Visualisation Support for Biological Bayesian Network Inference

A thesis
submitted in partial fulfilment
of the requirements of
Edinburgh Napier University,
for the award of
Doctor of Philosophy

by

Athanasios Vogogias

Edinburgh Napier University

August 2019

Abstract

Extracting valuable information from the visualisation of biological data and turning it into a network model is the main challenge addressed in this thesis. Biological networks are mathematical models that describe biological entities as nodes and their relationships as edges. Because they describe patterns of relationships, networks can show how a biological system works as a whole. However, network inference is a challenging optimisation problem impossible to resolve computationally in polynomial time. Therefore, the computational biologists (*i.e.* modellers) combine clustering and heuristic search algorithms with their tacit knowledge to infer networks. Visualisation can play an important role in supporting them in their network inference workflow. The main research question is: *"How can visualisation support modellers in their workflow to infer networks from biological data?"* To answer this question, it was required to collaborate with computational biologists to understand the challenges in their workflow and form research questions. Following the nested model methodology helped to characterise the domain problem, abstract data and tasks, design effective visualisation components and implement efficient algorithms. Those steps correspond to the four levels of the nested model for collaborating with domain experts to design effective visualisations. We found that visualisation can support modellers in three steps of their workflow. (a) To select variables, (b) to infer a consensus network and (c) to incorporate information about its dynamics.

To select variables (a), modellers first apply a hierarchical clustering algorithm which produces a dendrogram (*i.e.* a tree structure). Then they select a similarity threshold (height) to cut the tree so that branches correspond to clusters. However, applying a single-height similarity threshold is not effective for clustering heterogeneous multidimensional data because clusters may exist at different heights. The research question is: **Q1** *"How to provide visual support for the effective hierarchical clustering of many multidimensional variables?"* To answer this question, MLCut, a novel visualisation tool was developed to enable the application of multiple similarity thresholds. Users can interact with a representation of the dendrogram, which is coordinated with a view of the original multidimensional data, select branches of the tree at different heights and explore different clustering scenarios. Using MLCut in two case studies has shown that this method provides transparency in the clustering process and enables the effective allocation of variables into clusters.

Selected variables and clusters constitute nodes in the inferred network. In the second step (b), modellers apply heuristic search algorithms which sample a solution space consisting of all

possible networks. The result of each execution of the algorithm is a collection of high-scoring Bayesian networks. The task is to guide the heuristic search and help construct a consensus network. However, this is challenging because many network results contain different scores produced by different executions of the algorithm. The research question is: **Q2** *"How to support the visual analysis of heuristic search results, to infer representative models for biological systems?"* BayesPiles, a novel interactive visual analytics tool, was developed and evaluated in three case studies to support modellers explore, combine and compare results, to understand the structure of the solution space and to construct a consensus network.

As part of the third step (c), when the biological data contain measurements over time, heuristics can also infer information about the dynamics of the interactions encoded as different types of edges in the inferred networks. However, representing such multivariate networks is a challenging visualisation problem. The research question is: **Q3** *"How to effectively represent information related to the dynamics of biological systems, encoded in the edges of inferred networks?"* To help modellers explore their results and to answer Q3, a human-centred crowdsourcing experiment took place to evaluate the effectiveness of four visual encodings for multiple edge types in matrices. The design of the tested encodings combines three visual variables: position, orientation, and colour. The study showed that orientation outperforms position and that colour is helpful in most tasks. The results informed an extension to the design of BayePiles, which modellers evaluated exploring dynamic Bayesian networks. The feedback of most participants confirmed the results of the crowdsourcing experiment.

This thesis focuses on the investigation, design, and application of visualisation approaches for gaining insights from biological data to infer network models. It shows how visualisation can help modellers in their workflow to select variables, to construct representative network models and to explore their different types of interactions, contributing in gaining a better understanding of how biological processes within living organisms work.

Table of Contents

Acknowledgments

I would like to thank Edinburgh Napier University for funding my research. Also, I would like to thank the School of Computing for providing technical support, office space to work as well as space for collaboration and exchange of ideas with colleagues. I would like to express my sincere gratitude and deep appreciation to my supervisors Prof. Jessie Kennedy and Dr Daniel Archambault for their guidance and insightful mentoring, for motivating me and for helping me to grow as a researcher. Without their invaluable support, I would not be able to overcome difficulties and achieve my research goals. Also, I would like to thank all my collaborators, and especially Dr Anne Smith, for believing in the value of my research and for their continuous engagement and enthusiastic interest in applying my visualisation methods to their biological data. Without their feedback, my research would be meaningless. I would like to thank Dr Benjamin Bach for the inspiration, his comments and his generous assistance during our collaboration in the second half of my PhD. Finally, I would also like to thank my fellow PhD students for their companionship in this long and often lonely journey of scientific inquiry and my family and friends that always stand by me tolerating my ignorance. Their patience and love overwhelm me with satisfaction and joy.

List of Figures

List of Tables

1 Introduction

Networks are a mathematical model composed of nodes and edges useful for studying complex systems, including biological systems. Nodes in network models represent entities and edges between nodes represent interactions between entities. Instead of looking at single interactions, networks can describe multiple interactions between entities (*i.e.* patterns) in a clear and easy to interpret way, summarising how a system works as a whole [86]. Patterns of interactions between biological entities can be inferred from the data using heuristic search algorithms and depicted as a network [167]. However, scientists struggle to cope with the volume, the complexity and the dynamics of biological data [59], and the discipline of biological visualisation, which is a branch of bioinformatics, has emerged to help represent complex data and networks effectively [72]. This thesis argues that data visualisation can help scientists to gain insights about the state of the biological system and help infer networks that describe the interactions between its entities. Also, this thesis presents visualisation tools that support scientists in their workflow to infer networks from biological data.

The domain scientists or experts who analyse data to infer networks are called *modellers*. In the domain of biology, the modellers are computational biologists (*i.e.* bioinformaticians), so throughout this thesis, these terms are used interchangeably. For this thesis, we collaborated with a group of 5 computational biologists who wanted to use visualisation to support their network inference workflow. These collaborators were also the end-users of the presented visualisation tools. One of them was a senior lecturer specialised in probabilistic methods for inferring complex biological networks. The other 4 were research students that worked on individual projects supervised by the senior lecturer. Also, during the first period of this research, a senior statistician specialised in computational methods for analysing genetic data provided useful feedback. Our collaborators not only helped to identify requirements when the features of the visualisation tools were designed but also helped with their evaluation.

Using visualisation along with data mining techniques to extract valuable information from biological data, and to infer networks, is the main focus of this thesis. In Section 2.4, we describe the four levels of the nested model methodology [132], which helped to make this research successful. In particular, the nested model was used to help understand the characteristics of the domain problem, to abstract data and tasks, to design effective encodings and to implement efficient algorithms. The contributions of this thesis are presented as three separate design studies, each following the nested model methodology. In particular, we found that visualisation can help our modellers in three steps of their network inference workflow: (a) to organise variables into groups, (b) to infer representative networks and (c) to explore the different types of interactions which reveal information about the dynamics of the underlying biological system. The challenges involved in each of these workflow steps have led to the formation of the three research questions discussed in Section 1.4.

The different workflow steps require the visualisation of different types of data. For the first step (a), the data sets consist of potentially thousands of variables measured over time or in different experimental conditions, resulting in a multidimensional data set. Although such data sets can be gathered by the modellers as part of their experimental process, in this thesis, we used data sets stored in publicly accessible repositories. For the second step of the workflow (b), the data sets consist of collections of potentially thousands of networks. The networks were generated by our collaborators using BANJO [167], a package that contains different heuristic search algorithms for network inference. For the third step (c), the data generation process is similar to the second step, but the networks are more complex as they contain multiple edge types. More details about the data are explained in Section 2.2.

1.1 Biological Networks

Biological systems are composed of individual parts or entities which are interlinked. One of the challenges that scientists face when they analyse biological data is to understand the interactions between these entities. Networks are used to describe complex interactions simply and clearly, providing an abstract view of a biological system [86]. For instance, networks can describe interactions between chemical molecules within single cells by integrating genetic and environmental factors which both affect cell functions and the phenotype of a living organism [71]. The depiction of interactions between biological entities as a network is one of the first steps towards making sense of the data [138]. Thus, the study of networks has become intrinsic to modern biological research, finding its way into many applications in medicine, neuroscience, genetics, ecology etc.

In general, biological networks take their name from the different types of variables or interactions they model. For instance, the circuitry inside a living cell is commonly described by three types of molecular network: a) gene regulatory networks (GRNs), b) protein-protein interaction networks (PPIs) and c) metabolic pathways [138]. However, biological networks often include variables and interactions of different types, which are interlinked and influence each other [4]. For instance, biological networks often describe probabilistic relationships between aggregates of variables, experimental conditions, or other biological entities, providing a more high-level view of the system [54, 166]. This level of abstraction is particularly important when there is a lot of uncertainty about the underlying biological mechanisms that govern the behaviour of the system, and a more qualitative understanding is required. Biological networks are most useful when they are simple and can describe biological processes in a minimal and easy to interpret way.

1.2 Network Inference Workflow

The concept of *network inference* refers to the process followed by modellers for deriving networks from biological data [54]. Given a biological data set, the goal of the modeller is to find the network that optimally represents the interactions between biological entities found in the underlying data. During network inference, the modeller plays an active role in combining computational and mathematical heuristic methods with their tacit knowledge to identify interactions between biological entities that define the structure of the presented final network [54]. The final network is also known as the *consensus network* because it is often derived from the exploration, combination and comparison of multiple networks. The possible network solutions that modellers consider when they construct a consensus network are called *candidate* networks. In other words, during network inference, it is the modellers who decide on a method that determines the structure of the consensus network, combining edges from different candidate networks.

Inferring networks that describe the real interactions between biological entities is a challenging optimisation problem because many variables can affect the status of the biological system at any given moment [54]. A model that incorporates the effect of all these variables is difficult to find because the search space that includes all possible network solutions increases super-exponentially every time a new variable is considered [91]. Also, collecting biological measurements is financially expensive and technically complicated [91, 41]. This results in measurements which might be noisy, sparse, and can contain missing values. These factors add to the uncertainty about the state of the complex natural system (*i.e.* the biological network) which hence becomes inherently stochastic. Network inference cannot be resolved in polynomial time and finding an exact solution to the underlying computational problem is known to be NP-hard [48, 44]. Therefore, modellers use clustering and approximate heuristic search methods to find candidate networks.

Instead of considering all possible variables, modellers constrain the search space of solutions to the problem by filtering or aggregating variables so that only those that affect the status of the biological system are included in the model. Also, modellers use heuristic search algorithms that sample the search space of all possible network solutions. Each of these networks is assigned with a score that indicates how well it fits the underlying data. As part of their network inference workflow, modellers supervise the variable selection step, guide the heuristic search and then decide on a method that determines the structure of the consensus network. To construct a consensus network modellers often explore, combine and compare multiple candidate network solutions generated and assessed by the algorithm.

There are different steps involved in the process of inferring network models but as shown in Figure 1.1, the workflow usually starts with (1) forming hypotheses about the biological system based on previous evidence followed by the (2) design and execution of experiments for collecting relevant biological data measurements. Those measurements often contain variables with multiple dimensions or time points. Raw data (3) get pre-processed, cleaned and normalised to enable comparisons. Then modellers (4) reduce the number of variables selecting only those that become nodes in the network. Usually, this is done by removing redundant variables, or by aggregating similar variables after organising them into groups using clustering algorithms. The next step (5) involves the execution of heuristic search algorithms to infer the structure of the network. During this step, the modeller guides the heuristic search and decides on a method for constructing a consensus network. When the data set involves measurements over time (6), information about the dynamics of the system can be incorporated into the different types of edges in the network. Finally, the modeller shares the network with other scientists (7) who interpret this new knowledge, motivating future research and forming new hypotheses.

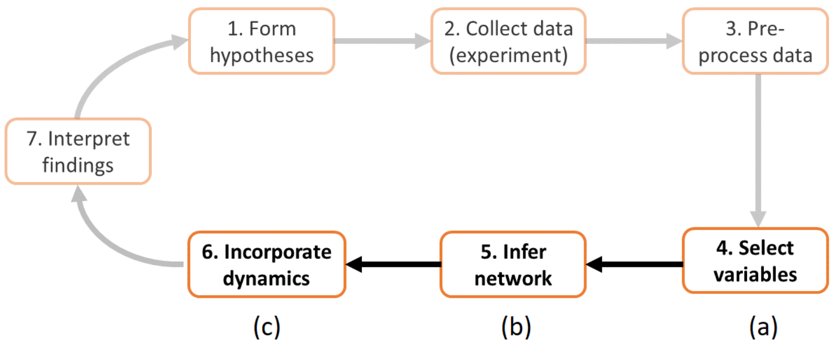


Figure 1.1: The workflow that describes the process of network inference with the three analysis challenges highlighted. Step (a): reduce the number of variables, Step (b): guide the heuristic search and construct a consensus network and Step (c): incorporate the dynamics about the system in the network model.

1.3 Biological Analysis Challenges

The workflow that computational biologists (*i.e.* modellers) often follow to infer network models involves the following three steps that appear highlighted in Figure 1.1 (a), (b) and (c). Those steps constitute the main biological analysis challenges targeted in this thesis.

- **Step (a):** Narrowing the search space by reducing the number of variables is an essential step for improving the performance of the computationally expensive heuristic search algorithms that follow. The challenge is to find the most important variables in large and complex data sets with multiple dimensions or time points. A hierarchical clustering algorithm is used to reduce the number of variables included in the model. The algorithm produces a tree structure which is called the *dendrogram*. Modellers have to inspect the dendrogram and the multidimensional data to decide which branches correspond to clusters. Selected variables, or clusters of variables, become nodes in the network model.

- **Step (b):** Heuristic search algorithms sample the search space of all possible networks generating collections of candidate networks. Also, multiple executions of the algorithm take place using different parameter settings and a network score that represents the fitness to the data. Modellers guide the heuristic search and decide on a method that determines the consensus network after consulting the candidate networks generated by the algorithm. However, it is difficult for modellers to explore, combine and compare large collections of candidate networks to infer a consensus network. To overcome this challenge, modellers first need to acquire an understanding of the shape of the solution space, consisting of the many high-scoring candidate networks produced by different heuristic search runs.

- **Step (c):** Modellers also attempt to infer the dynamics of the biological systems. Specialised heuristic search algorithms can analyse time-series data to infer information about the dynamics of the underlying biological system and incorporate this information into the network results. These algorithms produce collections of multivariate networks with multiple edge types which are hard to represent and explore. Therefore, supporting modellers in studying the characteristics of different edge types in the search results is important for understanding how biological processes evolve.

1.4 Research Questions

In the aforementioned three steps of modellers' workflow (Section 1.3), visualisation can play an important role in overcoming biological analysis challenges. This thesis focuses on the investigation, design, and application of visualisation approaches for supporting computational biologists in their workflow of inferring network models. The main research question that initiated the research for this thesis is the following: ***"How can visualisation help modellers in their workflow to infer networks from biological data"*** which immediately leads to the following questions:

- **Q1** ***"How to provide visual support for the effective hierarchical clustering of many multidimensional variables?"*** The first visual analysis challenge is to help modellers explore multidimensional data sets of multiple variables and allocate those variables into groups, using the method of hierarchical clustering. Selected variables, or clusters of variables, become nodes in the inferred network model.

- **Q2** ***"How to support the visual analysis of heuristic search results, to infer representative models for biological systems?"*** The second challenge is to help modellers guide the heuristic search and decide on a method that determines the final consensus network. Modellers are required to understand the shape of the solution space after sampling this space using a heuristic search algorithm, executed multiple times. The visual analysis challenge is to explore, combine and compare potentially hundreds of candidate networks for inferring the structure of a representative final consensus network.

- **Q3** ***"How to effectively represent information related to the dynamics of biological systems, encoded in the edges of inferred networks?"*** The third challenge is to help modellers understand the dynamics of the underlying biological system through the understanding of heuristic search results, which consist of networks with multiple types of edges. The visual analysis challenge is to identify effective visual encodings for multivariate networks in which the multivariate data is associated with the edges.

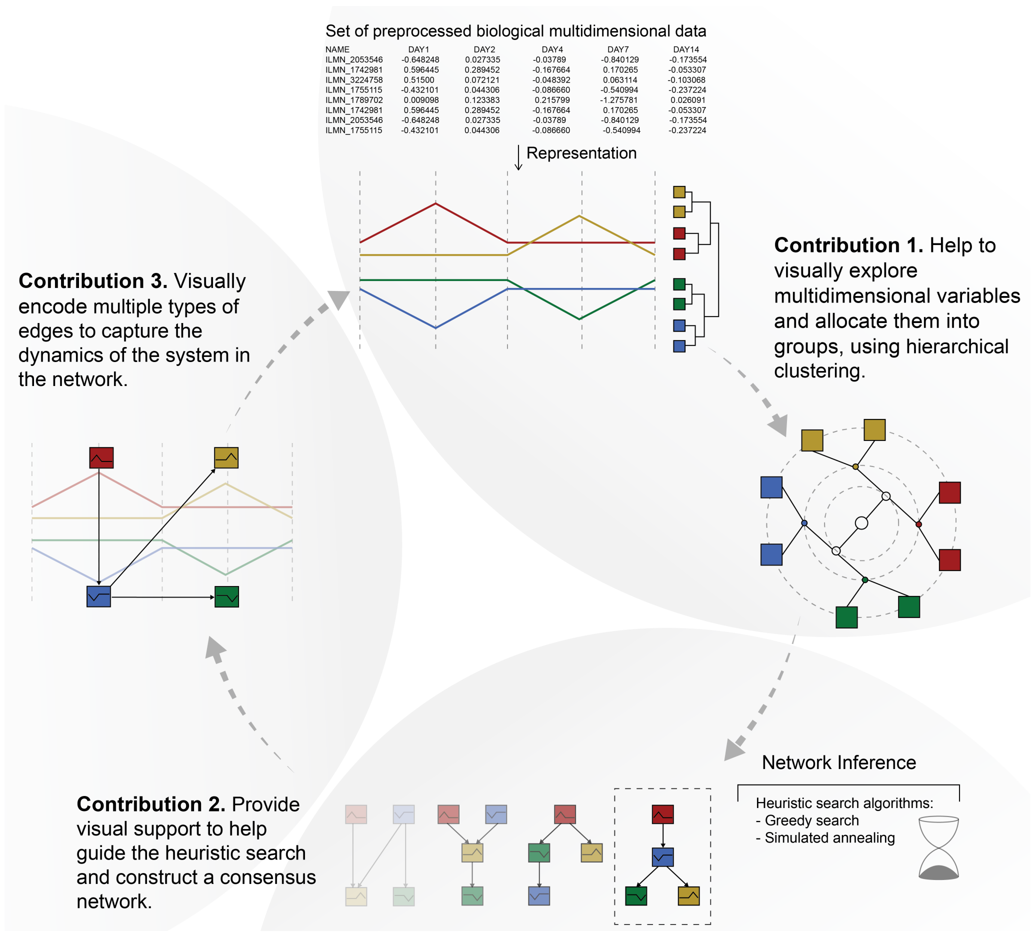
Supporting modellers in their workflow to overcome those challenges, constitutes the three main contributions presented in this thesis. In the following chapters, we provide answers to those three research questions. In Chapter 2, we cover some of the background regarding the domain scientists, the data and the methodology followed for answering the research questions. In Chapter 3, we discuss the research questions in detail and we present a literature review. Chapter 4 addresses the challenge of variable selection, which concerns the first research question (Q1). In Chapter 5, we address the second research question (Q2) which corresponds to the challenge of inferring a network structure from biological data sets. In Chapter 6, we address the challenge of representing information about the dynamics of the biological system, encoded in the different types of edges (Q3). The thesis concludes with Chapter 7, which presents an outline of each contribution in relation to the whole visual approach and also discusses future work.

1.5 Contribution to Knowledge

This thesis presents a novel visual analysis approach to the process of biological Bayesian network inference. Figure 1.2 summarises the three main contributions to knowledge which correspond to the three research questions (Section 1.4) derived from steps of the network inference workflow.

