niccolo'**, **Eleonora**, **Giovanni**, **Francesco**, **Mauro** for the musical emotions.

A thought to **Achille**, **Giulia**, **Irene**, **Omar** and all the other high-school friend who already graduate or are about to. Always been proud of you.

And **Gianluca**, **Matteo**, **Alessandro(s)**, **Lorenzo**, **Enzo** and all the other friends that made this time in Chicago enjoyable.

Adrianita, for your company and friendship during these months in Chicago.

TABLE OF CONTENTS

CHAPTER			PAGE
1	**INTRODUCTION**		1
2	**A MODEL FOR BIOLOGICAL PATHWAYS**		6
	2.1	Model	6
	2.2	Pathway Data Formats	8
3	**TASKS AND REQUIREMENTS ANALYSIS**		9
	3.1	Tasks	9
	3.2	Requirements	16
4	**RELATED WORK**		23
	4.1	Visual Encodings and Representations	23
	4.1.1	Graph visualizations	23
	4.1.1.1	Human curated node-link diagrams	24
	4.1.1.2	Automatically generated node-link diagrams	25
	4.1.2	Visualizing Categories on Node-Link Diagrams	27
	4.1.3	Hierarchical Techniques	30
	4.1.4	Matrix-Based Techniques	32
	4.2	Tools	33
	4.3	Future Challenges for Pathways Visualization Tools	38
	4.3.1	User Interaction and Better Visual Encoding	38
	4.3.2	Data Integration	39
	4.3.3	Experiment New Techniques	39
	4.3.4	Abstract Representations	40
	4.3.5	Visualizing Uncertainty	40
	4.3.6	Pathway Logic	40
	4.4	Final Remarks on Related Work	41
5	**BRANCHINGSETS**		42
	5.1	Design Study of BranchingSets	44
	5.1.1	Design Goals	44
	5.1.2	Visual Encoding	46
	5.1.3	Showing Category Information Only For Nodes	47
	5.1.4	Directed Graphs	48
	5.1.5	Protruding Links	49
	5.1.6	Interactions	49
	5.1.6.1	Highlighting and Labelling Categories	50

TABLE OF CONTENTS (continued)

CHAPTER **PAGE**

5.1.6.2 Hiding and Unhiding Categories . . . 50
5.1.6.3 Filtering by Keyword . . . 52
5.1.6.4 Expanding Neighbours of a Selected Node . . . 52
5.1.6.5 Navigating and Rearranging the Layout . . . 52
5.1.6.6 Finding Intermediate Steps between Two Nodes . . . 52
5.2 Applications . . . 53
5.2.1 Co-authorship Network . . . 54
5.2.2 Geospatial Data . . . 56
5.3 Final Remarks on BranchingSets . . . 56

6 EVALUATING BRANCHINGSETS WITH AN USER STUDY . . . 59
6.1 User Study . . . 59
6.1.1 Method . . . 59
6.1.2 Apparatus . . . 63
6.1.3 Tasks . . . 63
6.1.4 Hypotheses . . . 64
6.1.5 Results . . . 65
6.1.6 Qualitative Study . . . 68

7 BRANCHINGSETS WITH BIOLOGICAL PATHWAYS . . . 69
7.0.7 Related Work . . . 69
7.1 The Design . . . 71
7.1.1 Node-Link Diagram with BranchingSets . . . 72
7.1.2 Hierarchical Inspection . . . 73
7.1.3 Colors . . . 77
7.2 Curved Links . . . 79
7.2.1 Interactions . . . 80
7.2.1.1 Pathway Highlighting and Labelling . . . 80
7.2.1.2 Pathway Hiding and Unhiding . . . 82
7.2.2 Nested pathways inspection . . . 85
7.2.2.1 Keywords Filtering . . . 85
7.2.2.2 Upstream and Downstream Expansion . . . 85
7.2.2.3 Layout rearranging, zooming and panning . . . 87
7.2.2.4 Complex Structure Inspection . . . 87
7.2.2.5 Finding intermediate steps between two nodes in a pathway . . . 87
7.2.3 Additional Visual Encodings . . . 89
7.2.4 Animations . . . 90

8 IMPLEMENTATION NOTES . . . 92
8.1 The Prototype . . . 92
8.2 D3.js . . . 92
8.3 Data Structure . . . 92

TABLE OF CONTENTS (continued)

CHAPTER **PAGE**

8.4 Force Layout 93
8.5 Rectangles Packing 95
8.6 Labelling 97
8.6.1 The Concept 99
8.6.2 The Algorithm 100
8.6.3 Limitations 103
8.7 Drawing Curved Links 104

9 COMPARISON WITH OTHER SOLUTIONS 107
9.1 Comparison 107

10 CONCLUSIONS AND FUTURE WORK 114

CITED LITERATURE 117

VITA 126

LIST OF TABLES

TABLE		PAGE
I	LIST OF TASKS	10
II	LIST OF REQUIREMENTS TO SUPPORT THE TASKS *T1-11*	16
III	COMPARISON BETWEEN THE REQUIREMENTS IMPLEMENTED IN POPULAR TOOLS FOR PATHWAY VISUALIZATION	33
IV	Tasks used in the quantitative experiment	64
V	COMPARISON BETWEEN *BranchingSets* TECHNIQUE AND OTHER TECHNIQUES INCLUDED IN POPULAR TOOLS FOR PATHWAY VISUALIZATION	113

LIST OF FIGURES

FIGURE		PAGE
1	The image represents some techniques for representing categories on graph structures: **a)** The node is divided in multiple fans colored accordingly to the category **b)** Nodes and edges have colored contours **c)** KelpFusion **d)** LineSets .	27
2	The image represents the most common technique for representing tree structures: **a)** A traditional tree diagram **b)** tree diagram with a circular layout **c)** Tree Map **d)** H tree **e)** Circle Packing	30
3	A view of *Entourage* showing links between a larger "focus pathway" and several "context pathways."	35
4	Visualizing hundreds of nodes and links with Biofabric	36
5	Visualizing different gene expression with Reactome Pathway Browser . . .	37
6	Two simple examples of node-link diagrams enhanced with BranchingSets.	44
7	Coloring of the links in case of no information related to the membership. Since the node **B** and **C** does not have any category in common, the link which connect them is black and thinner	46
8	BranchingSets applied to a directed node-link graph. The link between **B** and **C** is bidirectional.	46
9	Node **E** belongs to the blue category but is visually disconnected from the other blue nodes.	48
10	BranchingSets **(a)** without protruding links **(b)** with protruding links.	49
11	The same graph with different visible categories **(a)** only violet **(b)** violet and green **(c)** violet, green and orange **(d)** violet, green, orange and blue	51
12	On the left, the user is dragging the mouse from one biological complex to another. On the right, the visualization is updated with all the connecting paths	53
13	Visualizing the co-authorship network of authors who published at least one paper at *IEEE Vis* from 2012 to 2014. **(a, b, c)** are zoomed views of sub-networks of the overall representation	54
14	Visualizing location and category of a group of restaurants in Milan with **(a)** LineSets **(b)** BranchingSets	57
15	Screenshoot of the interface of the user study. The image shows **(1)** the question **(2)** the answer **(3)** the screen area dedicated to the representation . . .	60
16	Examples of datasets used for the user study. **(a, b)** Representation of the same dataset with BranchingSets and LineSets. **(c, d)** Representation of the same dataset with BranchingSets and Bubble Sets	61
17	Examples of a dataset used in the user study for Task 5. **(a)** Representation of two graphs in separated views. **(b)** Representation of two graphs overlapped with BranchingSets	62
18	Chart showing the accuracy (left) and task completion times (right) for Tasks 1-4	65

LIST OF FIGURES (continued)

FIGURE **PAGE**

19 Subjective user ratings from the qualitative study 66
20 Chart showing the mean completion time for Tasks 5 67
21 The image shows the proposed technique to combine node-link diagram with *LineSet*. The arrangements of the nodes follows the topological order of the links 71
22 On the left, a reaction which involves two left components (A,B) and two right components (C,D). On the right, the figure explain the relationship between the four components considered in our technique 72
23 The figure demonstrates hierarchical inspection using the symbolic overview and the the pruned tree. In (a), the user selects protein *G* from the expanded node within the node-link diagram; this protein is highlighted within the pruned tree visualization, which also provides more information about its siblings (the proteins *D*, *E*, and *F*) and its parent and grandparent nodes (the complexes *C*, *B*, and *A*). In (b), the user selects a complex *B* in the expanded node, which is displayed in a similar manner in the pruned tree. 74
24 Two cases of inspection of the structure of a complex by means of two coordinated views 75
25 Screenshoot of the Color menu. The menu shows the available palettes and a preview of the selected palette. 79
26 The figure represent the same graph (a) with curved links (b) with straight links. The dashed circle (1) contains a cycle, (2) is not a cycle. 81
27 The figure represent two overlapping edges (a) with straight links (b) with curved links 82
28 When the user hovers on a component, the application will highlight only the relevant pathways 83
29 Screenshot of the interface of the prototype. On the left side from the top: the search field, the list of searched keywords and the list of pathways loaded by the application 84
30 Exploring the the pathway hierarchy. A sub-pathway is selected in orange, whereas the whole S-Phase pathway is selected in green. Inside S-phase, another subpathway is selected in blue. 86
31 Location of all the *MCM2-7* in the enlarged complexes 88
32 On the left, the user is dragging the mouse from one complex to another. On the right, the visualization is updated with all the connecting paths 89
33 Additional visual encodings embedded in our tool. Figure (a) shows (1) upstream and downstream markers (2) downstream marker (3) upstream marker. Figure (b) shows the downstream markers during the “finding intermediate steps between two nodes in a pathway” interaction. 90
34 ER-Diagram of the data structure used in the prototype 94
35 Figure (a) show the layout of a graph with a simple force layout. Figure (b) represents the same network with extra forces to show the directionality of the graph 95

LIST OF FIGURES (continued)

FIGURE		PAGE
36	*Padding* and *margin* parameters	96
37	Examples of rectangles packing generates with our algorithm with real biological complexes data	98
38	The rectangle A overlaps with B, and its vertical and horizontal traces overlap with B,C,D,E.	100
39	A scenario of a rectangle *A* placed on the plane, and its *horizontal overlapping profile*	101
40	Figure (a) shows an *overlapping profile*, the dashed line is a cubic *penalty function*. Figure (b) shows the combination of the *overlapping profile* with the *penalty function*. In both figures the global minimums are highlighted with red circles.	102
41	Quad-tree data structure representation for 5 rectangles	102
42	Figure (a) represents the angle α, the *thickness* parameter and $\vec{pd}, \vec{pa}, \vec{pr}, \vec{pp}$ vectors. Figure (b) represents the *tip_length* and *handle_length* parameters. The picture also show P1, P2, P3, P4 control points and the corresponding handles.	105
43	Sequence of screenshot of the first use case	110
44	Sequence of screenshots of the second use case	111

ABSTRACT

Understanding complicated networks of interactions between biochemical components is essential to solve contemporary problems in modern biology, especially in domains such as cancer and systems research. In these domains, biological pathway data is used to represent chains of interactions that occur within a given biological process. Visual representations can help researchers to understand and interact with complex pathway data in a number of ways.

Biologists make use of pathway visualizations for a range of tasks, including having an overview of large or multiple pathways, understanding inter-pathways connectivity and retrieving details concerning network entities and interactions. Some of these tasks require an understanding of the hierarchical nature of certain elements within the pathway or the ability to easily make comparisons between multiple pathways.

In this work, we present a taxonomy of tasks that are regularly performed by researchers who work with biological pathway data. From these tasks we generate a set of requirements that should be taken into consideration for the design of a pathway visualization software. Then, we present an overview of visualization techniques that are used for the representation of complex pathway data. We also perform a survey of a variety of pathway visualization tools that utilize different visualization techniques.

The most popular representation technique for visualizing the biological interconnections between components in a pathway is the node-link diagram. However, frequently the connectivity between elements is not the only information that it is necessary to display. In certain application domains, elements and links can be grouped in categories. To address these requirements, a number of techniques

have been designed for overlaying node-link diagrams with additional information. In this work we introduce *BranchingSets*, a novel technique for augmenting node-link diagrams with information about the categories which the nodes and links belong to. The technique enables user-driven interactions to procedurally examine the graph topology.

Furthermore, we present a pathway visualization tool which leverages *BranchingSets*, for the exploration of biological pathways incorporating both an intuitive graphical representation of multiple levels of information and we include a well-designed set of user interactions for selecting, filtering, and ordering the view of the data.

We provide a comparison between popular techniques for representing biological pathways and our technique. Our comparison shows that our technique provides functionality not currently enabled by current techniques, and that this functionality could be incorporated into existing visualization applications to enable additional visual analytics tasks.

ABSTRACT (ITALIAN)

Alcune ricerche ed esperimenti di biologia richiedono sempre più la comprensione e l'analisi di vaste e complicate reti di componenti biochimici, specialmente in settori come la ricerca per la cura del cancro o delle cellule staminali. In questi ambiti, i pathways biologici sono usati per rappresentare le catene di interazioni che avvengono all'interno di un processo biologico. Le rappresentazioni grafiche di un pathway biologico possono fornire un insostituibile supporto ai ricercatori per comprendere e interagire con questi dati in diverse maniere.

I biologi fanno uso di visualizzazioni di pathway in svariati casi, come ad esempio per avere una panoramica di uno o più pathway molto vasti, comprendere connessioni di alto livello tra diversi pathway, e ottenere dettagli aggiuntivi riguardo le diverse entità coinvolte e le interazioni tra esse. In alcune circostanze è necessario comprendere la natura gerarchica di alcuni elementi, in altre il biologo deve aver a disposizione uno strumento per comparare pathway differenti.

In questo lavoro di tesi verrà presentata una tassonomia di diverse operazioni che sono regolarmente eseguite dai ricercatori che abitualmente fanno uso di dati relativi ai pathway. Da questa lista di operazioni verrà estratta una serie di requisiti che devono essere presi in considerazione per il design di un software per la visualizzazione di pathway. Successivamente sarà presentata una panoramica di diverse tecniche di visualizzazione che sono usate per la rappresentazione di dati complessi relativi ai pathway. Verrà inoltre illustrata un'analisi di diversi strumenti per la visualizzazione di pathway e le corrispettive tecniche utilizzate.

ABSTRACT (ITALIAN) (continued)

La tecnica più popolare per la visualizzazione di pathway biologici è il node-link diagram. Sfortunatamente, le interconnessioni tra diversi elementi non è l'unica informazione significativa da mostrare. In alcune applicazioni, elementi e connessioni possono essere raggruppate in categorie. A fronte di cio', una serie di tecniche sono state affinate per aggiungere un ulteriore livello di visualizzazione aggiuntivo ai node-link diagram.

In questo lavoro verrà presentata una nuova tecnica per includere in un node-link diagram informazioni relative alle categorie dei nodi e delle connessioni: ci riferiremo ad essa con il nome *BranchingSets*. Questa tecnica include una serie di interazioni da parte dell'utente per permettere una esplorazione procedurale della rete.

Inoltre verrà presentato uno strumento per la visualizzazione di pathway che fa uso di *BranchingSets* per facilitare l'analisi di pathway biologici, permettendo sia una rappresentazione intuitiva di diversi livelli di informazione, sia una esplorazione procedurale della rete di interconnessioni tramite il filtraggio, la selezione e l'ordinamento dei dati.

Verrà inclusa anche un confronto tra BranchingSets e diverse tecniche per la rappresentazione di pathway biologici. Il confronto suggerisce che la tecnica presentata in questo lavoro offre alcune funzionalità che non sono attualmente incluse in strumenti esistenti.

CHAPTER 1

INTRODUCTION

Great effort has been made by biologists to organize the available knowledge on biological pathways. For instance, the Reactome Database (1) and the KEGG Pathway Database (2) are two examples of publicly accessible resources of biological data. Databases such as these, and the frameworks created to access, process, and query them, such as *Pathways Commons* (3), allow biologists to investigate multiple pathways simultaneously that may share common elements, such as biochemical reactions or protein complexes. The flexibility of these search tools, and the scale of the data that can be quickly retrieved, has motivated researchers to design new visualization tools to assist in a range of analysis tasks involving multiple pathways. A rich selection of requirements for pathways visualization tools is detailed in (4), which stresses the need of further research into interactive, dynamic solutions. Furthermore, it casts doubts on the possibility to satisfy all requirements with a unique visualization, and it encourages the design of balanced solutions. Indeed, the requirements for a pathway visualization tool include a variety of tasks, such as displaying related networks at different levels of abstraction, representing large networks and providing details about network entities and interactions. Accomplishing all these tasks in one representation may not be feasible. A visual technique usually chooses to address only few of these requirements.

In the last decade, analyses that involve thousands of proteins or genes have become conventional. Numerous attempts have been proposed to visualize and analyze large biological networks, with particular attention to the topology of the network and its hierarchical structure.

Biologists' analysis often require to merge information from different pathways, since pathways groups together participants and reactions that are often shared between other pathways. Furthermore, a single pathway presents an hierarchical structure of sub-pathways. The visualization of this hierarchy, as well as to manage to represent relationships between pathways, open doors to a new set of challenges.

This work will focus in particular on exploratory tasks which involve multiple pathways. Our purpose is to design a technique and to implement a prototype that can effectively display the similarities, the shared components and the interconnections between different pathways. Furthermore, we enable the user to efficiently explore the nested structure of a pathway and its components.

Typically, pathways are represented as directed graphs, where nodes in the graph represent biological "participants", such as proteins or protein complexes, and where the edges represent a biological functionality, such as a biochemical reaction. Often different shapes for arrows and nodes are used to differentiate between the different types of molecules or reactions.

In order to visualize several pathways on the same network, we present *BranchingSets*, a visual technique to represent categories on node-link diagram. *BranchingSets* provides group information on links and nodes without introducing visual clutter. Differently from other techniques to display category information on node-link diagrams, our technique integrates on the network representation, avoiding to occlude and clutter the visualization.

Indeed, *BranchingSets* colors nodes and links according to the category. When a node belongs to more category, multiple border are added around the node. If a link is shared between more categories, multiple line segment are placed side by side. Each pathway or sub-pawhay that is visible in the representation is assigned to a category which is bound to a color.

The representation of the topology of the network is still depicted with the usual conventions of node-link diagrams, which enables our technique to be accessible and straightforward to who is familiar with the traditional pathway representations. However, even though this type of visual encoding is the most widespread, node-link diagrams are known to have a number of issues. For instance, as the number of nodes or edges increases, it quickly becomes more difficult to make sense of the data (5).

The importance of a dynamic visualization which overcomes the limitations of a static representation has been recognized by the community for a long time ((6), (7)). Indeed, the few advantages of static images, such as a fine-tuned representation which carefully improves readability and fits the available space, are exceeded by the navigation methods typically supported by dynamic visualizations. Furthermore, it is important that the layout can be programmatically changed by the user, and that the existing representation can be extended with new components.

For the mentioned reasons, *BranchingSets* includes some well-designed user interactions to dynamically filter, expand and rearrange the visualization. The biologist can display only the biochemical participants that match a set of keywords, and then progressively expand the network of interconnections. Furthermore, the visualization can answer to more complex queries, such as displaying all the interconnections from a given entity to another.

Traditional node-link diagrams, even if enhanced with *BranchingSets*, do not provide the information to represent all the complexities of a pathway data structure. The data structure of pathways can be formalized with a hypergraph, a generalization of a graph with hyperedges that can connect any number of nodes. A biochemical reaction might involve multi-participant relationships. Moreover, a pathway usually contains multiple subpathways, and a single node can represent a nested structure of many bio-

logical entities. Some techniques for the visualization of pathways propose representations which limit the complexity of the data structure in favour of a simpler design. This is the case of the SIF binary network, which is often used to generate visualization using *Cytoscape* (8). This format represents only binary relationships and excludes rich biological semantics.

Inspired by the simplification adopted by the SIF binary network, we represent a biochemical reaction as a set of binary interconnections between components. This simplification makes easier to visually follow the chain of causal relationships between components and the cross-talk between pathways.

Lastly, in order to provide a representation of the hierarchy of the components, we have designed a specific technique which leverages two coordinated views. One view is a symbolic representation which is integrated in the network representation and provides a visual clue of the component structure as well as its position in the network. A separated view shows a "pruned tree', which similarly to a classical tree-diagram shows part of the hierarchical structure. The user can interact with the symbolic representation to change the information displayed in the "pruned tree" view.

The next chapter presents a brief description of the data structure of a pathway. Chapter 3 groups tasks and requirements of a pathway visualization tool, and Chapter 4 introduces the reader to the more popular tool for the analysis of biological pathways and the more common visualization techniques implemented.

Chapter 5 presents *BranchingSets*, a visualization technique for representing categories on node-link diagrams. Chapter 6 describes a user study that demonstrated that *BranchingSets* is effective at helping users to recognize similar patterns and identify differences between graphs as well as to identify nodes that belong to multiple categories. For other tasks, such as simple intersections or membership tasks, our

technique was comparable to *LineSets*. Even when there was no quantitative difference is accuracy or speed, users indicated that our solution produces representations which are more easily comprehended and less cluttered when compared with *Bubble Sets* or *LineSets*. Chapter 7 illustrates the features of a prototype which leverages *BranchingSets* to explore the pathways structure and the interconnections between the components. Chapter 8 focuses on some implementation details relevant to our prototype and Chapter 9 compares our prototype to already existing tools and explain two simple use cases that illustrate how the features enabled by our prototype can help to explore biological evidences. Chapter 10 draws the conclusions of our work.

CHAPTER 2

A MODEL FOR BIOLOGICAL PATHWAYS

In this chapter we explain the nature of the information stored in a pathway. Is not the purposes of this work to provide a formal and complete definition of the data structure of a pathway. Is instead our intention to introduce the reader to the pathway structure and fundamental entities, in order to better understand the requirements for a visualization technique and the kind of tasks requested by biologists. After an insight of the structure of the data, we present the most popular formats to store pathways information.

2.1 Model

The information stored in a biological pathways involves three major components:

- **Participant**

  A participant has a specific nature. It might be a protein, a compound, a gene. It is identified with a name and it is involved in one or more relationships;

- **Relationship**

  A relationships involve two or more components. Various kind of relationships with different biological meaning are present in a pathways. They differentiate according to the directionality and the cardinality;

- **Meta-data**

  The complex nature of the information stored in a pathway in enriched with additional data. A

meta-data usually represents information concerning the context of the pathway, links to additional resources or personal comments or notes of the author.

The presented classification of the components of a pathways can be further detailed. Indeed, we identify three families of relationships which differs from each other for the biological meaning.

- **Binary Relationships**

  A binary relationship involves two participants and is either directed or undirected. Example of directed binary relationship are *catalyst* and *reactions*, undirected relationships relationships are for instance simple interaction between proteins.

- **Many to Many Relationships**

  Biological reactions might be originated from more than one participant and might produce more than one product. Representing this kind of relationships is usually more challenging than *binary relationships*.

- **Many to one Relationships, or grouping**

  Participants of a pathways are often grouped in hierarchical structures. These are *many to one* relationships which identify whether a group of participants belongs or not to a given *set*. A *set* might by a *compound*, a *complex* or even a whole pathway.

Various kind of descriptions of the same pathway might be available according to the level of detail, the field, and the nature of the researches supported by the pathway. According to the different relationships and meta-data the description involves, the data structure can be modelled as a *graph*, a *metagraph* or an *hypergraph*. A *graph* involves only *participants* and *binary relationships*. Even if

this representation is extremely oversimplified, as explained later it is used to represents interactions between *participants* in large-scale networks.

A *metagraph* add to the *graph* the concept of **grouping**. One participant might group many *participants*. In a *hypergraph* relationships are no more restricted to be binary but can involve an arbitrary number of participants.

BranchingSets, the technique introduced in Chapter 5, shows binary relationships and grouping information. The binary relationships are encoded accordingly to the usual notation for directed node-link diagrams, whereas the grouping information is displayed by means of a smart coloring of links and nodes.

Furthermore, the tool presented in Chapter 7 integrates *BranchingSets* with an additional visualization to show hierarchical structures of biological components.

2.2 Pathway Data Formats

Pathway data can be stored in one of several file formats. In particular *BioPAX* (9), *KEGG* (10) and *SBML* are the most popular standards for storing the complex data structure described in the previous section. These formats are xml based and represent data as an ontology.

Other formats are employed for the visualization of biological pathways that are not specific to the field of biology. For instance *SIF Simple Interaction Format* used by *Cytoscape* (8) it is used to visualize undirected interactions between participants. Automated procedure to translate formats such as *BioPAX* to this simplified format are available.

CHAPTER 3

TASKS AND REQUIREMENTS ANALYSIS

In this chapter we detail the more relevant tasks performed by domain experts during the analysis of a biological pathways. From the description tasks we have inferred the requirements for a pathways visualization tool.

3.1 Tasks

In this section we compile a list of tasks relevant to biological pathway analysis. Task descriptions can be found in Table IV.

Task 1 Examine the upstream and downstream connections between two entities within a pathway.

Understanding how entities within pathways are connected are essential to most research related to pathway data. While some analyses involve undirected relationships between genes or gene products, studies of metabolic networks and other inter-cellular processes rely on directed relationships, which requires an understanding directed relationships between entities.

When discussing directed paths between entities, one entity is said to be *upstream* or *downstream* of another. As mentioned earlier, understanding upstream and downstream relationships is particularly

TABLE I. LIST OF TASKS

code	description
T1	Examine the upstream and downstream connections between two entities within a pathway
T2	Detect feedback loops of activation and inhibition within a pathway
T3	Examine high-level relationships between modules and pathways
T4	Analyze high-throughput *omics* data
T5	Discover potential "causal" mechanisms of up or down-regulation within a set of entities
T6	Simulate the effects of activation, inhibition, or knockout within a pathway
T7	Incorporate new entities or relationships into an existing pathway and examine their effects
T8	Incorporate experimental results with pathway data
T9	Curate, edit, and "debug" pathway data files, or construct user-defined pathways from scratch
T10	Reveal levels of confidence or uncertainty associated with relationships within a pathway

important to domains such as cancer drug research, where a drug may affect a small subset of genes or gene products, which in turn will affect various downstream processes.

In most cases, a directed relationship is meant to represent a biochemical reaction, where one entity is consumed as a reactant and another is produced as a product. Thus, an upstream entity may be connected to a downstream entity through a chain of several directed links. In the most basic sense, the "entities" mentioned above are genes, gene products (such as proteins or complexes), or other small molecules within a cell. A researcher may be interested in understanding the path of reactions (or other relationships) that connects two entities.

It is important to note, however, that directed relationships in biological pathways can take a variety of forms, such as entities that that modulate, *inhibit* or *activate* the catalysis of other biochemical reactions (9).

Task 2 Detect feedback loops of activation and inhibition within a pathway.

A task that follows directly from Task 1 is the detection of *feedback loops*. Feedback loops are common within metabolic activation networks, and they play a key role in processes related to uncontrolled cellular growth in cancerous cells.

Task 3 Examine high-level relationships between modules and pathways.

While observing relationships between two individual entities may be important in certain contexts, sometimes might be significant to reveal complex relationships that go beyond simple connections between individual entities. These complex relationships include abstract relationships between high-level subsets of a pathway as well as compound relationships (*many-to-many* or *many-to-one* relationships) between groups of entities.

High-level relationships are characterized by abstract subsets of a pathway. In BioPAX pathway data, abstract relationships between pathways and other entities can exist, such as when, for example, a biochemical reaction is connected to an entire pathway through the abstract *nextStep* relationship. These high-level groupings are similar to *reaction modules* described in other contexts (11). Systems biologists are particularly interested in seeing the high-level structure of biological pathways.

Discovering high-level links across pathways is important in a number of domain contexts. In cancer research, drugs can often influence many different gene products across several pathways. In some cases, existing drugs can be found to affect pathways related to several different therapeutic conditions. Understanding how several different gene products and pathways are affected by a drug is crucial to this research.

Task 4 Discover processes that are upstream or downstream of a given set of entities, and vice versa.

In a similar vein, researchers may be interested *many-to-many* or *many-to-one* relationships, called *compound* relationships, where an analysis is performed on a (potentially large) set of pathway entities.

This task is particularly relevant to researchers who work with high-throughput *omics* data involving hundreds or thousands of genes or gene products.

It is also important discovering downstream pathways that are common to a given set of proteins generated from mass spectrometry data. If, for example, a given set of proteins are all related to a certain cellular process via *downstream* connections, it may suggest to a researcher that the process in question is occurring within a given biological context. The process of discovering cellular processes related to a set of proteins (or other entities) can go in either direction; that is, a researcher may be interested in finding *processes* that are related to a given set of *entities*, or they may be interested in finding *entities* related to a given set of *processes*.

Task 5 Understand the hierarchial structure of protein complexes.

Compound relationships are also seen when examining protein complexes, which are represented as nested hierarchies of proteins. These hierarchies are particularly important to stem cell research.

Task 6 Analyze high-throughput *omics* data.

As mentioned in the discussion of Task 4, a researcher may want to analyze a large collection of hundreds or thousands of proteins (or other gene products). High-throughput *omics* data has increasingly become an essential part of modern biology (12). Researchers might need to incorporate large lists of entites in their analyses.

Task 7 Discover potential "causal" mechanisms of up or down-regulation within a set of entities.

Causal networks are important in the analysis of large-scale gene expression data. Ideally, a causal network would detect the likely regulators of a set of genes that are observed to be up-regulated or down-regulated in a particular setting (13) (14). This task is similar to Task 2, but requires a more complex algorithmic approach (such as in (14)) that can *detect* entities of interest based on their regulatory relationships within a pathway.

Task 8 Simulate the effects of activation, inhibition, or knockout within a pathway.

Simulation would be an especially useful tool to researchers, Indeed, the ability to simulate the effects of activation, inhibition, or *knockout* (removal of a gene or gene product) as being especially powerful in a number of analytic settings.

Task 9 Incorporate new entities or relationships into an existing pathway and examine their effects.

When exploring unknown relationships within pathways related to certain types of cancer, the ability to add new relationships to an existing pathway, and to examine the regulatory effects of the novel relationship, would be especially useful.

Task 10 Incorporate experimental results with pathway data.

Biological analyses are rarely focused solely on the structure and connectivity of pathway networks, and experimental data is at the crux of most biological studies. Pathway data alone would be essentially useless without the ability to incorporate the results of their experiments. For example, cancer research often involves comparing gene expression profiles between cancerous and healthy cells (15).

Task 11 Curate, edit, and "debug" pathway data files, or construct user-defined pathways from scratch.

Certain tasks are related to the curation, maintenance, and understanding of pathway data. For instance, would be useful being able to *debug* potentially flawed data, or create "personalized" pathways that only include a user-determined subset of entities and relationships, that can comes from different pathways and match a list of user-defined queries.

Task 12 Reveal levels of confidence or uncertainty associated with relationships within a pathway.

A relationship between two participants might be associated with a probability. This probability represents the degree of confidence that the interaction manifests and it is a relevant information to display. For example, each relationship within a BioPAX file is usually associated with a publication that provides evidence for its existence. Thus, some relationships may not be widely accepted within the scientific community, while others may have more robust empirical support.

3.2 Requirements

TABLE II

LIST OF REQUIREMENTS TO SUPPORT THE TASKS *T1-11*

code	description
R1	Effective labeling of pathway entities.
R2	Show individual meta-data on demand.
R3	Effective display of relationship type and directionality.
R4	Search, filter, and select.
R5	Network measures and complex queries.
R6	Simulation.
R7	Display hierarchical structures.
R8	Display compound relationships.
R9	Incorporate multiple interactive layouts.
R10	Analyses of multiple pathways.
R11	Incorporate multiple views.
R12	Visual incorporation of meta-data.
R13	Pathway curation, creation, and notation.
R14	Incorporate online databases.
R15	High-throughput data processing.

We have listed in the previos section the essential tasks necessary to perform useful analyses of biological pathway data. We now use these tasks to identify the requirements of a platform for effective pathway analysis. Table II lists these requirements and their relations to tasks mentioned in the previous section.

Requirement 1 Effective labeling of pathway entities.

The display of pathway entities is obviously essential to any analysis of pathway data. A particular challenge is the labeling of pathway entities and relationships. Biological entities such as protein complexes can be especially long, making the placement of labels within, for example, a node-link diagram especially challenging.

Requirement 2 Show individual meta-data on demand.

Any one entity or relationship within a pathway may be associated with a considerable amount of meta-data, such as chemical composition,

Requirement 3 Effective display of relationship type and directionality.

While gene regulatory networks may include only one or two types of relationship between entities, metabolic pathways can include a wide variety of relationship type. For example, a biochemical reaction

(e.g. in BioPAX pathways) is a relationship between multiple entities that could represent a variety of biological processes, such as *regulation*, *binding*, *dissociation*, *transport*, *activation*, *inactivation*, *phosphorylation*, *degradation*, etc. The number of relationship types makes visual encoding difficult — encoding relationship type with color, for example, would be visually overwhelming.

In metabolic networks, relationships are also directed, and therefore an effective visual tool must encode relationship directionality. Arrowheads may occlude each other and introduce visual clutter as the number of relationships increases. Certain visual techniques have incorporated color gradients or opacity to indicate direction, although this can also require additional visual parsing by the viewer.

Requirement 4 Search, filter, and select.

Given a potentially large amount of pathway data, the ability to search for entities and to visually filter pathway data is essential. Ideally, a textual search (or filter) would interactively and instantaneously display results. Users should also be able to easily create selections of entities and relationships by brushing or clicking, and selections should remain consistent across linked views (see Requirement 11).

This requirement could also be a complement to Requirement 3, as users may wish to filter or highlight a certain type of relationship or relationships. This requirement helps enable Tasks 8, 2, and 7, where an analyst may wish to highlight specific relationships of *activation*, *inhibition*, *upregulation*, or *downregulation*.

Requirement 5 Network measures and complex queries.

The ability to perform complex network queries is essential to many of the tasks listed in the previous section, including Tasks 8, 2, 4, and 7. As framed in the task descriptions, network measures could include shortest paths between two nodes (two entities), as well as complex queries aimed at identifying pathways or processes that are common to a selected set of entities.

Requirement 6 Simulation.

The ability to simulate particular events within a pathway would be particularly powerful, as discussed in relation to Task 7. Users could, for example, simulate activation, inhibition, or complete removal of a gene or gene product and observe the effects on the pathway.

Requirement 7 Display hierarchical structures.

Common pathway data formats such as BioPAX are inherently hierarchical. For example, pathways can be nested within other pathways, representing a biological hierarchy of low-level processes that act as components in high-level processes such as cellular replication. Task 3 specifically necessitates a way of viewing these hierarchical groupings, and an effective visualization would allow a user to interactively move between high-level and low-level views, e.g. by "collapsing" levels of the hierarchy.

Requirement 8 Display compound relationships.

Tasks 4 and 5 are both related to compound relationships, where many entities (e.g. proteins within a complex) are may be collectively connected to other entities (or other compound groupings). Representing these compound connections is not trivial. As mentioned earlier, the relationships themselves may be abstract, such as when entities and pathways are connected via the *nextStep* property in BioPAX data.

Requirement 9 Incorporate multiple interactive layouts.

One of the most important considerations of any pathway visualization tool are the layout mechanisms available to the user. Pathway layout algorithms must balance a number of factors. Contextual information within a pathway diagram is a popular solution. Indeed, many common pathway layouts include visual cues — such as membrane boundaries — indicating where certain entities are located within a cell. Including these contextual cues is useful to biologists who are accustomed to seeing illustrations of cellular structure.

On the other hand, representing cellular structure can be limiting. Abstract views of directed network data can reveal patterns and clusters that may not be apparent using other methods. Careful design of pathway layouts can also help mitigate labelling issues related to Requirement 1. Layouts can also harness more abstract patterns within pathway data, such as a *topology* of relationships, where *upstream* and *downstream* entities may be visually separated.

Ideally, layouts would also allow a user to interactively position nodes, and layout transitions would maintain object consistency (16).

Requirement 10 Analyses of multiple pathways.

A number of tasks (such as Tasks 1, 2, 4, and 7) inherently require the use of multiple pathways. In some cases, relationships may be found across "distant" pathways, while in other cases pathways may be merged or connected through network queries, as in Requirement 5.

Requirement 11 Incorporate multiple views.

Given the difficulty of designing an "ideal" layout, and with the requirement that a tool be able to analyze multiple pathways simultaneously, an effective visual tool will incorporate multiple linked views, which gives an analyst several "views" of a given pathway.

Requirement 12 Visual incorporation of meta-data.

Essential to many biological analyses is the incorporation of experimental meta-data, as mentioned in the discussion of Task 10. Meta-data may include gene expression levels or other experimental results related to specific entities. Careful integration of meta-data can enable effective analyses, but attention to design is crucial, as the potential for visual clutter is high.

Meta-data may also include visual representations of uncertainty, which would enable Task 12.

Requirement 13 Pathway curation, creation, and notation.

Task 11 leads directly to this requirement, which would allow an analyst to interactively manipulate and curate pathway data.

Requirement 14 Incorporate online databases.

The immensely large and highly connected nature of biological data makes manually loading a truly comprehensive dataset into an offline application difficult or impossible, meaning that any offline application must be limited to analyses of "local" datasets.

With a growing set of publicly-accessible biological databases with comprehensive APIs, an effective pathway visualization platform will allow an analyst to perform queries on these databases.

Requirement 15 High-throughput data processing.

Task 6 specifically necessitates the ability to analyze large sets of data at once. High-throughput *omics* data involves that analysis of hundreds or thousands of genes or gene producs, and an especially powerful pathway analytics platform would allow network queries and visual analytics to be performed across large sets of entities simultaneously.

CHAPTER 4

RELATED WORK

In this chapter we first introduce in Section 4.1 the traditional visual techniques commonly used to represents the types of data stored in a pathway, which are relational data, hierarchical data and grouping information. Section 4.2 presents an overview of different pathway visualization tools, which implements the mentioned visual techniques. Section 4.3 lists the future challenges of this field.

4.1 Visual Encodings and Representations

Having reviewed the tasks and requirements relevant to effective pathway visualizations, we now present an overview of different visualization techniques. We organize the techniques into three groups: graph, sets and hierarchies. Some of the methods discussed in this section are abstract visual representations that mat not yet be implemented in existing tools, but that fulfill requirements highlighted in the previous chapter.

4.1.1 Graph visualizations

Any pathway data can be represented in the abstract as a graph of vertices (such as genes, proteins, or complexes) and edges (such as biochemical reactions). The most common visual representation of a graph is a *node-link diagram*, where vertices are represented with shapes, such as circles or rectangles, and edges — or links — are line-segments that connect nodes together. In node-link representations, different shapes and colors can be used to encode additional information such as a node attribute or link direction.

A crucial aspect of any node-link diagram is the choice of node layout. Node-link representations fall into two broad categories: Diagrams that are automatically determined by an algorithm and diagrams where nodes are placed manually by human curators (such as KEGG (10) diagrams).

4.1.1.1 Human curated node-link diagrams

Publications in molecular biology frequently present biological pathways with human-generated figures. Human creators have the flexibility to arrange visual elements in ways that make representations human readable and this can allow authors to efficiently encode large volumes of complex information. The human-generated nature of these diagrams allows this complex information to be encoded through clear spatial layouts, organizing the pathway in a meaningful way.

The hand-made approach to pathway diagram creation has been replicated digitally in public databases, such as the Kyoto Encyclopedia of Genes and Genomes (10). The KEGG database is a very popular resource for human-generated pathway diagrams, and many tools incorporate KEGG diagrams in their visualizations, often in tandem with additional interactive functionality. This widely-used database allows for clear communication and dissemination of established pathways. Human-curated representations may also allow for interactive re-arrangement by the user.

While human-curated diagrams preserve the layout and presentation of pathway information, they have several drawbacks. Creating, updating, or modifying figures is a labor-intensive process. Any new data cannot be automatically applied to existing figures. Moreover, these human generated figures do not easily scale to large, complex pathways. As a pathway increases in size and complexity, human effort also increases considerably. These limitations are major problems in a research community that is continually generating and updating molecular pathway data.

4.1.1.2 Automatically generated node-link diagrams

Several algorithms exist which automatically produce interactive pathway visualizations from structured pathway data, rather than relying on layouts generated by hand. The visualizations produced by these tools mimic the style and visual encoding of human-generated figures in an attempt to to communicate complex network information efficiently. As the layout is computed algorithmically, it is easily updated with new data without requiring extra human effort as networks increase in size. Different automatic layout techniques may be designed to optimize some visual parameter, such the efficient use of a given space, a reduction in edge overlap, the recognition of certain clusters or patterns, or the visual separation of certain types of nodes or links. Clearly, a layout algorithm can aim to optimize only a few of these metrics.

A *Force-directed Layout* is a popular technique that computes the position of nodes with a relaxing layout algorithm which models nodes within the network as physical entities. This algorithm results in a graphical representation that tends to organize the graph based on the density of the connections within clusters of nodes. Even when force-directed algorithm does not directly aim to minimize edge crossings (some algorithms do), the resulting visual representation tends to bring highly connected groups of nodes close together, which in turn results in a reduction in overall edge overlap.

When graphs become large and dense, and the number of nodes and connections increases, edge overlap becomes more prevalent and often cannot be avoided. Layout techniques that aim to mitigate edge overlap include the *Circular Layout*, which tends to scale well when nodes are carefully ordered. However, the resulting representation tends to show overall clustering of connections, and detailed views of individual relationships are often occluded.

Other techniques aim to reduce edge crossings by duplicating nodes. However, as pointed out by Bourqui et al. (17), analysis tasks that rely on an understanding of the graph topology might be hampered by the adoption of node-duplication techniques.

Edge overlap and general visual clutter can also be reduced through *node reduction.* Kimelman et al. propose three possible approaches (18) to implement node reduction: *Ghosting*, *hiding* and *grouping*. *Ghosting* de-emphasizes nodes, *Hiding* completely removes a selection of nodes and *Grouping* joins nodes in new super-node representation.

Grouping nodes is frequently used in pathway visualizations, and pertains directly to Requirement 7. For instance, a pathway might be interactively collapsed into a single node.

Edge Bundling is a technique that can complement other graph layout algorithms. *Edge Bundling* is not itself a graph layout technique, but a technique for drawing edges between nodes in an attempt to reduce clutter and improve readability, and is especially useful in cases of excessive edge overlap (19). The technique trades readability for a lack of detail — it may be difficult or impossible to discern individual connections when edge bundling is used.

Another technique that is meant to enhance graph layout techniques is *fisheye distortion*, which operates as an interactive "lens," enhancing details over a user-defined portion of a graph, enabling better inspection of the nodes and interconnections in a given area.

Basic node-link diagrams are usually not comprehensive enough to represent an entire pathway data structure. As mentioned in the discussion of Requirement 8, a pathway often contains compound nodes and relationships that group many entities. A biochemical reaction can be characterized as a *hyperedge*, usually connecting two or more inputs to two or more outputs. Moreover, the links between nodes might

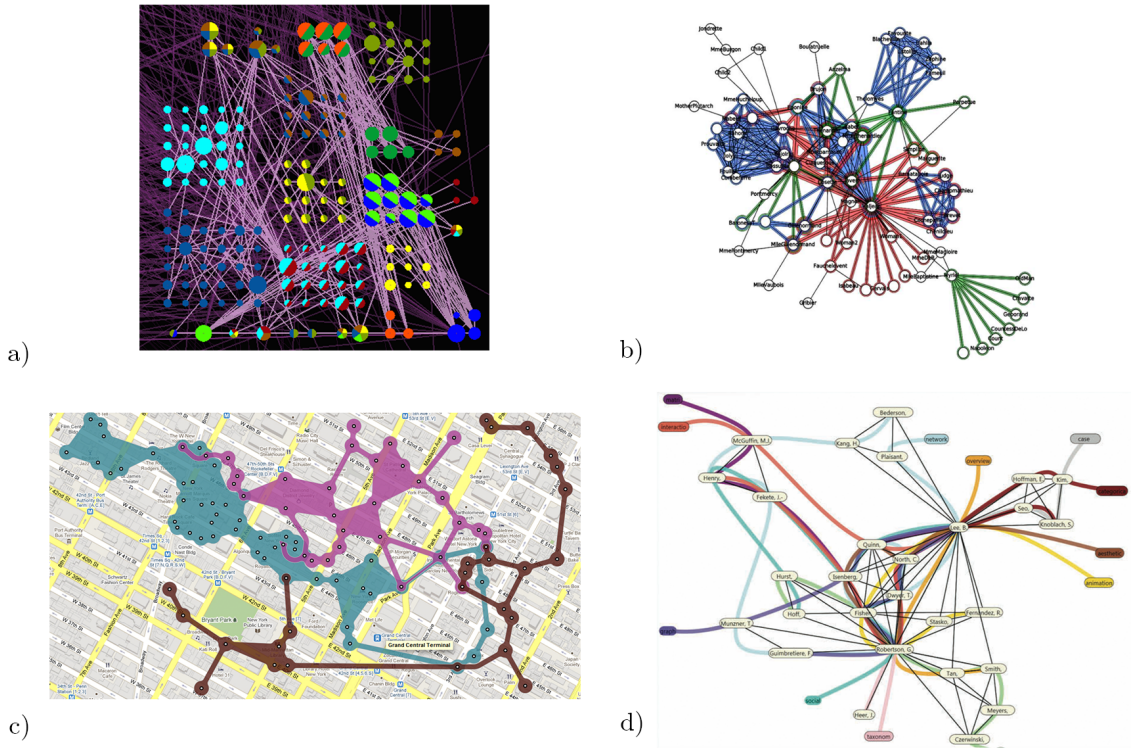

Figure 1. The image represents some techniques for representing categories on graph structures: **a)** The node is divided in multiple fans colored accordingly to the category **b)** Nodes and edges have colored contours **c)** KelpFusion **d)** LineSets

convey a wide variety of meanings, as discussed in Requirement 3. In the following sections we discuss alternative visualization techniques that can account for these complicated relationships in various ways.

4.1.2 Visualizing Categories on Node-Link Diagrams

Set visualization techniques can be used to encode the *many-to-one* relationships found in most biological pathways. These compound relationships are important to Requirement 8, and set visualizations can be combined with node-link approaches in an effective visualization platform.

A range of techniques have been introduced to represent group information within node-link diagrams. Some techniques assign different colors to nodes within different groups. Glyphs can used to indicate nodes that belong to multiple categories; Ware et al. (20) defines glyphs as "graphical objects designed to convey multiple data value". One example of a technique which makes use of glyphs is described by Itoh et al. (21), which subdivides nodes in multiple fans, similar to a pie chart, giving each fan a distinctive color and size to indicate that the node belongs to multiple categories and to indicate the strength of that membership (22).

Other techniques have been designed specifically to visualize the set membership of elements, but can be adapted to represent categories on node-link diagrams. Traditional *Euler Diagrams* and other techniques that make use of convex boundaries to group elements together (23) can lead to visually confusing representations if the data contains multiple set intersections that cause the elements to not fit within the boundaries of the set. *GMap*, introduced by Gansner et al. (24), is another method for overlaying group information over node-link diagrams. It uses a map metaphor to group members into "countries", "seas", and "lakes". A limitation of this technique is the inability to visualize multiple intersections. For this reason a range of alternative visualization techniques have been developed to display elements belonging to more than one set.

Bubble Sets (25), *LineSets* (26), and *KelpFusion* (27) are techniques that address the challenge of representing set membership even when there are multiple intersections. Each of these techniques is designed, or can be adapted, to be overlaid on top of existing visualizations (for example, on top of geographical maps). *Bubble Sets* displays set relations using isocountours, which can produce problem-

atic representations when an element belongs to many multiple sets, in some cases causing elements to appear included in the wrong sets.

A technique by Alper et al. called *LineSets* (26) represents sets as smooth curves and uses distinct colors to indicate set membership. All the nodes which belong to the same set are connected by curved lines, and nodes which belong to multiple sets are located at the intersection of multiple lines. Identifying all the nodes that belongs to a given set can be done by finding the respective curve and visually following all the nodes placed on the line, similar to finding which subway stops are on a particular subway line. Intersection between sets can be identified by looking for the nodes placed on curve intersections. This solution generally offers better readability when multiple sets overlap than *BubbleSets*.

KelpFusion (27) uses continuous boundaries made by lines and hulls. The visual appearance is generally comparable or better than *LineSets*, but it strongly depends on the spatial arrangements of elements. Hulls are used to group together elements which are both spatially close and belong to the same set. They are less effective when applied to node-link diagrams that don't have an intrinsic geospatial meaning.

Other solutions are explicitly designed for both the representation of sets and for the identification of patterns in a graph representation. Antoine Lambert et al. (28), for example, propose a solution which assigns a color to every set and simultaneously colors nodes according to the set to which they belong. When a node belongs to multiple sets, multiple colored contours are used. A similar technique, which also uses node coloring, is adopted by Takayuki Itoh et al. (21). In this solution, nodes which belong to multiple categories are subdivided in multiple "fans." With these techniques, the visual cost – in terms of clutter and color overlap – increases with the number of categories in the data.

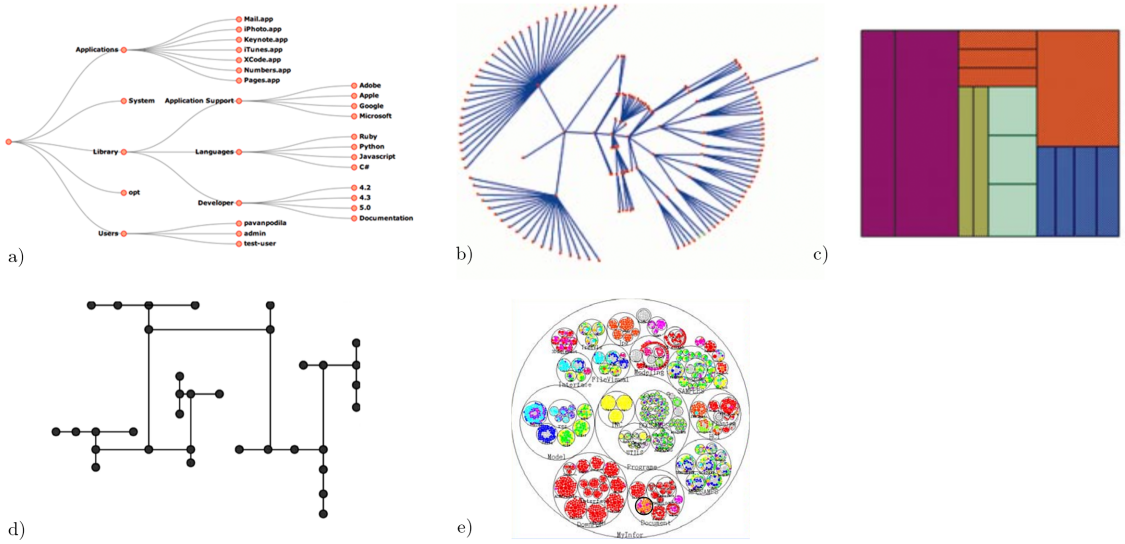

Figure 2. The image represents the most common technique for representing tree structures: **a)** A traditional tree diagram **b)** tree diagram with a circular layout **c)** Tree Map **d)** H tree **e)** Circle Packing

Figure 1 illustrates some of the techniques detailed in this section.

4.1.3 Hierarchical Techniques

As mentioned earlier in the discussion of Requirement 7, pathways are often deeply hierarchical. As mentioned in Requirement 7, a pathway usually contains multiple sub-pathways, and a protein complex can represent a are often nested hierarchy of constituent proteins. Conventional representations of hierarchical structures with tree diagrams are prone to visual clutter and poor readability as the amount of leaves and the depth of the tree increases. For this reason, different visualization techniques have been developed to efficiently represent complex hierarchical structures.

A *Radial View* is a tree diagram that uses a circular layout in an attempt to more efficient use of a two-dimensional space. With deeply nested trees, a *Radial View* can suffer from the same issues seen with more "traditional" tree diagrams. An alternative representation is seen in the *H-tree* layout, which positions nodes by using only orthogonal and perpendicular links, which helps eliminate edge crossings and optimize space.

Circle Packing (29) is a technique which presents hierarchical elements as nested circles. This technique provides an efficient overview of a tree structure, however it may be difficult to adapt to more complex and deeply-nested trees. In terms of biological pathways, circle packing layouts may be useful for the understanding of hierarchial structures such as proteins that are nested within complexes, as in Requirement 7. However, a pathway might contain complexes with hundreds of proteins and dozens of sub-complexes. Circle Packing layouts also present a challenge when it comes to readable labels (Requirement 1). Furthermore, understanding the path through a hierarchy from the root node to to a leaf element is challenging with a circle packing layout.

Another representation of hierarchical structure is the *Tree Map*. A tree map places components within rectangles with areas proportional to some given metric. This technique is popular in representing, for example, the memory usage in hard drive, but it may have only limited utility in representing biological pathway data. The spactial organization of a *Tree Map* enables an intuitive understanding of the organization of a structure, but is very difficult to parse when a tree is more than two or three levels deep.