- **Contribution 1:** Answering the first research question (Q1), we developed a novel visualisation tool, called MLCut, which enables the hierarchical clustering of multidimensional data by cutting dendrograms at multiple levels. The interface of the tool incorporates two coordinated views, one for representing multidimensional data sets as parallel coordinates and a second for representing the dendrogram using a scalable design. The visual encoding can represent potentially large multidimensional data sets and dendrograms. Moreover, an interactive mechanism for cutting dendrograms at multiple heights was implemented. More details about this contribution are presented in Chapter 4. The paper for MLCut was published in the *Computer Graphics and Visual Computing (CGVC)* conference [180].

- **Contribution 2:** As part of a contribution that concerns the second research question (Q2), we developed a novel visualisation tool, called BayesPiles, for providing visualisation support for Bayesian network structure learning. The tool enables the exploration, combination and comparison of potentially large collections of scored, directed networks. The tool can visualise directed networks as matrices and supports an overview of multiple networks, capabilities for sorting networks, node reordering and node/edge filtering. Most importantly, the modeller can inspect the shape of the solution space, interactively select networks from


Set of preprocessed biological multidimensional data
Representation
Contribution 1. Help to visually explore multidimensional variables and allocate them into groups, using hierarchical clustering.
Network Inference
Heuristic search algorithms:
- Greedy search
- Simulated annealing
Contribution 2. Provide visual support to help guide the heuristic search and construct a consensus network.
Contribution 3. Visually encode multiple types of edges to capture the dynamics of the system in the network.

NAME	DAY1	DAY2	DAY4	DAY7	DAY14
ILMN_2053546	-0.648248	0.027335	-0.03789	-0.840129	-0.173554
ILMN_1742981	0.596445	0.289452	-0.167664	0.170265	-0.053307
ILMN_3224758	0.51500	0.072121	-0.048392	0.063114	-0.103068
ILMN_1755115	-0.432101	0.044306	-0.086660	-0.540994	-0.237224
ILMN_1789702	0.009098	0.123383	0.215799	-1.275781	0.026091
ILMN_1742981	0.596445	0.289452	-0.167664	0.170265	-0.053307
ILMN_2053546	-0.648248	0.027335	-0.03789	-0.840129	-0.173554
ILMN_1755115	-0.432101	0.044306	-0.086660	-0.540994	-0.237224

Figure 1.2: The visual analysis circle that describes the contributions of this thesis for inferring biological networks from multivariate data.

the collection and construct a consensus network manually. More details about this contribution are presented in Chapter 5. The paper for BayesPiles was published in the *ACM Transactions on Intelligent Systems and Technology (TIST)* journal [181].

- **Contribution 3:** As part of the contribution that concerns the third research question (Q3), we explored a large design space of possible visual encodings based on a literature review, feedback from modellers and perceptual principles related to primary visual variables. The main contribution is a quantitative evaluation of a selected subset of the most promising encodings in matrices. The results of this study informed the design of BayesPiles to also support dynamic Bayesian networks. Except biology, the identification of effective encodings has application to several other domains that utilise multidimensional networks, such as neuroscience, social networks and software engineering. More details about this contribution are presented in Chapter 6. The paper for the evaluation study was accepted for presentation in the *VIS 2019 Workshop on the Visualization of Multilayer Networks* [182].

In the following Chapter 2, we describe in more detail the domain scientists and the data involved in this thesis. Also, we cover the background knowledge related to the domain of visualisation as well as details about the design methodology followed for answering the research questions.

2 Background

In this thesis, we describe how visualisation can support computational biologists (*i.e.* domain scientists) in their workflow to infer networks from biological data. We present visualisation tools that can help reduce the number of variables (Figure 1.1 (a)), infer the structure of a final network model (Figure 1.1 (b)) and understand information about the dynamics of the interactions (Figure 1.1 (c)). This chapter, covers some of the background knowledge required for understanding the context of these contributions. Section 2.1, introduces the domain scientists who collect and visually analyse biological data to infer networks. Section 2.2 provides a description of the data visualised in this thesis. The chapter continues with an overview of concepts and definitions related to the fields of information and biological visualisation (Section 2.3). Section 2.4, presents the methodology followed for identifying biological and visualisation analysis challenges addressed in this thesis.

2.1 Domain Scientists

The visualisation tools presented in this thesis were designed for scientists who are interested in inferring networks that model biological systems. In other words, the end-users of the contributed tools are scientists who use visualisation to gain insights from biological data and combine mathematical and computational methods with their tacit knowledge to infer networks. These scientists can be statisticians, mathematical modellers, bioinformaticians (*i.e.* computational biologists) or any other scientists interested in inferring biological networks. Because they can come from different backgrounds, throughout this thesis the terms *end-users, specialists, modellers, analysts, computational biologists, bioinformaticians, domain scientists* and *domain experts* are used interchangeably. Most commonly, we refer to the end-users of our tools with the generic term: *"modellers"*. Moreover, because we collaborated with a specific group of five computational biologists, we often refer to them as *"our collaborators"* or *"our biologists"*.

Our collaborators included a senior lecturer in the field of biology, who is an expert in the field of complex biological network inference, and four research students, who were supervised by the senior lecturer. Each of the research students worked independently on individual projects, focusing on different aspects of the biological network inference workflow (Figure 1.1). Our collaboration with the research students lasted for as long as they were working on each of their projects. The duration of that period ranged from six weeks up to six months, depending on the project. Our collaboration with one of the research students lasted for approximately six months, and it was related to the first two steps of the network inference workflow (Figure 1.1(a) and (b)). A second student, who was a neuroscientist, only collaborated during a short period of six weeks for a research project that focused on the second challenge (Figure 1.1 (b)). The projects of the other two students were mostly focusing on challenges related to the third step of the workflow (Figure 1.1 (c)). Our collaboration with one of them lasted for three months, while the second student (who was a PhD candidate) collaborated during six months on an occasional basis (3-4 meetings in total). The senior lecturer collaborated in all steps of the workflow for the three years of this thesis. Our collaboration for some periods was very close (meeting weekly), while for other periods it was occasional (meeting monthly). During six months and while we were addressing the first step of the workflow (Figure 1.1 (a)), we also received feedback from a senior statistician specialised in computational methods for analysing genetic data.

For the step of variable selection in the analysis workflow (Figure 1.1 (a)), our collaborators used the implementation of the agglomerative hierarchical clustering algorithm (average-linkage) found in the R package TSclust [131]. They executed the algorithm through the command line, and they inspected the results in the form of the static dendrogram visualisation the package provides. For the steps of network inference (Figure 1.1 (b)) and the integration of the dynamics of interactions (Figure 1.1 (c)), our collaborators used a particular software package for Bayesian network structure learning, called BANJO [167]. BANJO provides a command-line interface through which common heuristic search algorithms such as greedy search and simulated annealing become easily accessible. Users of BANJO can easily set parameters editing a configuration file. Such parameters include discretisation policies for transforming continuous data into discrete, setting-up the heuristic search algorithms, the maximum number of parents per node permitted, edges between pairs of nodes which are already known, the range of latency types (*i.e.* Markov lag) for dynamic Bayesian networks etc. The output of BANJO is a text file that reports on the networks that the execution of the heuristic search algorithm has found. The networks are encoded in a textual format and sorted based on score, with the top-scoring network appearing first. Our collaborators combined many of these textual reports to identify edges that appeared in high-scoring networks and were missing from the lower-scoring ones. Also, in their effort to construct a consensus network, they used Graphviz [65] to visualise more than one networks.

2.2 The Data

As part of the second step of the network inference workflow (Figure 1.1), scientists perform experiments collecting biological data to test their hypotheses. Typically, the purpose of these experiments is to record the state of a biological system by collecting quantitative measurements of variables in different conditions or time points [115]. There are many ways that data can be collected and the experimental design may involve both manual and automated steps. Modern automated methods, such as next-generation sequencing technologies [125, 149], enabled scientists to massively collect genetic data from organisms, resulting in large data tables that contain potentially thousands of variables measured in different experimental conditions or time points. However, because collecting biological data is financially expensive and technically complicated [91], in real data sets, the number of experimental conditions or time points usually ranges from 2 to 20, with each corresponding to a different column in the data table. In this thesis, we present case studies that involve data sets that fall within this range. Also, we do not use the same data set throughout the whole analysis workflow (Figure 1.1) because each of our collaborators was focusing on analysing data in specific steps of the pipeline. However, the tools presented in this thesis can be used sequentially for the same data set throughout the whole network inference workflow.

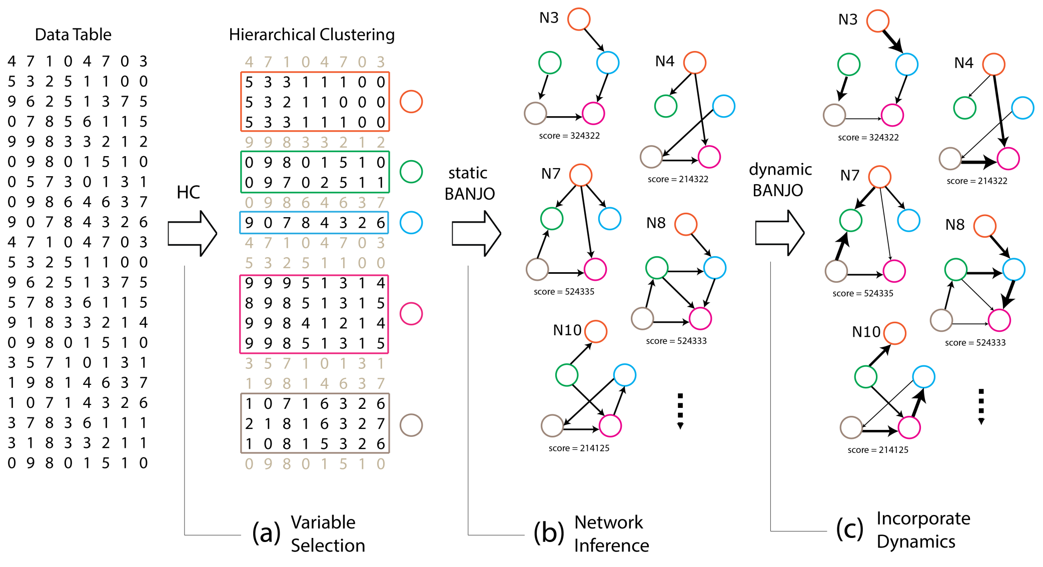
In a typical data table, recordings are summarised in data tables in which rows correspond to variables, while columns correspond to the different experimental conditions or time points (Table 2.1). Thus, each variable can be described as a vector of multiple attributes, each of which corresponds to a different condition or time point, resulting in a high-dimensional data set. The numerical values of real data sets are usually continuous measurements of different ranges. Their range depends on the experimental design and the technology used for each data set in the sampling (*i.e.* measuring) process. However, these continuous measurements always get normalised in a pre-processing step to enable comparisons.

NAME	DAY1	DAY2	DAY4	DAY7	DAY14
ILMN_2053546	-0.64824	0.02733	-0.03789	-0.84012	-0.17355
ILMN_1742981	0.59644	0.28945	-0.16766	0.17026	-0.0533
ILMN_3224758	0.515	0.07212	-0.04839	0.06311	-0.10306
ILMN_1755115	-0.43211	0.0443	-0.08666	-0.54099	-0.23722
ILMN_1789702	0.00909	0.12338	0.21579	-1.27578	0.02609
ILMN_2053546	-0.64824	0.02733	-0.03789	-0.84012	-0.17355
ILMN_1784217	-0.17191	-0.37519	-0.25647	-0.97748	0.07347
ILMN_1652082	-0.07852	-0.01369	-0.12612	-0.67727	0.15559
ILMN_2053178	-0.43855	0.0289	0.08289	-0.57347	-0.02958
ILMN_1656625	-0.35504	-0.00547	-0.19932	-0.45741	-0.12073
ILMN_1665909	-0.43526	0.072852	-0.18364	-0.58188	-0.08157
ILMN_3246292	-0.39645	0.04943	-0.12847	-0.53187	-0.10791
ILMN_1803429	0.12951	0.13822	0.20856	0.80324	0.22108
ILMN_1670881	0.17555	-0.06011	0.0919	0.92469	0.1284
ILMN_2051519	-0.36165	0.07544	0.06387	-0.51779	0.00425

Table 2.1: The first 15 variables of a time-series gene expression data set with 5 time points. Each numerical value corresponds to the mean log2 fold change.

After the data-collection step and before any representation or analysis of the data, there is a data pre-processing step (the third step in Figure 1.1) which ensures that the data can be used for comparisons and further processing. As part of this step, scientists perform a variety of methods to clean, filter and format the data, remove noise and normalise the measurements across the set, making them comparable [147]. By the end of this step, a ready to use and share data set gets created. The raw data used in this thesis were normalised using the *z-score*, and the entries in the resulting data table were the means of the *log2* fold change for each variable in every condition or time point tested, as shown in Table 2.1. Examples of such data sets can be found online in publicly accessible repositories such as the Gene Expression Omnibus (GEO) [64].

After the variable selection step of the biological workflow (Figure 2.1 (a)), the number of variables is reduced by either removing or aggregating rows in the data table. The data table is reduced from containing hundreds or even thousands of rows, to contain just a few dozen (30 to 50 rows). The resulting data table of this smaller set of variables is used as input in the network inference step (Figure 2.1 (b)). However, an additional discretisation step is first required. Although real data sets contain continuous variable types, most network inference methods, such as BANJO [167], require that the measurements in the data table are discrete values [56]. In those cases, a discretisation policy is applied to the data before networks can be inferred. BANJO offers several policies for discretising continuous data sets at different granularities.



Data Table

4	7	1	0	4	7	0	3
5	3	2	5	1	1	0	0
9	6	2	5	1	3	7	5
0	7	8	5	6	1	1	5
9	9	8	3	3	2	1	2
0	9	8	0	1	5	1	0
0	5	7	3	0	1	3	1
0	9	8	6	4	6	3	7
9	0	7	8	4	3	2	6
4	7	1	0	4	7	0	3
5	3	2	5	1	1	0	0
9	6	2	5	1	3	7	5
5	7	8	3	6	1	1	5
9	1	8	3	3	2	1	4
0	9	8	0	1	5	1	0
3	5	7	1	0	1	3	1
1	9	8	1	4	6	3	7
1	0	7	1	4	3	2	6
3	7	8	3	6	1	1	1
3	1	8	3	3	2	1	1
0	9	8	0	1	5	1	0

Hierarchical Clustering

4	7	1	0	4	7	0	3
5	3	3	1	1	1	0	0
5	3	2	1	1	0	0	0
5	3	3	1	1	1	0	0
9	9	8	3	3	2	1	2
0	9	8	0	1	5	1	0
0	9	7	0	2	5	1	1
0	9	8	6	4	6	3	7
9	0	7	8	4	3	2	6
4	7	1	0	4	7	0	3
5	3	2	5	1	1	0	0
9	9	9	5	1	3	1	4
8	9	8	5	1	3	1	5
9	9	8	4	1	2	1	4
9	9	8	5	1	3	1	5
3	5	7	1	0	1	3	1
1	9	8	1	4	6	3	7
1	0	7	1	6	3	2	6
2	1	8	1	6	3	2	7
1	0	8	1	5	3	2	6
0	9	8	0	1	5	1	0

Figure 2.1: The flow of data in the three steps of modellers' workflow: (a) variable selection through hierarchical clustering (HC), (b) network inference and (c) incorporating the dynamics of interactions.

The output of network inference algorithms is a collection of potentially hundreds of candidate networks (Figure 2.1 (b)). A scoring metric assesses the merit of each network based on its statistical fit to the data [69]. BANJO generates plain text reports as an output which contain the candidate networks found by the heuristic search algorithm printed in a tabular format as lists of edges between pairs of nodes (Table 2.2). The networks are ordered based on their score, calculated using the BDe metric [87], with the top-scoring network appearing first. In this format, the first line indicates the rank of the network, its score and the iteration it was first encountered by the algorithm. The second line indicates the number of nodes in the network. In each of the

```
Network #1, score: -1168.0524, first found at iteration 11642630
12
 0 1 10
 1 4 0 2 4 10
 2 3 0 6 10
 3 3 5 7 10
 4 3 3 6 10
 5 0
 6 3 0 3 7
 7 3 0 5 10
 8 3 4 9 11
 9 3 7 10 11
10 0
11 4 0 3 4 5
```

Table 2.2: The format BANJO uses to represent Bayesian networks as text.

remaining rows, the first column indicates the *id* of the node of reference. The second column indicates the number of the incoming edges to that node (*i.e.* parents) and the rest of the columns indicate the ids of the incoming edges. For example 0 1 10 means that node with id=0 has 1 parent which is the node with id=10. Each execution of the algorithm can generate hundreds of candidate networks, and modellers often compare networks across different runs to construct a consensus network. Therefore, the data sets that modellers handle, contain results from multiple runs, and up to a thousand networks in total.

When the data contains measurements over time, it is also possible to infer networks which incorporate information about the dynamics of the interactions (*i.e.* dynamic Bayesian networks), as shown in Figure 2.1 (c). Such networks involve multiple types of edges, and an example of a network generated by BANJO is shown in Table 2.3. In this thesis, we visualise data sets that contain networks with up to four types of edges. To be able to visualise such networks, first, we had to parse those text files and transform them into a more flexible JSON format. For this purpose, we wrote custom text-formatting scripts that convert BANJO report files into JSON. In the next section, we present an overview of basic concepts and definitions about visualisation and its role in helping scientists make sense of their data.

2.3 Visualisation

Visualisation can be defined as the scientific discipline which aims at helping humans to gain a better understanding of data, through the sense of sight, using visual means. In other words, visualisation is about creating visual representations and interactive interfaces that can augment our perception and help us explore and understand reality based on evidence found in collected data. Practically, when visualisation is applied to real situations, gaining insight is mainly achieved by exploring, explaining or confirming information and knowledge found in the data. Therefore,

```
Network #1, score: -15756.4344, first found at iteration 25555
12
 0   0:   0               1:   2 0 7          2:   0
 1   0:   0               1:   1 1            2:   1 1
 2   0:   1 0             1:   2 1 2          2:   0
 3   0:   0               1:   2 2 3          2:   0
 4   0:   0               1:   2 1 4          2:   0
 5   0:   0               1:   2 4 5          2:   0
 6   0:   0               1:   1 6            2:   1 6
 7   0:   0               1:   2 3 7          2:   0
 8   0:   0               1:   2 3 8          2:   0
 9   0:   1 5             1:   1 9            2:   0
10   0:   1 9             1:   2 8 10         2:   0
11   0:   0               1:   2 10 11        2:   0
```

Table 2.3: The format in which a dynamic Bayesian network with three types of edges is printed in the output of BANJO.

visualisation can be *exploratory*, *explanatory*, or *confirmatory*. In the first case, the aim is to help users form new hypotheses about reality, based on information they find after interacting with the data [176]. In the second case, the aim is to present information in a clear way to the users and promote a better understanding of what was already found in the data [175]. In the third case, the aim is to provide an evidence-based verification and confirmation of previous knowledge found in new data [176]. In any case, a successful visualisation can help to deal with practical problems in a more efficient and effective way. In this thesis, we mostly focus on problems related to the exploration and representation of biological data. However, the visualisation tools presented in this thesis can be also used for the visual explanation and confirmation of results.

2.3.1 Biological Visualisation

In the field of biological visualisation, the discipline of biology provides challenging problems that originate and motivate hypotheses and research questions, while the discipline of visualisation provides a suitable vehicle for approaching these biological challenges. The aim is to gain a better understanding of biological processes, using visualisation techniques for exploring biological data and for representing findings more effectively [59]. Visualisation aims at developing useful tools for conducting scientific research more efficiently and effectively. Inherently, biological visualisation aims at assisting researchers in finding better solutions to biological challenges, such as representing biological networks and data effectively [72]. Biological visualisation overlaps with many research disciplines, but because it relies heavily on the use of computer systems, it is often considered to be part of the wider interdisciplinary field of bioinformatics.