4.1.3 compares "traditional" tree diagrams to alternative techniques.

4.1.4 Matrix-Based Techniques

Matrix representations act as an alternative to node-link diagrams in representing graphs. Matrix representations excel in their ability to show densely connected graphs while minimizing visual clutter.

Matrix representations present a significant challenge for pathway data, as paths connecting more than two entities can be very difficult to follow. However, matrix views may be an important alternative view of very densely connected biological pathways, such as with large gene regulatory networks.

4.2 Tools

TABLE III. COMPARISON BETWEEN THE REQUIREMENTS IMPLEMENTED IN POPULAR TOOLS FOR PATHWAY VISUALIZATION

Tools	R1	R2	R3	R4	R5	R6	R7	R8	R8	R9	R10	R11	R12	R13	R14	R15
Entourage	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓			✓	✓
Reactome	✓	✓	✓	✓		✓	✓	✓	✓		✓		✓		✓	
VisAnt	✓	✓	✓		✓		✓	✓		✓	✓	✓			✓	
VidaPad	✓	✓	✓		✓			✓			✓				✓	
ChiBe	✓	✓	✓	✓			✓	✓		✓	✓	✓				
BioFabric	✓									✓		✓				
MetaViz	✓	✓	✓					✓		✓		✓				

In this section we examine and compare several existing tools for the visualization of biological pathways. Particular attention is given to which requirements detailed in Section 3.2 are implemented in each tool. Table V summarizes these relationships.

Broadly speaking, current tools make use of one of three primary visual representations: node-link diagrams, node-link diagrams with compound nodes, and adjacency matrices. We look at *Entourage* (15), *Reactome Pathway Browser* (30), *VisAnt* (31), *MetaViz* (17) and *VitaPad* (32), each of which leverages a traditional node-link diagram.

ChiBE (33) is notable for being one of the first pathway visualization tools to employ *compound nodes*, which can contain either the proteins contained within a complex, or can indicate the cellular location of many pathway compontents. *ChiBE* provides a basic pannable and zoomable view of a pathway, and includes the abillity to display node details on demand through user interaction. Although certain limited analytical functions are available, *ChiBE* is more well-suited to the construction or *curation* of pathways, rather than to insightful pathway analysis.

Reactome Pathway Browser is a tool for visualizing pathway diagrams included in the *Reactome* database. It offers basic navigation and limited interactivity. It presents a very fixed layout and it does not enable the layering of different information levels in the same representation. It makes use of side panels to enable the inspection of complexes structure and analysis data. The *Reactome Pathway Browser* does not enable the simultaneous visualization of multiple pathways but, given a component, it allows the user to navigate through its related pathways.

Entourage (15) – also a part of *Caleydo* (34) – allows for the display of both pharmacological and genomic data related to selected gene products that span multiple pathways. Combining functional data with multiple pathway views makes *Entourage* a powerful tool for molecular genomics research.

Entourage (15) is a component of the *Caleydo* (34) framework that displays relationships between proteins and other gene products across multiple pathways. To this end it employs a novel algorithm that extracts *contextual pathways* from larger pathways. Contextual subsets show small portions of a pathway that contain selected proteins of interest. Because the display size of each contextual subset is relatively small, many can be displayed at once, which allows *Entourage* to provide linked views of several pathways that share certain proteins of interest (see Figure 3). Multiple pathways and sub-

pathways are represented in isolated views, which preserves the pathway structure but introduces node redundancy. Links connect proteins or complexes that are shared across pathways, and unless requested by user interaction, these links are reduced to partial "stubs" in order to reduce visual clutter. The *Bubble Set* technique is used to highlight selected pathways and visual links are used to connect nodes that are actually the same node.

Entourage also allows for the display of both pharmacological and genomic data related to selected gene products that span multiple pathways. Combining functional data with multiple pathway views makes *Entourage* a powerful tool for molecular genomics research.

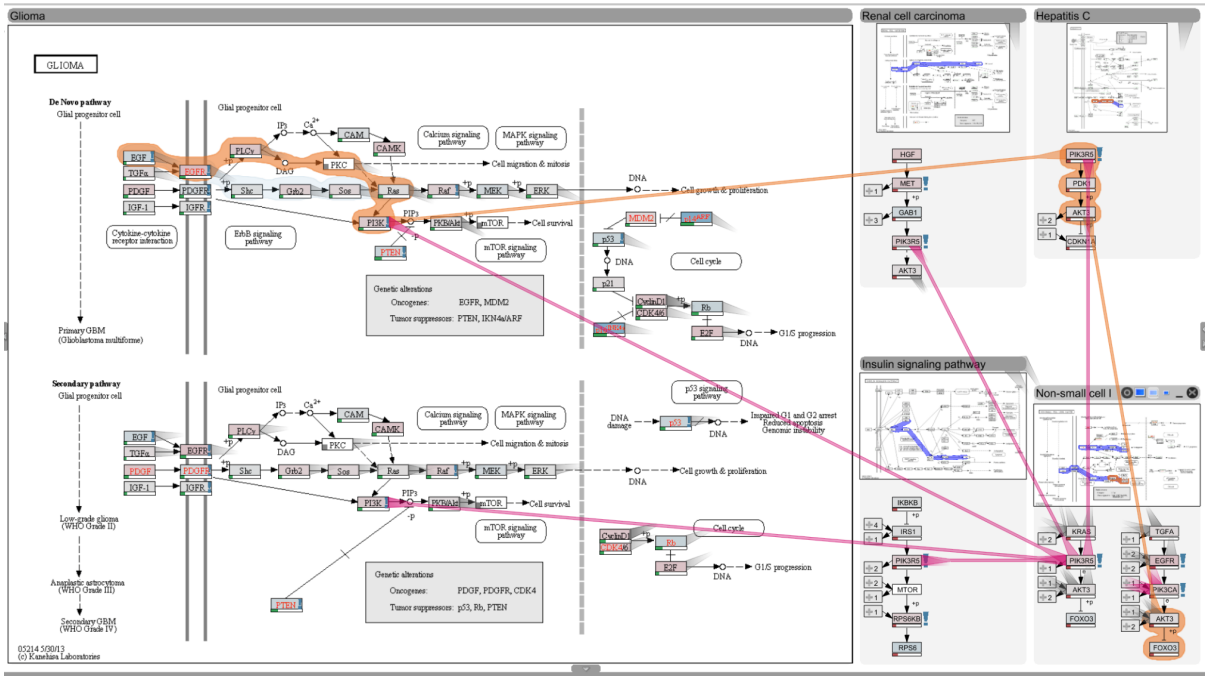

Figure 3. A view of *Entourage* showing links between a larger "focus pathway" and several "context pathways."

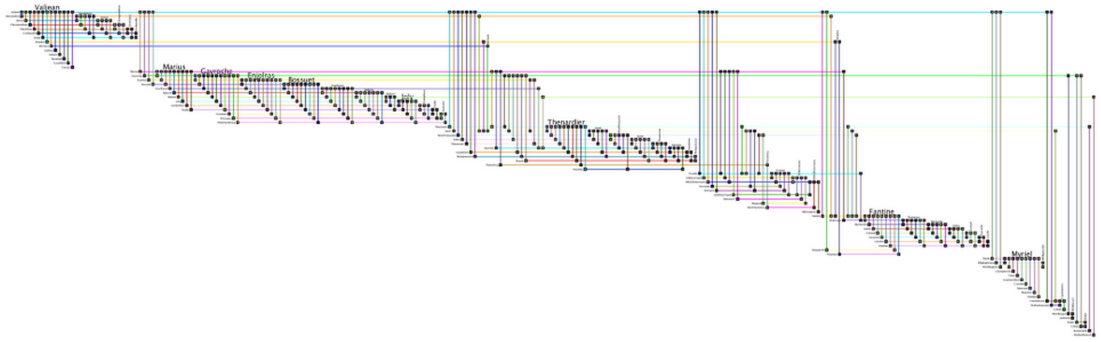

Figure 4. Visualizing hundreds of nodes and links with Biofabric

Furthermore, both *Entourage* and *Reactome Pathway Browser* enable the user to integrate experimental data. *Entourage* prefers to visualize the experiment information as side panels, whereas *Reactome* combine the path visualization with additional visual encoding to show gene expression and other values (Figure 5)

Other tools implement functionalities that enable the representation of multiple pathways in the same view, such as *MetaViz*. *MetaViz* visualizes multiple metabolic pathways simultaneously, which allows topological analysis and avoids the duplication of biological components. This method offers different ways to present metabolisms depending on the pathway that the user wishes to focus on. The desired pathway will be the only one to follow a representation which flows from the top to the bottom accordingly the topological ordering of components.

BioFabric also enables the exploration of large biological networks composed by multiple pathways without replicating nodes. This somewhat unconventional tool is meant to visualize extremely large

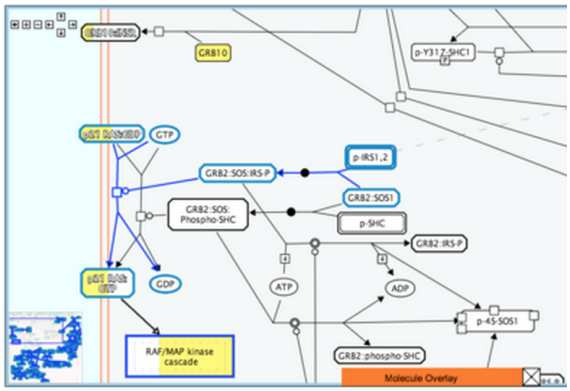

Figure 5. Visualizing different gene expression with Reactome Pathway Browser

networks composed by up to thousands of nodes. However, its design is not specifically conceived for pathway visualization and then it does not implements any pathways-specific task. However, this tools can import SIF files which are commonly used for pathways visualization. Figure 4 represents a network with hundreds of nodes and links. Elements are represented with horizontal lines, links with vertical lines. Altough this tools is considerably less intuitive than node-link diagram, it enable the representation of large network without losing deatils.

VisAnt (31) enables a wide range of exploratory questions and topology tasks. The tasks it allows include the search of the shortest paths between two nodes (35) and the individuation of dense nodes. The positions of the nodes in the graph are computed with a relaxing layout algorithm which models the network as a physical entities. This algorithm builds a graphical representation that organize the graph by density of the connections between cluster of nodes. Diagrams can include abstract nodes that extend beyond the typical components found in a biological pathway. Diseases, genes, therapies, and drugs are presented together in the same graph with more “typical” pathway components such as

proteins and complexes. *VisAnt* permits a dynamic exploration of a biological network composed by multiple pathways. However, if more than one pathway includes the same node, multiple instances of the node will be present in the representation. Furthermore, layouts are restricted to either force-directed or hub-spoke algorithms.

Although in the majority of available tools the visual encoding of a biological network remains limited to the conventional graphs representations, pathways analysis tools often decorate node-link representations with experimental data or side information. *VitaPad* (32) allows the user to incorporate microarray data into pathways and offers a flexible visualization.

4.3 Future Challenges for Pathways Visualization Tools

Design a visualization for biological pathways hides a variety of challenges. Those challenges can be broken down in visual task, for instance representing large node-link diagram networks, or complex hierarchical structures, which already well-know to the information visualization community.

A variety of solution have been proposed, some of which has been implemented in pathways visualization tool. However, the challenge to design a tool that implements all the requested feature is still open. We identified three different “frontlines of improvements”.

4.3.1 User Interaction and Better Visual Encoding

The first category of improvements includes small improvements in visual encoding and in adding user interactivity. Indeed, some tools still present fixed layout and implement a limited interactivity. Certainly tools might improve in the near future by implementing more well-designed user interactions and functionality. However, the majority of pathway representation is still bounded to the conventional node-link representation.The visual encoding of node-link diagrams can be augmented with additional

technique, such as the case of bubble set layered on a graph to provide information about categories. Furthermore, techniques such as *edge bundling* or *fisheye distortion* that has proved to be easily embedded in graph representation should be more in pathway visualization tools.

4.3.2 Data Integration

Pathways data has been collected in public accessible database. KEGG and Reactome offers very basic tools for the visualization of the pathway. Usually the bioinformatician needs to merge pathway information from different source in order to get an higher quality pathway. Some tools enable the user to import the standard formats which can exported from these database or to directly copy the link to the resource. However, in the future tool should enable the exploration of all the pathway knowledge, interactively searching for participants and assembling the portion of the biological network of interest. High quality data representation must be supported by high quality data.

4.3.3 Experiment New Techniques

The visual representation of pathways has been for decades paper-oriented, where the representation were the user interaction were completely absent and the representation was limited to the traditional notation for node-link diagrams. The growth of the data to visualize and the support of computer has introduced new requirements and opened doors to new visual solution. However, only a small amount of the techniques detailed in Section 4.1 have been implemented in pathway visualization tools. This can be explained with the propensity to persist with the traditional visual encoding to simplify the adoption of the tool, but this tendency should be stopped. The adoption of new representation from the important pathways visualization plataform might persuade the community to a change.

4.3.4 Abstract Representations

Nearly every tool in this review uses some type of node-link diagram. The few exceptions are matrix based (*BioFabric* and *Compressed Adjacency Matrices*) but they can only be used for very specific types of networks, and they do not provide solutions relevant to pathway data specifically, such as discovering paths between two proteins of interest. Further work should be done to explore the possible benefits of abstract representations, as almost every tool in this review relies on automatic or human-curated node-link diagrams.

4.3.5 Visualizing Uncertainty

Especially considering domain expert feedback, tools generally do not attempt to visualize the "uncertainty" behind a connection in a pathway, as expressed by the first domain expert. This is a challenging task, as even the definition of "uncertainty" may be difficult to operationalize. However, data formats such as *BioPAX* do have robust support for citations, allowing published references to be connected to entities and relationships within a pathway. A tool that could effectively encode "uncertainty data" into a visualization may be very valuable to systems researchers who work with the results of hundereds or thousands of separate publications.

4.3.6 Pathway Logic

Pathway Logic is an approach to create a model and to perform analysis of biological processes with rewriting logic (36). Pathway Logic is a field in expansion which is currently hampered by two factors: the high computational efforts required for some tasks, and the necessity to provide formalization of every details form the chemical reaction to the pathway structure.

Talcott in (36) explores the potential to use *Maude*, a state of the art implementation of rewriting logic (37) for executing a number of tasks on a pathway structure. For instance it is possible to formulate and submit high-level queries to find pathways, for example, activating one protein, or compare two pathway. Moreover it is possible to find knockoutsproteins, which prevents reaching a state of interest, and explore step-by-step the network of interconnections according to a set of rules.

There are also examples of models of the evolution of a biological pathway by using *Petri-Nets* (38).

4.4 Final Remarks on Related Work

While a wide variety of pathway visualization tools exist, there is still plenty of room for innovative tool development. Many tools tend to greatly overlap each other with respect to the analytical tasks available, and attempts to address the most challenging aspects of pathway data analysis are few and far between. With the increasing popularity of web-based tools that integrate multiple databases and reduce accessibility costs, there is great potential for the development of innovative and useful tools for complex pathway analytics.

CHAPTER 5

BRANCHINGSETS

Node-link diagrams can be an effective way to visually represent data elements and the relationships between them. Traditional node-link diagrams are usually augmented with additional information layers. Different shapes and colors are often employed to differentiate between types of nodes, and links are encoded with different strokes, widths, or colors in order to indicate the different types of connections between nodes. However, for many data sets, the type of element and the connectivity between elements often is not the only relevant information. It can also be important to visualize the category or categories that each node or link is a member of. As a simple example, it could be meaningful to indicate the original source of particular elements when multiple data sets are merged together; nodes from different datasets could be grouped into different categories and visually distinguished based upon their origin.

A more complex, real-world example comes from the domain of systems biology. A main task for biologists is to analyze signaling pathways in order to understand the interactions between cellular components. These complex pathway structures may be pieced together from multiple experiments drawn from various publications, and it can be important for the biologist to have a clear idea of the provenance of the various elements and relationships in the pathway (39).

Dynamic graphs are another example of a node-link visualization that may require a clear representation of which set links and nodes are members of. A user might be interested in which links and nodes appear, disappear, or remain over a sequence of time (40). A particular range of time could be

represented by a different category, and a link or node could be part of this set if it overlapped that span of time.

Different solutions have been proposed to provide a visual clue of the membership of items such as the traditional *Venn Diagrams*, *KelpFusion* (27), *Bubble Sets* (25), and *LineSets* (26). In particular, *Bubble Sets* and *LineSets* are intended to be used to provide additional information to node-link diagrams. Inspired by these approaches, we introduce *BranchingSets*, a novel technique that also aims to show the set membership of elements within node-link diagrams.

Our work is inspired by the results of *LineSets* and by some of the visual solutions included in the design of *KelpFusion*. Indeed, sparse *KelpFusion* representations on top of node-link diagram can produce results which looks similar to our technique. However, our visual encoding has some differences that makes both the implementation and representation clearer. Furthermore, our technique includes a set of user interactions which enable an intuitive and interactive exploration of the data.

Our purpose is to introduce a more effective technique for integrating categorical data with node-link diagrams, and to make it easier to perform visual analysis tasks involving the memberships of nodes and links.

The following section introduces our technique and in Section 5.2 we present two real-world applications as case studies for which *BranchingSets* has been successfully introduced. Section 5.3 concludes the chapter and delineates future research directions. The next chapter will describe how we leverage *BranchingSets* in the more complex domain of system biology.

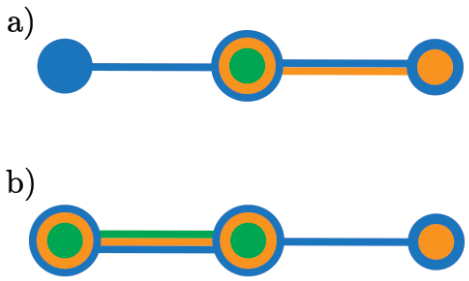

Figure 6. Two simple examples of node-link diagrams enhanced with BranchingSets.

5.1 Design Study of BranchingSets

BranchingSets is a technique which integrates set membership information with node-link diagrams, while at the same time minimizing visual clutter and occlusion. Our technique is able to categorize both nodes and links between nodes into multiple categories. Additionally, it enables a range of useful interactions. In this section we describe our design goals and the details of the technique in detail.

5.1.1 Design Goals

Allow users to identify the membership of each node and link (G1). We consider is important to represent the membership of nodes as well as representing the membership of links. As explained in Section 5.2, we have been motivated by two possible scenarios. The first one is when the representation combines relational information from multiple data sources, and it is necessary to identify the data sources which contains a given node or link. Another case is when comparing the same graph over time. We want to highlight the existence of nodes and link at a given time interval, as well as comparing the evolution of the graph over time.

Allow users to identify the sub-graph composed by all nodes and links which belong to a given category (G2). Given a category, the user should be able to efficiently understand which is the portion of the graph whose links and nodes belongs to that category. Indeed, frequent tasks require to locate all elements from a certain data source and understand the relationships between them.

Allow the users to identify the sub-graph resulting from the intersection or the complement of different categories (G3). Given two or more categories, the user should be able to identify which is the intersection sub-graph (i.e. the portion of the graph shared between the categories), and which is the complement sub-graph, that is the graph which belongs to a given category but not to the others. This kind of tasks are very common in a variety of applications

Utilize the visual representation of node-link diagrams (G4). We want to design a solution which does not introduce additional visual clutter to the node-link diagram and does not require to rearrange the layout of the nodes of the network for improving the visual appearance of the technique. Even in the case of node-link diagrams where the position of the nodes does not have geospatial relevance, the topology of the network usually entails some other important meanings, or is computed to minimize other metrics (e.g. minimize edge overlapping). Changing the position of nodes, for instance to group together nodes which belongs to the same category, might compromise the effectiveness of the representation.

Reduce the complexity of the visualization (G5) we want to enable the user to narrow down the representation to display only relevant information to his or her research, and to interactively explore a complex network reducing the visual clutter to the minimum. A clear visual encoding might not be sufficient to provide a readable visualization. In case of large relational datasets with many categories associated we need to leverage user interactions to provide a meaningful representation.

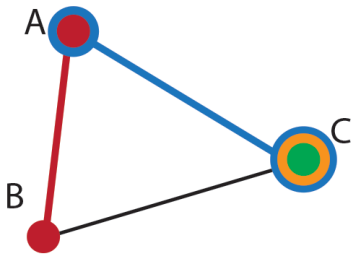

Figure 7. Coloring of the links in case of no information related to the membership. Since the node **B** and **C** does not have any category in common, the link which connect them is black and thinner

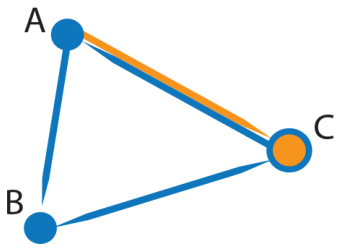

Figure 8. BranchingSets applied to a directed node-link graph. The link between **B** and **C** is bidirectional.

5.1.2 Visual Encoding

We assign a different color to each category, and additionally we define an ordering of the colors. As explained below, the order of colors is used to simplify the recognition of nodes which belongs to the same categories. If the categories have no intrinsic hierarchy, an ordering is arbitrary, but consistent.

Each node is assigned to the color of the category it belongs to. If a node belongs to multiple categories, we use all the respective colors, but give the node the primary color that comes first in the

colors ordering. A supplementary colored border is added to the node for every additional category, and the colors are assigned with respect to the colors ordering previously defined. This solution, while not original, creates a visual correspondence between nodes which belong to the same categories. Indeed, nodes which are member of exactly the same categories will present an identical sequence of colors; and nodes which share only a subset of categories will have the visual advantage of presenting the shared colors in the same order. Figure 6 shows an example of two networks with three nodes which belong to different combinations of categories.

BranchingSets also shows group information associated with links. Similarly to the strategy designed for visualizing nodes memberships, links are colored with the corresponding color of the category. In case that the same link belongs to multiple categories, multiple lines are placed side by side, one for each color (Figure 6).

In contrast to *LineSets*, *Bubble Sets*, and *KelpFusion*, *BranchingSets* does not ensure that elements which belongs to the same category are visually connected. Figure 9 shows a case where a node is visually disconnected from other nodes of the same category.

5.1.3 Showing Category Information Only For Nodes

In the case of a dataset that does not include membership information for edges, the links are colored to assist the visual grouping of nodes that belong to the same category. A link between two nodes is displayed with a colored line segment for each category that the nodes have in common. If two nodes are connected with a link but they do not have any category in common, the line segment has a thinner stroke width and a black color (Figure 7). This convention does not ensure that all the nodes which

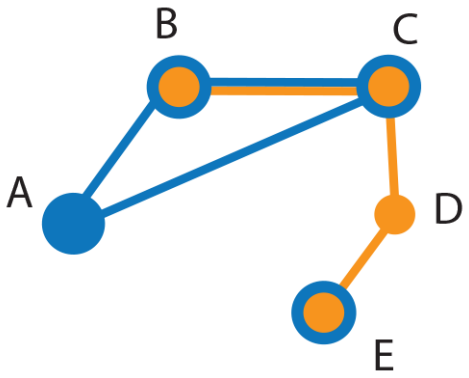

Figure 9. Node **E** belongs to the blue category but is visually disconnected from the other blue nodes.

belongs to the same category are connected with links of the corresponding color. It depends on the network topology and the distribution of categories on the nodes.

5.1.4 Directed Graphs

Figure 8 presents an example of how *BranchingSets* can be used to show the grouping of links and nodes in directed graphs. We do not use the traditional techniques for visualizing directed edges, such as tapered lines, gradients or arrows (41), as tapered lines reduce the surface of the link, which makes more difficult to identify its color. Any technique which makes use of color blending or an opacity gradient could similarly be difficult to use in conjunction with a color coding for visualizing categories. The thick tip of regular arrows introduces problems when multiple links must be placed side by side, as our technique does. For these reasons in our solution we have adopted sharp-pointed lines, medium-sized line segments with a sharp tip. In case of a bidirectional link the line segment has a sharp tip on both sides.

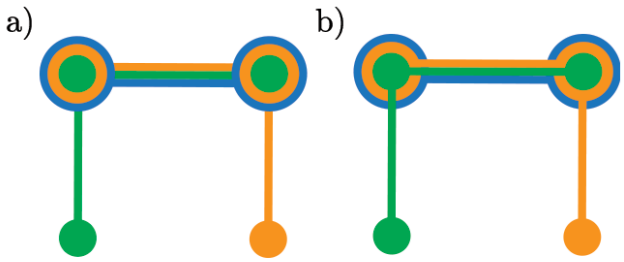

Figure 10. BranchingSets **(a)** without protruding links **(b)** with protruding links.

5.1.5 Protruding Links

By default, *BranchingSets* represents links with line segments beneath the nodes. This leads to the visual result that line segments "disappear" behind the circular shape of the node. An alternative is to assign links and nodes of the same color to the same graphical layer. Graphical layers are ordered according to the colors ordering, in a way that the first color is on the foreground. The resulting effect is that there is a continuity between the colored border of the node and the line segment. We have called this technique *protruding links*.

Figure 10 shows the same graph with and without *protruding links*. In the case of *protruding links* we believe that there is a stronger visual interconnection between nodes and links of the same color, and we consider that it might be helpful in tasks such as identifying sub-networks that belongs to the same category.

5.1.6 Interactions

We have enhanced *BranchingSets* with a number of user interactions for exploring, extending and inspecting a complex network. The interaction tasks, discussed below, include the following:

- Highlighting and labelling categories;
- Hiding and unhiding categories;
- Filtering by keyword;
- Expanding neighbours of a selected node;
- Navigating and rearranging the layout;
- Finding intermediate steps between two nodes.

5.1.6.1 Highlighting and Labelling Categories

By default all node labels are hidden. This improves the usability and avoids the inconvenience of too many labels that might overlap and clutter the representation. When the user hovers with the mouse pointer on a node, only the labels of the nodes which are included in the same category are displayed. Moreover, the category which includes the hovered component stands out because all the nodes and links which does not belong to that category change to a gray color. If a node which is included in more than one category is hovered, all the involved categories will be highlighted.

5.1.6.2 Hiding and Unhiding Categories

The user can dynamically explore the data by hiding or revealing different categories. A given node or link is visible if it belongs to at least one visible category, and it shows the memberships related only to visible categories. Figure 11 shows an example of a network with an increasingly number of visible categories.

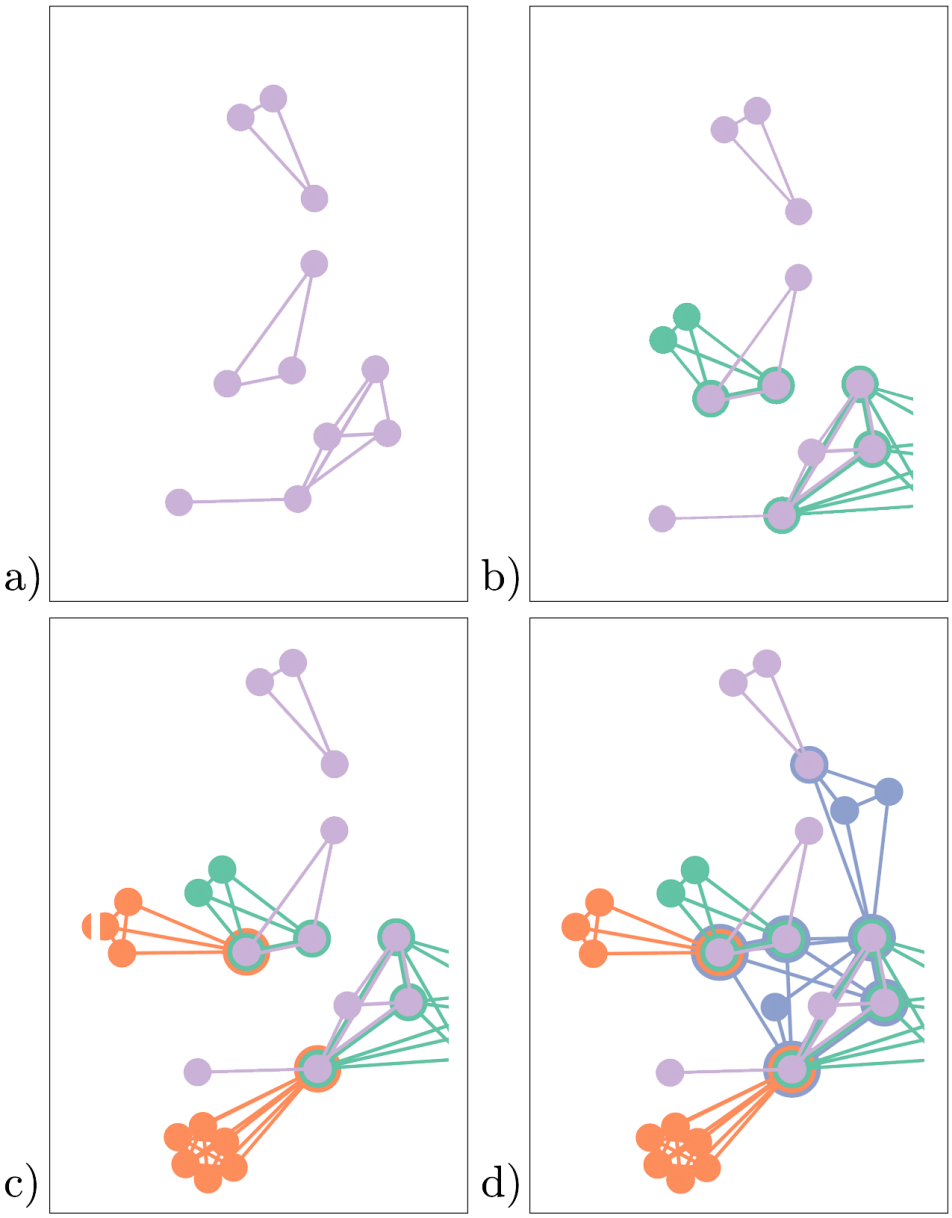

Figure 11. The same graph with different visible categories **(a)** only violet **(b)** violet and green **(c)** violet, green and orange **(d)** violet, green, orange and blue

5.1.6.3 Filtering by Keyword

We enable users to search for keyword by typing one or more words in a search field. The keywords can be added and removed from a list. The visualization will display only the nodes whose labels matches at least one of the searched keywords. The possibility to search for keywords and filter categories, together with the *expanding neighbours* described below, enables an efficient way to progressively explore large datasets.

5.1.6.4 Expanding Neighbours of a Selected Node

By clicking on a node the visualization will be expanded to display all the nodes connected to it. This user interaction enables the following workflow: the user searches for a set of nodes of interests, then he or she progressively explores the network of interconnections by revealing the neighbours.

5.1.6.5 Navigating and Rearranging the Layout

The visualization of large datasets can easily exceed the boundary of the screen. For this reason we enable the user to interactively zoom in, zoom out and pan across the viewport. Since the representation may lead to edges that intersect, the user can drag and drop any component to a preferred location to improve viewability.

5.1.6.6 Finding Intermediate Steps between Two Nodes

In addition to the *expanding neighbours* interaction, the user can expand the set of visible nodes and links in another manner. By dragging the mouse while holding the right button from one node to another, the visualization will update the visual representation with all the sequences of nodes and links that starting from the first node, ends with the latter. In directional graphs, only the sequences of links from the first node to the target node will be revealed. Some applications of this interaction could

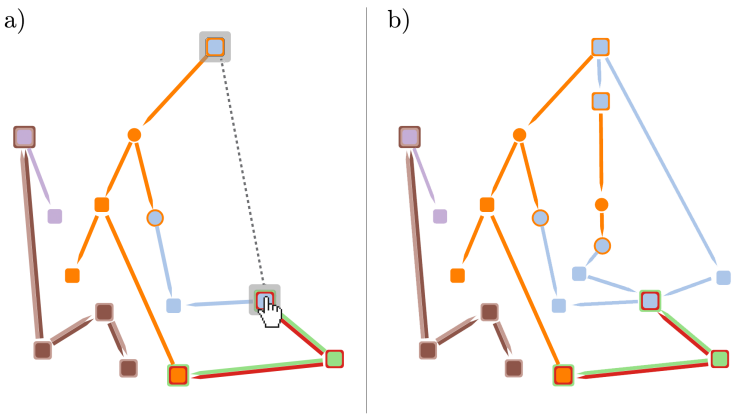

Figure 12. On the left, the user is dragging the mouse from one biological complex to another. On the right, the visualization is updated with all the connecting paths

include: revealing the biological elements between two proteins a biochemical pathway; showing the stages of an industrial process in a workflow diagram; or displaying the sequence of web-pages that can be visited to navigate from one page to another. Figure 12 shows an example of displaying intermediate steps applied to a directional graph.

5.2 Applications

In this section we present two application from different domains that utilize *BranchingSets*. The first example shows how *BranchingSets* can be effectively employed to visualize how a network changes over time. The second example is a demonstration of *BranchingSets* adapted to show geospatial data on a map. Furthermore, Chapter 7 introduces a pathway visualization tool which leverages the visual encoding of *BranchingSets* to represent the interconnections between pathways and integrates the user interaction detailed in this chapter to enable a dynamic exploration of the data.

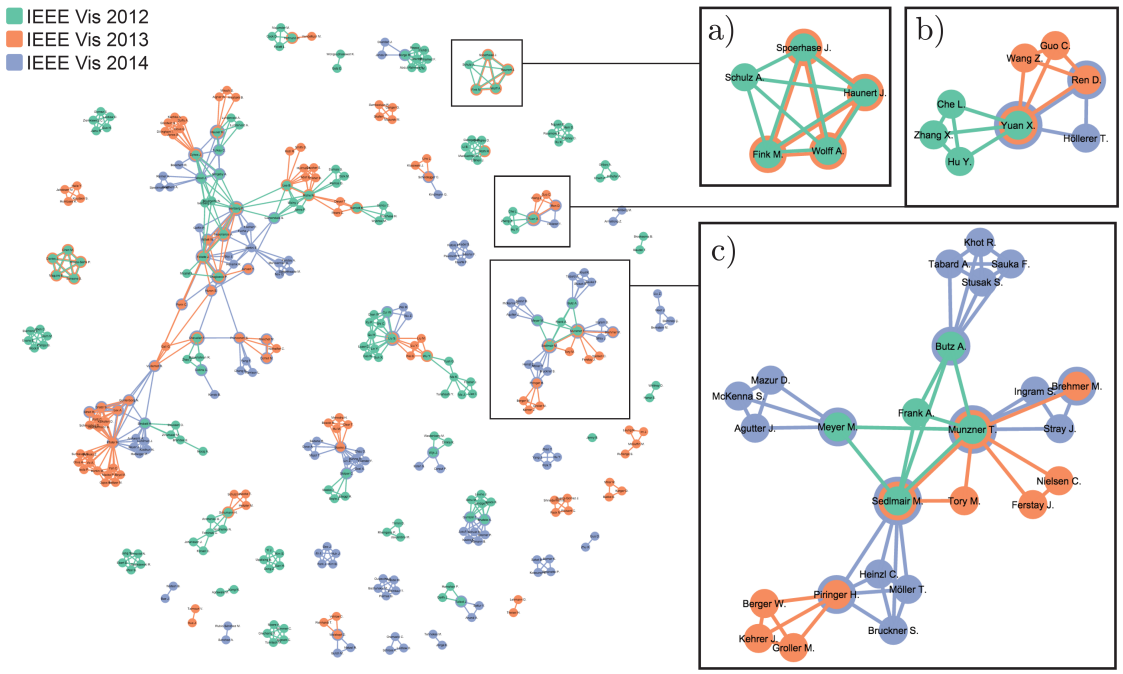

Figure 13. Visualizing the co-authorship network of authors who published at least one paper at *IEEE Vis* from 2012 to 2014. **(a, b, c)** are zoomed views of sub-networks of the overall representation

5.2.1 Co-authorship Network

We used *BranchingSets* to visualize changes of a co-authorship network over time. Our data includes all the authors who published at least one paper at *IEEE InfoVis* from 2012 to 2014. Every node of the network is an author, and a link exists between two nodes if the two authors are co-authors of the same paper.

Some solutions have been proposed to visualize additional information on co-authorship networks, such as cluster information (42) and publication year (43). In our solution, we want to address the

following questions: *Which authors published in all three years? What is the topology of their network of collaborators? Do authors works with the same people over time?*

Figure 13 shows the visualization of the co-authorship network with the *BranchingSets* technique. In this example the year of publication represents the category information to be displayed; we used a green, orange and blue color to indicate 2012, 2013 and 2014, respectively. Nodes are colored accordingly to the year when the author published a paper. If an author published in more than one year we used multiple colored border, according to the *BranchingSets* guidelines. Links connect co-authors and are colored accordingly to the year when the two authors have published a work together. Multiple colored lines are placed side by side if two authors have been co-authors in more than one year.

Figure 13(a) shows a group of five authors who have published together in 2012. Form the representation it is immediately evident that four of them wrote a paper together also in 2013, but one of them, Shultz. A., was not included among the authors of the latter work.

Since *BranchingSets* adds an extra border around the nodes for each category, it is easy to spot authors which have published in multiple years because they are graphically bigger and therefore stand out respect the others.

Figure 13(b) represents the network of collaborators of Yuan X., who published a paper in all three conferences. However, he worked with different researchers every year with the exception of Rend D., who published a paper with Yuan X. in both 2013 and 2014.

Figure 13(c) depicts the complex relationships between the collaborators of two authors who published in all three years: Munzner T. and Sedlmair M. The representation also shows that these two authors worked together only in 2012 and 2013.

5.2.2 Geospatial Data

Visualizing geo-located elements on a map can easily lead to cluttered representations, especially if the technique uses visual cues for associating elements of the same category, such as *Bubble Sets* and *LineSets*. These kinds of representations can leave a big footprint on the map visualization. In this example we want to propose an implementation of *BranchingSets* to visualize a set of geo-located elements on a map. We considered a dataset of 20 restaurants belonging to three overlapping categories: pizzeria restaurants, vegetarian restaurants, and restaurants which have received positive reviews from the costumers.

This data does not include any connectivity information, however we used a minimum spanning tree algorithm (MST) to connect all the nodes of the same category minimizing the use of "ink", according to the Tufte's rule (44). We applied *BranchingSets* to visualize the position and the categories of restaurants on the map, and the links generated with MST provide a visual connection between restaurants of the same category.

Figure 14 shows the resulting effect, comparing the representation made with *BranchingSets* with an analogous representation using *LineSets*.

5.3 Final Remarks on BranchingSets

In this paper we described *BranchingSets*, a visualization technique that shows category information within node-links diagrams. We applied our techniques to a variety different datasets: biological pathways, co-authorship networks and geospatial data.

BranchingSets is specifically designed to be integrated with node-link diagrams, which leads to concise and less cluttered representations compared to other techniques that show the category infor-

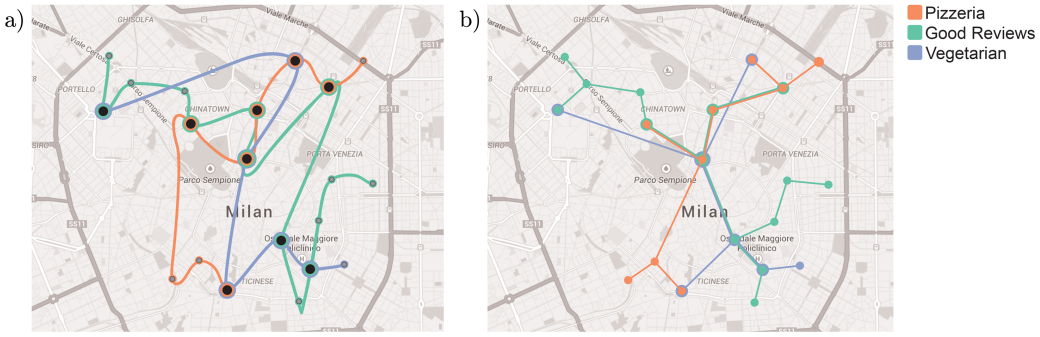

Figure 14. Visualizing location and category of a group of restaurants in Milan with **(a)** LineSets **(b)** BranchingSets

mation within a spatial structure. Additionally, our technique displays the set membership of links, information which is not provided with techniques such as *Bubble Sets* and *LineSets*. In some cases this category information related to links is required to produce meaningful insights, as demonstrated in the co-authorship example.

Furthermore, *BranchingSets* includes a set of user-driven techniques for interactively exploring, extending, and inspecting a complex network. These interactions have proven to be helpful when delving into large datasets, such as described in our biological pathways example.

We believe that *BranchingSets* is effective at helping users to recognize similar patterns and identify differences between graphs as well as to identify nodes that belong to multiple categories, and that our solution produces representations which are more easily comprehended and less cluttered when compared with *Bubble Sets* or *LineSets*.

The future direction or our research includes a an user study to explore how our solution scales when we increase the number of nodes, connections, categories, and intersections. Moreover, we are planning to conduct a study to highlight the effectiveness of the user-driven interactions to manage the exploration of large data.

CHAPTER 6

EVALUATING BRANCHINGSETS WITH AN USER STUDY

6.1 User Study

We have conducted an user study to evaluate the performance of *BranchingSets* compared with *Bubble Sets* and *LineSets*. A quantitative study has measured the accuracy and the completion time for a number of task on the different techniques. A qualitative study has collected the users' subjective evaluation in terms of ease of comprehension and visual clutter.

6.1.1 Method

In order to perform the controlled user study we recruited 17 people from different backgrounds. The age of the users ranged from 20 to 34, with a mean of 24, and the gender distribution was 9 females and 8 males.

Before starting the test, we made sure that every user had the background knowledge to perform the test. Five of them said that they were not familiar with graph layouts, but everyone was giving a short training session that described the fundamental concepts of node-link diagrams. Following this short introduction to node-link diagrams, all the users were introduced to the *Bubble Set*, *LineSets*, and *BranchingSets* technique by showing and explaining figures representing the same dataset with different techniques. To avoid bias, especially on the qualitative part of the user study, we told the users that the study had the aim to evaluate different techniques, but the users were unaware that *BranchingSets* was the focus of our research.

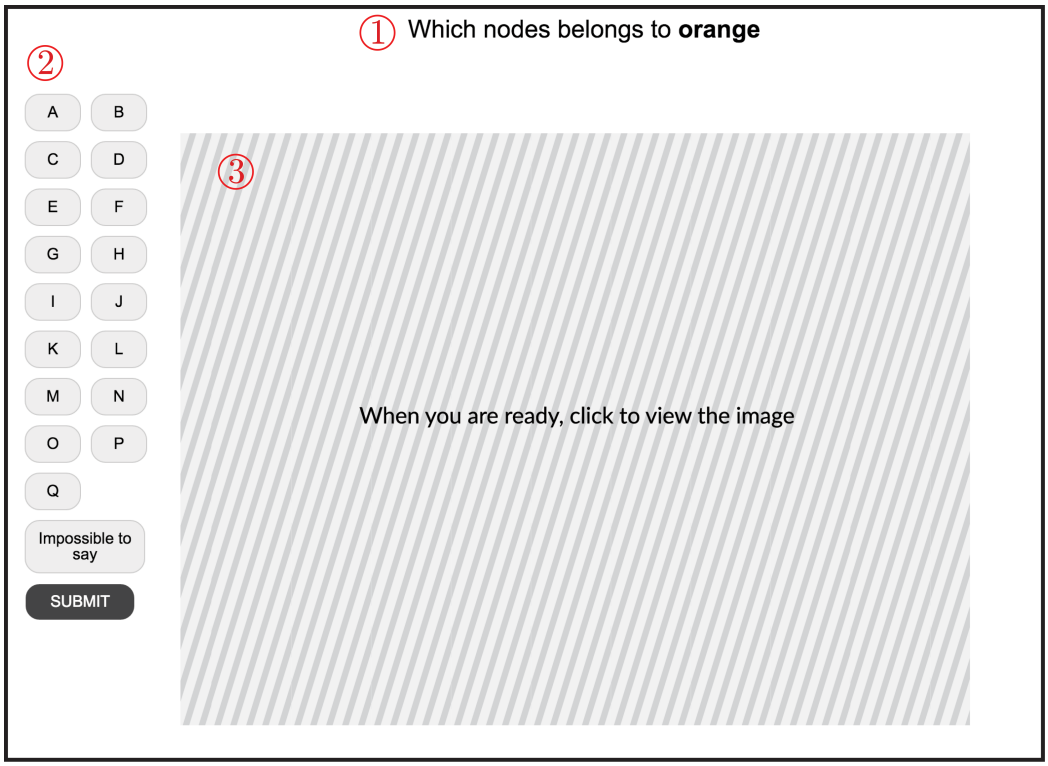

Figure 15. Screenshoot of the interface of the user study. The image shows **(1)** the question **(2)** the answer **(3)** the screen area dedicated to the representation

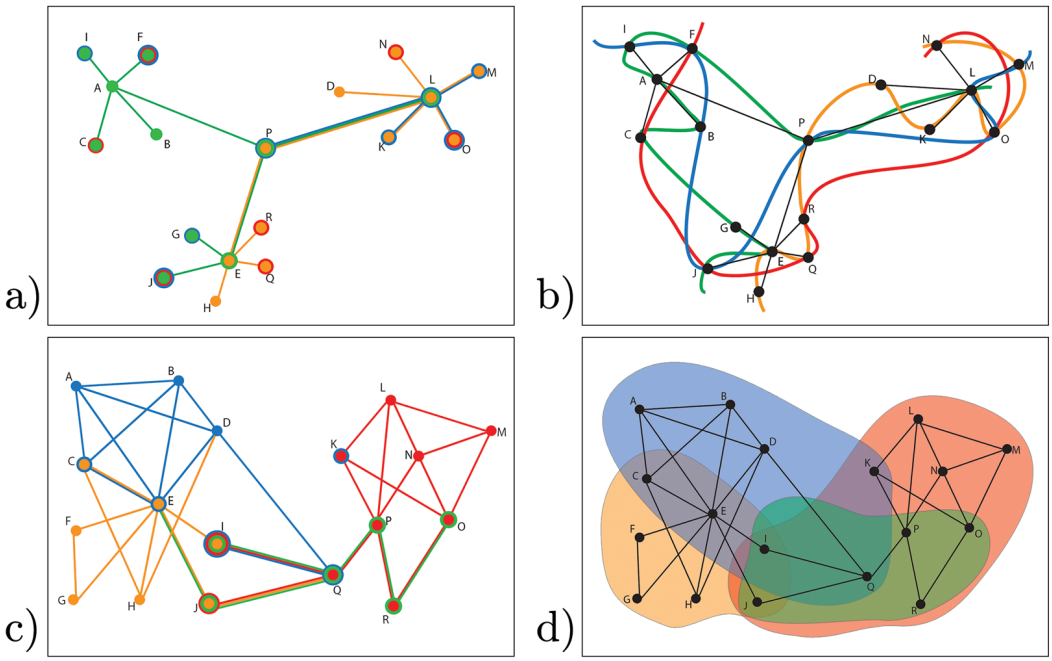

Figure 16. Examples of datasets used for the user study. **(a, b)** Representation of the same dataset with BranchingSets and LineSets. **(c, d)** Representation of the same dataset with BranchingSets and Bubble Sets

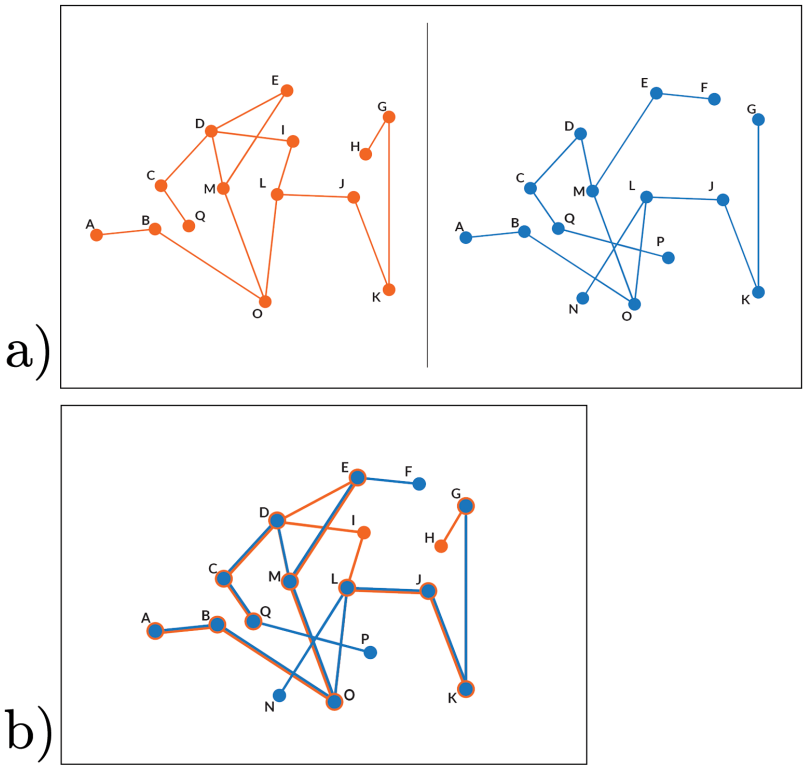

Figure 17. Examples of a dataset used in the user study for Task 5. **(a)** Representation of two graphs in separated views. **(b)** Representation of two graphs overlapped with BranchingSets

We have considered in our study both accuracy and completion time to be significant metrics to evaluate a technique. For a first set of tasks, which aim to compare the performance of *Bubble Sets*, *LineSets* and *BranchingSets*, we used three different datasets, each dataset was represented with all the three techniques. The dataset used has been made especially for the purpose of this user study, and they try to represent networks of different complexity and topology. Figure Figure 16 presents a sample of the figures shown during the user study.

6.1.2 Apparatus

The study was performed on a 15 inch monitor, and the figures each were 1000x800 pixels in size. Before starting the study, the users had the time to get familiar with the interface. Furthermore, when a new question was displayed on the monitor the user was able to read the question and the possible answers before starting the task. Indeed, for every question the figure was hidden and the measurement of the time started only when the user revealed the image. Figure Figure 15 shows a screenshoot of the interface developed for the user study.

6.1.3 Tasks

We have based our quantitative study on five common tasks that can be performed on graphs with categories. Table Table IV lists the tasks and a sample question for each task. Tasks 1-4 have been conducted on three different techniques: *Bubble Sets*, *LineSets* and *BranchingSets*. Task 5 measures the completion time for finding the differences between two graphs. In a first case the two graph are colored with two different colors and represented in two separated view. In a second case the two graph are merged with *BranchingSets* (Figure Figure 17).

TABLE IV

Tasks used in the quantitative experiment

	Task Type	Task Text
T1	Size of a set	"Which group contains more nodes?"
T2	Membership	"Which nodes are in the red group?"
T3	Intersection	"Which nodes are contained in both the red and green group?"
T4	Multiple Intersections	"Which nodes are contained in exactly 3 groups?"
T5	Comparison	"Which nodes/links are present in the green group but not in the orange group?"

6.1.4 Hypotheses

We have based or study on the following hypotheses:

(H1) No significant difference will be found between *LineSets* and *BranchingSets* in terms of accuracy and completion time for tasks (T1, T2). We believe that the continuous smooth curves of *LineSets* will perform comparably to the visual encoding of *BranchingSets*

(H2) BranchingSets will outperform both LineSets and Bubble Sets in terms of accuracy and completion time for tasks (T3, T4). The less cluttered representation offered by *BranchingSets* will help the identification of intersections between two or more categories.

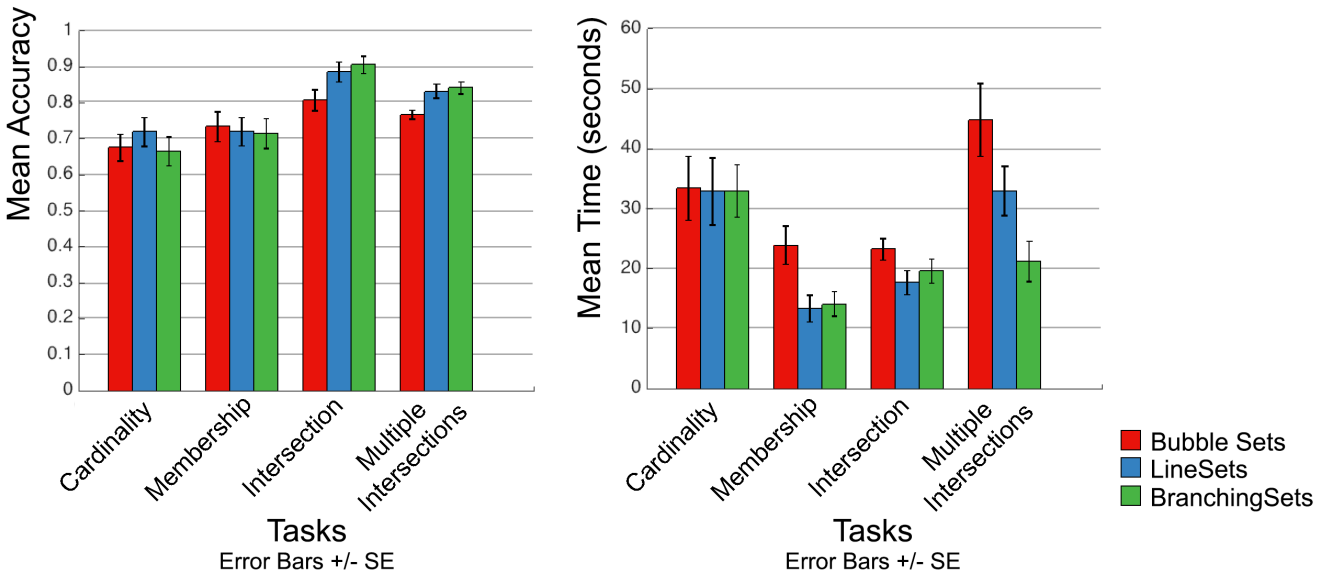

Figure 18. Chart showing the accuracy (left) and task completion times (right) for Tasks 1-4

(H3) BranchingSets will outperform a representation with separated views in terms of completion time for task T5. We believe that *BranchingSets* will make easier to identify the sub-graphs which contain nodes and links in the same category. The sub-graphs are visually stacked and will make it easier to spot similarities and differences.