2.3.2 *Visual Analytics*

The exploration of the data typically follows the Visual Analytics loop, which starts with the execution of an algorithm followed by a representation of the results, as shown in Figure 2.2 (a). A visual interface enables the user to interact with the results, select data and refine parameters informing the next execution of the algorithm [102]. This process creates a loop which is repeated several times in an iterative fashion, as shown in Figure 2.2 (b) and leads to the gradual improvement of the results (*i.e.* the model). Depending on the speed of the algorithm, the user can interact either sequentially or in real time with its execution. Usually, in both cases, the user can receive visual feedback through the representation of the data in real time.

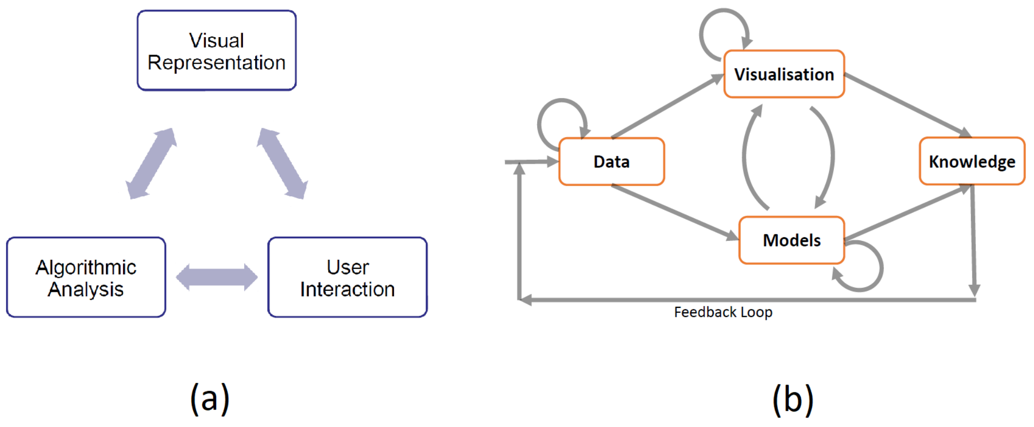


Figure 2.2: (a) The visual analysis loop. (b) The visual data-exploration loop [102].

When applied to biological data, Visual Analytics produces interactive visualisation software tools, algorithms, classifications, techniques and/or methodologies, which can be used for completing different biological analysis goals [102]. For example, as mentioned in Chapter 1, systems biologists try to model biological processes by observing changes within a living cell at a molecular level. Usually, biologists collect observations under multiple experimental conditions and then try to integrate findings in a model, which is often represented as a network. In the case of probabilistic methods, such as Bayesian networks, there is a need for visualisation tools which can help modellers to make sense of their data and improve their network inference workflow. The sense-making loop for Visual Analytics, shown in Figure 2.3, places visualisation between the user and the data. The users try to make sense out of data by combining their knowledge with what they perceive visually. Thus, the process of sense-making is repetitive and involves elements from both exploratory and confirmatory data analysis methods since it combines new with old knowledge [176].

When the sense-making loop for Visual Analytics is applied to systems modelling, it takes the form of the visual data-exploration loop (Figure 2.2 (b)). The user interacts with a visual

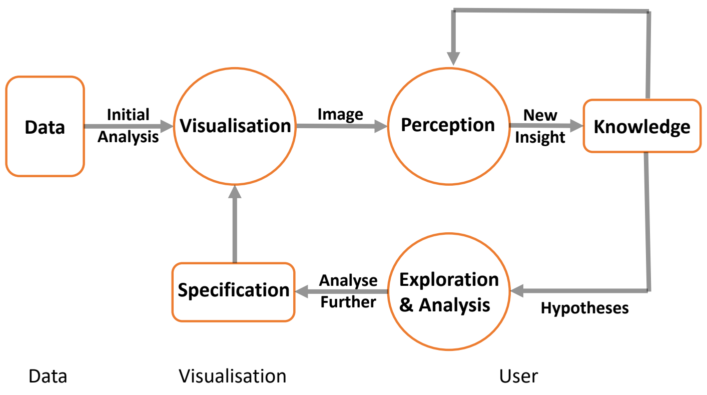


Figure 2.3: The sense-making loop for Visual Analytics [102].

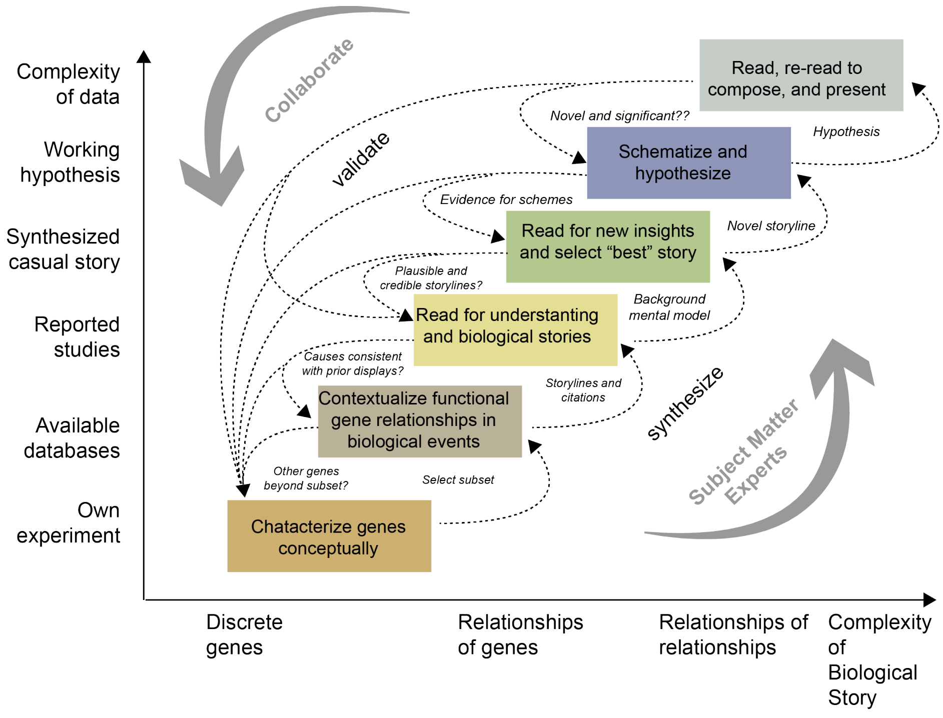
representation of the data to tune parameters of the suggested model in an iterative way. This process is repeated in a loop until the model is further refined in a way that more knowledge about the real system is acquired from the data.

In the context of exploring data from biological experiments for the purpose of unravelling functional relationships between molecules, biological visualisation provides users with interactive data visualisations that give insights through a process of sense-making. For instance, Figure 2.4 presents a model which describes the sense making loop for exploring interactions in disease-related processes in a molecular level [130].

This thesis focuses on the application of visualisation methods for gaining insights from biological data to create better network models. Visualisation plays an important role because it provides opportunities for developing tools for data exploration, complexity reduction and pattern detection, which can help modellers gain insights about the biological system [80]. However, given a biological problem, there are many ways that information can be visualised and it is difficult to identify which design would be the most effective [133]. In the following section, we describe the methodology followed for creating the visualisations presented in this thesis.

2.4 Visualisation Design Methodology

The outcome of effective visualisation design can augment human judgement by combining the ability of humans' visual system in detecting patterns quickly, with the power of modern computers in storing, processing and displaying information. Thus, visualisation designers must take into account limitations related to the visual perception of humans, to create successful visual encoding and interaction techniques [188]. Usually, a successful visualisation design is achieved by developing effective graphical representations which reduce the cognitive overload and visual clutter of naive representations [29].



Complexity of data
Working hypothesis
Synthesized casual story
Reported studies
Available databases
Own experiment
Collaborate
validate
Read, re-read to compose, and present
Novel and significant??
Hypothesis
Schematize and hypothesize
Evidence for schemes
Novel storyline
Read for new insights and select "best" story
Plausible and credible storylines?
Read for understanting and biological stories
Background mental model
Causes consistent with prior displays?
Storylines and citations
Contextualize functional gene relationships in biological events
synthesize
Subject Matter Experts
Other genes beyond subset?
Select subset
Chatacterize genes conceptually
Discrete genes
Relationships of genes
Relationships of relationships
Complexity of Biological Story

Figure 2.4: Sense-making model for exploring molecular level influences on disease-related processes [130].

Due to advances in computer science, it became possible for the user to interact with the display and get visual feedback in real time. This development created new opportunities and challenges for developing powerful visualisation systems based on computers. Computer-based visualisations of data can help people carry out tasks, that cannot be automated [133]. However, there are many possible ways that data can be represented visually and the design space is huge. To deal with all these possibilities, visualisation designers must take into account limitations in computers, displays and humans [133].

Computer systems are considered an invaluable resource for the discipline of visualisation because they can process and represent data sets that would be infeasible to draw manually. In addition, computers provide capabilities for manipulating visual representations interactively. However, real-world data sets may be composed of hundreds of Gigabytes and current computer systems cannot always handle them efficiently. Therefore, there are limitations in computational resources available to deal with the increased demands that those data sets impose. Therefore, the path of visually gaining insights from large, complex and dynamic data, partly coincides with the one already followed by the disciplines of high-performance computing, data mining and optimisation [157, 95]. For example, scientific discovery often uses visualisation in conjunction with computational methods as part of a larger workflow.

Regarding display limitations: it is true that during the last decades screens became larger and of a higher resolution. However, their capabilities are still limited compared to the size of the data available for analysis. There are visualisation approaches for utilising larger and higher resolution displays [15] and also approaches that utilise the hardware architecture of the most advanced graphics processing units (GPUs) [30]. Such technological solutions are important for advances in the discipline of visualisation but they are outside the scope of this thesis, in which we are mostly concerned with creating visualisations for users with a standard set of computational resources (an average PC).

In visualisation approaches, except for technological limitations, there are also limitations inherent to human visual perception. For example, the human brain has certain limitations in matters of memory and attention. In order to comply with those limitations, appropriate visual encoding has to be selected for creating effective external representations. Such representations help us to surpass our cognitive limitations and augment our capacity to take decisions and solve problems. Following design principles is important so that information will become more sensible, will be communicated more effectively, with increased precision and within reasonable time constraints [103, 188, 133]. This is how the discipline of visualisation has been shaped; by considering the limitations in human perception and cognition, in conjunction with the technological limitations.

2.4.1 The Nested Model

The methodology followed for determining the visualisation designs presented in this thesis was driven by the *nested model*, proposed by T. Munzner [132]. This model was selected to be the core methodological approach during this thesis, mainly because of its generality and its widespread adoption by the visualisation community. Contrary to other frameworks [78, 178, 160, 151], the nested model has been applied successfully to all kinds of visualisation approaches and application domains, including case studies for multi-attribute rankings [79], social media data [38], climate change data [144] and even poetry [1]. Most importantly, the nested model has been used extensively in design studies for biological visualisation [76, 169, 67, 163].

There are many possible ways to visually encode information to pictures (the design space is huge) and the danger of creating ineffective visualisations is very high [133]. The nested model can help designers avoid threats that often lead to bad design decisions, identify and clarify interesting problems in the application domain and create effective visualisations. In this section, we overview the basic concepts of this model and in the next chapters, we describe how we used it to create visualisations presented in this thesis.

According to this model, there are four nested levels, summarised in Figure 2.5, which should be addressed and evaluated separately: (1) target users to identify domain analysis objectives, (2) select appropriate data structures and operations, (3) justify visual encoding and interactions and (4) develop efficient algorithmic implementations.

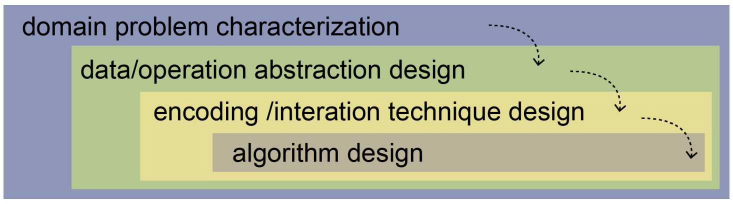

domain problem characterization
data/operation abstraction design
encoding /interation technique design
algorithm design

Figure 2.5: The four levels of the nested model. In the first level, the domain problem is characterised and understood. In the second level, data and tasks are abstracted and clarified. In the third level, the visual encoding and the interaction technique are designed. In the fourth level, an efficient algorithm for manipulating the data is designed. [132].

- **Level 1: target users to identify domain analysis objectives.** The first level in the nested model involves characterising the domain problem. Designers target domain users to understand and clarify the underlying domain problem. The identified analysis goals are described using the terminology of the domain in which the problem occurs. For that step to be successful, close communication with domain users should be established, to receive feedback, refine and prioritise requirements and avoid potential misunderstandings related to the objectives of the users. The first level of the nested model mainly answers the questions: "**why** this domain problem cannot be solved using automation (algorithm) and why there is a need for a visualisation approach to address its challenges?"

- **Level 2: select appropriate data structures and operations.** In the second level of the nested model, the way that the most relevant data is structured is decided and operations applied to the data are identified. This level requires the understanding of the most relevant aspects of the data to the domain problem. To achieve that understanding, the designer should receive feedback from users about potential data abstractions and operations. A good way to receive feedback is through the development and presentation of sketches and prototypes followed by discussions with end users. This level mainly answers the questions: "**what** data should be visualised and which operations should be applied to this data to support the domain problem?"

- **Level 3: justify visual encoding and interactions.** In the third level of the nested model, the designer selects how the data should be encoded visually, using a combination of different visual variables and also how the operations on the data could be performed interactively. There are different ways of testing potential encodings, e.g. after evaluation through a usability study with a group of participants. Receiving user feedback is also important for testing the effectiveness of the proposed approaches. This level answers the questions: "**how** the data are going to be encoded in the display and how the users will interact with the data?"

- **Level 4: develop efficient algorithmic implementations.** Finally, the last step is that of the algorithm design and implementation and it is common in every computer science project. In that step, the designer has to deal mostly with technical issues to ensure that the right application programming interface and software libraries have been selected and that the data can be handled, visualised and controlled as expected. Those technical issues may include speed and memory measurements and they can be extended to involve other performance benchmarks (e.g. to test performance for larger data sets). Dealing with such issues is common in software engineering and in the design of information technology management systems. Systems could be evaluated according to their scalability, robustness, portability, interoperability, internationalisation and other characteristics. This level of the nested model involves practical questions such as: "how to improve the speed of reading data from the database? How can this data transformation take place in real time every time the user moves a slider or selects this option?"

Moreover, for each level of the nested model, there is a number of common threats, which have to be addressed and validated throughout the whole process. The threats are shown in Figure 2.6. One of the ways to avoid those threats is by the continuous refinement of the requirements and the design choices until the domain problem is clear, the data and their operations are abstracted, the visual encoding and the interactivity are effective and the algorithm design is performing well.

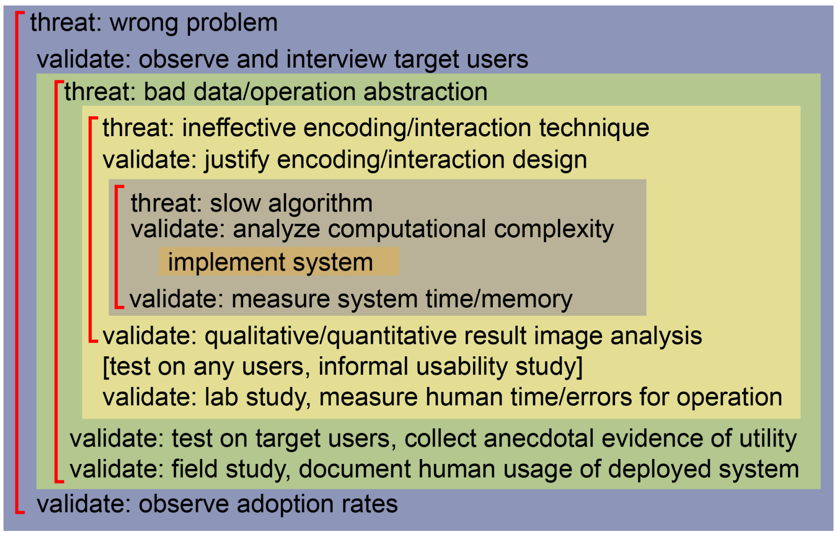

threat: wrong problem
validate: observe and interview target users
threat: bad data/operation abstraction
threat: ineffective encoding/interaction technique
validate: justify encoding/interaction design
threat: slow algorithm
validate: analyze computational complexity
implement system
validate: measure system time/memory
validate: qualitative/quantitative result image analysis
[test on any users, informal usability study]
validate: lab study, measure human time/errors for operation
validate: test on target users, collect anecdotal evidence of utility
validate: field study, document human usage of deployed system
validate: observe adoption rates

Figure 2.6: Threats and validation in the nested model [132].

Because biology is an applied field that poses specific challenges to the field of visualisation, our visualisation research approach was *problem-driven* rather than *technique-driven*. Hence, one of the most important threats was the characterisation of the domain problem. For this reason, it was important to collaborate closely with domain experts to understand their requirements and design effective visualisation tools that could help them in specific steps of their analysis workflow.

The problem of network inference from biological data is located between two extremes. On the one hand, modern biological data collection methods create opportunities. But on the other hand, the resulting data sets are difficult to explore and make sense of, while at the same time automation is not possible since the network cannot be inferred in polynomial time using computational methods. Thus, the modeller plays an important role in steering the network inference process. For those reasons, a design study was found suitable for addressing the underlying biological problem. Figure 2.7 shows the area in which the approach of a design study is suitable (between two axes: task clarity and information location). While the nested model methodology was used throughout this thesis, the contributed visualisation tools were developed and presented as design studies.

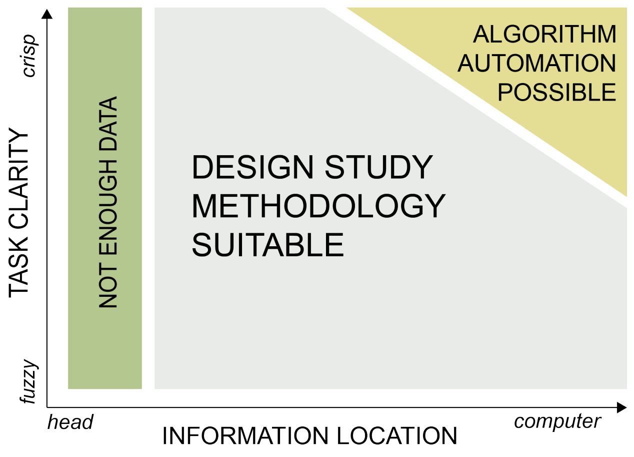


Figure 2.7: The task clarity and information location axes as a way to analyse the suitability of design study methodology. Green and yellow areas mark regions where design studies may be the wrong methodological choice [159].

In the next chapter, we describe in more technical detail the three challenges addressed in this thesis and we discuss related visualisation tools and techniques. Each challenge corresponds to a different research question (Section 1.4) and a step in the network inference workflow (Figure 1.1). In Section 3.1, we discuss related work to visualising dendrograms and the challenge of cutting dendrograms constructed by a hierarchical clustering algorithms to select variables that constitute nodes in the inferred network. In Section 3.2, we present related work to the challenge of learning the structure of Bayesian networks. Finally, in Section 3.3, we discuss work related to the challenge of exploring multiple types of edges in collections of inferred dynamic Bayesian networks.

3 Related Work

In this chapter, we present related work to the biological and visual analysis challenges that correspond to the three research questions which motivated this thesis (Section 1.4). For each of those challenges, visualisation approaches found in the literature are reviewed. Section 3.1, covers work related to the challenge of variable selection (Figure 1.1 (a)) using hierarchical clustering. In Section 3.2, related work to the challenge of network inference (Figure 1.1 (b)) is presented. Section 3.3, covers work related to the third challenge (Figure 1.1 (c)) of incorporating information about the dynamics of interactions through the representation of multivariate networks. We conclude with a summary of the visualisation challenges and an overview of what follows next.