6.1.5 Results

We used multiple paired t-test with a within subject design to compare accuracy and completion time of users tasks on different techniques. We computed the logarithm of times, as usual for reaction times, to normalize the skewed distribution of the data.

On average, 76% of the answer were correct. We found a significant difference in the accuracy between *LineSets* and *Bubble Set* ($p < .05$), and between *BranchingSet* and *Bubble Set* ($p < .03$) in

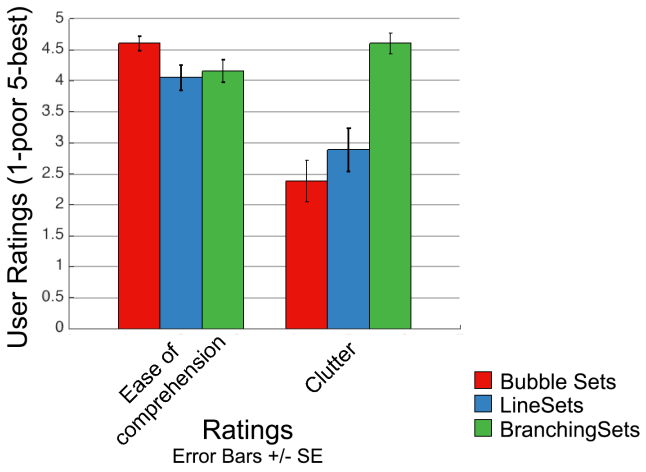

Figure 19. Subjective user ratings from the qualitative study

performing the multiple intersections task (T4). Figure Figure 18 on the left compares the mean values for the accuracy.

Regarding the completion time, we have found a significant difference between *Bubble Set* and *LineSets* for the intersection tasks ($p < 0.01$). Furthermore, we have detected a significant difference in the completion time for the multiple intersection task (T4) between *BranchingSets* and *Bubble Set* ($p < 0.02$).

Conversely from our hypothesis we have found a significant difference in the completion time for membership task (T2) between *BranchingSets* and *Bubble Sets* ($p < .01$). Less surprisingly, as already proved in (26), *LineSets* is faster then *Bubble Sets* for membership tasks ($p < .01$). The mean time for the completion of tasks T1-T4 is depicted in Figure Figure 18 on the right.

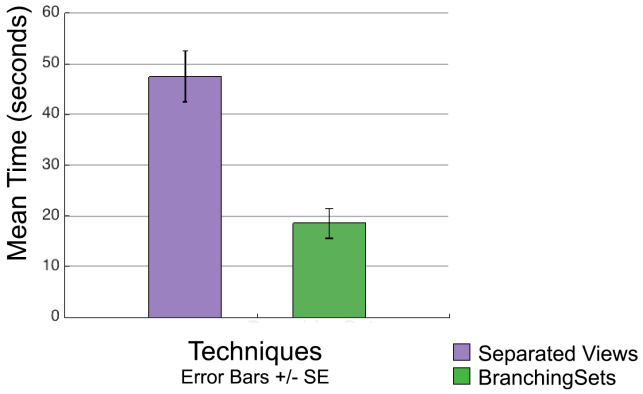

Figure 20. Chart showing the mean completion time for Tasks 5

BranchingSets has revealed to perform comparably to *LineSets* in term of accuracy and completion time for tasks T1-T3. For multiple intersections task (T4) *LineSets* has performed on average 26.4% faster than *Bubble Sets* and *BranchingSets* has performed faster than *Bubble Sets* and *LineSets* respectively than 35.7% and 52.7%.

On average, task T5 took 47.6 seconds with two separated view and 18.6 seconds with BranchingSets. Our hypothesis (H3) was confirmed ($p < .0001$). *BranchingSets* technique merges together the representation of multiple graphs, which we proved to absolutely outperform a traditional visualization which represents the graph on separated view performing more than 150% faster. Especially for tasks that require to find the changes between two networks, having a representations that overlaps multiple representations make easier the identification of the differences. Although this result is obvious, we wanted to provide a quantitative measure of the performance gap between the two techniques. The results are depicted in Figure Figure 20.

6.1.6 Qualitative Study

In addition to the quantitative tasks discussed above, we also asked users to answer two qualitative questions for each technique involved in the study. The questions aim to measure the users' preferences in terms of visual clutter and ease of comprehension using a Likert scale from 1 to 5. We were interested especially to learn whether or not users considered it easy to comprehend the layout and visual encodings that our technique presented. We provided the following guidelines for the rating: 1 means that the technique is very difficult to understand, even with an explanation, 5 means that the technique is understandable even without any explanation.

Users noted that the visual encoding of *Bubble Sets* is generally easier to understand than *LineSets* and *BranchingSets*, whereas *LineSets* and *BranchingSets* resulted to be comparable. Users overall indicated that they found our technique less cluttered than *Bubble Set*. The average grade was 4.6 with a mode of 5. The users considered *BranchingSets* to provide a clean and neat representation even in the presence of numerous categories and multiple intersections. Figure Figure 19 summarizes the results of the qualitative study.

CHAPTER 7

BRANCHINGSETS WITH BIOLOGICAL PATHWAYS

7.0.7 Related Work

For various reasons, it can be challenging to represent multiple pathways simultaneously in traditional node-link diagrams, both due to issues of scalability and also due to difficulties in defining a layout that accurately and efficiently displays the topology of these pathways. In order to reduce the visual clutter introduced by dense networks with many edge crossings, many visualization tools rely on *node duplication*, which considerably reduces the overlapping of edges and enables a visually appealing arrangement of the nodes. However, as pointed out by Bourqui et al. (17), analysis tasks that rely on an understanding of the pathway topology might be hampered by the introduction of duplicated nodes. Furthermore, the combination of multiple pathways within a single visualization introduces the need of displaying the relationships between biological entities and the pathway or pathways they are participants of. That is, biochemical reactions, proteins, or other biological compounds can be included in more than one pathway. This raises the further challenge of correctly displaying this membership information on top of the traditional node-link diagram without introducing visual clutter.

The traditional *Euler Diagrams* might lead to visually confusing representations if the data contains multiple set intersections and complex positioning of the elements. For this reason a range of alternative visualization techniques has been developed to display elements belonging to multiple sets (23). *Bubble Sets* (25), *LineSets* (26), and *KelpFusion* (27) are all techniques that address this challenge. Each

of these techniques are designed to be overlaid on top of existing visualizations, such as geographical maps, and specifically some limitations prevent both *Bubble Sets* and *KelpFusion* from being applicable to pathway visualization. *Bubble Sets* displays set relations using isocountours, which produces problematic representations when an element belongs to multiple sets, in some cases causing elements to appear in included in the wrong sets.

Developed by Alper et al., *LineSets* is a technique that represents sets as smooth curves and uses colors to indicate membership (26). This solution offers better readability when multiple sets overlap than *BubbleSet*. *KelpFusion* uses continuous boundaries made by lines and hulls. The visual appearance is generally comparable or better than *LineSets*, but it strongly depends on the spatial arrangements of elements. Hulls are used to group together elements which are both spatially close and they belong to the same set. They can easily became more confusing than useful when applied in biological networks. For this reason we consider *LineSets* as a effective technique which we can extend to be used to visualize pathways (45).

In this chapter we use the *BranchingSets* technique presented in Chapter5, which introduces the inclusion of hierarchical structure and which is presented as a more abstract node-link diagrams, without a direct spatial foundation. The presented technique particularly addresses the biological tasks involved in the pathways analysis. It can be considered as an extra information layer added onto the node-link diagram to represent the intricate relationship between different pathways while avoiding node duplication. After presenting a prototype implementation and the major design choices, we compare *BranchingSets* to the most commonly used techniques found in pathway visualization tools in Chapter 9.

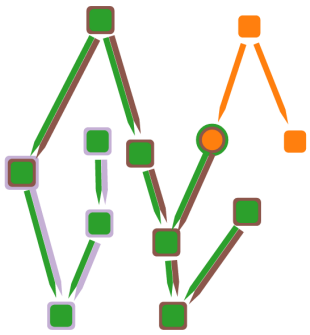

Figure 21. The image shows the proposed technique to combine node-link diagram with *LineSet*. The arrangements of the nodes follows the topological order of the links

7.1 The Design

In this section we describe the two main representations included in the *BranchingSets* technique adopted by our prototype. The first is a combination of node-link diagrams and *LineSets* that allows the user to concurrently identify the set memberships and visualize directional binary relationships of some elements. The second component is a hierarchical representation which enables the inspection of nested structures.

Next, we detail how these two components are composed together for an effective interactive visualization of pathway structure. Finally, we introduce our prototype which leverages the described technique and implements a set of user-driven interaction to enable various tasks.

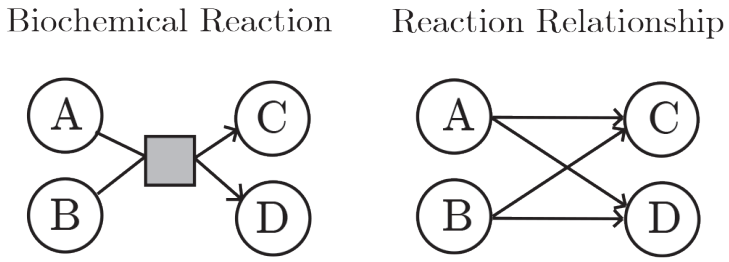

Figure 22. On the left, a reaction which involves two left components (A,B) and two right components (C,D). On the right, the figure explain the relationship between the four components considered in our technique

7.1.1 Node-Link Diagram with BranchingSets

This representation technique is designed to visualize a directed graph where nodes and edges belong to one or more sets. We identify every set with a color, and every element is either a square or a circle. Edges are shaped as sharp-pointed lines (Figure 21).

The elements and edges are colored according to the sets they belong to. If the same component is included in more than one set, a colored border is added around the node for every additional pathway it is associated with. Moreover, if more than one set includes an edge between the same elements, then multiple sharp-pointed line are placed side by side. The line direction and colors indicate the corresponding set and the direction of the edge. The segment connecting two components has a sharp tip on both the extremities if two opposite edges in the same set involve the same elements.

This representation has some substantial differences from a classical *LineSets* implementation. Firstly, the sets are not lines but rather branches. Secondly, the layout of the elements shows a flow from the top

to the bottom which matches, when it is possible, the direction of the relationships. However, a completely consistent representation cannot be achieved when the graph contains cycles, in this case there will be directed edges pointing upwards, but the general layout will still mostly adhere to the flow from the top to the bottom of the viewport. Additionally, the layout of the elements is designed to minimize nodes overlapping and edges crossing. The resulting visualization combines the efficient identification of elements with sets from the *LineSets* technique with the visual usability of the traditional node-link graph representations.

Our technique avoids node duplications and effectively shows the relationships between different sets. In the proposed visualization, pathways are the sets. The elements of the sets are protein and complexes. Each pathway is assigned a color, whereas the edges between elements are *reaction relationships*. A *reaction relationship* exists between component A and B in the pathway P if the pathway P has at least one reaction which left side contains A and right side contains B. Figure 22 shows how a biochemical reaction which involves two participants on both the left and the right side is converted in four binary directed relationships according to the previous definition. Furthermore, the shape of a node is either a circle or a square according to the type of component the node stands for. The circle-shaped nodes are proteins whereas the squares represent complexes.

7.1.2 Hierarchical Inspection

The visualization and inspection for hierarchical structures is supported by an interactive visualization. Similar to the traditional notation for tree structures, we refer to the biggest component of the structure as *root*, whereas the elements with no children are *leaves*. The *parent* of a element is the element which directly contains it. However, the conventional representations of hierarchical structures

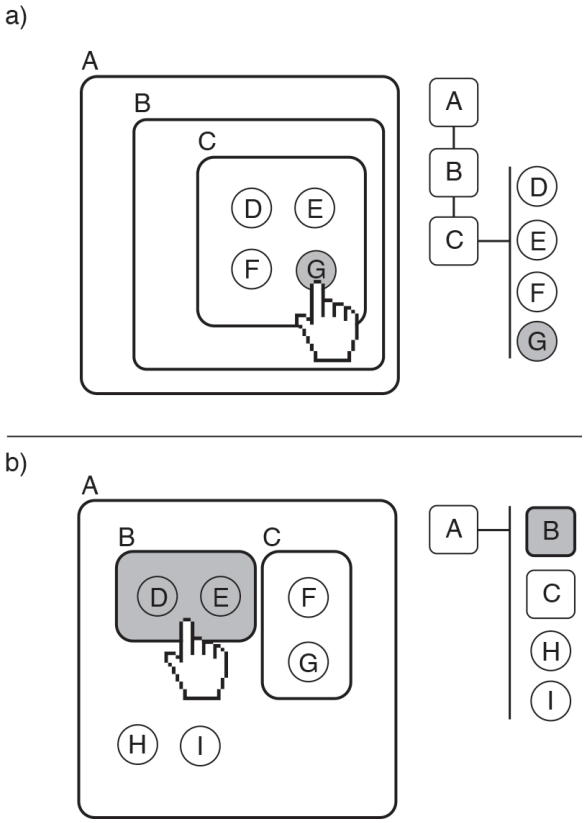

Figure 23. The figure demonstrates hierarchical inspection using the symbolic overview and the the pruned tree. In (a), the user selects protein *G* from the expanded node within the node-link diagram; this protein is highlighted within the pruned tree visualization, which also provides more information about its siblings (the proteins *D*, *E*, and *F*) and its parent and grandparent nodes (the complexes *C*, *B*, and *A*). In (b), the user selects a complex *B* in the expanded node, which is displayed in a similar manner in the pruned tree.

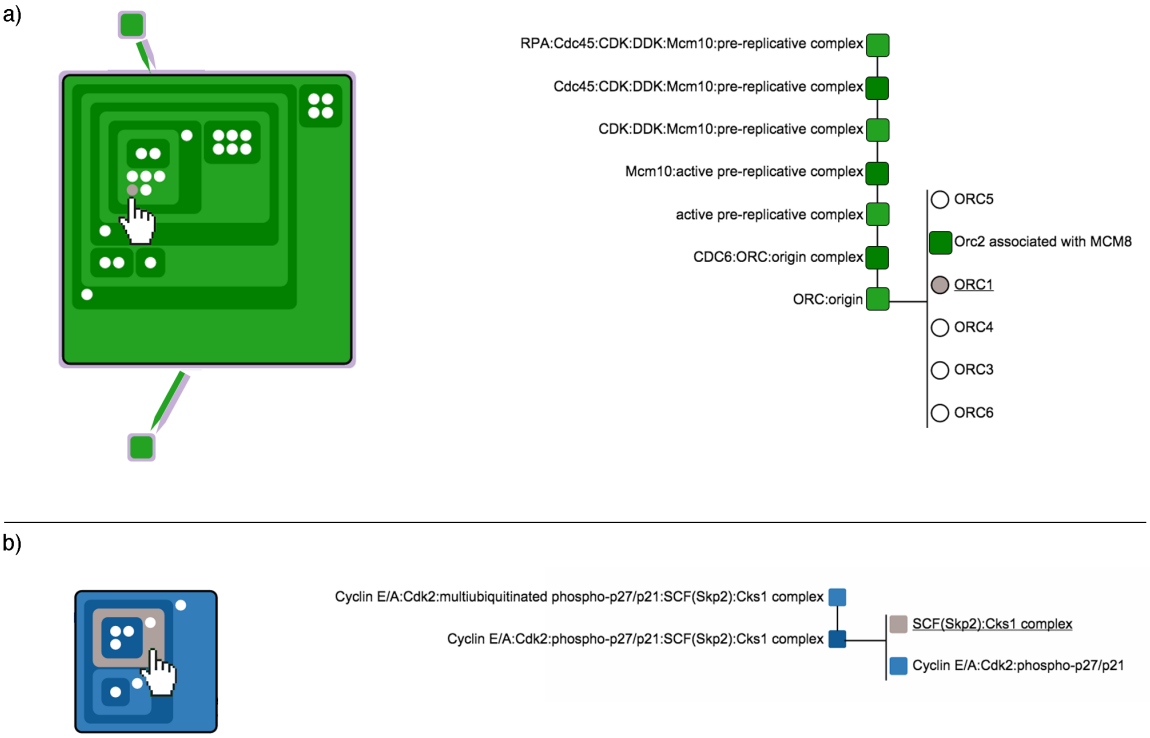

Figure 24. Two cases of inspection of the structure of a complex by means of two coordinated views

with traditional tree diagram is prone to visual clutter and poor readability as the amount of leaves and the depth of the tree increases.

The proposed solution is composed by two coordinated views: an overview and a schematic pruned tree structure. The overview is a symbolic portrait of the structure by means of a rectangle packing layout, whereas the pruned tree is a detailed representation of the hierarchy. No labelling is used in the overview to improve viewability.

When the user hovers with the mouse pointer on a element in the overview, the pruned tree is updated to reflect the hierarchy of elements from the *root* to the hovered element. The tree is called *pruned* because it does not show the whole structure, but it only displays the straight hierarchy of nested elements from the *root* to the *parent* which contains the selected element. At the same time the pruned tree enables the inspection of all the components directly included in the *parent*.

Figure Figure 23 illustrates how the symbolic overview and the pruned tree are coordinated in order to enable the inspection of a hierarchical structure. In Figure Figure 23(a), the user selects protein G, which brings up a pruned tree that indicates the hierarchy of parent complexes that contain G, fist A, then B, and then the direct parent C. The right side of the pruned tree shows protein G and its siblings D, E and F. Figure Figure 23(b) illustrates a similar case for complex B and its corresponding pruned tree.

We adopt this technique to represent the hierarchical structure of a complex. The overview representation is meant to provide a compact overview of the structure of the complex, where the squares represent complexes and the white dots are proteins. Figure 24 shows a protein hovered with the mouse pointer. The hovered protein or complex changes its color to gray.

This pruned tree structure is a straightforward but effective representation of the hierarchy of complexes, showing the structure from the *root complex* to the protein currently selected (via a mouse hover). The symbols on the side of the component's names provides a graphical reference to the symbolic view. The shape and the color of the symbols matches the ones used in the symbolic view. By inspecting the pruned tree in Figure 24, the reader can easily read the name of the *parent complex* of the selected protein, *ORC:origin*, and the related complex and five proteins. For the sake of clarity, the hovered component in the symbolic view is underlined and emphasized in the pruned tree.

This visualization is designed to be integrated in the graph visualization previously described. For this reason, the color palette used to differentiate nested complexes is limited to one color hue and two different color intensities. This minimizes the confusion with other tonalities used in the rest of the visualization. The proposed solution for analyzing the structure of a complex attempts to find an optimal solution between the complexity of the structure displayed and completeness of the information. In the following subsection we give more space to a detailed description of the prototype and the the user interactions involved in it.

7.1.3 Colors

BranchingSets uses colors to display information related to the category on the node-link diagram. Our prototype assign a category on each *selected* pathway, and then we need to assign a color to each of them. The effectiveness of our technique relies on a choice of the colors that make lines and shapes distinguishable even if drawn next to each other.

According to the type of the analysis performed by the users, the number of pathways or sub-pathways made visible at the same time by the user might vary. For this reason we can not generate one single color palette of the requested size. Furthermore, use more colors than necessary leads to a color palette with presents colors more similar to each other. Dynamically changing the colors when the user request to display more categories might be confusing, because the user might have memorized certain color associated to some pathways.

Our solution is to let the user to choose the color palette that fit its request. By opening the *settings* menu, the user can select the color palette of the preferred size. Figure 25 shows the color menu. Above the buttons to change color palette, a preview of the palette is displayed.

The colors has been generated with the *I Want Hue* on-line tool (46) by configuring the hue, color and lightness to have a good contrast with the white background of the tool. We also offer a three-colors color-blind friendly, which has been taken from the *Colorbrewer* on-line tool (47).

Moreover, in some contexts the color coding used in the application uses a lighter version of the color of the pathway (such as in the symbolic representation of the complex structure and in the sub-pathways selection menu). To avoid confusion, the color palettes offered by the application includes colors which combines different values of hue and and color, but present almost the same lightness.

In order to facilitate the user to remember the association between color and pathway, we try to preserve the correspondence between pathway and color when possible. For instance, if the user select a *pathway A* and it is assigned the orange color, when the user deselect and select again *pathway A*, it will have the same color. Even if the user deselect *pathway A* and then select other pathways, we do not

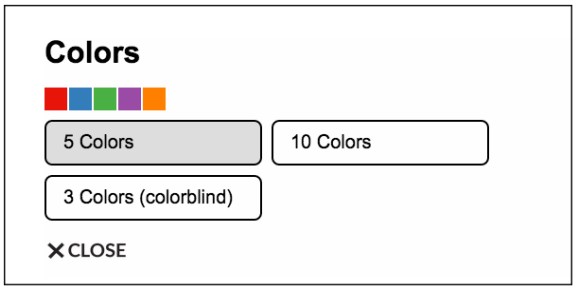

Figure 25. Screenshoot of the Color menu. The menu shows the available palettes and a preview of the selected palette.

assign the same orange color to any other pathway until the number of selected pathways reach the size of the color palette. In this case, the orange color is bound to another pathway.

If the user select more pathways or sub-pathways than the colors available in the color palette, an informative message is displayed suggesting to change the color palette in the *settings* menu. If the user change color palette and the colors available are less then the currently selected pathways, a message will warn the user than some pathways will be automatically deselected.

7.2 Curved Links

Curved links in a node-link diagram can be used to provide an additional visual clue of the directionality of the link and to reduce edges overlapping. Figure 27 compares the case of a small network with a node placed on an edge with and without curved links.

Curve's clockwise orientation is used to provide directionality, in combination with the tapered tip of the line. This visual encoding can be useful to detect cycles. Figure 26 shows a network with one cycle of four nodes (1) and six nodes interconnected connected (2) comparing the representation with curved

links and with straight links. In the first case, the user can follow the bend of the link to distinguish real cycles from other similar interconnections. This task is more difficult in the second case, because with straight lines the user have to visually follow the directionality of the link one by one.

The user can interactively choose between straight link and curved link under the *settings* menu, according to what he or she prefers.

7.2.1 Interactions

Based on the the interactions described in Section 5.1.6, we designed a set of user interactions for exploring, extending and inspecting the pathways at different level of detail. The interaction tasks, discussed below, include the following:

- Pathway highlighting and labelling;
- Nested pathways inspection
- Pathway hiding and unhiding;
- Keyword filtering;
- Upstream and downstream expansion;
- Layout rearranging, zooming, and panning;
- Complex structure inspection;
- Finding intermediate steps between two nodes in a pathway.

7.2.1.1 Pathway Highlighting and Labelling

By default all the complexes and proteins labels are hidden. This design choice has been made to improve the usability and to avoid the inconvenience of too many labels that might overlap and impair

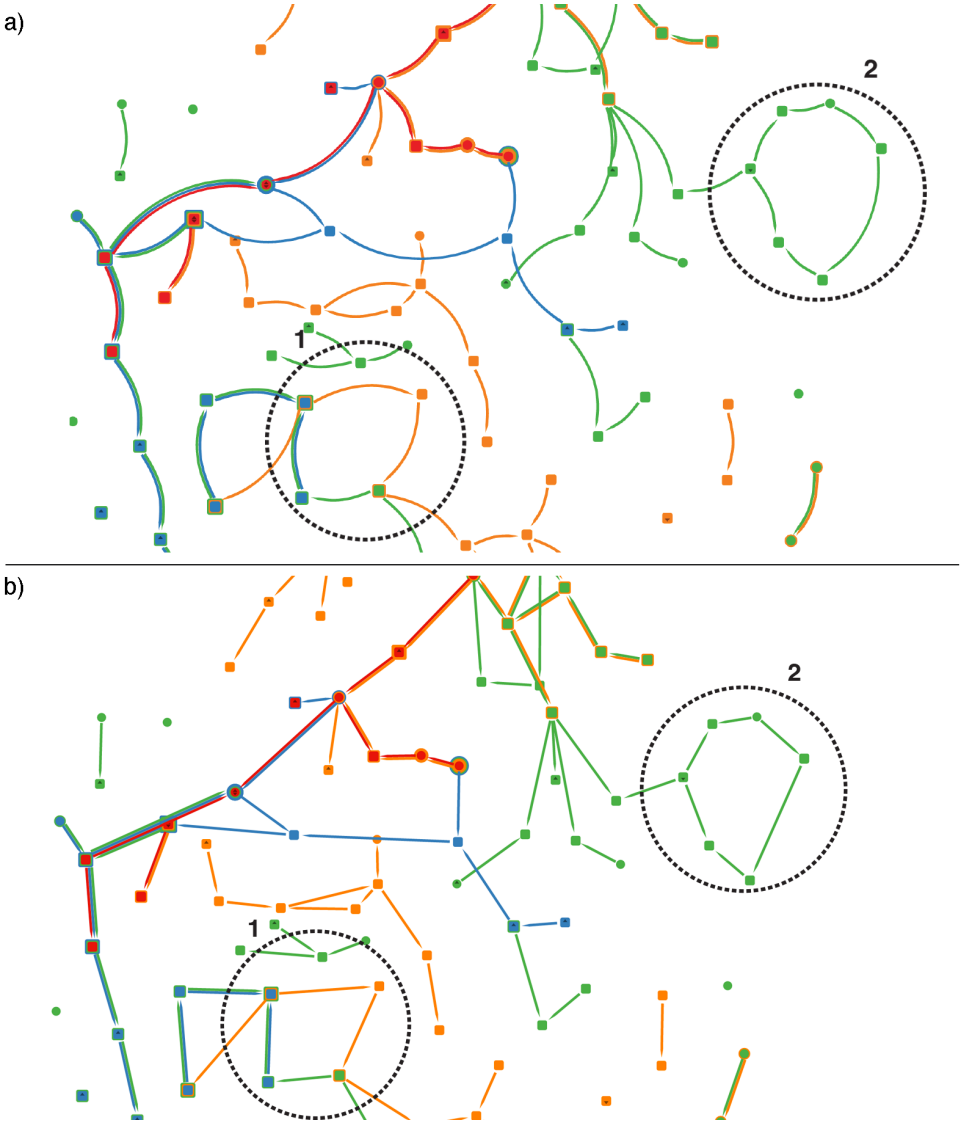

Figure 26. The figure represent the same graph (a) with curved links (b) with straight links. The dashed circle (1) contains a cycle, (2) is not a cycle.