3.1 Hierarchical Clustering Analysis

Hierarchical clustering algorithms are used in many applications for "grouping" data records into a number of non-overlapping sets (*i.e.* clusters). Those algorithms take as input a distance matrix with estimated pairwise dissimilarity scores (*i.e.* distances) between all data records. Dissimilarities between records are calculated using an appropriately selected metric or measure, such as Euclidean distance or correlation coefficient, which matches the purpose of the intended analysis [46]. The output produced is a simplified hierarchical structure, known as the *dendrogram*, which encapsulates the rationale followed by the hierarchical clustering algorithm. For instance, agglomerative hierarchical clustering algorithms repeatedly find and merge pairs of similar data records into composite objects, forming a hierarchy of dissimilarity levels, until the point that all records are merged into a single group: the root of the tree (Figure 3.1). The way in which the distance between composite objects is calculated determines the different types of hierarchical clustering algorithms (such as average-linkage, single-linkage and complete-linkage) [193].

Except for the bottom-up approach of agglomerative clustering, there is also the divisive approach which starts with placing all data records in one group. Then, in each step of the algorithm, the group is divided into two separate groups and this continues until the point that every data record belongs to a separate cluster [152]. There are several types of hierarchical clustering algorithms but all of them result in a dendrogram in which the intermediate nodes show the dissimilarity levels at which a merging/division took place and the leaves represent the original data records.

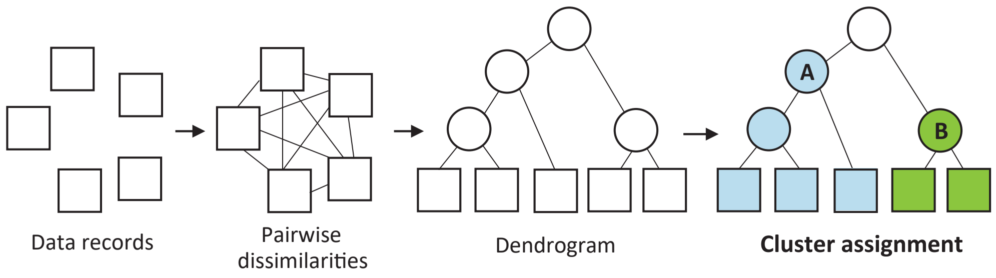


Figure 3.1: The flow of constructing the dendrogram. Pairwise dissimilarities between data records are calculated and used by the hierarchical clustering algorithm to construct a dendrogram. The branches A and B correspond to clusters.

In the following Section 3.1.1, we investigate how trees and dendrograms have been visualised in the literature.

3.1.1 Visualising Dendrograms

The result of the hierarchical clustering algorithm is encapsulated in the dendrogram. Dendrograms are traditionally represented as top-down trees with the length of each branch encoding its distinctiveness. The total height of the dendrogram from the leaves until the root corresponds to the range between the minimum and the maximum dissimilarity levels and it can be normalised so that the leaves correspond to the dissimilarity of 0 and the root of 1, as this is shown in the vertical axis of Figure 3.2. One of the problems with this representation is that for large data sets the dendrogram becomes too long and its exploration becomes difficult. The tree does not fit in a typical monitor display, the use of scrollbars often becomes necessary, while the lower branches often appear cluttered. Such a representation makes it difficult to compare branches or encode additional information such as clustering assignments to the leaves of the tree which represent the data records [43].

Except from the top-down layout that uses a node-link diagram (Figure 3.3 (a)), there are more options for representing trees visually. The dendrogram can be rotated and its leaves can form a list, as shown in Figure 3.3 (b). Alternatively, the node-link diagram can be shown in a radial layout (Figure 3.3 (d)). Higher-level nodes can get a larger layer area, either extending horizontally (Figure 3.3 (c)), or forming concentric circles (Figure 3.3 (e)). The tree structure can be enclosed to form nested circles (Figure 3.3 (f)) or a TreeMap (Figure 3.3 (g)). Finally, indentation can be used to outline the structure of the tree, as shown Figure 3.3 (h). Evaluation studies have shown that radial layouts are more space efficient that top-down and left-to-right representations [124], but they take longer to read [37].

There are several tools and techniques for visualising trees, but most of them do not focus on the task of hierarchical clustering analysis. For instance, *TreeJuxtaposer* [134] is a tool designed for biologists who want to perform a structural comparison of large trees, such as phylogenetic

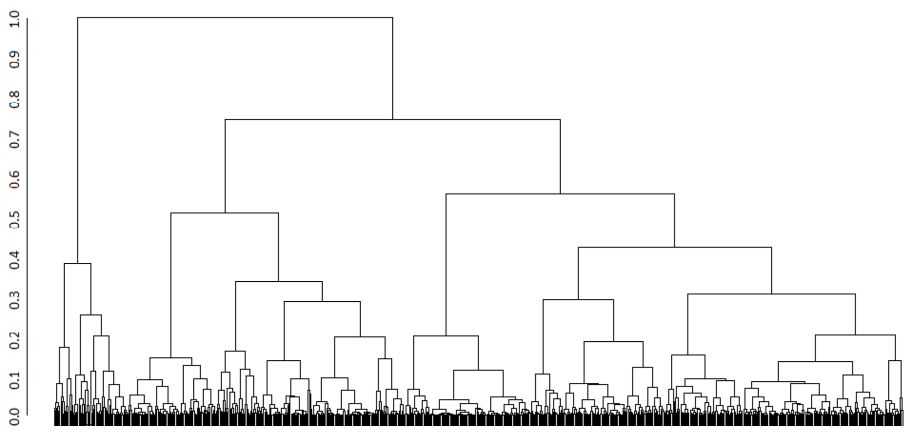


Figure 3.2: A traditional top-down representation of a dendrogram generated using R. The axis on the left shows the dissimilarity levels ranging between 0 (leaves) and 1 (root).

trees, which look similar to dendrograms. *MizBee* [126] uses three views that correspond to three levels of detail in order to provide support for comparing whole genomes. Both tools can be used to compare different data sets. However, our task is different as we focus on the challenge of finding patterns in the original data, through the interactive exploration of a single dendrogram. A technique presented in Chen *et al.* [43] uses a uniform threshold to provide improved visibility by simplifying the dendrogram representation. This is a useful technique for summarising the dendrogram in a selected level of detail and making it fit in smaller displays. However, it does not provide support for exploring or cutting the dendrogram at different heights.

3.1.2 Cutting the Dendrogram

Since overlapping is not permitted between clusters in hierarchical clustering, often there is ambiguity regarding the allocation of variables into clusters. The analyst, therefore, has to decide whether several records form one or more clusters, or if they are all part of a larger cluster (*i.e.* a nested cluster). This ambiguity is more evident in large and complex data sets, where clusters may exist at different heights of the dendrogram. For data sets which consist of relatively distinct and homogeneous subsets, deciding a single similarity threshold, which cuts the tree at a uniform height, could be sufficient for determining representative clusters. For larger dendrograms, which often consist of heterogeneous and less distinct subsets, a more flexible approach that involves multiple-level cuts, would produce more representative results. However, it is not clear how the analyst could explore the data and the different clustering scenarios to identify groups of similar records in large and complex data sets.

Dynamic Tree Cut (DTC) [112] can cut the branches of the dendrogram at different levels automatically, based on their shape, their length or other criteria. However, heuristic criteria are tailored to describe pre-determined shapes and patterns in the dendrogram and they cannot identify

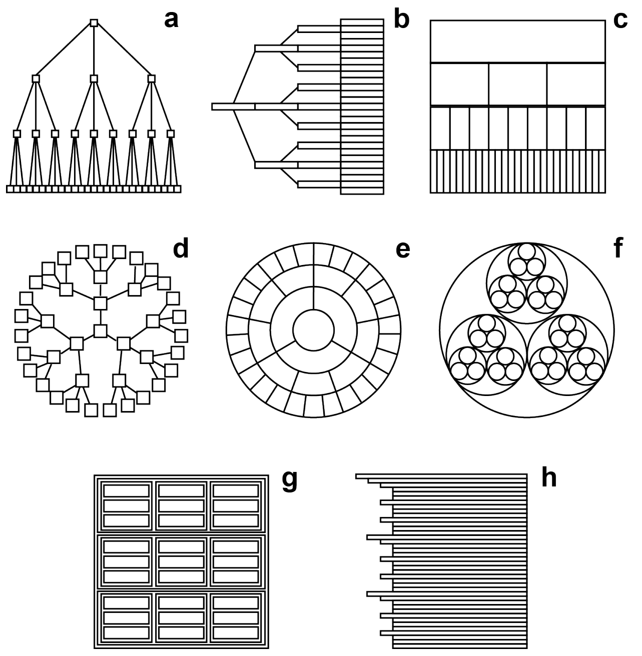


Figure 3.3: Different tree layouts as presented by McGuffin *et al.* [124]. (a) node-link, (b) a variation on (a) to support long labels, (c) icicle, (d) radial, (e) concentric circles, (f) nested circles, (g) treemap and (h) indented outline.

new patterns in the data. Thus, heuristic approaches are rarely optimal because they cannot capture all the pattern variations which can be observed in real data sets. Moreover, semi-supervised approaches such as the ones presented by Dotan-Cohen *et al.* [60], Navlakha *et al.* [137] and in *HCsnip* [140] that integrate prior knowledge into the algorithm to detect clusters, require that the data records are first labeled. The configuration cannot be generalised for unstructured data sets without assuming any background knowledge related to them. Hence, the clustering results rely on additional information about the system, which is usually missing from most data sets, and the assumption that the added labels determine how similar the data records are.

In the real world, there is no *"one-size-fits-all"* solution and it is common to ignore special characteristics of clusters [101]. Within the same data set, some clusters may be tight (low pairwise dissimilarity), while some others may be loose (high pairwise dissimilarity). For instance, biologically associated genes may follow a similar expression pattern either constantly or only for a time period, as reported by Mahanta *et al.* [120] and Craig *et al.* [50]. Therefore, adding the *"human in the loop"* is needed to visually explore the dendrogram and the data records in different levels of detail and select potential clusters manually [165].

Visual support tools have been always used in the analysis of biological data. An evaluation of microarray visualisation tools has been presented by Saraiya *et al.* [156] and a more recent survey has been presented by Pavlopoulos *et al.* [142]. Specialised tools for microarray data analysis often incorporate visualisation features for different analysis tasks, including hierarchical clustering analysis. *Chipster* [98] and *Mayday* [22] are two open source microarray data analysis platforms that support hierarchical clustering. Due to the importance of time-course gene expression data, there is also a number of tools that target the clustering of such data sets. For instance, *STEM* [66] is a software tool for automatic profiling and clustering of short time-series data. The tool creates profiles for possible temporal patterns that can occur and then matches those patterns with what is found in the data set. However, it is difficult to capture all variation that could possibly exist in a multivariate time-series data set. A flexible and user-driven approach, in matters of statistical analysis capabilities, is provided by *PESTS* [165]. All of those tools support some visualisation features for hierarchical clustering analysis but they provide little or no support for interactive exploration of the data.

There are several tools which perform hierarchical clustering analysis, but only a few of them provide visual feedback or support the interactive exploration of the clusters. However, due to the increasing complexity and size of the data, visualisation becomes an important aspect of performing clustering analysis. The relatively new paradigm of visual analysis is founded on the idea that expert users are capable of steering the analysis to produce more successful results [139]. The actions of the users are often driven by tacit knowledge which cannot become part of an algorithm. Therefore, involving a human for taking decisions and for guiding the analysis is essential.

At the highest level, a view of the clusters as part of the whole dendrogram should be supported and at the lowest level, the original multidimensional data should be visualised and linked to their clustering assignment to enable the visual comparison of data records. The idea of *drilling-down* to see more detail in the data is common to many visual analysis tools. Similar steerable approaches have been investigated in the past for exploring graph structures, as in Archambault *et al.* [13] and in Abello *et al.* [2]. To provide flexibility and control over the clustering process, modellers required a method that would enable them to combine hierarchical clustering results with their own tacit knowledge to take decisions about the allocation of variables into clusters. During the analysis, modellers need to be able to cut branches of the dendrogram at different heights, as shown in (Figure 3.4).

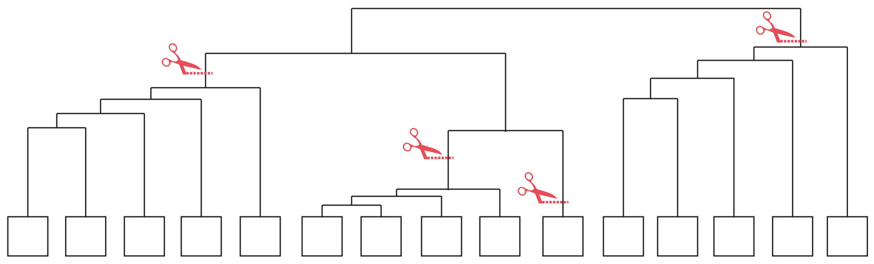


Figure 3.4: Multi-level cuts in a heterogeneous dendrogram. The red icons indicate four locally applied similarity thresholds which cut the tree in four branches that form the same number of non-overlapping clusters. This clustering scenario could not be achieved using any single-height similarity threshold.

Moreover, it is hard to evaluate the quality of hierarchical clustering results while clusters are selected, mainly because the underlying data and the rationale of the clustering algorithm remain hidden to the final user, who only relies on the dendrogram for selecting clusters. However, visualising the original multivariate data in addition to the dendrogram can help to reveal interesting patterns and relationships between data records which are not always obvious in the dendrogram. Thus, enabling the representation and comparison of the original data records is also important for improving and confirming the selection of clusters. However, visualising the original data records can be complicated, especially when they are large and contain multiple dimensions or time points. In the following Section 3.1.3 we discuss approaches for visualising data sets of variables that contain multiple dimensions or time points.

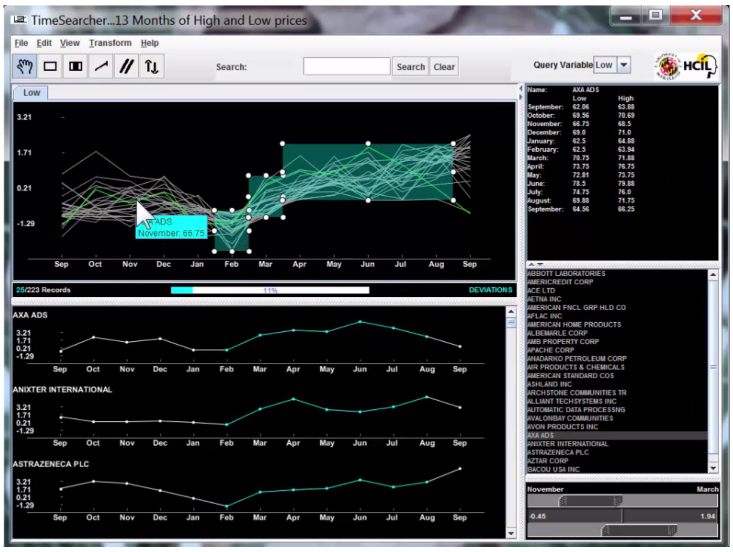
3.1.3 Representing Multivariate Data

Hierarchical clustering algorithms are often applied to data sets that contain records with multiple dimensions or time points. Representing the original multivariate data is challenging but there are several visualisation tools and techniques that target this problem. The most common approaches, either use heatmaps or parallel coordinates [93] to represent multivariate data. For instance, there are several visualisation tools that aim at the discovery of time patterns in large data sets of time-series. However, most of those tools are generic and not designed specifically for representing and exploring temporal profiles in biological data.

A representative software tool in the wider category of time-series visualisation analysis is Timesearcher [89], Figure 3.5. The user can specify the time pattern of interest by applying dynamic queries. In addition the pattern can be sought interactively as the user drags the pattern to the different areas of the time-series representation. Even if this tool is very useful when the pattern is already known, it fails to detect unknown patterns automatically. The user should know the pattern beforehand, or at least perform a sequence of queries until the desired pattern gets revealed.

TIALA [94] is a tool designed specifically for exploring microarray time-series expression data in order to investigate biological mechanisms (Figure 3.6). It can align multiple time-series data from different experiments. Those can be visualised both in 2D using small multiples and in 3D using superimposed time-series on the same axes. This is a very useful tool for comparing a small number of time-series experiments (e.g. experiments from different cell lines). This tool is designed to support comparisons between different time-series data sets and not to support the representation, exploration and hierarchical clustering of a single data set, which is the first analysis challenge.

Contrary to TIALA, Interactive Horizon Graphs [143] is a visualisation approach aiming at the parallel comparison of a much larger number of time-series data sets. The user can search for time patterns when the multiple time-series are displayed vertically. Interactive zooming and panning techniques help the user to inspect several time-series at the same time. However, this approach cannot scale well for thousands of time-series, which is usually the case with many biological data



Name:	AXA ADS	
	Low	High
September:	62.06	63.88
October:	69.56	70.69
November:	66.75	68.5
December:	69.0	71.0
January:	62.5	64.88
February:	62.5	63.94
March:	70.75	71.88
April:	73.75	76.75
May:	72.81	73.75
June:	78.5	79.88
July:	74.75	76.0
August:	69.88	71.75
September:	64.56	66.25

ABBOTT LABORATORIES
AMERICREDIT CORP
ACE LTD
AETNA INC
AMERICAN FNCL GRP HLD CO
AFLAC INC
AMERICAN HOME PRODUCTS
ALBEMARLE CORP
AMB PROPERTY CORP
APACHE CORP
ANADARKO PETROLEUM CORP
AIR PRODUCTS & CHEMICALS
AMERICAN STANDARD COS
ASHLAND INC
ARCHSTONE COMMUNITIES TR
ALLIANT TECHSYSTEMS INC
AUTOMATIC DATA PROCESSNG
AVALONBAY COMMUNITIES
AVON PRODUCTS INC
AXA ADS
ANIXTER INTERNATIONAL
ASTRAZENECA PLC
AZTAR CORP
BACOU USA INC

Figure 3.5: Applying brushing technique in Timesearcher software to specify time pattern [89].

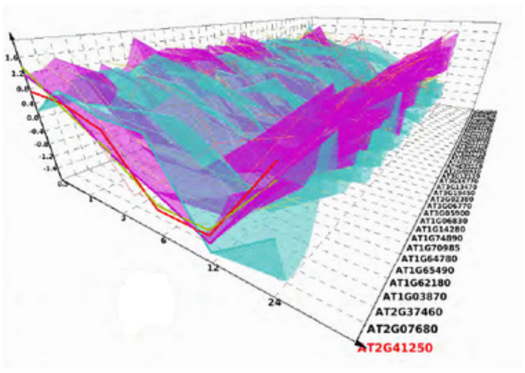


Figure 3.6: Time-series represented in 3D with TIALA [94].

sets.

The gene expression time-series explorer (MaTSE) [50], utilises the technique of animated scatterplots, to discover local patterns in complex time-course gene expression data (Figure 3.7). The user can select a time period from the whole time span and then drag the selection across the time-series plot visualisation. At the same time, an animated scatterplot can help the user to identify motion, which occurs in locations where there is an interesting turbulence in the time pattern. Interesting local events can be identified, while clustering usually smoothens such perturbations in the data. MaTSE performs a very finely grained analysis of time segments and it is useful for exploring data sets with many time points. However, this tool has been designed for a very specific type of analysis, while real-world data sets often include only a small number of time points.

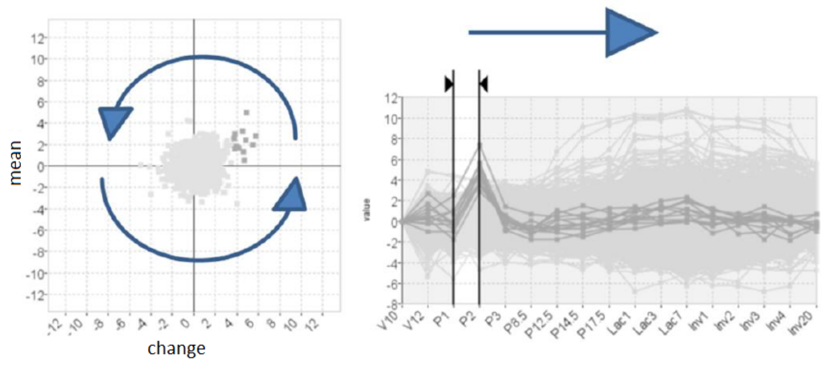


Figure 3.7: Using animated scatterplots to identify patterns in temporal data using MaTSE [50].