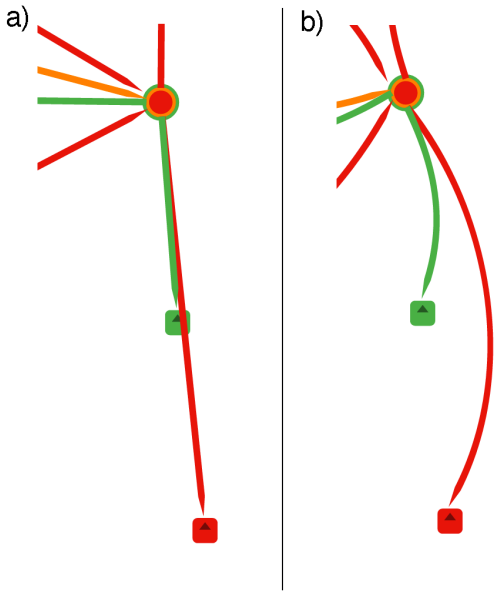

Figure 27. The figure represent two overlapping edges (a) with straight links (b) with curved links

the graphical representation. When the user hovers with the mouse pointer on a component, only the labels of the components which are included in the same pathway are displayed. Moreover, the pathway which includes the hovered component stands out because all the components and reactions belonging to any other pathways change to a gray color (see Figure 28). If a component which is included in more than one pathway is hovered, all of the involved pathways will be highlighted.

7.2.1.2 Pathway Hiding and Unhiding

The user can hide or reveal any pathway by clicking on its name on the left of the screen (see Figure 29). When a pathway is hidden, its name is colored gray and the visualization is updated to show only the components which are included in at least one visible pathway.

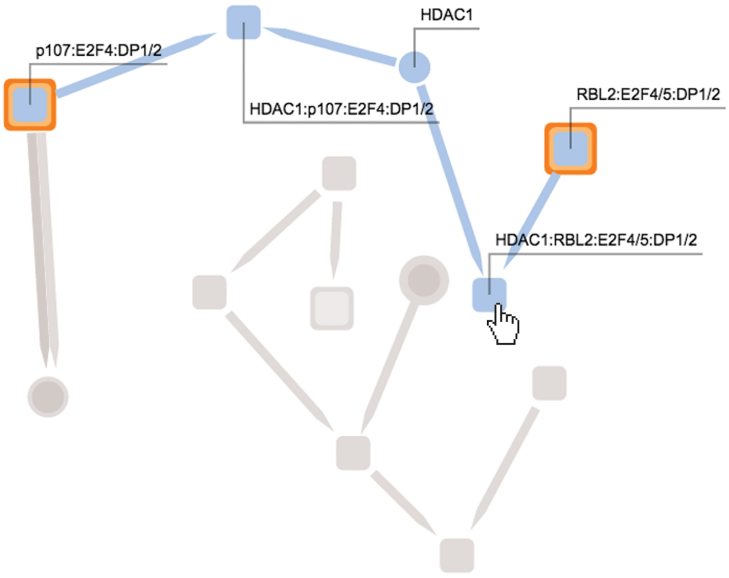

Figure 28. When the user hovers on a component, the application will highlight only the relevant pathways

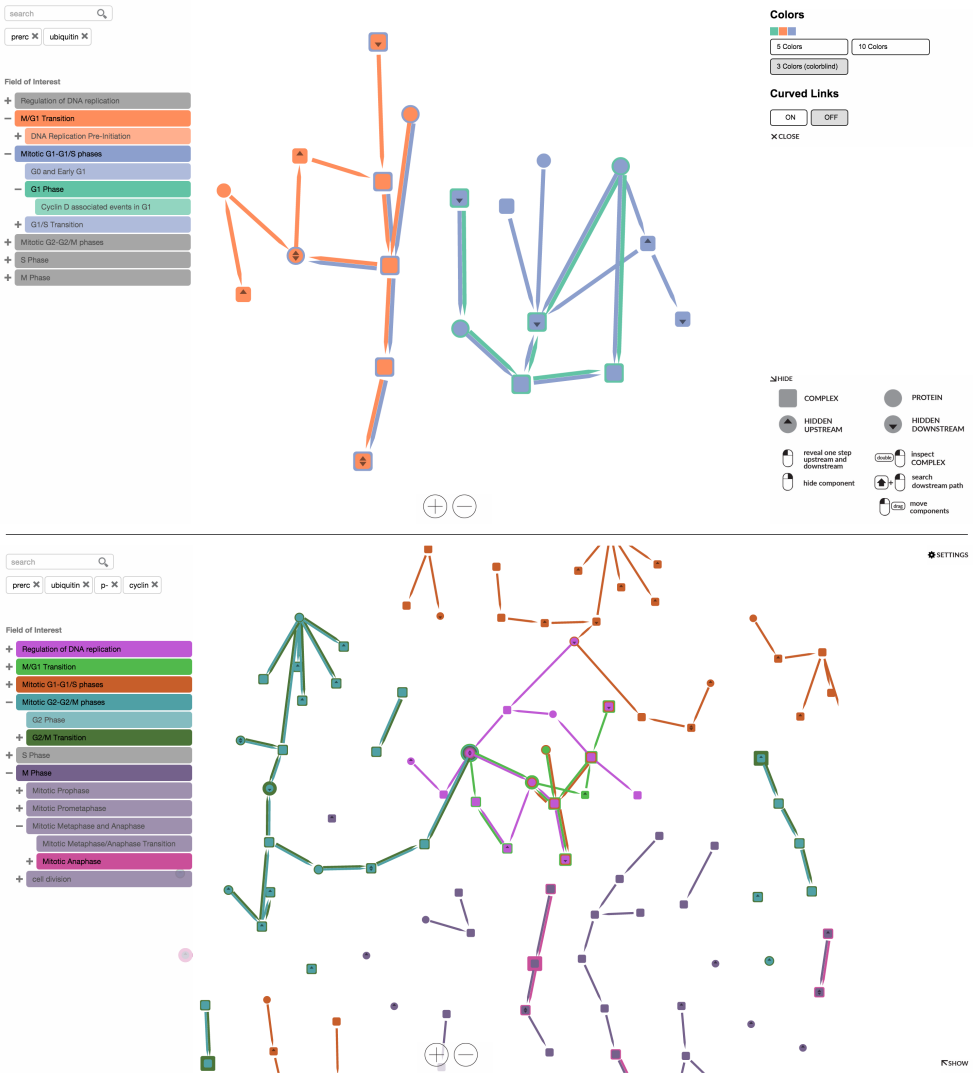

Figure 29. Screenshot of the interface of the prototype. On the left side from the top: the search field, the list of searched keywords and the list of pathways loaded by the application

7.2.2 Nested pathways inspection

The pathways are organized in an hierarchical structure. To let the user do delve into the structure and select interesting sub-pathways, we have organized the structure in a expandable list. Figure 30 shows a screenshot of the user interface. Every pathway which contains sub-pathways has a '+' symbol, and when clicked will show all the sub-pathways below their parent.

The indentation of the pathway names and the coloring provide a visual clue of the hierarchy. Indeed, sub-pathways are translated to the right with respect to their parents, and their coloring has the same hue of the parent but is lighter. If a sub-pathway is selected, its color will change to a new color.

7.2.2.1 Keywords Filtering

The user can add a keyword by typing one or more words in the search field. The keyword is added to the list of searched keywords. A keyword can be removed by clicking on the *x* icon. The visualization will display only the components whose name matches at least one of the searched keywords and the complexes which contain a sub-complex or a protein that meets the searched values.

7.2.2.2 Upstream and Downstream Expansion

By clicking on a protein or complexes the tool will expand the visualization with one *upstream step* and one *downstream step* starting from the clicked component. A *downstream step* of one protein or complex is composed by all the *reaction relationships* which start from the given component. Similarly, an *upstream step* involves all the *reaction relationships* which ends to that component.

The previously described user interactions enables the following workflow: the user searches for a set of proteins and complexes, then he or she progressively explores the network of interconnections by revealing the upstream and downstream.

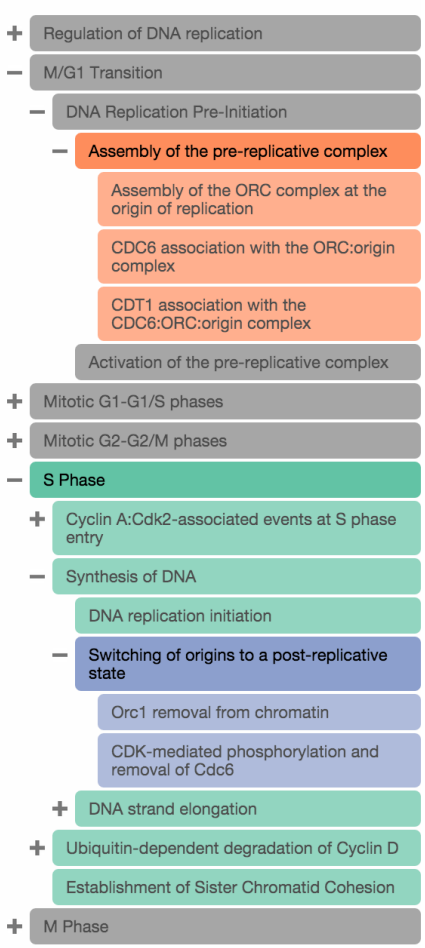

Figure 30. Exploring the the pathway hierarchy. A sub-pathway is selected in orange, whereas the whole S-Phase pathway is selected in green. Inside S-phase, another subpathway is selected in blue.

7.2.2.3 Layout rearranging, zooming and panning

Considering that the number of complexes and proteins involved might lead the visualization to exceed the boundary of the screen, the user can interactively zoom in, zoom out, and pan across the viewport. Since the representation might occasionally lead to unavoidable edge intersections, the user can drag and drop any component on a preferred location to improve viewability.

7.2.2.4 Complex Structure Inspection

As mentioned in Section 7.1, the visualization of the complex structure is meant to be integrated in the rest of the visualization. A complex can be enlarged by double-clicking on it. The enlarged complex reveals its inner structure and all the other components are pushed away to avoid unwanted overlapping. When hovering on the hierarchy structure the pruned tree is updated on the right side of the screen. A single click on an enlarged complex will hide the hierarchical structure, causing it to shrink the complex to the default size. More than one complex can be concurrently enlarged, and this enables the user to easily jump from the inspection of one complex structure to another one, and to find the same sub-complex in multiple complexes. Indeed, when the user hovers with the mouse on a component inside a complex, the same component will be highlighted in all the enlarged complexes if present. Figure 31 shows the *preRC* enlarged complexes and other downstream and upstream complexes, highlighting the location in the hierarchical structure of *MCM2-7* and its six related polypeptides in all the expanded complexes.

7.2.2.5 Finding intermediate steps between two nodes in a pathway

Beside the upstream and downstream expansion previously described, the user can expand the set of visible components and *reaction relationships* in another manner. By clicking on a component and

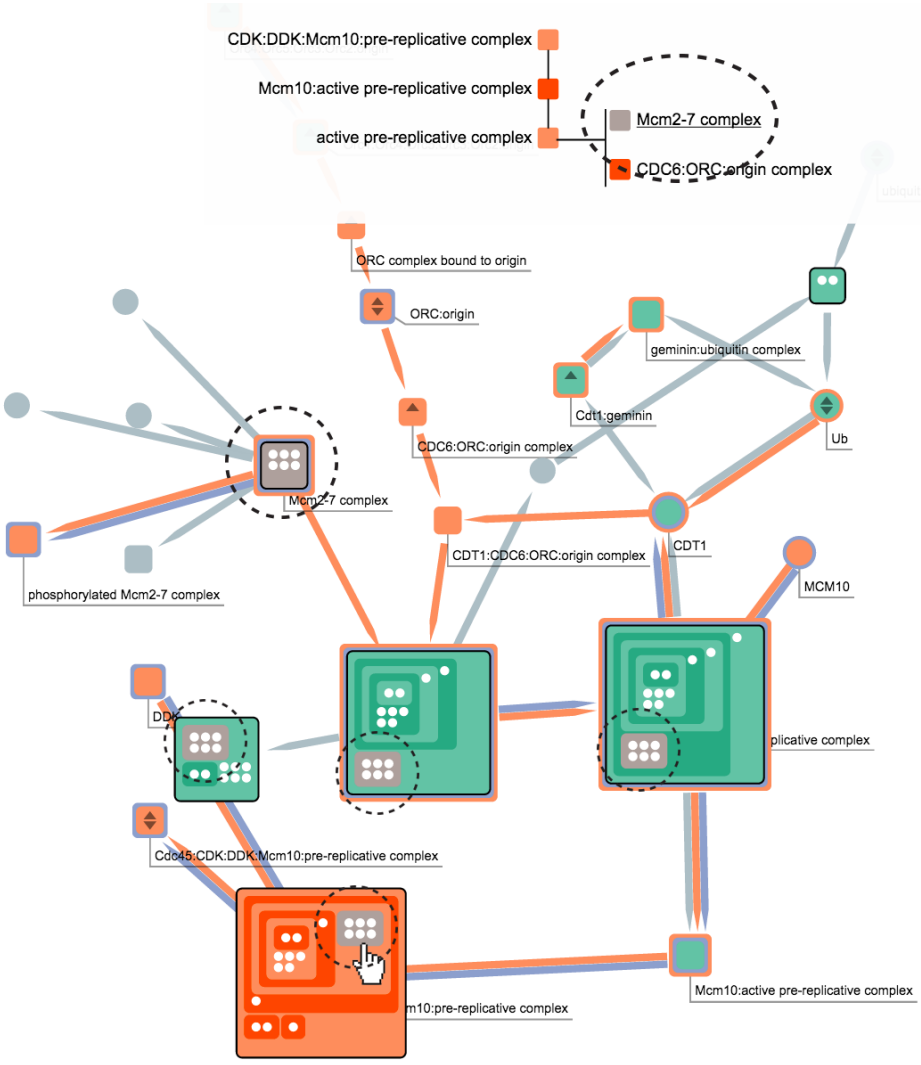

Figure 31. Location of all the *MCM2-7* in the enlarged complexes

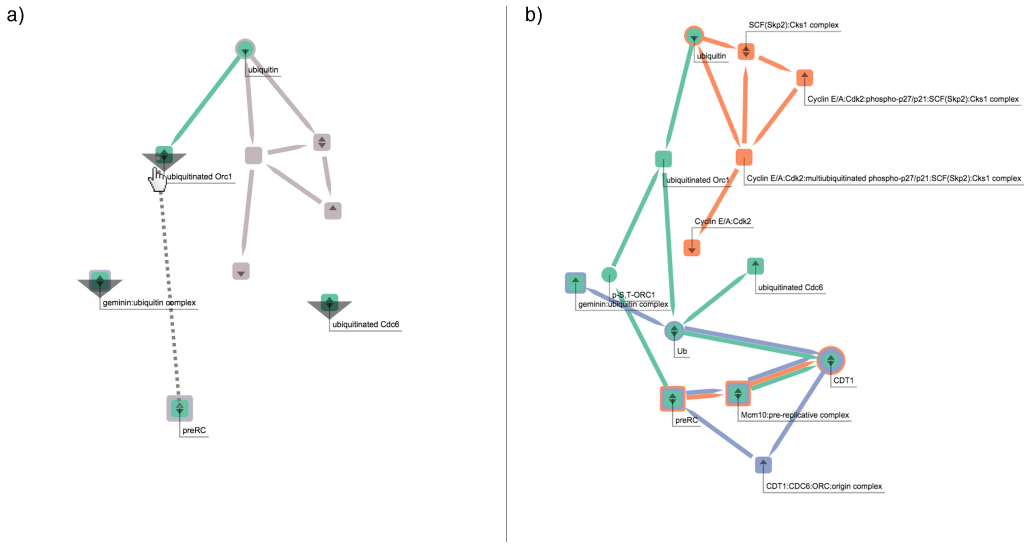

Figure 32. On the left, the user is dragging the mouse from one complex to another. On the right, the visualization is updated with all the connecting paths

start dragging the mouse while holding the shift button, the visualization will highlight with a triangle pointing downwards all the components that are downstream to the given component. The sequences of reactions and intermediate components might be hidden, and when the mouse is released on a downstream component, the tool will update the visual representation with all the sequences of *reaction relationships* that start from the first component ad end with the latter (Figure 32). Clearly, the *reaction relationships* that are revealed belong only to visible pathways.

7.2.3 Additional Visual Encodings

The interaction with the visualization is enhanced with two additional visual encodings. To facilitate the "upstream and downstream expansion" we place on the node a triangle-shaped maker pointing

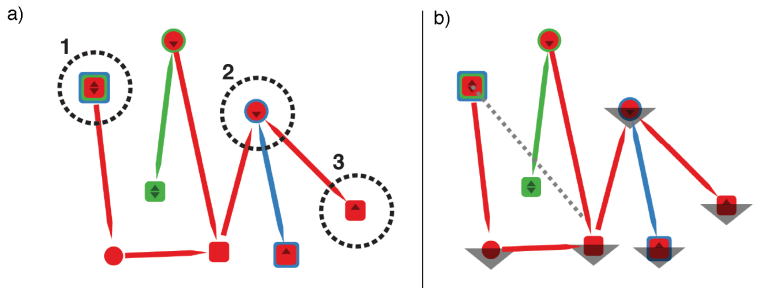

Figure 33. Additional visual encodings embedded in our tool. Figure (a) shows (1) upstream and downstream markers (2) downstream marker (3) upstream marker. Figure (b) shows the downstream markers during the "finding intermediate steps between two nodes in a pathway" interaction.

upwards if the node has an hidden upstream, and a triangle-shaped maker pointing downstream if the node has a hidden downstream. Figure 33(a) shows a screenshot of the application.

During the "finding intermediate steps between two nodes in a pathway" interaction all the nodes which are downstream of the starting node are highlighted with a bigger semi-opaque triangle pointing downward.

7.2.4 Animations

When the user perform one of the interactions described in Section 7.2.1, the tool updates the visualization accordingly. New nodes and links will appear or disappear, and the layout algorithm is run again to place the nodes in a suitable location.

Animated transitions can be considerably more effective than "jumps" in helping the user to track the changes occurring in the visualization (48). For this reason, nodes and reactions changes position smoothly with an animation. Furthermore, nodes that appear and disappear on the representation will

fade in and fade out by mean of a transition. When a new reaction appear on the scree, it is revealed progressively from its starting point to its destination.

CHAPTER 8

IMPLEMENTATION NOTES

8.1 The Prototype

In this section we present the framework used to realize our tool and the implementation choices that we have faced during the development, such as the organization of the data-structure. Furthermore, we describe few algorithms which design present some challenges, such as the representation of the nested structure of a biological complex and the layout algorithm to display the labels of the components.

8.2 D3.js

For the development of the application we have used *D3.js* (49). *D3.js* is a JavaScript library for manipulating the DOM of a web-pages with on data-driven approach. *D3.js* presents a own code-style and guidelines to build visualization components, and offers basic tools to build a new representations.

The *D3.js* library uses JavaScript functions to select HTML elements and create SVG objects. The SVG standard is used to represent objects and shapes with vector graphics. *D3.js* includes helper functions to style the object, or add transitions and other dynamic effects and interactions.

The dataset is bound to SVG objects using D3 functions which describes how the single piece of data should be represented according to the value.

8.3 Data Structure

The pathway data is stored in the *BioPax* format (9). This format leverages the *web ontology language (owl)* (50) to represent the knowledge.

The prototype presented in Chapter 7 requires to frequently query the data. For instance, the operation of expanding a node requires to consider first all the reactions that have that node as participants, and then all the other components included in the reaction. The operation of finding intermediate steps between two components requires to perform a breath-first search on the graph.

The ontology-based description provided in the *owl* file is a complete description of the pathway structure, but its querying is neither time-efficient nor helps the design of the algorithms. For this reason we parse the *owl* file and we build a data structure suitable for our application. The data structure incorporates only the relevant information and relationships relevant for our purposes.

The *ER-Diagram* depicted in Figure 34 represents the organization of the components. We classify the element in two major categories: *pathways* and *components*. A pathway might include other *pathways*, and contains one or more *components*. A component is either a *reaction* or a *participant*. A reaction has *left participants* and *right participants*, according to weather they are the input or the output of the reaction.

A *participant* is either a *protein* or a *complex*. A *complex* is an hierarchical structure of other *proteins* or *complexes*. We added the redundant information to each *participant* of which are the next and previous participants. A *participant A* is contains a given *previous participant B* if *B* is on the left side of a reaction which contains *A* as a *right participants*.

8.4 Force Layout

Organizing then node of the node-link diagram on the screen avoiding overlapping and edge-crossing is one of the more challenging problems. The solution adopted in this work uses a *force-layout*, which is a physic-based simulation that create forces between nodes to compute a suitable arrangement. The

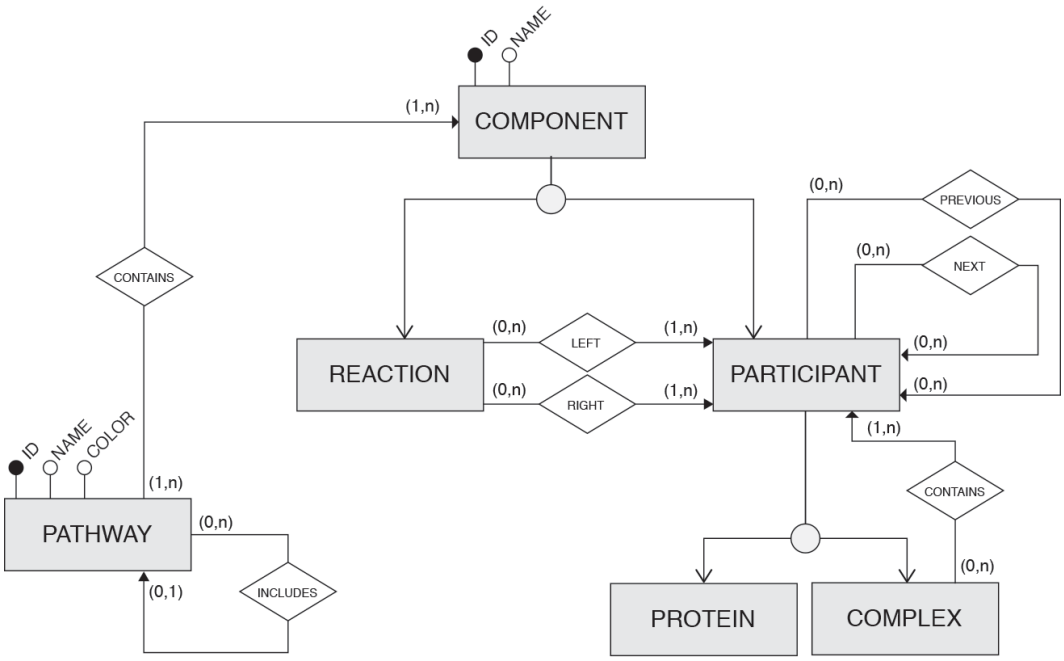

Figure 34. ER-Diagram of the data structure used in the prototype

algorithm used leverages the popular physic-based heuristic proposed by Barnes and Hut (51) which has a complexity of $n \times log(n)$ where n is the number of nodes.

A force exists between two nodes if the nodes are connected on the node-link diagram, and a *gravity* factor constrain the nodes around the center of the canvas. In order to provide a visual clue of the directionality of a sequence of interconnections, we add an external force which pushes down the leaves and push up the roots.

Figure 35 compares the arrangement of the nodes with a simple force-layout and with the influence of external forces which highlight the directionality of the graph.

Forces to push elements

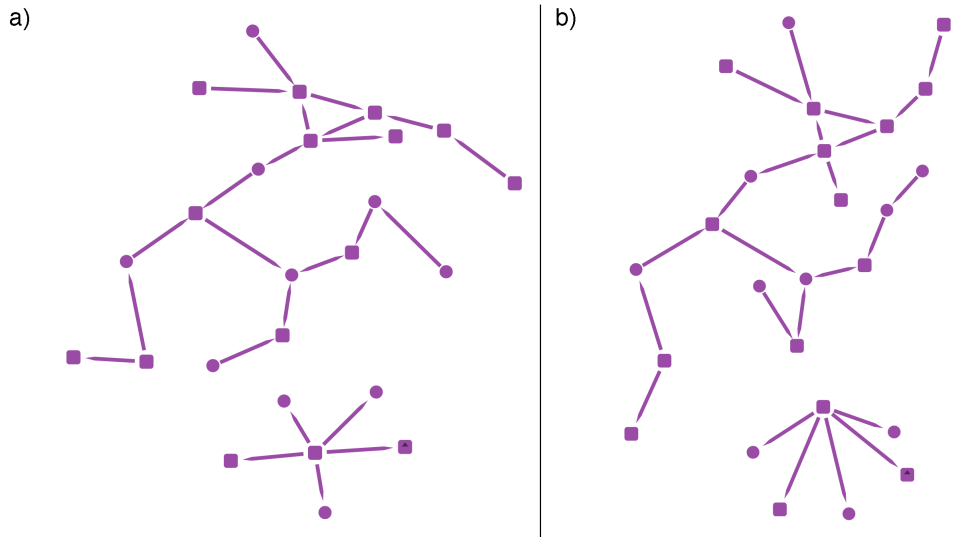

Figure 35. Figure (a) show the layout of a graph with a simple force layout. Figure (b) represents the same network with extra forces to show the directionality of the graph

8.5 Rectangles Packing

In order to display the structure of a biological complex described in Section 7.1.2 we need to represent a hierarchical nesting of complexes and proteins.

The problem of computing a layout for a hierarchical structure of rectangles has already been faced in other works such as the *Tree-Map* proposed by Shneiderman et al. in (52). Furthermore, there are available plenty of algorithms to compute an optimal arrangement of rectangles inside a rectangle ((53), (54), (55)).

However, the mentioned solutions minimize the space of the representation, which is not our objective function. Indeed, even tough we want to provide a compact visualization, our first motivations

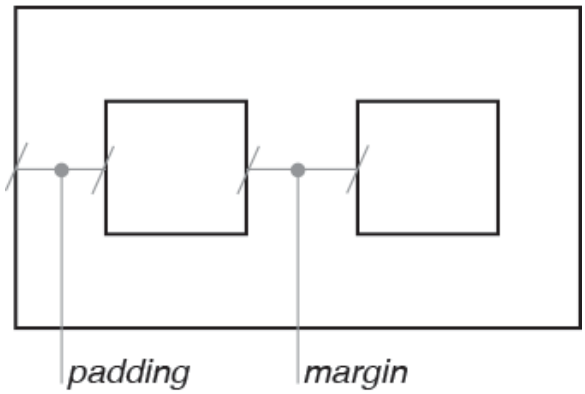

Figure 36. *Padding* and *margin* parameters

are to provide a readable visualization. For this reason we have developed our solution, which aims to arrange a concise and readable portrait of a nested structure.

We consider two parameters: *margin* and *padding*. *Margin* is the distance between the border of a node and a sibling, whereas *padding* is the distance from the border of a node and its parent (Figure 36).

The algorithm is composed by three functions: **RecursivePacking(*node*)**, **PackElements(*nodes*)**, **GridPack(*nodes*)**. To start, the routine **RecursivePacking(*node*)** is called on the *root* of the hierarchical structure. This routine operates as follow:

1. for each children $child_i$ of *node*, call **RecursivePacking(**$child_i$**)**;
2. call the routine **PackElements([**$child_1, child_2 ... child_n$**])** on the array of children of the *node*;
3. set the initial position of the node

   $node.x = 0, node.y = 0$;

4. set the width and the height of the node to

   $node.width = max(child_i.x + child_i.width) + padding$

   $node.height = max(child_i.y + child_i.height) + padding$;

The routine **PackElements(*nodes*)** execute the following steps:

1. find all the leaves contained in the *nodes* array, make a new array *leaves* and remove them from the *nodes* array;
2. execute the routine **GridPack([**$leaf_1, leaf_2 ... leaf_n$**])** on the array of leaves;
3. append to *nodes* the returned value of GridPack;
4. initialize two variable $x = padding, y = padding$
5. for each $node_i$ in *nodes* execute the following steps:
   (a) translate the element (and all its sub-elements) of x, y;
   (b) **if** $x + node_i.width \leq y + node_i.height)$ **then** $x+ = node_i.width + margin$
   else $y+ = node_i.height + margin$;

We omit a formal description of **GridPack(*leaves*)**. The routine organizes the nodes in a grid style layout, and return a new node which contains all the leaves.

Figure 37 shows some outputs of the algorithm with different biological complex data.

8.6 Labelling

As described in Chapter 3, an effective labelling of the components of the visualization is required to provide in order to build a successful representation. The labels show names of a protein and complexes

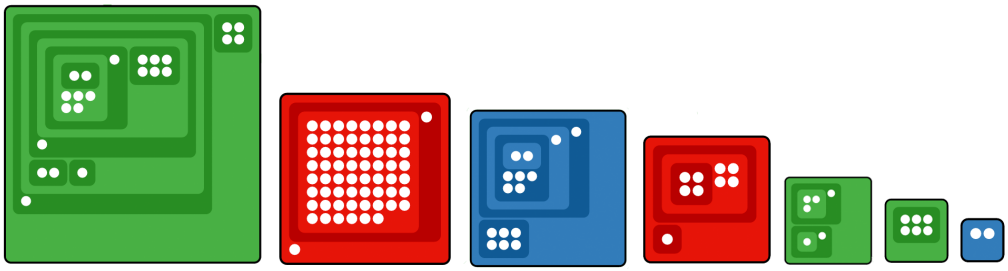

Figure 37. Examples of rectangles packing generates with our algorithm with real biological complexes data

and should be placed next to the corresponding component and should not overlap to each other to enhance readability. Indeed, text elements which overlaps are remarkably hard to read, and labels too far from the element they are related to might result to be confusing.

The challenge of placing text elements in the right position is made simpler by providing a visual connection between the label and the component. Indeed, our implementation draws a line segment from the component to the closest bottom corner of the label and from that bottom corner to the opposite one, so that it underlines the text. In this way the user has a visual clue of the correspondence between the label and the component even if the label is reasonably far from the component.

Furthermore, thanks to the interactive labelling of the components described in Chapter 7, only a subsets of all the components are labelled. Although this design decision has been made to reduce visual clutter, it moderately helps the task of finding a suitable position for each text element.

In this section we address the problem of designing a computationally-efficient algorithm to the correct position of labels. We approximate the text with its bounding-box, and we consider the problem

of placing a set of n rectangles of different sizes on a two-dimensional surface by minimizing two objective functions. The first function is the sum of distance from a rectangle and the assigned location, the second function is the sum of the overlapping areas of the rectangles.

According to the implementation requirements, the two objective functions can be combined in such a way to privilege the distance over the overlapping area and vice versa. Moreover, different distance function might be considered, such as the Manhattan distance, or the euclidean distance to the power of two.

Straightforward implementation are *NP-hard* (56). Chirstensen et al. in (56) propose different algorithm, which differs from methods and performance. In this section we describe our implementation, which complexity is $n \times log(n)$ and presents some similarities with the greedy, *gradient descend* algorithm proposed by Chirstensen et al.

8.6.1 The Concept

The idea behind the algorithm is to place the rectangles one by one in the assigned position on the place. If a rectangle is placed in a position which overlaps on other rectangles, it is translated on the X-axis or the Y-axis in order to reduce the overlapping surface and to not get too far from the assigned position.

Figure 38 shows a rectangle placed in a position which overlaps with other rectangles. The figure represents in green the *vertical and horizontal traces* of the rectangle. The trace represent all the possible locations where the rectangle can be considered to be re-positioned to minimize overlapping.

The algorithm generates two *overlapping profiles* of the translation of the rectangle respectively for the X-axis and the Y-axis. An *overlapping profile* is a function which maps the translation on the

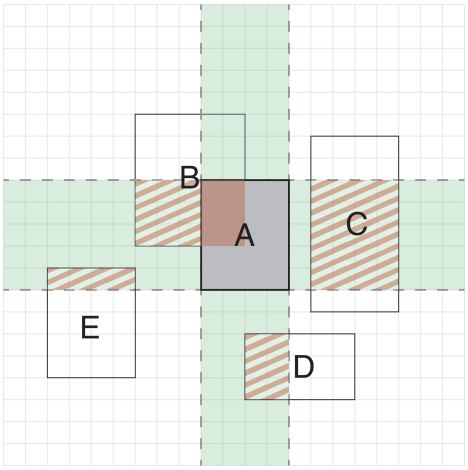

Figure 38. The rectangle A overlaps with B, and its vertical and horizontal traces overlap with B,C,D,E.

given axis to the corresponding overlapping area. The algorithm sums the *overlapping function* with a *penalty function* which increases with the distance from the assigned position, then it finds the absolute minimum point the two modified *overlapping profiles*. The rectangle will be translated on the axis which contains the minimum point of the corresponding value.

Figure 39 shows the horizontal trace of a rectangle placed on the plane, and the corresponding *overlapping profile*. Figure 40 shows how the *overlapping profile* is distorted with a *penalty function* to prefer points close to the initial position of the rectangle.

In the next section we provide further details of the algorithm and a brief analysis of its complexity.

8.6.2 The Algorithm

An efficient retrieval in a logarithmic time of elements placed in the plane is made possible thanks to a quad-tree data structure (57). Figure 41 represents the structure of the tree for a scenario with

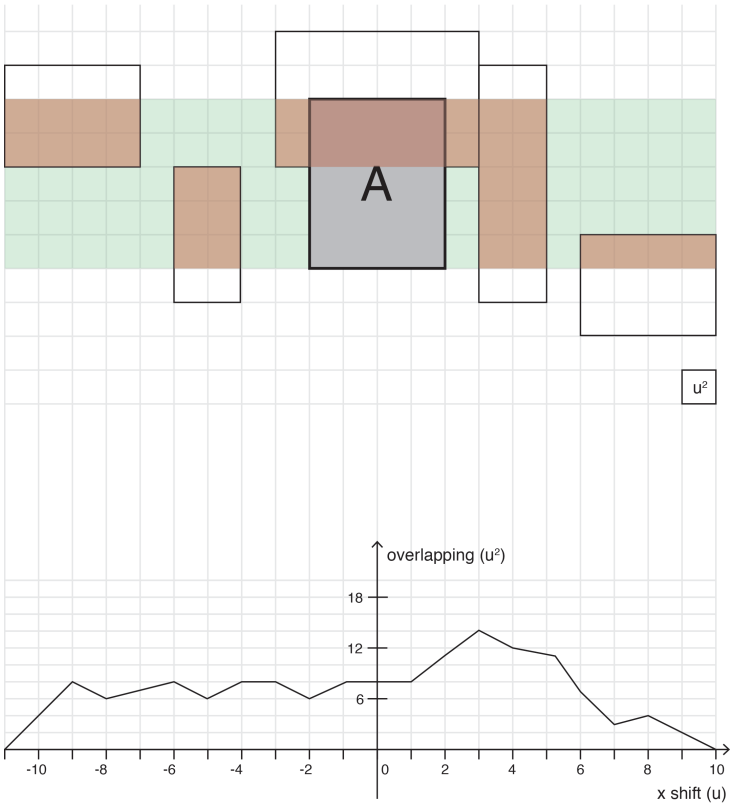

Figure 39. A scenario of a rectangle *A* placed on the plane, and its *horizontal overlapping profile*

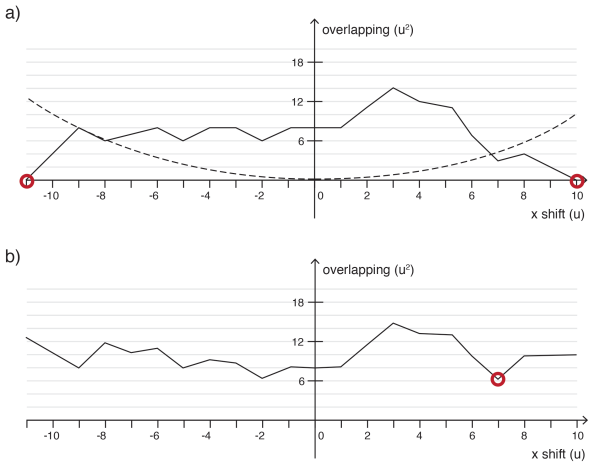

Figure 40. Figure (a) shows an *overlapping profile*, the dashed line is a cubic *penalty function*. Figure (b) shows the combination of the *overlapping profile* with the *penalty function*. In both figures the global minimums are highlighted with red circles.

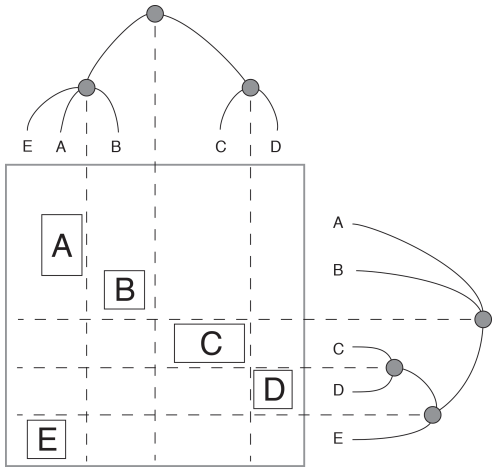

Figure 41. Quad-tree data structure representation for 5 rectangles

5 rectangles. We consider $r_1 ... r_n$ rectangles and the corresponding positions $pos_1 ... pos_n$. For each rectangle r_k we performs the following steps:

1. check weather r_k overlaps on any of the other $r_j, j < k$ rectangles already placed;
2. find all the $r_1^{over} ... r_t^{over}$ rectangles which overlaps with its *horizontal and vertical traces*;
3. for each of the t overlapping rectangles, consider its position and size and update the *overlapping profiles*. At the some time, consider also the penalty function;
4. find the minimum in the two *overlapping profiles*;
5. place the rectangle r_k and update the quad-tree.

Finding all the rectangles which intersects the horizontal and vertical traces rectangle is accomplished in $O(log(n))$ thanks to the quad-tree data structure.

We consider the space of the *horizontal and vertical traces* to be negligible with respect to the whole plane, and then the number t of overlapping rectangles to be constant. The *overlapping profile* is stored as an array of length $t \times 2$, and finding the minimum requires a constant amount of time.

Generating the vertical and horizontal overlapping profile is $O(t)$ and then requires a constant time. Updating the quad-tree can be accomplished in a logarithm time, thus the overall time complexity is $O(n \times log(n))$.

8.6.3 Limitations

The proposed algorithm utilizes a greedy approach and does not perform a exhaustive search of the space for finding the optimal location.Indeed, only the vertical and horizontal translations are taken in consideration. This solution might be adapted to consider also diagonals translations of rectangles. In

order to do so, we should instantiate another quad-tree which helps the search of rectangles by splitting the available surface with diagonal regions.

8.7 Drawing Curved Links

The graphical representation is controller with *SVG* elements. *SVG* supports the drawing of common shapes such as rectangles and circles, and more complex drawing can be realized with a combination of *Bézier* curves. A *Bézier* curve defines a path in the 2D plane with a set of points and handles. The path is constrained to pass trough the points, whereas the handles provide indication of the tangential direction of the path.

To draw the sharp-pointed lines implemented in the prototype described in Chapter 7 we use a mixture of line segments and *Bézier* curves. Here we describe the approach used to calculate the position of the points and handles given the starting point and destination of the arrow.

The arrow is drawn according to the the following parameters, which have been tuned in the design phase to realize a visually pleasant and clear visualization:

- **thickness**: the thickness of the arrow;
- **arrow_tip**: the length of the tip of the arrow which is tapered to suggest the directionality of the arrow;
- **alpha**: the starting angle of the curved links;

A graphic representation of the mentioned parameters can be viewed in Figure 42 (1).

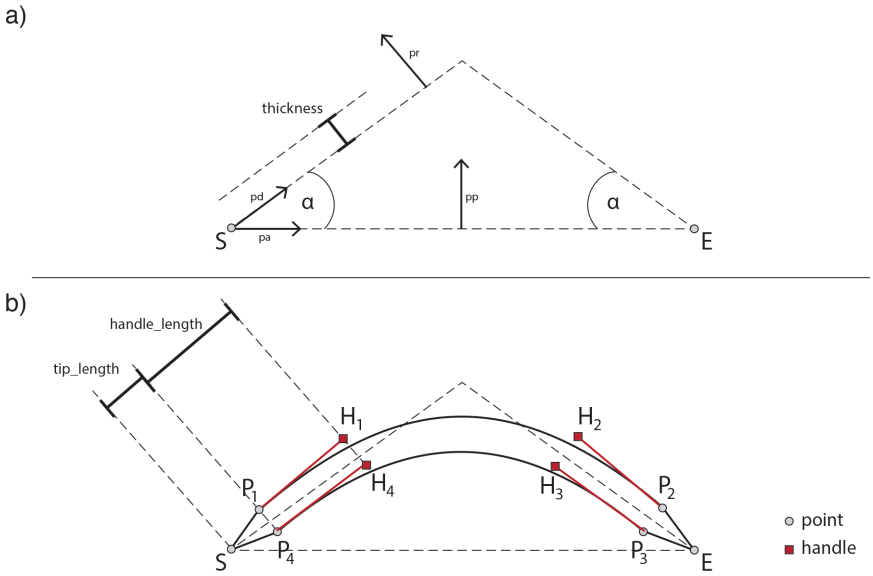

Figure 42. Figure (a) represents the angle α, the *thickness* parameter and $\vec{pd}, \vec{pa}, \vec{pr}, \vec{pp}$ vectors. Figure (b) represents the *tip_length* and *handle_length* parameters. The picture also show P1, P2, P3, P4 control points and the corresponding handles.

Given $\vec{S}, \vec{E}$ the coordinates of the start and the target position of the arrow, and the normalized vectors $\vec{pa}, \vec{pd}, \vec{pp}, \vec{pr}$ which geometrical meaning is depicted in Figure 42 (1), the position coordinates of the points and handles are computed as follow:

$$\vec{P_1} = \vec{S} + tip\_length \cdot \vec{pd} + thickness \cdot \vec{pr}$$

$$\vec{P_4} = \vec{S} + tip\_length \cdot \vec{pd} - thickness \cdot \vec{pr}$$

$$handle\_length = sin(alpha) \cdot (\frac{length(\vec{(}E) - \vec{(}A))}{2} - tip\_length)$$

$$\vec{H_1} = \vec{P}1 + handle\_length \cdot \vec{pd}$$

$$\vec{H_4} = \vec{P}4 + handle\_length \cdot \vec{pd}$$

In a similar way there are computed $\vec{P_2}, \vec{P_3}, \vec{H_2}$ and $\vec{H_3}$. The command to generate the path is then generated using the *SVG* conventions. The "L" prefix denotes a line segment, "C" indicate a cubic *Bézier*, "M" the starting point of the path and "z" close the path. The resulting command is assembled as follow

"**M** $\vec{S}$ **L** $\vec{P_1}$ **C** $\vec{H_1}$ $\vec{H_2}$ $\vec{P_2}$ **L** $\vec{E}$ **L** $\vec{P_3}$ **C** $\vec{H_3}$ $\vec{H_4}$ $\vec{P_4}$ **L** $\vec{S}$ **z**$''$

CHAPTER 9

COMPARISON WITH OTHER SOLUTIONS

9.1 Comparison

In this section we compare the *BranchingSets* technique to the visualization techniques featured in current tools for pathway visualization. In so doing, we highlight the tasks that are enabled by our technique. We explore both the differences inherent to the representation technique adopted as well those that are due to implementation-specific decisions. Our goal is not to provide a comprehensive analysis of all existing tools, but rather to identify the main visualization approaches used to represent biological pathways. Table Table V summarizes the considerations made in this section.

Broadly speaking, current tools make use of one of three primary visual representations: node-link diagrams, node-link diagrams with compound nodes, and adjacency matrices. We look at *Entourage* (**?**), *Reactome Pathway Browser* (30), *VisAnt* (31), *MetaViz* (17) and *VitaPad* (32), each of which leverages a traditional node-link diagram. We also look at *ChiBe* (**?**), which extends the node-link representation by additionally displaying compound nodes that indicate the composition of complexes. However, the tool does not provide an hierarchical inspection of complexes, but instead shows all the proteins contained. This approach might result in a limiting factor for complicated structures, because a nested structure represented with plain compound nodes can easily lead to unwieldy visualizations. The tool has a "merge pathways" functionality which combines the visualization of multiple visualization in the same view. Although this approach is similar to our technique, when multiple pathways are displayed there

is no visual correspondence between a component and the pathways it belongs to. Finally, we examine *BioFabric* (58), a tool that relies on an technique similar to an adjacency matrix.

Reactome Pathway Browser is a tool for visualizing pathway diagrams included in the *Reactome* database. It offers basic navigation and limited interactivity. Different from our method, it presents a very fixed layout and it does not enable the layering of different information levels in the same representation. It makes use of side panels to enable the inspection of complexes structure and analysis data. The *Reactome Pathway Browser* does not enable the simultaneous visualization of multiple pathways but, given a component, it allows the user to navigate through its related pathways.

Entourage is a tool for pathways visualization that enables the exploration of the "cross-talk" between pathways by partitioning the pathway structure. This "artificial partitioning" helps to reduce the complexity of the visualization. Multiple pathways and sub-pathways are represented in isolated views, which preserves the pathway structure but introduces node redundancy.

Other tools represents multiple pathways in the same view, such as *MetaViz*. *MetaViz* visualizes multiple metabolic pathways simultaneously, which allows topological analysis and avoids the duplication of biological components. This method offers different ways to present metabolisms depending on the pathway that the user wishes to focus on. The desired pathway will be the only one to follow a representation which flows from the top to the bottom accordingly the topological ordering of components. Our work has been partially inspired by *MetaViz*, but our technique tries to organize the representation of all pathways according to the topological ordering and it allows an hierarchical inspection of nodes. *BioFabric* also enables the exploration of large biological networks composed by multiple pathways without replicating nodes. This somewhat unconventional tool is meant to visualize extremely large

networks composed by up to thousands of nodes. However, its design is not specifically conceived for pathway visualization and then it does not implements any pathways-specific task.

VisAnt (31) enables a wide range of exploratory questions and topology tasks. The tasks it allows include the search of the shortest paths between two nodes (35) and the individuation of dense nodes. The positions of the nodes in the graph are computed with a relaxing layout algorithm which models the network as a physical entities. This algorithm builds a graphical representation that organize the graph by density of the connections between cluster of nodes. A similar approach has been adopted in this work. Furthermore, *VisAnt* permits a dynamic exploration of a biological network composed by multiple pathways. However, if more than one pathway include the same node, multiple instances of the node will be present in the representation.

Although in the majority of available tools the visual encoding of a biological network remains limited to the conventional graphs representations, pathways analysis tools often decorate node-link representations with experimental data or side information. *VitaPad* (32) allows the user to incorporate microarray data into pathways and offers a flexible visualization. *Entourage* enables the integration of experimental data, whereas *VisAnt* permits to store annotation in the nodes of the network.

Our technique emphasizes the accurate layout of the network topology and includes the ability to dynamically interact with participants to understand the hierarchical nature of the pathways. We believe that existing tools could be enhanced by our approach, and that our visualization technique would enable biologists to more effectively carry out analysis tasks that involve multiple pathways and require an investigation of the hierarchy of the biological participants.

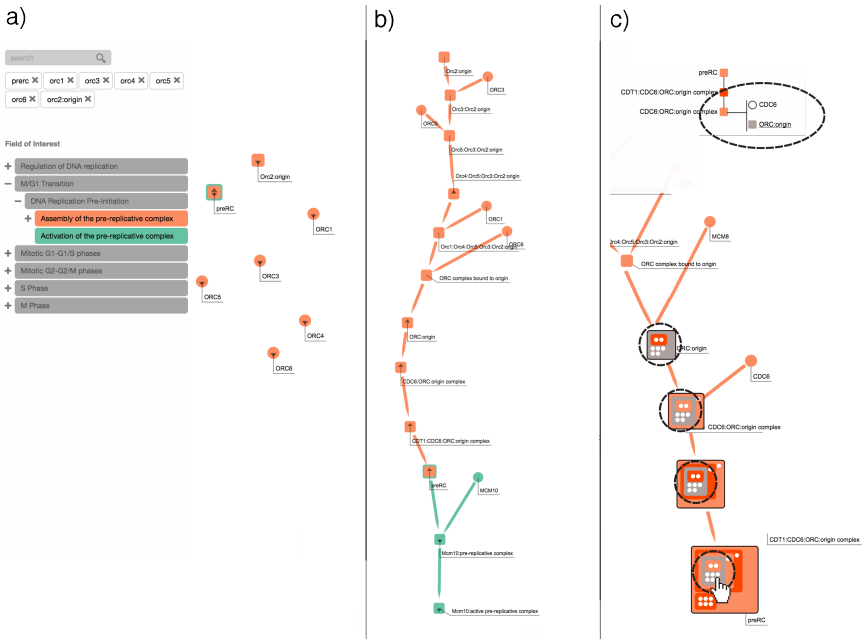

Figure 43. Sequence of screenshot of the first use case

Use Cases

Below we present two use cases to illustrate possible workflows enabled by our application, allowing a user to inspect a subset of entities and interactions within a biological pathway. We also report expert feedback by two biologists who gave us detailed comments regarding our prototype.

Exploring the role of ORC1-6 in assembly of the pre-replication complex

In this use case we explore some of the interactions involved in the assembly of the *pre-replication* complex *(preRC)* (59). We pre-load the prototype application with six pathways involved in the cell cycle. The user expands the menu on the left to view the subpathways of the *M/G1 Transition* pathway and activates the *Assembly of the pre-replicative complex (As-preRC)* and *Activation of the pre-replicative*

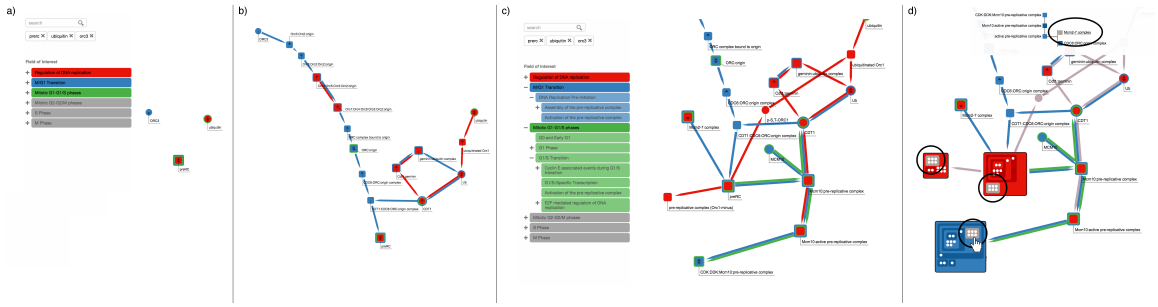

Figure 44. Sequence of screenshots of the second use case

complex (Ac-preRC) subpathways. Then, the user searches for *ORC1-6* and *preRC* in the text field. The visualization is updated with the six *ORC* proteins and the *preRC* one complex that match the keywords entered by the user. The color coding shows that the *ORC* proteins are involved in the *As-preRC* pathway, whereas *preRC* in involved in both *As-preRC* and *Ac-preRC*.

The user then interacts with the prototype application to reveal the intermediate steps from the *ORC* proteins to the *pre-replication* complex. Furthermore, the user expands the network to view the immediate downstream elements of *preRC*, revealing reactions involved in *Ac-preRC* pathway, such as *Mcm10:preRC*, which is expanded again to show the upstream *MCM10* protein and the downstream *Mcm10:active preRC* protein.

The user then expands the upstream of *ORC:origin*, revealing the *MCM8* protein, and expands the downstream complex (*CDC6:ORC:origin*), revealing the upstream protein *CDC6*. Then the user chooses to inspect the hierarchical structure of *preRC*, *ORC:origin*, and other two complexes. By hovering over a sub-complex of *preRC*, the user can see that these same sub-complexes appear in all the visible complex structures (since our application automatically highlights them). This enables the user

user to better understand the steps involved in the *preRC* assembly. This use case is depicted via the three screenshots shown in Figs. Figure 43a–c.