RankExplorer [164] is a visualisation method for revealing ranking changes in time-series data (Figure 3.8), together with other changes in different types of data. This is an approach for visualising changes in multiple attributes of data which are linked to time-series. This method can visualise more than one attribute related to each time point of the time-series. While this approach is interesting, it is more common for biological data sets to involve either multiple attributes or multiple time points of one attribute. Moreover, the tool does not provide visual support for the results of hierarchical clustering algorithms.

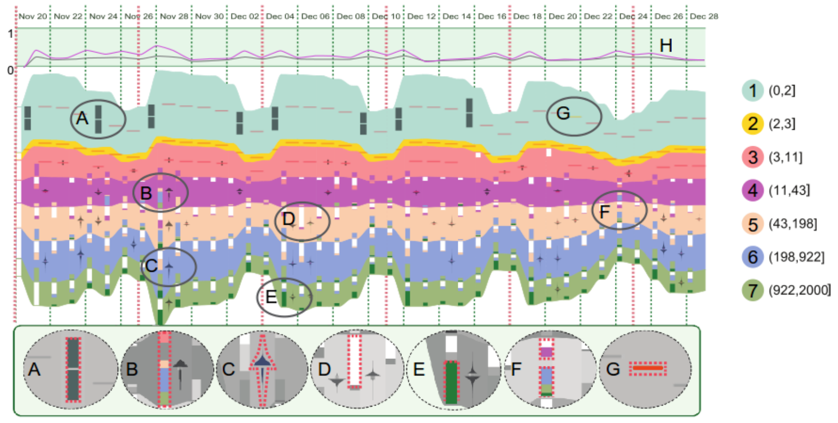


Figure 3.8: Visualisation of ranking changes in large time-series data with RankExplorer [164].

Although we did not find any visual support tools that enable the exploration of the dendrogram in different levels of abstraction to target the problem of ambiguity in hierarchical clustering analysis, there are tools that provide this kind of visual support for other clustering algorithms. For instance, *Spark* [139] provides two views, one in the level of clusters (found by *k-means*) and one in the level of regions within clusters. However, we focus on visualising the output of hierarchical clustering algorithms rather than *k-means*. *Spark* does not directly address the particular challenges related to hierarchical clustering analysis (*i.e.* exploring the dendrogram structure in coordination with the original data, to allocate variables into clusters).

The most relevant tool to our challenge is the *Hierarchical Clustering Explorer (HCE)* [161], which has been designed for supporting interactive genomic microarray data analysis. It provides a dendrogram linked to a heatmap. It enables dynamic querying using a minimum similarity bar, which specifies a single similarity threshold in which the dendrogram is cut. In contrast to other tools, HCE provides interactivity and it is a powerful visualisation tool for hierarchical clustering analysis. However, the dendrogram representation does not easily fit in a standard display without producing visual clutter and it does not support multi-level cuts.

Summary of challenge 1: The step of variable selection is essential for simplifying the network inference process by narrowing the search space within which heuristic search algorithms look for network solutions. To reduce the number of variables, biologists often use hierarchical clustering algorithms for selecting only "the most important" variables [54]. The output of a hierarchical clustering algorithm is a dendrogram and clusters correspond to its branches. However, cutting the tree at a single height is not sufficient for performing effective allocation of variables into clusters because clusters can exist in different heights. This problem is more evident for large and complex data sets which involve variables of multiple dimensions or time-points. Therefore, a more flexible method that enables cutting the dendrogram at multiple heights was needed. In this section, we described the challenge of cutting the dendrogram at multiple levels, which requires an effective representation of both the dendrogram and the original data, we reviewed visualisations of trees and dendrograms as well as the most relevant visualisations for complex data found in the literature. In Chapter 4, we present our contribution to this challenge by introducing a novel visualisation tool, called MLCut, which enables the exploration of multi-level cuts in dendrograms in coordination with a view of the original data.

3.2 Network Inference

After the most important variables are selected, the next challenge for the modellers is to infer a representative network structure (Figure 1.1(b)). However, the route of inferring networks from biological data passes through a difficult optimisation problem which is known as the *"curse of dimensionality"* [27], which occurs because the search space of all possible network models increases super-exponentially with the number of variables [91, 54]. For instance, there are thousands of genes that could affect a biological process. However, it is practically impossible to check how well all possible network models fit the underlying data (a process which is known to be computationally expensive). Thus, inferring the structure of the best network is a known combinatorial optimisation problem which cannot be solved in polynomial time [44, 48]. To render this problem more manageable, hierarchical clustering is used to select only a small number of variables that become nodes in the model. Then, modellers use a combination of mathematical and computational methods to infer the structure of the network model.

There are many computational methods for learning the structure of biological networks and each has its own strengths and weaknesses [86, 194, 115, 88]. Different modelling methods, such as regression [172], correlation, Boolean networks, Ordinary Differential Equations (ODE), Bayesian Networks (BNs), mutual information methods and others, are used for inferring networks in systems biology [194]. Each method has its own advantages and disadvantages. Different algorithms and methods are evaluated in an annual competition, which aims to assess the performance of algorithms in finding known networks, and a combination of methods was found to be a successful strategy for improving results [121]. Also, finding the most appropriate method depends on the aim and scope of the analysis and the data available [121]. For instance, weighted gene co-expression network analysis (WGCNA) [111] became a popular method for constructing genetic networks.

To model such complex and uncertain biological systems, probabilistic graphical models gained popularity over deterministic methods, due to their ability to handle uncertainty and their ease of interpretation. The structure of probabilistic graphical models contains a lot of information about the underlying system because it can describe complex relationships between variables, such as statistical dependencies, in a minimal way [117]. Probabilistic models provide a clear and compact representation of the data and encapsulate details such as the parameterisation of the probability distribution. Their simplicity highlights their qualitative nature and enables modellers to combine mathematical and computational machine-learning methods with their own tacit knowledge to take decisions about the final structure of the model [105]. Other advantages are their ability to deal with confounding (hidden) variables and that they can integrate prior knowledge about the system in their structure [115]. Eventually, these models are key in supporting modellers in gaining insight into biological systems and to develop better and more useful graphical models from experimental data. Thus, graphical models can describe complex systems effectively and therefore find applications in medical prognosis and treatment, social network analysis, natural language processing and robotics [105].

In this thesis, we focus on one type of graphical model, Bayesian networks (BNs), and their usage for modelling biological systems. We focus on this particular type because our collaborators were using Bayesian methods to model their biological systems. In the following, we explain the foundations of Bayesian networks in biology research and challenges modellers are engaged in when they infer Bayesian networks, as well as the specific challenges that relevant visualisation interfaces address in the literature.

3.2.1 Bayesian Networks

Bayesian networks are directed acyclic graphs (DAGs) with nodes representing random variables and being allocated a conditional probability distribution (CPD) that depends on the parents of a node. Edges between nodes show direct statistical dependencies which can be used to reason about causal relationships between the variables. The main disadvantage of Bayesian methods is that inferring (*i.e.* learning) the structure of the network is computationally expensive. There are many possible networks in the search space and the process of selecting and evaluating candidate networks is slow as finding the best network is an NP-hard problem [48, 44]. The search space grows super-exponentially with the number of variables and for networks with more than fifty nodes, given the sparsity of data measurements, it is almost infeasible to achieve good results. However, the quality of a search result has been shown to be better for smaller networks between ten and fifty nodes (*i.e.* variables) [121]. Therefore, in most cases, Bayesian networks used in biology are smaller than fifty nodes.

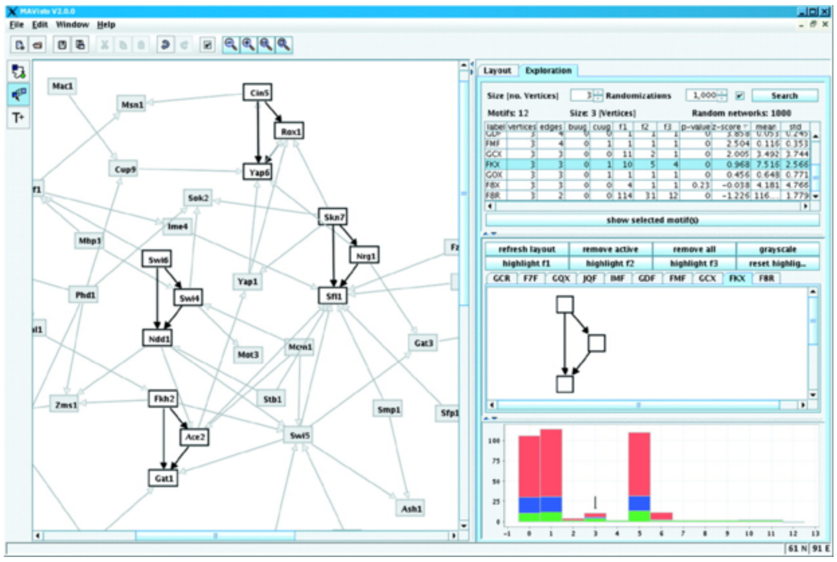
In our case, Bayesian networks describe biological systems, where nodes represent biomarkers and edges represent relationships between them. To find Bayesian networks, our biology collaborators rely on two heuristic search algorithms which are currently used by BANJO [167], a software package that specialises in Bayesian network structure learning. These methods are *greedy search* and *simulated annealing*. The heuristic search algorithms of BANJO check billions of networks but only return a relatively small collection of the highest scoring ones, usually around 100 networks per run (each network has a score representing how well it fitted internal evaluation criteria given the data set). Because of the variety and complexity of the search space, a "good" network structure must be controlled and evaluated by a human analyst. This involves evaluating the performance of multiple runs and understanding the shape of the solution space (distribution of optima). This also means that the analyst is left with the task of *manually* (visually) comparing networks and their respective scores.

Eventually, the analyst's main task is to create a *consensus network*. A consensus network can be either a single network that has been found appropriate, or it can be a logical combination of a set of networks [153]. While BANJO supports the automatic creation of a consensus network, the number of networks taken to determine the consensus network can have a huge impact on quality. Analysts often choose this number "blindly" or decide to include a larger number of networks, hoping this will lead to an improved solution. Also, including networks with similar topology but different edge directions as well as topologically different networks, which can further distort the consensus network. Instead, the analysts need to check the topologies of the highest ranked networks—possibly across several runs of the optimization algorithm. If these highest scoring networks do not satisfy the analyst, e.g., because the solution space contains multiple optima, the analyst has to select multiple networks for combination: every link present in at least one selected network is also present in the final consensus network. However, there can be cases that require filtering specific links. Thus, finding and evaluating consensus networks does not just require visualising all the networks to overview their topologies, but it implies an explorative strategy of searching for an appropriate consensus network and heavily relies on interactive visual previews of the consensus network. In the following Section 3.2.2, we describe visualisation tools and techniques we found relevant to the challenge of network inference.

3.2.2 Exploring and Comparing Networks

There are several automated community detection methods that decompose complex networks in their building blocks, which constitute patterns (motifs) of potentially interesting relationships between elements [129]. MAVisto [158] (Figure 3.9) is a representative tool, which uses algorithms and visualisation for exploring motifs in biological networks. There are many other similar methods, which mostly focus on the task of motif discovery such as: the one proposed by Song *et al.* [168], FANMOD [191], POWRS [52] and SeAMotE [5]. For instance, PheNetic [53] supports the interpretation of molecular profiling data using networks. Other tools suggest motif simplification [62] or summarisation [136] techniques, to enhance network representation, exploration and analysis. Most of those motif discovery methods use heuristics and their results are often neither accurate nor stable. In addition, many of those methods only support the analysis of a single large and complex network, while the challenge of Bayesian network inference requires the combination of features from many different networks, which usually, are relatively small (up to fifty nodes). A more flexible approach would be to use consensus clustering as a framework for combining results [110]. However, it is not clear how to optimally combine results in an automated way because there is a lot of variation in the data, many network formats and many possible network structures. Most importantly, it is hard to integrate tacit knowledge of domain experts in a fully automated method. Thus, there are no sufficiently good algorithmic solutions for detecting the best network model in any data set. Visualisation approaches can provide flexibility in exploring heuristic search results and in constructing consensus networks which are the two main challenges in Bayesian network inference.

Research in network visualisation has yielded a plethora of tools and techniques to improve layout readability, visualise networks with specific attributes, as well as visualise and explore network series (dynamic networks) [183, 24]. Techniques exist for the comparison of two data sets [75] as well as for the visualisation of series of data sets in temporal data [19]. The challenge of finding differences between two graphs has been previously studied and the most common approach is small multiples in which the topology of each network is shown clearly, however comparing networks becomes a cognitive task. Archambault *et al.* present an algorithm which uses the difference map between two graphs to decompose their nodes and edges in order to create a hierarchical structure which can be used to find differences more easily [11, 14]. Semantic Graph Visualiser (Figure 3.10) aims at the comparison and merging of two different networks by superimposing common nodes and by using colour to encode their different properties [10]. ManyNets aims at the comparison of multiple networks by visualising network metrics and metric distribution in table format [68]. To help with the comparison of multiple networks, Hascoët and Dragicevic [85] propose an interactive select-and-hide method to allow comparing multiple topologies by colouring networks and allowing the user to enable or disable networks. The main problem with these methods is scalability with respect to network density (the more links the network contains and the more networks are "superimposed", the more line-crossings occur) as well as the number of networks.



Size [no. Vertices] 3 Randomizations 1,000 ✓ Search

Motifs: 12 Size: 3 [Vertices] Random networks: 1000

label	vertices	edges	buug	cuug	f1	f2	f3	p-value	z-score	mean	std
GDP	3	4	0	0	1	1	1	0	3.858	0.053	0.245
FMF	3	4	0	1	1	1	1	0	2.504	0.116	0.353
GCX	3	3	0	0	11	2	1	0	2.005	3.492	3.744
FKX	3	3	0	1	10	5	4	0	0.968	7.516	2.566
GOX	3	3	0	1	1	1	1	0	0.456	0.648	0.771
F8X	3	3	0	0	4	1	1	0.23	-0.038	4.181	4.766
F8R	3	2	0	0	114	31	12	0	-1.226	116...	1.779

Figure 3.9: Screen-shot of MAVisto analysing a transcriptional regulatory network of Saccharomyces cerevisiae with different perspectives to explore motifs. On the left-hand side, the network is shown with the motif-preserving layout of highlighted matches of the feed-forward loop motif. On the right-hand side, all discovered motifs can be further analysed. Detailed information is presented in the motif table (top), the structure of the currently active motif is displayed in the motif view (middle) and the motif frequency spectrum is shown in the motif fingerprint (bottom) (as found in Schreiber *et al.* [158]).

Besides automated and general network visualisation approaches, there are visualisation tools which specifically target the analysis of Bayesian networks. VisNet [196] has been explicitly designed to visualise properties of a single Bayesian network using a node-link representation. Elvira [109] uses a similar approach but gives more emphasis to the interpretation of the Bayesian networks. Kadaba *et al.* [97] use animation to show causal relationships in networks. NetEx [49], which is a Cytoscape plug-in, targets the problem of visualising large Bayesian networks as node-link diagrams. The Visual Causality Analyst [184] provides a GUI that supports causal reasoning and it also uses node-link diagrams to represent Bayesian networks. CompNet [108] (Figure 3.11) facilitates the comparison of networks visually and via metrics. The tool presents an overlay of a number of networks and statistics on the presence or absences of nodes in given clusters of this union. However, all these tools work only for one or a small number of Bayesian networks and none of them supports any specific visualisation or interaction capabilities for the exploration of heuristic search results and the creation of consensus networks. Thus, the limitation is visual scalability in terms of the number of networks they can show in a readable manner. This thesis targets the task of exploring the output of heuristic search algorithms, such as the ones included in BANJO, which can generate hundreds of networks in a single run.

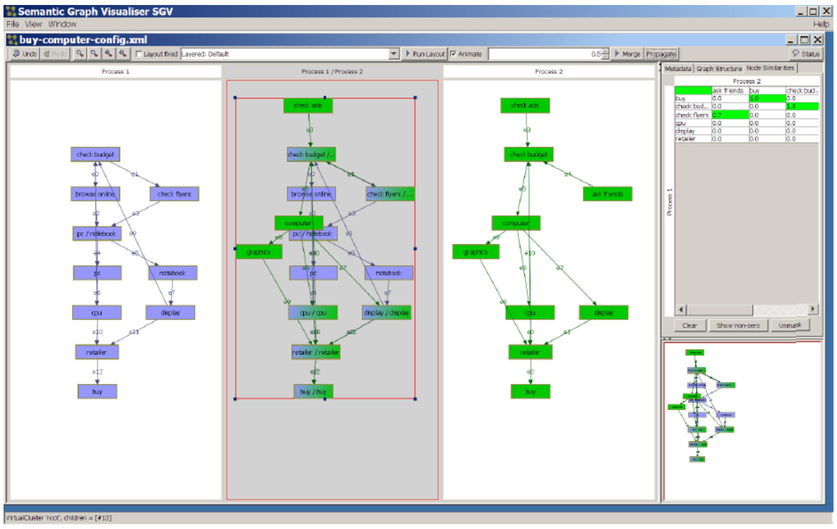


Figure 3.10: The Semantic Graph Visualizer (SGV) comparing two process graphs representing workflows involved in buying a computer. (as found in Andrews *et al.* [10]).

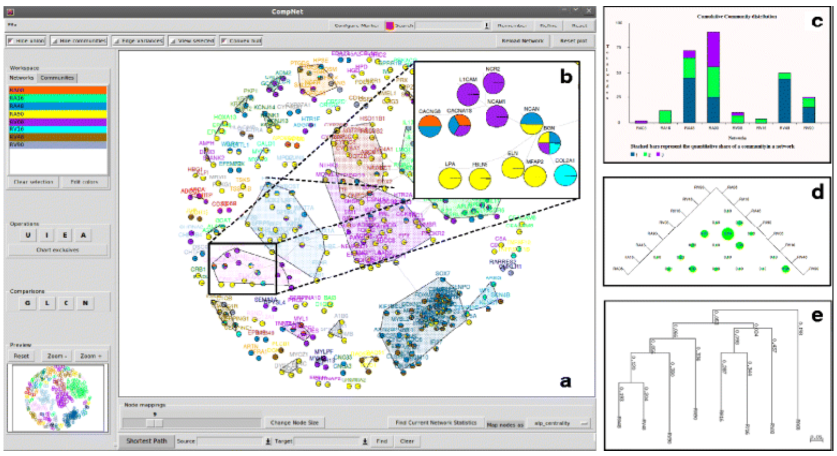


Figure 3.11: (a) CompNet canvas displaying the union of eight protein-protein interaction networks. The names of nodes belonging to different communities are marked with different colours. (b) The 'pie-nodes' representation enables to identify the presence/absence of individual nodes across the compared networks. (c) The cumulative community distribution plot (d) Bubble chart representing similarity between networks (e) Hierarchical tree built using network similarity (as found in Kuntal *et al.* [108]).

There are many network analysis tools in the literature, which are either generic or they are tailored to perform specific visualisation tasks. Also, there are several reviews that compare the different features of those tools, mainly focusing on tasks that those tools can perform [154, 141, 171]. Although there is a rich literature in graph comparison tools and algorithms for motif discovery, there is a lack of tools that address the challenge of network inference [7]. Probably this is because our targeted challenge is only relevant to the post-processing of results produced by network inference methods, which generate relatively large numbers (possibly hundreds) of candidate networks. However, we found that the design of some already existing visualisation tools and techniques could be extended to also support the exploration and comparison of heuristic search results used for inferring Bayesian networks. In Chapter 5, we describe how we evaluated some of those approaches with domain experts who wanted to infer networks from their data. Then we present the design of BayesPiles, a novel visual analytics tool which has been created based on the design of an already existing tool for exploring dynamic graphs, called MultiPiles (Figure 3.12) [18]. BayesPiles can be used as a visual analysis component within the experimental process that modellers often follow to infer the structure of biological Bayesian networks.