Pathway Interconnections in the Cell Cycle

In this use case, the user selects three pathways from the larger set of pre-loaded pathways: *Regulation of DNA replication (RDR)*, *M-G1 Transition (MG1)* and *Miotic G1-G1/S phases (G1S)*. The user searches for the *preRC* complex, *ORC3* and *Ubiquitin*. (Each pathway is in BioPAX format and was taken from the Reactome database.)

The user first searches for known elements by typing in specific keywords, which initially matches two proteins and one protein complex. By selecting two nodes and dragging between them, the user is able to explore the missing steps between, first, *ORC3* and *preRC* and, second, between *Ubiquitin* and *preRC*. The user can then easily detect which proteins and complexes are involved in all three pathways, such as the *DNA replication factor (Cdt1)* and the *Mini chromosome maintenance complex (MCM2-7)* complex. Using the integrated visualization of the hierarchy of elements within protein complexes, the user can further inspect the structure of *preRC* and the other complexes, showing, for example, the location within the hierarchical structure of *MCM2-7* and its six related polypeptides in all the expanded complexes. This use case is explained via the four screenshots shown in Figs. Figure 44a–d.

TABLE V. COMPARISON BETWEEN *BranchingSets* TECHNIQUE AND OTHER TECHNIQUES INCLUDED IN POPULAR TOOLS FOR PATHWAY VISUALIZATION

	BranchingSets	Entourage	Reactome	VisAnt	VidaPad	ChiBe	BioFabric	MetaViz
Tasks								
Interactive layout rearranging	Yes	No	No	Yes	Yes	Yes	No	Yes
Multiple pathways	Yes	Yes	No	Yes	No	Yes	Yes	Yes
No Node Redundancy	Yes	No	-	No	-	Yes	Yes	Yes
Hierarchical inspection	Yes	Yes	Yes	Yes	No	No	No	No
Interactive Exploration	Yes	Yes	No	Yes	No	No	No	No
Integration with experimental data and notes	No	Yes	No	Yes	Yes	No	No	No

CHAPTER 10

CONCLUSIONS AND FUTURE WORK

In this work we have presented tasks and requirements for a pathway visualization tool. The requisites changes according to the research field and the type of analysis, but they all entail representations of relational-data, categories, hierarchical structures and and managing large data visualizations.

Our work has focused on inspecting the hierarchical structure of biological components and displaying cross-talk between pathways. To provide a better understanding of the biological components and reactions shared among different, we have designed an ad-hoc technique: *BranchingSets*.

BranchingSets is a novel technique which displays category information on node-link diagram and enables the interactive exploration of the network. Since it integrates with the traditional node-link representation, it leads to less cluttered representations with respect to other visualization techniques designed for the same purpose, such as *LineSets* and *Bubble Sets*.

Colors are assigned to pathways for a clear visual encoding. The user can dynamically focus on a limited set of pathways to reduce the amount of information and the number of different colors displayed at the same time. User driven interactions enable a procedural exploration of the network, with the aim of limiting the extension of the graphical representation to avoid visual clutter and minimize excessive edge crossings. Furthermore, the dynamic labeling of components allows the user to focus on the pathway topology in general to reduce the complexity of the visualization. Finally, the inspection of the hierarchical structure of complexes permits the user to change the level of detail of the representation, enabling the simultaneous presence of different abstraction layers. Our initial prototype of *Extended*

LineSets has proven to be an effective representation for a range of tasks common to domain experts in biology and bioinformatics.

BranchingSets has been implemented in a prototype visualization tool for biological pathways. Colors are assigned to pathways for a clear visual encoding. The user can dynamically focus on a limited set of pathways to reduce the amount of information and the number of different colors displayed at the same time. User driven interactions enable a procedural exploration of the network, with the aim of limiting the extension of the graphical representation to avoid visual clutter and minimize excessive edge crossings. Furthermore, the dynamic labeling of components allows the user to focus on the pathway topology in general to reduce the complexity of the visualization. Finally, the inspection of the hierarchical structure of complexes permits the user to change the level of detail of the representation, enabling the simultaneous presence of different abstraction layers. Our initial prototype of *BranchingSets* has proven to be an effective representation for a range of tasks common to domain experts in biology and bioinformatics.

Our prototype will require further work in order to become a full-fledged application for pathway visualization. Currently our tool assumes that all relevant pathways are pre-loaded. The selection of pathways could instead be dynamically loaded from a public database of pathways. Our prototype already reads the *BioPax 3* format, and a future improvement includes better integration with the *Pathway Commons* Web API (3).

Although our technique eliminates node redundancy, multiple edges often originate from the same biological reaction. New interactions and visual paradigms should be designed to better represent this information.

When our tool is used to find the intermediate steps between two nodes in a pathway, it reveals all the possible sequences of *reaction relationships*. This could potentially lead, in certain cases, to a jarring increase in the number of elements displayed in the network, especially when dealing with larger pathways with dense interconnections. Future investigations will include the possibility of choosing only the paths that match specified metrics, or by displaying only the shortest path between any two given nodes.

When the user benefits of feature for an interactive exploration of the network, he reveals all the possible sequences of *reaction relationships*. This leads to a potential uncontrolled extension of the visual representation when dealing with larger pathways. Future investigations will include the possibility of selectively narrowing down the explorations of paths to the ones which match different metrics, such as minimizing the number of steps between any two given nodes. Furthermore, for enabling more complex queries we should leverage *Pathway Logic*, which provide a more complete and robust framework for supporting tasks such like these.

Finally, the inspection of the hierarchical structure of complexes permits the user to change the level of detail of the representation, enabling the simultaneous presence of different abstraction layers.

CITED LITERATURE

1. Joshi-Tope, G., Gillespie, M., Vastrik, I., D'Eustachio, P., Schmidt, E., de Bono, B., Jassal, B., Gopinath, G., Wu, G., Matthews, L., et al.: Reactome: a knowledgebase of biological pathways. Nucleic acids research, 33(suppl 1):D428–D432, 2005.

2. Qiu, Y.-Q.: Kegg pathway database. Encyclopedia of Systems Biology, pages 1068–1069, 2013.

3. Cerami, E. G., Gross, B. E., Demir, E., Rodchenkov, I., Babur, Ö., Anwar, N., Schultz, N., Bader, G. D., and Sander, C.: Pathway commons, a web resource for biological pathway data. Nucleic acids research, 39(suppl 1):D685–D690, 2011.

4. Saraiya, P., North, C., and Duca, K.: Visualizing biological pathways: requirements analysis, systems evaluation and research agenda. Information Visualization, 4(3):191–205, 2005.

5. Herman, I., Melançon, G., and Marshall, M. S.: Graph visualization and navigation in information visualization: A survey. Visualization and Computer Graphics, IEEE Transactions on, 6(1):24–43, 2000.

6. Hu, Z., Mellor, J., Wu, J., Kanehisa, M., Stuart, J. M., and DeLisi, C.: Towards zoomable multidimensional maps of the cell. Nature biotechnology, 25(5):547–554, 2007.

7. Klukas, C. and Schreiber, F.: Dynamic exploration and editing of kegg pathway diagrams. Bioinformatics, 23(3):344–350, 2007.

8. Shannon, P., Markiel, A., Ozier, O., Baliga, N. S., Wang, J. T., Ramage, D., Amin, N., Schwikowski, B., and Ideker, T.: Cytoscape: a software environment for integrated models of biomolecular interaction networks. Genome research, 13(11):2498–2504, 2003.

9. Demir, E., Cary, M. P., Paley, S., Fukuda, K., Lemer, C., Vastrik, I., Wu, G., D'Eustachio, P., Schaefer, C., Luciano, J., et al.: The biopax community standard for pathway data sharing. Nature biotechnology, 28(9):935–942, 2010.

10. Kanehisa, M. and Goto, S.: Kegg: kyoto encyclopedia of genes and genomes. Nucleic acids research, 28(1):27–30, 2000.

CITED LITERATURE (continued)

11. Barba, M., Dutoit, R., Legrain, C., and Labedan, B.: Identifying reaction modules in metabolic pathways: bioinformatic deduction and experimental validation of a new putative route in purine catabolism. BMC Syst. Biol., 7(1):99, 2013.

12. Gehlenborg, N., O'Donoghue, S. I., Baliga, N. S., Goesmann, A., Hibbs, M. A., Kitano, H., Kohlbacher, O., Neuweger, H., Schneider, R., Tenenbaum, D., and Gavin, A.-C.: Visualization of omics data for systems biology. Nat. Methods, 7(3 Suppl):S56–68, March 2010.

13. Felciano, R. M., Bavari, S., Richards, D. R., Billaud, J.-n., Warren, T., Panchal, R., and Krämer, A.: Predictive systems biology approach to broad-spectrum, host-directed drug target discovery in infectious diseases. In Pacific Symposium on Biocomputing, volume 18, pages 17–28. World Scientific, 2013.

14. Krämer, A., Green, J., Pollard, J., and Tugendreich, S.: Causal analysis approaches in ingenuity pathway analysis (ipa). Bioinformatics, page btt703, 2013.

15. Lex, A., Partl, C., Kalkofen, D., Streit, M., Gratzl, S., Wassermann, A. M., Schmalstieg, D., and Pfister, H.: Entourage: Visualizing relationships between biological pathways using contextual subsets. Visualization and Computer Graphics, IEEE Transactions on, 19(12):2536–2545, 2013.

16. Heer, J. and Robertson, G.: Animated transitions in statistical data graphics. IEEE Trans. Vis. Comput. Graph., 13(6):1240–1247, November 2007.

17. Bourqui, R., Cottret, L., Lacroix, V., Auber, D., Mary, P., Sagot, M.-F., and Jourdan, F.: Metabolic network visualization eliminating node redundance and preserving metabolic pathways. BMC systems biology, 1(1):29, 2007.

18. Kimelman, D., Leban, B., Roth, T., and Zernik, D.: Reduction of visual complexity in dynamic graphs. pages 218–225, 1995.

19. Holten, D.: Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. Visualization and Computer Graphics, IEEE Transactions on, 12(5):741–748, 2006.

20. Ware, C.: Information Visualization: Perception for Design. San Francisco, CA, USA, Morgan Kaufmann Publishers Inc., 2004.

CITED LITERATURE (continued)

21. Itoh, T., Muelder, C., Ma, K.-L., and Sese, J.: A hybrid space-filling and force-directed layout method for visualizing multiple-category graphs. In Visualization Symposium, 2009. PacificVis' 09. IEEE Pacific, pages 121–128. IEEE, 2009.

22. Vehlow, C., Reinhardt, T., and Weiskopf, D.: Visualizing fuzzy overlapping communities in networks. Visualization and Computer Graphics, IEEE Transactions on, 19(12):2486–2495, 2013.

23. Riche, N. H. and Dwyer, T.: Untangling euler diagrams. IEEE Transactions on Visualization and Computer Graphics, 16(6):1090–1099, 2010.

24. Gansner, E. R., Hu, Y., and Kobourov, S. G.: Visualizing graphs and clusters as maps. IEEE Computer Graphics and Applications, 30(6):54, 2010.

25. Collins, C., Penn, G., and Carpendale, S.: Bubble sets: Revealing set relations with isocontours over existing visualizations. Visualization and Computer Graphics, IEEE Transactions on, 15(6):1009–1016, 2009.

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

27. Meulemans, W., Riche, N. H., Speckmann, B., Alper, B., and Dwyer, T.: Kelpfusion: A hybrid set visualization technique. Visualization and Computer Graphics, IEEE Transactions on, 19(11):1846–1858, 2013.

28. Lambert, A., Queyroi, F., and Bourqui, R.: Visualizing patterns in node-link diagrams. In Information Visualisation (IV), 2012 16th International Conference on, pages 48–53. IEEE, 2012.

29. Stephenson, K.: Introduction to circle packing. The Theory of Discrete Analytic Functions, Cambridge University Press, Cambridge, 2005.

30. Croft, D., Mundo, A. F., Haw, R., Milacic, M., Weiser, J., Wu, G., Caudy, M., Garapati, P., Gillespie, M., Kamdar, M. R., et al.: The reactome pathway knowledgebase. Nucleic acids research, 42(D1):D472–D477, 2014.

31. Hu, Z., Mellor, J., Wu, J., and DeLisi, C.: Visant: an online visualization and analysis tool for biological interaction data. BMC bioinformatics, 5(1):17, 2004.

CITED LITERATURE (continued)

32. Holford, M., Li, N., Nadkarni, P., and Zhao, H.: Vitapad: visualization tools for the analysis of pathway data. Bioinformatics, 21(8):1596–1602, 2005.

33. Babur, O., Dogrusoz, U., Demir, E., and Sander, C.: Chibe: interactive visualization and manipulation of biopax pathway models. Bioinformatics, 26(3):429–431, 2010.

34. Lex, A., Streit, M., Kruijff, E., and Schmalstieg, D.: Caleydo: Design and evaluation of a visual analysis framework for gene expression data in its biological context. In Pacific Visualization Symposium (PacificVis), 2010 IEEE, pages 57–64. ieeexplore.ieee.org, March 2010.

35. Hu, Z., Chang, Y.-C., Wang, Y., Huang, C.-L., Liu, Y., Tian, F., Granger, B., and DeLisi, C.: Visant 4.0: integrative network platform to connect genes, drugs, diseases and therapies. Nucleic acids research, 41(W1):W225–W231, 2013.

36. Talcott, C.: Pathway logic. In Formal Methods for Computational Systems Biology, pages 21–53. Springer, 2008.

37. Clavel, M., Durán, F., Eker, S., Lincoln, P., Martı-Oliet, N., Meseguer, J., and Quesada, J. F.: Maude: Specification and programming in rewriting logic. Theoretical Computer Science, 285(2):187–243, 2002.

38. Heiner, M., Koch, I., and Will, J.: Model validation of biological pathways using petri netsdemonstrated for apoptosis. Biosystems, 75(1):15–28, 2004.

39. Saraiya, P., North, C., and Duca, K.: Visualizing biological pathways: requirements analysis, systems evaluation and research agenda. Information Visualization, 4(3):191–205, 2005.

40. Archambault, D., Purchase, H. C., and Pinaud, B.: Animation, small multiples, and the effect of mental map preservation in dynamic graphs. Visualization and Computer Graphics, IEEE Transactions on, 17(4):539–552, 2011.

41. Holten, D. and van Wijk, J. J.: A user study on visualizing directed edges in graphs. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pages 2299–2308. ACM, 2009.

42. Santamaría, R. and Therón, R.: Overlapping clustered graphs: co-authorship networks visualization. In Smart Graphics, pages 190–199. Springer, 2008.

CITED LITERATURE (continued)

43. Börner, K., Dall'Asta, L., Ke, W., and Vespignani, A.: Studying the emerging global brain: Analyzing and visualizing the impact of co-authorship teams. Complexity, 10(4):57–67, 2005.

44. Tufte, E. R. and Graves-Morris, P.: The visual display of quantitative information, volume 2. Graphics press Cheshire, CT, 1983.

45. Paduano, F., Dang, T., Murray, P., and Forbes, A. G.: Extended linesets: A visualization technique for the interactive inspection of biological pathways. In Proceedings of the 5th IEEE Symposium on Biological Data Visualization. IEEE, 2015. To appear.

46. Jacomy, M.: I want hue - colors for data scientists. Retrieved Apr 2015, from `http://tools.medialab.sciences-po.fr/iwanthue/`.

47. Harrower, M. and Brewer, C. A.: Colorbrewer. org: an online tool for selecting colour schemes for maps. The Cartographic Journal, 40(1):27–37, 2003.

48. Heer, J. and Robertson, G. G.: Animated transitions in statistical data graphics. Visualization and Computer Graphics, IEEE Transactions on, 13(6):1240–1247, 2007.

49. Bostock, Michael Ogievetsky, V. H. J.: D3.js - data-driven documents. Retrieved Feb 2015, from `http://d3js.org/`.

50. Sean, B., Frank, v. H., Jim, H., Ian, H., Deborah, L. M., Peter, F. P.-S., and Lynn, A. S.: Owl web ontology language reference. Retrieved Apr 2015, from `http://www.w3.org/TR/owl-ref/`.

51. Barnes, J. and Hut, P.: A hierarchical o (n log n) force-calculation algorithm. 1986.

52. Shneiderman, B.: Tree visualization with tree-maps: 2-d space-filling approach. ACM Transactions on graphics (TOG), 11(1):92–99, 1992.

53. Korf, R. E.: Optimal rectangle packing: New results. In ICAPS, pages 142–149, 2004.

54. Huang, W. and Chen, D.: An efficient heuristic algorithm for rectangle-packing problem. Simulation Modelling Practice and Theory, 15(10):1356–1365, 2007.

55. Leung, J. Y., Tam, T. W., Wong, C. S., Young, G. H., and Chin, F. Y.: Packing squares into a square. Journal of Parallel and Distributed Computing, 10(3):271–275, 1990.

CITED LITERATURE (continued)

56. Christensen, J., Marks, J., and Shieber, S.: An empirical study of algorithms for point-feature label placement. ACM Transactions on Graphics (TOG), 14(3):203–232, 1995.

57. Samet, H.: The quadtree and related hierarchical data structures. ACM Computing Surveys (CSUR), 16(2):187–260, 1984.

58. Longabaugh, W. J.: Combing the hairball with biofabric: a new approach for visualization of large networks. BMC bioinformatics, 13(1):275, 2012.

59. Yoshida, K.: Hierarchical order of initiator protein assembly in pre-replication complex. Protein Conformation: New Research, page 201, 2008.

60. Ellson, J., Gansner, E., Koutsofios, L., North, S. C., and Woodhull, G.: Graphvizopen source graph drawing tools. pages 483–484, 2002.

61. Xia, J. and Wishart, D. S.: Metpa: a web-based metabolomics tool for pathway analysis and visualization. Bioinformatics, 26(18):2342–2344, 2010.

62. Paley, S. M. and Karp, P. D.: The pathway tools cellular overview diagram and omics viewer. Nucleic Acids Research, 34(13):3771–3778, 2006.

63. Walworth, Bosco, O.: Rb-e2f1 pathway. Retrieved Mar 2015, from `http://www.reactome.org/PathwayBrowser/#DIAGRAM=453279`, 2005.

64. Kitware, Inc.: The Visualization Toolkit User's Guide, January 2003.

65. Grinstein, G., Keim, D., and Ward, M.: Information visualization, visual data mining, and its application to drug design. IEEE Visualization 2002 Course #1 Notes, October 2002.

66. Max, N.: Optical models for direct volume rendering. IEEE Transactions on Visualization and Computer Graphics, 1(2):99–108, June 1995.

67. Ware, C.: Information Visualization: Perception for Design. Morgan Kaufmann Publishers, second edition, 2004.

68. Kindlmann, G.: Semi-Automatic Generation of Transfer Functions for Direct Volume Rendering. Master's thesis, Cornell University, 1999.

69. Levoy, M.: Display of Surfaces from Volume Data. Doctoral dissertation, University of North Carolina at Chapel Hill, 1989.

CITED LITERATURE (continued)

70. Lorensen, W. E. and Cline, H. E.: Marching cubes: A high resolution 3d surface construction algorithm. In Computer Graphics (Proceedings of SIGGRAPH 87), volume 21, pages 163–169, July 1987.

71. Nielson, G. M. and Hamann, B.: The asymptotic decider: Removing the ambiguity in marching cubes. In Visualization '91, pages 83–91, 1991.

72. Sarkar, M. and Brown, M. H.: Graphical fisheye views of graphs. In Proceedings of the SIGCHI conference on Human factors in computing systems, pages 83–91. ACM, 1992.

73. Dinkla, K., Westenberg, M. A., and van Wijk, J. J.: Compressed adjacency matrices: Untangling gene regulatory networks. IEEE Trans. Vis. Comput. Graph., 18(12):2457–2466, December 2012.

74. Longabaugh, W. J. R.: Combing the hairball with BioFabric: a new approach for visualization of large networks. BMC Bioinformatics, 13:275, 27 October 2012.

75. Elliott, B., Kirac, M., Cakmak, A., Yavas, G., Mayes, S., Cheng, E., Wang, Y., Gupta, C., Ozsoyoglu, G., and Meral Ozsoyoglu, Z.: PathCase: pathways database system. Bioinformatics, 24(21):2526–2533, 1 November 2008.

76. Funahashi, A., Matsuoka, Y., Jouraku, A., Morohashi, M., Kikuchi, N., and Kitano, H.: CellDesigner 3.5: A versatile modeling tool for biochemical networks. Proc. IEEE, 96(8):1254–1265, August 2008.

77. Latendresse, M., Paley, S., and Karp, P. D.: Browsing metabolic and regulatory networks with BioCyc. Methods Mol. Biol., 804:197–216, 2012.

78. Salazar, G. A., Meintjes, A., Mazandu, G. K., Rapanoël, H. A., Akinola, R. O., and Mulder, N. J.: A web-based protein interaction network visualizer. BMC Bioinformatics, 15:129, 6 May 2014.

79. Pcviz. Retrieved Jan 2015, from `http://www.pathwaycommons.org/pcviz/`.

80. Breitkreutz, B.-J., Stark, C., and Tyers, M.: Osprey: a network visualization system. Genome Biol., 4(3):R22, 27 February 2003.

81. Fjukstad, B.: Kvik: Interactive exploration of genomic data from the NOWAC postgenome biobank. 2014.

CITED LITERATURE (continued)

82. Suderman, M. and Hallett, M.: Tools for visually exploring biological networks. Bioinformatics, 23(20):2651–2659, 15 October 2007.

83. Wang, Y.-T., Huang, Y.-H., Chen, Y.-C., Hsu, C.-L., and Yang, U.-C.: PINT: Pathways INtegration tool. Nucleic Acids Res., 38(Web Server issue):W124–31, July 2010.

84. Hanahan, D. and Weinberg, R. A.: Hallmarks of cancer: the next generation. cell, 144(5):646–674, 2011.

85. Cairns, R. A., Harris, I. S., and Mak, T. W.: Regulation of cancer cell metabolism. Nature Reviews Cancer, 11(2):85–95, 2011.

86. Kitano, H.: Systems biology: a brief overview. Science, 295(5560):1662–1664, 2002.

87. Luo, J., Manning, B. D., and Cantley, L. C.: Targeting the pi3k-akt pathway in human cancer-rationale and promise. Cancer cell, 4(4):257–262, 2003.

88. Reya, T., Morrison, S. J., Clarke, M. F., and Weissman, I. L.: Stem cells, cancer, and cancer stem cells. nature, 414(6859):105–111, 2001.

89. Raju, R., Nanjappa, V., Balakrishnan, L., Radhakrishnan, A., Thomas, J. K., Sharma, J., Tian, M., Palapetta, S. M., Subbannayya, T., Sekhar, N. R., Muthusamy, B., Goel, R., Subbannayya, Y., Telikicherla, D., Bhattacharjee, M., Pinto, S. M., Syed, N., Srikanth, M. S., Sathe, G. J., Ahmad, S., Chavan, S. N., Kumar, G. S. S., Marimuthu, A., Prasad, T. S. K., Harsha, H. C., Rahiman, B. A., Ohara, O., Bader, G. D., Sujatha Mohan, S., Schiemann, W. P., and Pandey, A.: NetSlim: high-confidence curated signaling maps. Database, 2011:bar032, 29 September 2011.

90. Dogrusoz, U., Erson, E. Z., Giral, E., Demir, E., Babur, O., Cetintas, A., and Colak, R.: PATIKAweb: a web interface for analyzing biological pathways through advanced querying and visualization. Bioinformatics, 22(3):374–375, 1 February 2006.

91. Nikitin, A., Egorov, S., Daraselia, N., and Mazo, I.: Pathway studio—the analysis and navigation of molecular networks. Bioinformatics, 19(16):2155–2157, 1 November 2003.

92. Köhler, J., Baumbach, J., Taubert, J., Specht, M., Skusa, A., Rüegg, A., Rawlings, C., Verrier, P., and Philippi, S.: Graph-based analysis and visualization of experimental results with ONDEX. Bioinformatics, 22(11):1383–1390, 1 June 2006.

CITED LITERATURE (continued)

93. Baitaluk, M., Sedova, M., Ray, A., and Gupta, A.: BiologicalNetworks: visualization and analysis tool for systems biology. Nucleic Acids Res., 34(Web Server issue):W466–71, 1 July 2006.

94. Kono, N., Arakawa, K., Ogawa, R., Kido, N., Oshita, K., Ikegami, K., Tamaki, S., and Tomita, M.: Pathway projector: web-based zoomable pathway browser using KEGG atlas and google maps API. PLoS One, 4(11):e7710, 11 November 2009.

95. Xia, J., Benner, M. J., and Hancock, R. E. W.: NetworkAnalyst–integrative approaches for protein-protein interaction network analysis and visual exploration. Nucleic Acids Res., 42(Web Server issue):W167–74, July 2014.

96. Junker, B. H., Klukas, C., and Schreiber, F.: VANTED: a system for advanced data analysis and visualization in the context of biological networks. BMC Bioinformatics, 7:109, 6 March 2006.

97. Tufte, E. R.: ENVISIONING INFORMATION. Optom. Vis. Sci., 68(4):322, April 1991.

98. Barsky, A., Munzner, T., Gardy, J., and Kincaid, R.: Cerebral: visualizing multiple experimental conditions on a graph with biological context. IEEE Trans. Vis. Comput. Graph., 14(6):1253–1260, November 2008.

99. Shannon, P., Markiel, A., Ozier, O., Baliga, N. S., Wang, J. T., Ramage, D., Amin, N., Schwikowski, B., and Ideker, T.: Cytoscape: a software environment for integrated models of biomolecular interaction networks. Genome Res., 13(11):2498–2504, November 2003.

100. Smoot, M. E., Ono, K., Ruscheinski, J., Wang, P.-L., and Ideker, T.: Cytoscape 2.8: new features for data integration and network visualization. Bioinformatics, 27(3):431–432, 1 February 2011.

101. Partl, C., Lex, A., Streit, M., Kalkofen, D., Kashofer, K., and Schmalstieg, D.: enroute: Dynamic path extraction from biological pathway maps for in-depth experimental data analysis. In Biological Data Visualization (BioVis), 2012 IEEE Symposium on, pages 107–114, October 2012.

VITA

NAME: Francesco Paduano

EDUCATION: Bachelor's degree in Computer Science, Politecnico di Milano, 2013

Master of Science in Computer Science, University of Illinois at Chicago, 2015

Alta Scuola Politecnica, Politecnico di Milano and Torino, 2015

Laura Magistrale in Computer Science, Politecnico Di Milano, 2015