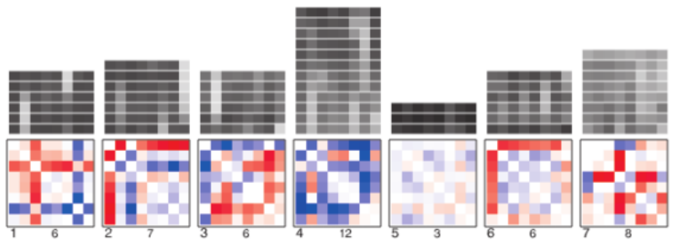


Figure 3.12: Small multipiles (*i.e.* MultiPiles) create lists (piles) of similar dense graphs in a time line, visualised using the technique of adjacency matrix. Larger piles indicate longer occurrences of the graph in the time line [18].

Summary of challenge 2: Heuristic search algorithms sample the search space of all possible network models based on parameter settings and a network score that encodes the fitness of its structure to the underlying data. The purpose is to find a final consensus network which is representative and explains the observations collected from the biological system. In this process, the role of the modeller is to guide the heuristic search (choosing the algorithm and setting its parameters) and to decide on a method that determines the structure of the final consensus network. Our modellers were interested in exploring sets of multiple candidate networks generated by a Bayesian network algorithm [166], implemented and distributed freely as part of the software package called BANJO [167]. The process of exploring such data sets was found to be very complicated and time-consuming. Thus, there was a need for developing a visualisation approach which can strengthen the argument for choosing a particular consensus network as the final model, and which can also speed up the process of finding it, by providing visual feedback to the user. Moreover, a visual analysis approach can facilitate the reproduction of networks by different modellers, supporting the cross-validation of research results. Based on a literature review and modellers' feedback, we identified MultiPiles [18] as the most promising technique for exploring many networks. In Chapter 5 we present BayesPiles, a novel tool inspired by MuliPiles which can support domain-specific visual analysis tasks for inferring Bayesian networks.

3.3 Multivariate Network Analysis

3.3.1 Dynamic Bayesian Networks

When the data contain measurements over time (time-series), it is possible to infer information about the dynamics of the interactions in the biological system [104]. When such information is incorporated into the edges of a Bayesian network then the resulting network is called a dynamic Bayesian network (DBN). Using the terminology from mathematical graph theory, static-BNs are directed acyclic graphs (DAGs), while dynamic-BNs (DBNs) permit feedback loops, which result in directed graphs (DGs) (Figure 3.13). Thus, DBNs are an extension of Bayesian networks that can model dynamic systems [135]. Although the structure of a DBN does not change over time, it contains different types of edges which encode information about the dynamics of interactions in the underlying system. In other words, a DBN is a temporal probabilistic model which not only can describe the state of a system at a certain time point (as a BN does), but also it can describe interactions in the system over a sequence of discrete time-slices, including self-correlation [104].

A traditional DBN (Figure 3.13) can be described by a series of time-slices with edges that not only connect pairs of nodes in the same time-slice (*i.e.* intra-time-slice edges), but also edges that connect pairs of nodes that belong to different time-slices (*i.e.* inter-time-slice edges) [135]. In other words, edges in DBNs can skip time-slices and connect nodes of previous time-slices with nodes of future time-slices. The number of time-slices skipped is an integer (usually between 0 and 3) that indicates the delay of an interaction. Those delays are called *Markov lags* (MLs) and correspond to different edge types in DBNs.

Picking the frequency of time-slices (time granularity) is important, to capture the dynamics of a system and it usually coincides with the rate in which measurements are collected from the system (i.e. the sampling rate). For example, if the sampling rate is one sample per day and this corresponds to one time-slice in the model, then an edge of type ML1 (ML = 1) would only connect nodes between previous and current slices (skipping one slice) and the delay of the interaction would correspond to the time period of one day.

Edges of ML1 are called persistence edges because they persist their state from one time-slice to another. In Figure 3.13 (a) there are two persistence edges (of type ML1) between node D_{t0} and nodes B_{t1} and F_{t1}. When a DBN includes many persistence edges, it means that the sampling rate was set in an effective way for finding important interactions which have a constant effect to the state of the system over a long period of time. However, there are more types of MLs that can appear in a DBN.

When an edge is of type ML2 (ML = 2) then two time-slices are skipped and a link appears to connect two nodes every two time-slices. This is the case for the edge that connects node D_{t0} with

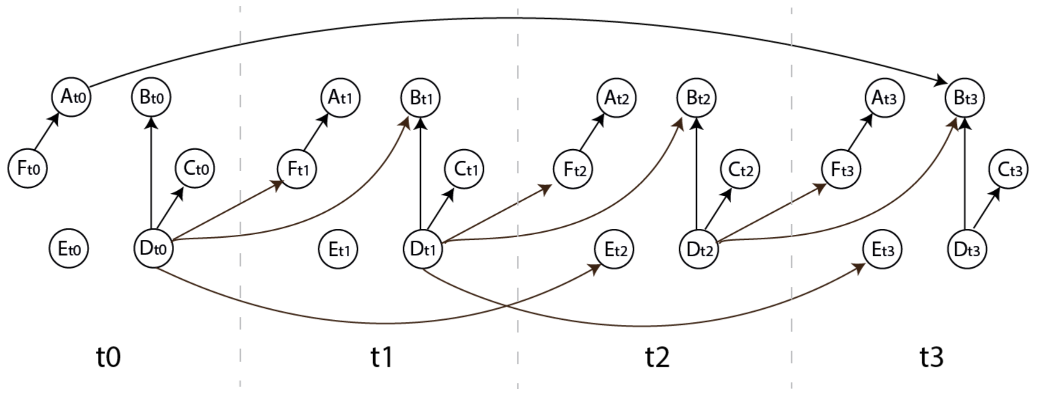


Figure 3.13: The traditional representation of a DBN with edges connecting nodes in different time-slices. Edges that show self-correlation have been removed to create a more clear diagram. Self-correlation edges create a feedback loop that starts and finishes in the same node.

node E_{t2} in Figure 3.13. This edge indicates that the interaction between those two nodes is two times slower than the sampling rate. So for a sampling rate of one sample per day, this interaction delays for two days before it is observed in the data.

In a similar way, an edge of type ML3 (ML = 3) indicates that the interaction is three times slower than the sampling rate. An example of an edge of type ML3 is shown in Figure 3.13 between nodes A_{t0} and B_{t3}. When there are many edges of this type in the model it means that the sampling rate (time granularity) is too fine. MLs larger than 3 are usually not considered in the heuristic search by biologists unless oversampling took place. In that case, sub-sampling or the aggregation of sequential time-slices in which samples have the same value may be considered. A coarser sampling rate would result in DBNs with edges appearing in lower MLs (between 0 and 3). However, oversampling is rare because taking sample measurements of biological data is usually expensive.

Finally, when the edge is of the type ML0 (ML = 0) then it connects nodes of the same time-slice in the network. Edges of this type are *intra-time-slice edges*, while edges of any other type are *inter-time-slice edges* [105]. For example in Figure 3.13 there are three edges of type ML0 connecting nodes F_{t0} with A_{t0}, D_{t0} with B_{t0} and D_{t0} with C_{t0}. Edges of this type indicate interactions that are instantaneous. Those dependencies in the model react faster than the sampling rate. However, it is not clear how much faster those interactions are. Thus, having many of those edge types in the model is not very informative about the dynamics of the system (static-BNs only contain edges of this type). To get a better sense of how fast these interactions occur, a more fine-grained experiment that collects samples in a faster time rate would be needed. In summary, the types of edges (Markov lags) commonly found in DBNs are usually interpreted in the following way by the modellers.

ML0: The sampling rate was slower than the speed of the interaction.

ML1: The sampling rate was the same as the speed of the interaction.

ML2: The sampling rate was 2 times faster than the speed of the interaction.

ML3: the sampling rate was 3 times faster than the speed of the interaction.

Although this is how Markov lags are usually interpreted, the sampling rate may differ during an experiment due to technical, practical or ethical reasons. Thus, modellers often interpret the various edge types taking into account possible variation in the sampling rate which determines the period between time-slices.

Visualising DBNs effectively is a challenging problem because it is difficult to encode multivariate data associated with the edges in a network structure. Thus, the resulting visualisations often appear cluttered. Consequently, although DBNs are important for modelling a variety of dynamic systems in many scientific domains, there is a lack of visual analysis tools that can support modellers in inferring DBNs. In the following section, we present a review of the most relevant tools and techniques found in the literature, for visualising DBNs or other multivariate networks.

3.3.2 Visualising Multivariate Networks

Several techniques have been proposed in the literature for representing and exploring multivariate networks. PivotGraph [189] uses a layout that arranges a node-link diagram between the X and Y axes that encode two categorical dimensions (Figure 3.14). GraphTrail [63] uses a hybrid bar chart which can encode additional dimensions using arcs to connect the bars. Pretorius *et al.* [145] propose an approach that integrates lists of edge labels with clusters of node attributes in a single view using links, colour and size to encode relationships and additional information. Shamir *et al.* propose the application of interactive queries to retrieve information from multivariate networks [162]. However, the exploration is limited to a particular set of available queries.

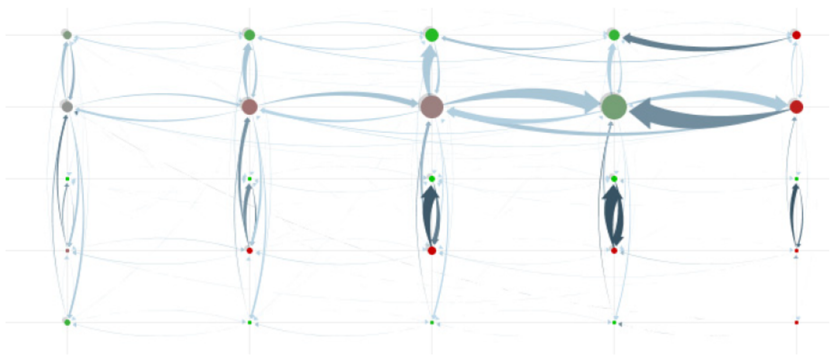


Figure 3.14: A PivotGraph visualisation of a large graph rolled up onto two categorical dimensions [189].

Although these techniques are useful, they often suffer from multiple edge crossings when networks get denser (Figure 3.15 left view). Therefore, several approaches use node or edge aggregation to deal with the visual clutter in node-link diagrams. For instance, Clustervis [39] uses a radial ring layout to arrange aggregates of nodes. Elzen *et al.* [177] use intuitive overviews to represent clusters of nodes to guide user selection (Figure 3.15). Grouseflocks [13] supports a hierarchy attached to a network that describes its partitions as multiple abstraction layers. Although there has been an attempt to maintain the readability of the node-link topology after clustering nodes [12] or edges [150], useful information can easily get lost and the representation can become too simple for low-level tasks such as network comparisons, or for finding nodes and edges in the network.

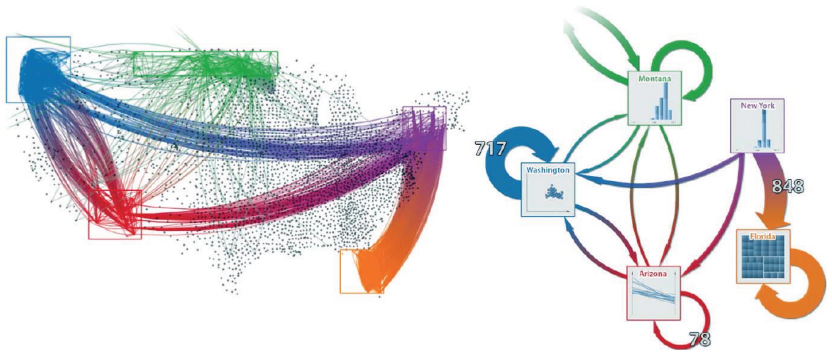


Figure 3.15: Multivariate network exploration using selections of interest, detail view (left) and high-level infographic-style overview (right) [177].

As an alternative to node-link diagrams, matrices have also been used to represent multivariate networks. For instance, designs for encoding weighted networks of two edge types have been examined for comparing brain connectivity networks [9] (Figure 3.16). In another study, edge-weighted encoding in matrices has been explored [42]. Such approaches usually encode weight using opacity or size. However, it is unclear how those encodings could scale to support more than two types of edges between a pair of nodes.

Using glyphs is a way for encoding more than two variables in a small area (*e.g.* the cell of a matrix) and there are several reviews for methods that represent multivariate data as glyphs [186, 70, 33]. For instance, they have been used to represent cliques [62], while Gestaltlines [36] have been used in matrices to represent longitudinal social networks [35].

Small Multipiles [18] and Matrix Cubes [17] can represent how the structure of a network changes over time. Those visual encodings for dynamic networks in matrices can be relevant to multivariate networks. However, in our case, the challenges are different because in DBNs a pair of nodes can contain multiple types of edges and modellers were interested mostly in tasks related to the distribution and comparison of edge types rather than structural changes over time.

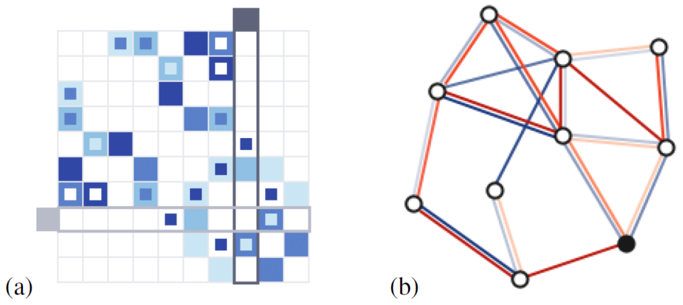


Figure 3.16: Alternative superimposed (a) matrix and (b) node-link visualisations supporting weighted graph comparisons [9].

Except for integrated visualisations, multiple coordinated views (MCV) have also been used to deal with the increased complexity of multivariate networks. For instance, Detangler targets the task of group cohesion in networks with multiple edge types using dual linked views and two abstraction levels [148]. GraphDice [31] on the other hand, extends the X, Y layout of PivotGraph adding interactive features such as animated transitions between three dimensions of categories. Pairs of categorical dimensions are shown as two-dimensional small multiples in a coordinated view (Figure 3.17). Whereas MCV methods are useful, integrated approaches are preferred for visualising multivariate networks because they save space by containing both the structure and the data in the same view [96]. This also makes them more efficient since users are not required to look at separate views for finding relations in the network [77]. However, embedding additional multivariate data into standard network representations that use node-link diagrams or adjacency matrices is not intuitive and it can often result in visual clutter.

In Chapter 6, we present a user study for formally evaluating the effectiveness of visual encodings in matrices when users were performing common visual analysis tasks related to DBN inference. The results of this study informed the design of BayesPiles which was extended to also support the exploration and analysis of DBNs.

Summary of challenge 3: Time-series data can be used to infer information about the dynamics of interactions in biological systems. Extending Bayesian networks, DBNs encode this information in their multiple types of edges, called Markov lags. Modellers are interested in exploring the results of heuristic search runs that produce multiple DBNs, to understand the dynamics of the underlying biological system. However, it is hard to represent and analyse collections of DBNs because they contain edges of different types. In Chapter 6, we present a formal user study that tested the effectiveness of visual encodings for edges in matrices. The results informed the design of a visual analysis tool (BayesPiles) for exploring DBNs and the tool was evaluated by domain experts.

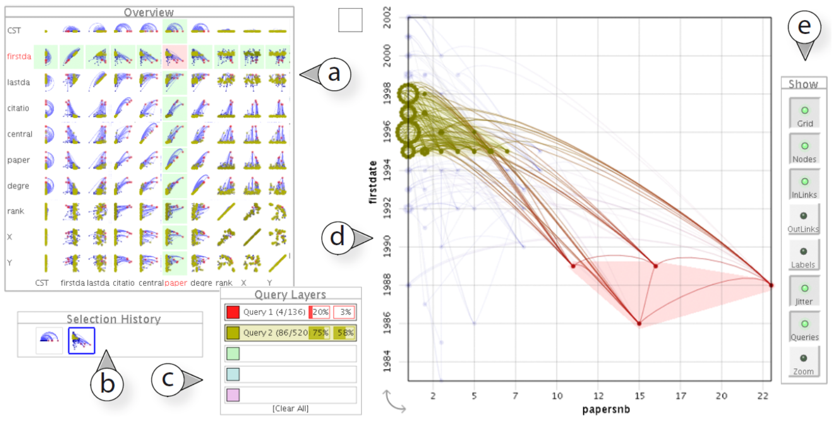


Figure 3.17: Exploration of the InfoVis 2004 Contest co-authorship data set using GraphDice. On the left is the main visualisation window of GraphDice including (a) an overview plot matrix, (b) a selection history tool, (c) a selection query window, (d) the main plot, and (e) a toolbar [31].

3.4 Summary

Three research challenges have been identified in modeller's workflow of inferring probabilistic biological networks. Those challenges were the main research objectives targeted in this thesis. The first challenge is related to the variable selection step during which hierarchical clustering is used for allocating variables into clusters. In our literature review, we found that HCE [161] is the most relevant tool for performing this step. However, HCE does not support multiple-level cuts and large dendrograms often appear cluttered. In Chapter 4, we present MLCut, a tool that supports an effective dendrogram representation and clustering analysis tasks, such as cutting the tree at multiple levels.

The second challenge is related to the network inference step during which heuristic search algorithms sample the space of all possible networks. The challenge is to help modellers guide the heuristic search and decide on a method that determines the final consensus network. Our literature review showed that MultiPiles [18] is the most relevant visualisation approach for supporting this challenge. In Chapter 5 we present how BayesPiles extended MultiPiles to support domain-specific analysis tasks, such as the manual construction of a consensus network.

The third research challenge is related to the incorporation of information about the dynamics of the biological system into the inferred network model. Representing this information visually is challenging because the resulting networks contain edges of multiple types. Our literature review showed that although matrix-based representations can be effective for representing dense networks, there was no previous study that evaluated the performance of visual encodings for multivariate networks in matrices, in which the multivariate data is associated with the edges. In

Chapter 6, we present a formal user study that investigates effective encodings for performing network analysis tasks related to the inference of dynamic Bayesian networks and how the results of this study informed the design of BayesPiles. Then we present how this extension was evaluated by modellers.

This thesis suggests that modellers' efficiency can be improved when visualisation is used as part of their network inference workflow. Following the nested model framework, three biological analysis challenges were identified and mapped to visualisation challenges. For each of those challenges, we presented an overview of necessary background knowledge and a literature review which describes the most relevant visualisation techniques and tools. In the following, we present how each of the three research challenges has been addressed in practice. Each of the following three chapters presents a contribution to knowledge. In the next chapter, we present the design of MLCut, a novel visualisation method for performing hierarchical clustering analysis.

4 Exploring Multi-Level Cuts in Dendrograms with MLCut

In the previous chapter, we presented related work for each of the three research questions addressed in this thesis. In this chapter, we focus on answering the first research question: **Q1** ***"How to provide visual support for the effective hierarchical clustering of many multidimensional variables?"*** To investigate how visualisation could help to address this question, we applied the nested model methodology [132], which led to the following questions:

Q1a What are the requirements of domain scientists who select variables based on the results of hierarchical clustering algorithms?

Q1b Which design decisions and operations can lead to effective visualisations for exploring the results of hierarchical clustering algorithms applied to many multidimensional variables?

We discuss the design of a novel visualisation tool, MLCut following the nested model methodology approach [132]. MLCut, enables the interactive exploration of different clustering scenarios by allowing its users to cut the dendrogram at multiple heights. Clustering assignments are shown in a coordinated view of the original multivariate data, which is updated in real time as the user interacts with the derdrogram. In Section 4.1, we describe how the first level of the nested model was applied to the hierarchical clustering analysis challenge, followed by the second level in Section 4.2. By the end of these two steps of our methodology, we answer Q1a summarising a list of requirements. Regarding Q1b, in Section 4.3 we present the design and implementation of two MLCut prototypes and how they satisfy the requirements (third and fourth levels of the nested model). Also, in Section 4.4, we discuss how MLCut has been evaluated with real time-series gene expression data in a case study. Finally, in Section 4.5, we present a summary of the contribution. A paper describing MLCut was published in the *Computer Graphics and Visual Computing (CGVC)* conference [180].

4.1 Domain Problem Characterisation

As part of the domain problem characterisation, which is the first level of the nested model (Figure 2.5), it was important to establish a shared understanding of the domain problem with the domain scientists [159]. We received anecdotal feedback from three computational biologists with experience in hierarchical clustering, following the example of other successful design studies [159, 126]. Over a period of two months, we held two one-hour informal meetings with each of them (six meetings in total). During these meetings we let them describe their data sets, the method they were using to organise them in groups, as well as the problems they were facing during the step of variable selection. Two of our analysts were interested in reducing the number of variables in their data sets using hierarchical clustering and then infer network models. In particular, they were interested in finding data records of similar temporal profiles and then aggregate them taking their averages so that a single variable would be created from each cluster.

Our first collaborator was an experienced computational biologist who often uses hierarchical clustering to reduce the number of variables and infer probabilistic network models. The second was a biology graduate with experience in computational methods who wanted to infer models from time-series gene expression data as part of a research project. Our third collaborator was a senior statistician specialised in computational methods for analysing genetic data and hierarchical clustering was used to map collections of DNA markers with a set of chromosomes.

Inferring network models is challenging mainly because of the large number of variables affecting the state of the biological system [48]. To infer useful networks, computational biologists first need to reduce the number of variables, including the most important and excluding those that are redundant [57]. The challenge is to find and organise variables that are similar into groups. Computational biologists commonly use the unsupervised method of hierarchical clustering to allocate biological variables into groups. This is a popular method among scientists because they do not need to specify in advance the number of clusters. Also, the algorithm produces a *dendrogram* (i.e. a tree) that shows similar objects close to each other, encouraging the exploration of the results. The main task for the user is to explore the data and decide which branches of the tree correspond to the different clusters (*i.e.* groups). Usually, this is done selecting a similarity threshold that determines a single height for cutting the tree. However, for large and complex data sets, applying a single similarity threshold is not sufficient for effectively allocating variables to clusters because these may exist at different heights of the dendrogram.

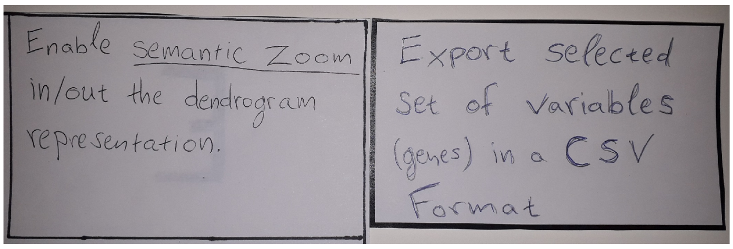
The size and complexity of the data set often result in cluttered dendrogram representations that do not help to decide where to cut the branches that correspond to different clusters. Finding more representative clusters required an interface that could support the exploration of the original data in coordination with the dendrogram. Our challenge was to create a visualisation tool that would allow computational biologists to observe their data and cut the dendrogram so that branches correspond to clusters. Each of the identified clusters of variables could be either studied as a separate module, or it could form an aggregate represented by a single node in the network (usually taking the mean).

4.2 Requirements

After understanding the domain problem of variable selection in the context of hierarchical clustering, we proceeded with the second level of the nested model during which the data were abstracted from the tasks. For this step, it was important to collaborate closely with the computational biologists, to understand their data and tasks and derive a list of requirements that our tool should support.

To identify, refine and prioritise requirements we performed four *card sorting* sessions with two of the computational biologists consulted in the previous level of the nested model. We performed two sessions with each participant separately, following the three steps of *preparation*, *execution* and *analysis* described by Sakai *et al.* [155]. At the end of these sessions, we performed an additional short session during which both computational biologists were asked to collaborate in order to organise and refine requirements. Before conducting the card sorting sessions, we followed the application procedure for acquiring cross-university ethical approval.

During the first session, the computational biologists were asked to verbally describe operations and tasks they would like to perform when they select variables using hierarchical clustering. Then, they were asked to write these requirements on empty cards (Figure 4.1). By the end of the first session, we had two collections of requirements related to hierarchical clustering that our collaborators had generated.



Enable semantic Zoom in/out the dendrogram representation.

Export selected set of variables (genes) in a CSV Format

Figure 4.1: A picture showing two of the handwritten cards after a card sorting session with one of the participants.

As a preparation step before the second session we gathered all cards creating a single collection of requirements. We found that many of the requirements were common between the two participants. The next step was to refine them, remove duplicates, type them and print them to make them easily readable for the next session. During the second session, participants were given all the cards from the previous sessions (four cards in total) and they were asked to revise and update their content as required. Also they were asked to hand-write new requirements on empty cards. The participants generated four requirements during the first session and another four during the second. At the end of the second session, both participants were asked to collaborate and organise these eight cards in groups depending on how similar they were. Complementary and duplicate requirements could be easily detected as they would appear in the same group (Figure 4.2). By the end of the second session, we had an updated collection of requirements. Duplicates were removed and similar or complementary requirements were merged to form more concise requirements. Answering Q1a, we found that the design of our tool should support the following five requirements:

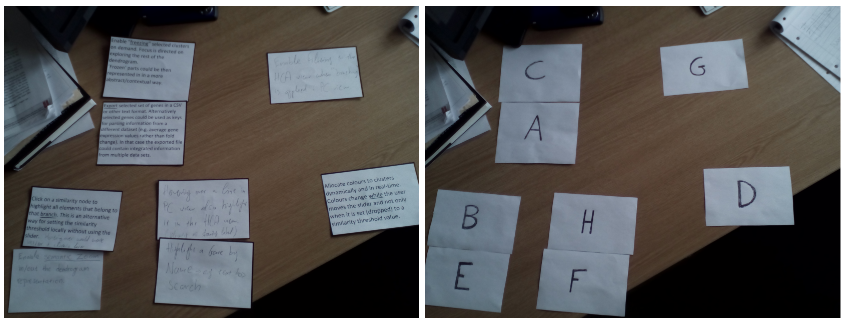


Figure 4.2: Pictures from the second card sorting session. New requirements and requirements found in the first session were grouped based on their relevance.

R1 - An effective dendrogram representation that scales well for large data sets. Analysts were interested in a scalable dendrogram representation which could fit in a standard monitor display and could also show which are the different clusters.

R2 - An effective representation of the original data, in addition to the dendrogram representation. Analysts wanted to be able to visually inspect their multidimensional data in coordination with the dendrogram.

R3 - The ability to interactively explore the dendrogram and the representation of the original data, in different levels of detail. Analysts wanted a method that would allow them to steer the exploration of the dendrogram and their data so that they could detect and select potential clusters on demand.

R4 - The ability to maintain multiple cluster selections on display during the exploration process, without losing a view of the whole data set. Analysts expected to have flexibility and control over their variable selection process and to be able to compare clusters while they were interacting with the interface.

R5 - The ability to export selected clusters so that they could be used for further analysis: this was important for the analysts so that they could proceed easily to next steps of their analysis workflow.

These requirements have implications for interface design including scalability to large dendrograms, representing the original multidimensional data, showing selected clusters, support for cutting the dendrogram at different heights and comparing clusters. The data that needed to be visualised to satisfy these requirements involved both the dendrogram and the original multidimensional data. In the following Section 4.3, we describe how the third level of the nested model was applied to design effective visual encodings and interactive operations for satisfying these requirements and for answering research question Q1b.

4.3 Design and Implementation

Applying the third level of the nested model, we investigated and reviewed potential designs for dendrograms and representations of multidimensional data found in the literature. We evaluated design ideas and we developed prototypes for the interface while we were receiving anecdotal feedback from the computational biologists. To improve the design of our tool, we followed an iterative process of continuously refining design decisions and evaluating results. For a period of four months, we arranged to meet twice a month with our collaborators to receive feedback regarding how well our design decisions supported the requirements listed in Section 4.2. However, while most of the design decisions were made after consulting the end users, some of the design choices were based purely on studies about human perception and cognition. For instance, the visual encoding was based on principles related to the effectiveness of visual variables for representing certain types of data [119, 47, 28], while the colour palette was created using *ColorBrewer* [84]. Details about design decisions related to visual encoding and user interface (UI) controls are explained in the next sections which describe the two phases for developing MLCut, focusing on the dendrogram design and the way it was linked with a representation of the original multidimensional data. Those two phases led to the design and implementation of two prototypes of MLCut.

4.3.1 Design Process

The first step in the design process of MLCut was to consider potentially effective representations for the dendrogram and the original multidimensional data, conducting a literature review as described in Section 3.1. We found that the most relevant tool to our problem was the *Hierarchical Clustering Explorer (HCE)* [161], which uses two coordinated views to represent a top-down dendrogram linked to a heatmap of the original multidimensional data. However, the top-down dendrogram representation did not scale well and zooming and panning were often needed. Also, while the heatmap was found useful for representing multidimensional data, our collaborators found it less intuitive than parallel coordinates when it was used to visualise time-series data. Most importantly, HCE supports only single-level cuts for allocating variables into clusters, while one of our requirements was to support multiple-level cuts.

Two main phases of design and development took place, which resulted in two prototypes. Each of those prototypes was used by different research groups for analysing different data sets. The design of the first prototype was mostly focusing on the effective exploration of the dendrogram. The challenge was to visually represent and explore large dendrograms in a scalable way.

The design of the second prototype was mostly focusing on linking the dendrogram with a representation of the original multidimensional data. The challenge was to create an interface that would enable users to interactively cut the dendrogram in multiple levels and see the effect of their actions in the original data.

4.3.2 Dendrogram Design

The first phase of developing MLCut addressed the problem of exploring large dendrograms. To satisfy R1, described in Section 4.2, we considered alternative two-dimensional (2D) tree layouts such as node-link, icicle, radial, concentric circles, nested circles, treemap and indented outline, described in McGuffin *et al.* [124]. In one of our first attempts to visualise the dendrogram, we implemented a version of the top-down node-link tree representation (Figure 3.3 (a)). The user could zoom in to inspect branches of interest in more detail or zoom out to see the whole dendrogram, as shown in Figure 4.3. This dendrogram layout was suitable for identifying outliers and clusters based on the length of their branches. Long branches were probably part of the same cluster, while data records that did not belong to any of the branches indicated possible outliers. This layout made it easy to compare between branches when the whole dendrogram was displayed on the screen. However, considering the limitations of a standard monitor display, this layout did not always produce intelligible results. This was mainly because, above approximately 400 leaves, the size of the hierarchical structure was too large to browse. In this layout, it was unrealistic to expect users to visually compare edge lengths in large dendrograms of potentially thousands of edges. Moreover, clustering assignments encoded on the leaves or the branches of the tree were not easily discernible because of their small size, while using labels resulted in visual clutter.

The updated design of our dendrogram was based on known perceptual principles regarding the suitability of certain visual variables (such as "position", "size", "shape" etc.) in encoding information for performing perceptual tasks [119, 47, 28]. In our updated design, we adopted a space-efficient radial layout that enabled additional encoding for the clustering assignment using "colour hue". Whilst Burch *et al.* [37] shows that radial node-link representations take longer to read than top-down node-link views, at least up to 500 nodes, radial node-link layouts are more space efficient than top-down node-link representations [124], and also more efficient than left-to-right node-link representations if labels are not shown, as is the case here. The radial layout utilizes better the space available for displaying the data and limits the use of scrollbars. Because the hierarchy is wide, the top-down tree layout in Figure 4.3 would not be visible in a single view without scrollbars. Another option is to zoom out to a scale where the dendrogram was entirely visible, but the colour of the nodes, used to encode the clusters, would not be distinguishable. For the same task, a radial dendrogram is much more compact, allowing the different colours to be distinguished, while the whole data set can fit in a standard monitor display. Identifying the different clusters in the dendrogram is a nominal perceptual task for which "colour" is a well-suited visual encoding [119]. Following the same logic, edge lengths were encoded using the visual variable of "size". In addition, we quantified the property of edge length and we enabled

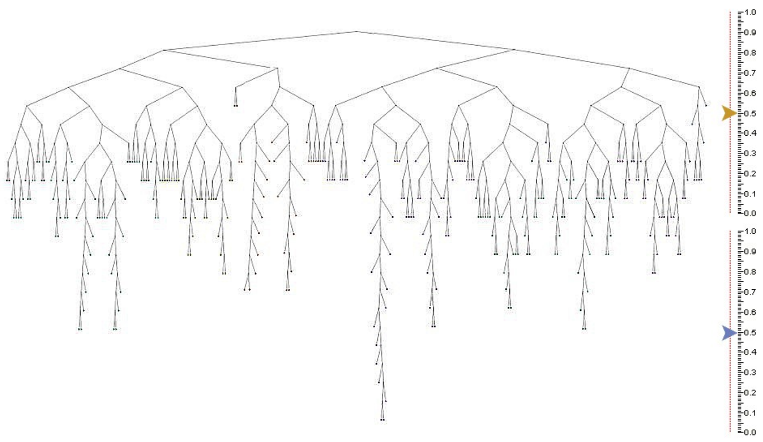
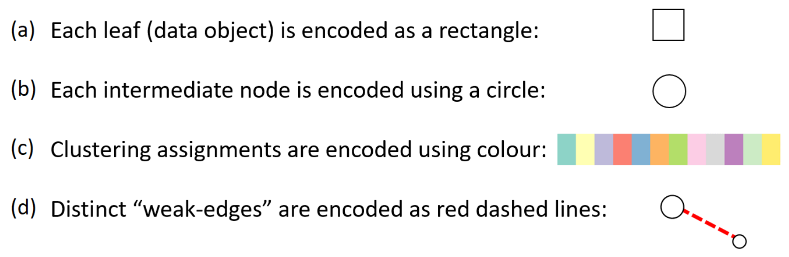


Figure 4.3: Dendrogram displayed in an earlier version of MLCut using a version of the top-down node-link tree layout. This dendrogram layout is suitable for identifying clusters and outliers based on the length of the branches.

users to apply a distinctiveness threshold using a *dynamic slider* [6]. Longer than the threshold edges (*i.e.* "weak"), were encoded using easily discernible red, dashed lines. Dynamic sliders are discussed in more detail in Section 4.3.3. The visual variable of "shape" was used to distinguish between the two different categories of leaves and intermediate branch levels. Data records, which are leaves in the dendrogram, are represented as rectangles and dissimilarity scores, which are always intermediate branch nodes, are represented as circles of diameter proportional to their value. The node with the largest diameter and the highest dissimilarity score is the root of the tree. A summary of the visual encoding used in the dendrogram is shown in Figure 4.4.


(a) Each leaf (data object) is encoded as a rectangle: □

(b) Each intermediate node is encoded using a circle: ○

(c) Clustering assignments are encoded using colour: 🟩🟨🟪🟥🟦🟧🟩🟪⬜🟪🟩🟨

(d) Distinct "weak-edges" are encoded as red dashed lines:

Figure 4.4: A summary of the visual encodings used for representing the dendrogram in MLCut.

Interacting with the dendrogram is simple as scientists can use the middle wheel of their mouse or buttons in the user interface to easily apply *semantic zooming* to achieve a better view on branches of interest. Hovering over elements of the dendrogram reveals more detail, such as their dissimilarity score (for circles), or their name and vector of values (for rectangles). Clicking on a rectangle selects it, while clicking on a circle selects the whole branch. A thicker border is used to highlight the selected items. Moreover, to partly support R4 and enable comparisons, variables of particular interest can be "locked" and remain highlighted by double-clicking on their mapped rectangle at the dendrogram. This action would add texture to the rectangles, as shown in Figure 4.5 and the selection would persist until the user would double-click again on those elements to "unlock" them.

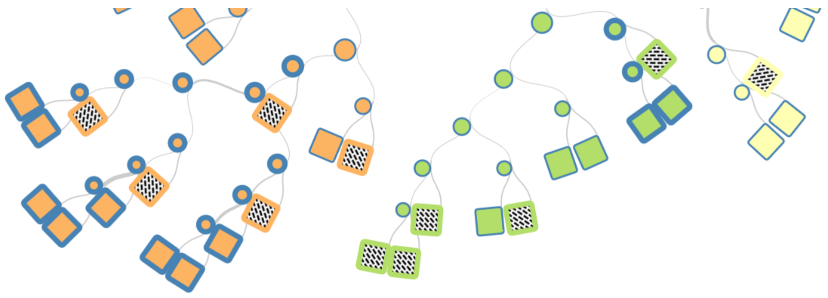


Figure 4.5: Texture added to data records that have been double-clicked. Those records will remain highlighted ("locked") in the interface and not affected by any cluster selection.

4.3.3 Dynamic Sliders

To enable the exploration of the dendrogram in a controlled and reproducible way, and thus, partially support requirements R3 and R4, we implemented two dynamic sliders for setting similarity and distinctiveness thresholds respectively. A global similarity threshold can be applied using the first slider. Branches that belong to the same cluster get the same colour. In Figure 4.6, snapshots I and II demonstrate the process of merging a large number of smaller clusters into three main groups by moving the top slider, which controls the similarity threshold. This interaction is useful for identifying the main clusters and also for testing the different scenarios that the single-height approach can investigate in a consistent and reproducible way. The algorithm that dynamically allocates colours to clusters as the user moves the slider, rotates between the 12 colours of a palette (Figure 4.4 (c)) in a way that only new clusters get different colours, while existing clusters maintain their colour to make the merging/division of clusters easier to spot. The efficient allocation of colours to clusters, complies with the fourth (and last) level of the nested model as the algorithm updates cluster selection quickly and the interaction with the interface takes place in real time.

The second dynamic slider can be used to identify long (or "weak") edges that indicate heterogeneity within clusters. It helps to point out nested clusters, which appear considerably more distinctive than the larger ones in which they may seem to belong to. In large dendrograms, it is difficult to compare edge lengths. Hence, the second slider sets the maximum allowed similarity

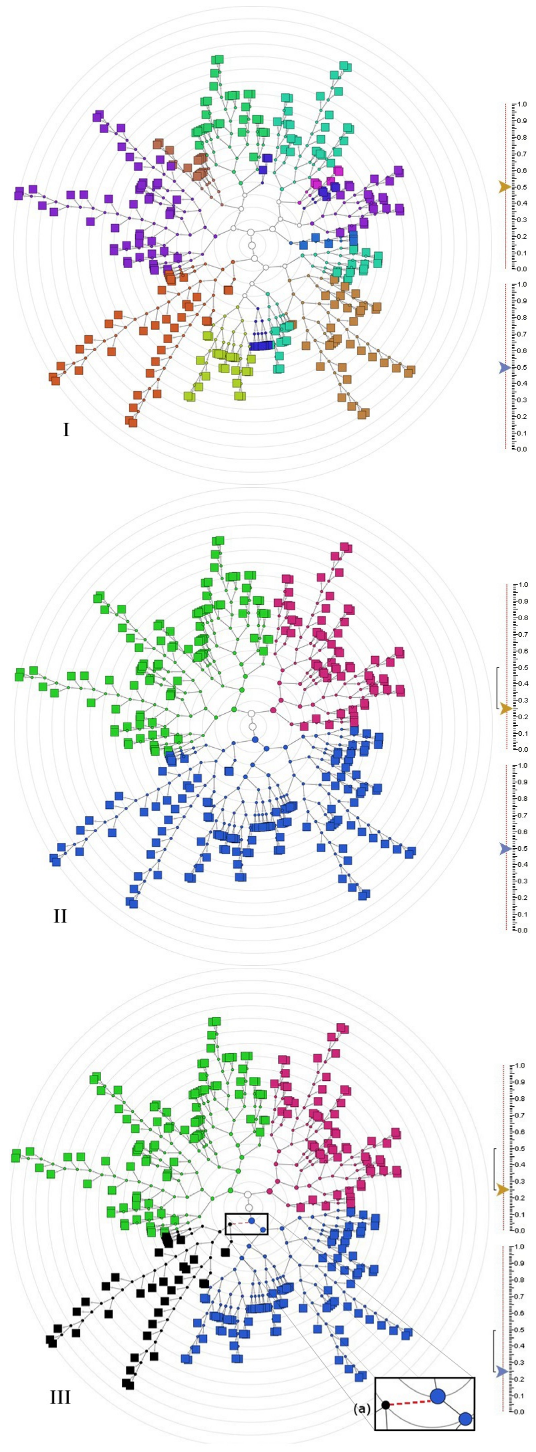


Figure 4.6: Dynamic query sliders in use. The top slider in II sets the similarity threshold and the bottom slider in III sets the distinctiveness threshold.

distance between a parent main cluster and a child sub-cluster. Distinctive *"weak-edges"* between neighbouring nodes are shown as *thicker*, *dashed* and coloured *red*. Experimenting with different *distinctiveness* thresholds can help users identify potential outliers and nested clusters. Figure 4.6 III demonstrates the identification of a distinct nested cluster (shown in *black*) by moving the bottom slider, which controls the *distinctiveness* threshold. In the first prototype, possible outliers and nested clusters found using the distinctiveness slider were shown in *black*. However, this encoding changed in the final prototype because using the same colour was confusing when multiple *"weak-edges"* were found close to each other in the dendrogram. Therefore it was decided to retain the colour encoding that characterises the parent cluster and only show the *"weak-edges"* using thick, dashed, red lines.

4.3.4 Coordinated Views

The first prototype supported the exploration of large dendrograms but it did not support requirement R2. The original data records were only represented in an abstract way as the leaves of the dendrogram while their values were shown in a static tabular format within the tool. Common algorithms for hierarchical clustering ignore special characteristics of the underlying data records in the data sets. Usually, hierarchical clustering algorithms only use the distance matrix to merge or divide clusters and form the dendrogram, which may lead to some wrong merging/dividing decisions. If the wrong combination of distance metric/measure, hierarchical clustering algorithm and its type is used, important information could be lost and the clustering results could be even misleading. Clustering results cannot be evaluated simply by looking at the dendrogram. Therefore, the computational biologists also asked for an effective representation of the original multidimensional data, in coordination with the dendrogram (R2 and R3). It was important to enable users interact with the dendrogram while seeing the effect their choices on the original data.

We were looking for an intuitive way to represent multidimensional data and time-series in particular, which were very common in the data sets of our collaborators. As part of this step, we showed our collaborators different examples of visualisation techniques (discussed in Section 3.1.3) and we asked them to review them providing anecdotal feedback. In this way, different techniques were informally evaluated in matters of their relevance to users' data sets and tasks. The techniques that our collaborators found the most useful for representing their data (*i.e heatmaps* and *parallel coordinates*) were explained in more detail. Further discussions with the computational biologists led to the clarification of operations and interactivity the interface needed for supporting the requirements (fourth level of the nested model). Although our collaborators found the *heatmap* useful for representing multidimensional data, they did not find it intuitive when the data set consisted of time-series. On the other hand, they found *parallel coordinates* more intuitive for representing time-series and also suitable for satisfying requirements R2 and R4, as they found it easier to compare variables represented as lines in the parallel coordinates, rather than squares of different opacity in the heatmap.

In our effort to satisfy requirements R2 and R3, we also adopted the approach of multiple coor-

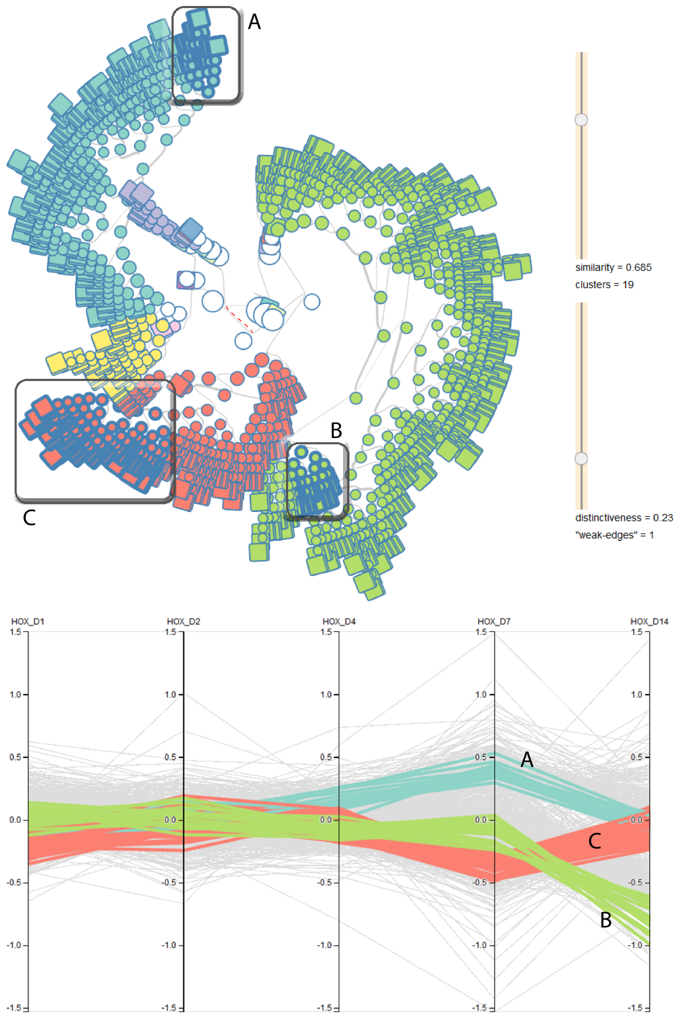


Figure 4.7: Three sub-clusters of genes (A, B and C) that exhibit distinctive time patterns. Each sub-cluster belongs to a larger main cluster, visually encoded using colour.

dinated views in the design of MLCut. The user interface is composed of two separate but linked components (Figure 4.7). The top view is a radial representation of the dendrogram, while the bottom shows the original multidimensional data as *parallel coordinates* [93]. Each data record, represented as a rectangle in the dendrogram (*i.e. top*) view, is linked to a line in the parallel coordinates (*i.e. bottom*) view. Data records are normalised, and every axis in the parallel coordinates is scaled to the same minimum and maximum values to enable comparisons. To support R3 and R4, the user can explore clustering assignments using the two sliders and also interact with the branches of the dendrogram to explore potential multi-level cuts, shown in the parallel coordinates view. The interaction is done hovering over the circles of intermediate branch nodes. This gives a real-time preview of the effect the branch-cut would have in the original data. Clicking on a circle, selects the whole branch, including its leaves. Previewing and selecting branches can be done interactively at any level of detail: from the whole tree down to a single leaf. This flexibility enables the exploration of potential sub-clusters within the main clusters identified using the sliders. Selected branches are highlighted with thicker borders at the top view and with thicker lines at the bottom, as shown in Figure 4.7.

To reduce the visual clutter at the bottom view, variables that are not selected are shown in light grey. The top view supports most of the interactivity, and the colour encoding is preserved to enable the further exploration of the dendrogram. Finally, to support R5, each selected cluster or sub-cluster can be exported as a comma separated values (CSV) file by double-clicking on any of the lines of the bottom view.

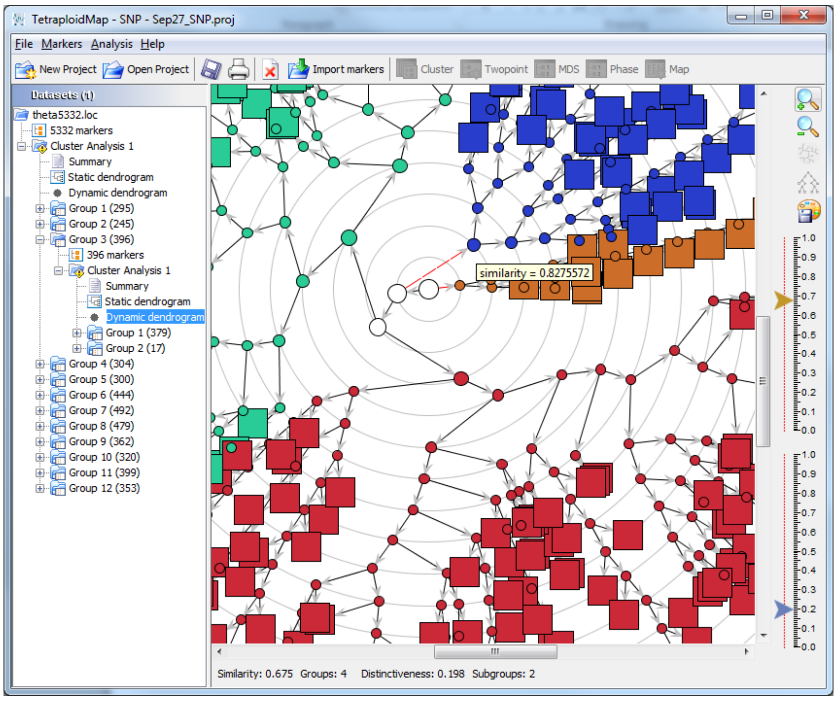
4.3.5 Releases

The development of MLCut took place in two main phases that led to the implementation of two software prototypes. The first prototype was incorporated into a larger software package used for the analysis of genetic data in tetraploid populations, called *TetraploidSNPMap* [82] (Figure 4.8). The second prototype is the most mature version that supports all features of MLCut [180] (Figure 4.7) and it can be used as a standalone visualisation tool.

The first prototype was developed in Java. It does not include an implementation of coordinated views between the dendrogram and the original data but it supports a top-down tree layout for the dendrogram (Figure 4.3). The second and final prototype was written in JavaScript using *D3* [34] and it can be accessed online through a web browser. The source code can be found in a repository as described in Appendix A. The implementation lacks the top-down tree layout representation for the dendrogram, but it includes the implementation of the coordinated views that show the dendrogram and the original multidimensional data as parallel coordinates. The preprocessing and hierarchical clustering analysis of the original multidimensional data was done in R using the TSclust package [131].

4.4 Evaluation

Usability testing was done to validate the design of MLCut during and after the development of the second software prototype. We followed the method of a design study [159], receiving informal feedback from three biological domain experts who used MLCut to visualise and cluster their data. A senior statistician tested the usability of the two dynamic sliders and the dendrogram design, confirming previous clustering results. The first prototype became part of a larger software package, TetraploidSNPMap [82]. This package, not only supports hierarchical clustering but also provides tools for genetic linkage analysis. The usability testing of the first prototype was done using the default data set of DNA markers supplied in the release of TetraploidSNPMap. The second (and final) prototype of MLCut was evaluated by two computational biologists who were interested in clustering time-series gene expression data as part of a research project. The testing



Similarity: 0.675 Groups: 4 Distinctiveness: 0.198 Subgroups: 2

Figure 4.8: A screenshot from TetraploidSNPMap showing MLCut as an integrated clustering component [82].

of the second prototype also confirmed previous clustering results. In the following, we present the application of both MLCut prototypes for clustering real biological data.

4.4.1 Clustering SNPs to Chromosomes

Modern sequencing technologies enable thousands of single nucleotide polymorphisms (SNPs) to be measured in genetic mapping populations. The first step in genetic linkage analysis is to cluster the SNPs into separate chromosomal groups so that SNPs in different groups are inherited independently.

The first prototype of this tool (part of TetraploidSNPMap shown in Figure 4.8) was used to partition over 5000 SNPs measured on 190 offspring in a cross between two tetraploid potato lines. The distance metric between each pair of SNPs was estimated from the significance of a χ^2 test for independence [118], and average-linkage was used as the type of the hierarchical clustering method.

The user found that the main clusters agree well with position information from the sequenced potato genome. Detail within the clusters shows SNPs located on the different homologous chromosomes within each linkage groups. Full details of the genetic mapping are given in Hackett *et al.* [81].

4.4.2 Clustering Time-series Gene Expression Data

In the context of gene expression analysis, hierarchical clustering algorithms are used for data partitioning and variable selection. When analysing gene expression data to infer network models, a subset of variables (from thousands) is selected for inclusion in the network [122]. The clusters or representative single genes selected, become variables represented as nodes in the network model. Towards this analysis goal, the second prototype of MLCut was developed in collaboration with a small group of two computational biologists for clustering short time-series gene expression data. For such data sets, each of the clusters corresponds to a characteristic profile or temporal pattern [185].

A real usage scenario took place in which a gene expression data set with short time-series was explored [179]. The data set consisted of the fold change of 800 differentially expressed genes in five time-points and it is publicly available in the *Gene Expression Omnibus (GEO)* [64] repository with accession number *GSE49577* [106].

Initially, different distance measures were used for calculating pairwise dissimilarities between time-series such as: Euclidean distance, autocorrelation coefficient and dynamic time warping. Also, different agglomerative hierarchical clustering algorithms have been tested using the *TSclust* [131] package in R. The combination of Euclidean distance with an average-linkage hierarchical clustering algorithm was selected as the best option for the task.

Using the second and final version of MLCut, the users managed to find three distinct temporal profiles of late gene expression (Figure 4.7). This was achieved by interactively exploring the branches for potential sub-clusters, and eventually by cutting the dendrogram in multiple levels. The difference in gene expression patterns occurs between the third and the fifth parallel coordinates, which correspond to time-points. Gene expression in the cluster shown in Figure 4.7*A* first increases and then decreases, while in Figure 4.7*C* the opposite happens (first decreases and then increases). Gene expression in the cluster shown in Figure 4.7*B* remains stable between the third and fourth time-points and decreases after that. These patterns not only agree well with clusters related to late gene expression as reported in Koussounadis *et al.* [106], but also provide a more clear cluster assignment scenario.

The case study demonstrated the benefits of MLCut in practice and helped to test and refine the implementation of the prototype. Finally, anecdotal feedback was given through emails and also verbally during our discussions with the users. One of the users wrote: *"I cannot seem to download the gene list by clicking on the genes. It is still working with the HOX data set but for some reason, it will not let me in the OV data sets. Thank you for all of your help so far and other than this the tools are excellent!"*. Another user wrote: *"I really like how the tool lets you see both the expression lines and clusters, and how this changes as you change the clustering. I can really see the applications for being about to choose sub-clusters based on visual match rather than having to blindly slice the tree at one level only, and am looking forward to seeing what we can discover using this method of clustering the data"*.

4.5 Summary of Contribution

To answer Q1, we developed MLCut, an interactive visualisation tool that enables analysts to select clusters manually by applying multi-level cuts on demand. There are two types of thresholds: a *global* single-height similarity threshold that applies to the whole dendrogram and *local* distinctiveness threshold that applies between pairs of two linked nodes, enabling a more finely-grained exploration. In addition, the interactive exploration of the dendrogram is coordinated with a representation of the original data, shown as *parallel coordinates*. The analysis process involves three steps. First, a single-height similarity threshold can be applied using a dynamic slider to identify the main clusters. Second, a distinctiveness threshold can be applied using a second dynamic slider to identify long edges (*i.e.*"weak-edges"), that indicate heterogeneity within clusters. Third, the user can interact with the dendrogram and the original data to manually cut the branches of the tree at multiple levels. This step is important for detecting nested clusters and outliers. Interactive drilling-down is supported using mouse events such as hovering, pointing and clicking on elements of the dendrogram.

The design of MLCut followed a synergistic approach that combines the strengths of hierarchical clustering algorithms with the ability of humans to visually detect patterns and anomalies in the data. It was developed in close collaboration with analysts that had experience in using hierarchical clustering as part of their workflow. The tool was evaluated while being used for allocating single nucleotide polymorphisms (SNPs) to chromosomes of tetraploid species and for finding temporal patterns in time-series gene expression data. Anecdotal feedback from the analysts suggested that MLCut is a promising method for clustering which could lead to scientific discoveries. In the following chapter, we discuss how selected variables or clusters of variables become nodes in the inferred network. As part of the second step in modellers' workflow (Figure 1.1 (b)), we address the challenge of learning the interactions between pairs of nodes that constitute the structure of the network.

5 Bayesian Network Inference with BayesPiles

Clustering is an important step for reducing the number of variables and for selecting those that become nodes in the inferred network model. However, modellers still need to understand how all these variables interact with each other to form the structure of the network. The step of inferring the structure of the network is the most challenging in modellers' workflow (Figure 1.1 (b)), because there is a lot of uncertainty about the state of the natural system, resulting in many possible network solutions for a given data set. Therefore, modellers apply heuristic search algorithms that sample the solution space of all possible networks, based on algorithm parameters and a network score that encodes the statistical fit to the data. Then, modellers combine the heuristic search results using their own tacit knowledge to take decisions about the final structure of the model [105, 117]. In this chapter, we focus on answering the second research question: **Q2** ***"How to support the visual analysis of heuristic search results, to infer representative models for biological systems?"*** To investigate how visualisation could help to address this question, we applied the nested model methodology [132], which led to the following questions:

Q2a How can visualisation help modellers understand the shape of the solution space in order to guide the heuristic search in finding better network solutions?

Q2b How can visualisation help modellers decide on a method that determines the structure of a final consensus network?

This chapter presents BayesPiles, a novel interactive visual analytics tool which was developed to support computational biologists (i.e. modellers) in exploring, comparing and combining Bayesian networks (BNs) interactively. BayesPiles builds upon the design of an already existing tool for analysing dynamic networks visually, called MultiPiles. In Section 5.1, we describe how the the first level of the nested model was applied to characterise the domain problem. In Section 5.2, as part of the second level of the nested model, we present a list of common tasks related to Bayesian network inference. In Section 5.3, as part of the third and fourth levels of the nested model, we describe the design of BayesPiles and how it extends MultiPiles to support the identified tasks. In Section 5.4, we present an evaluation of our tool with domain experts, who used BayesPiles as part of their network inference workflow in three case studies that involved real biological data sets. Finally, at the end of the chapter (Section 5.5), we present a summary of the contribution and we discuss findings. The paper for BayesPiles was published in the *ACM Transactions on Intelligent Systems and Technology (TIST)* journal [181].

5.1 Domain Problem Characterisation

Following the design study methodology [159], we applied the first level of the nested model (described in Section 2.4) to understand the problems computational biologists often face when they infer Bayesian networks from biological data. Inspired by other successful design studies [127, 126], we collaborated closely with three domain experts (*i.e.* computational biologists) who wanted to infer Bayesian networks. Our first collaborator was an experienced academic researcher, specialised in computational biology using Bayesian methods. The second was a PhD candidate in computational biology who used Bayesian networks as part of a research project. Our third collaborator was a student in neuroscience who mostly provided feedback at a later stage when BayesPiles was used for analysing neuroimaging data. Two more research students provided feedback when BayesPiles was extended to support dynamic Bayesian networks (Chapter 6).

As part of the first two levels of the nested model, our priority was to understand the domain problem and abstract the data and the main biological analysis tasks. Over a period of two months, we held four one-hour informal meetings with the most experienced of our collaborators and two one-hour meetings with the PhD student (six meetings in total). During the first month we met only with the most experienced of our collaborators. The second (less experienced) computational biologist provided feedback during the second month to confirm and refine the characteristics of the domain problem.

During these meetings our collaborators described the process they followed to infer networks and the tools they used to explore, combine and compare them to construct a final consensus network. We conducted a literature review to better understand the domain problems and extend our background knowledge on the field of Bayesian network inference. Through our literature review (Section 3.2) and anecdotal feedback from our collaborators, we understood that heuristic search algorithms, such as *greedy* search and *simulated annealing*, are used for finding and scoring hundreds of possible network solutions. Then, modellers have to explore, compare and combine results from different runs to infer a final consensus network. However, this task is difficult because heuristic search algorithms run multiple times and produce potentially hundreds of networks which are hard to explore and compare without visualisation support.

The main task for the computational biologists was to guide the heuristic search and decide on a method that determines the structure of a final consensus network, usually by selecting the top-scoring network or constructing the consensus network from a collection of high-scoring networks.

A "good" network has to explain the observations—a process that requires the combination of both data and expert knowledge, which in turn require an understanding of the shape of the solution space and the comparison of potentially hundreds of individual network *solutions*. As a system can give scores to networks, we use the term *solution space* to refer to the set of highest scoring networks as generated and assessed by the system. On a general level, the human task is twofold: *i)* answer Q2a assessing the individual quality of network solutions (with respect to their scores) and *ii)* answer Q2b creating consensus networks by combining individual solutions.

5.2 Tasks

We often had to contact our analysts for clarifications, but we gradually managed to refine our understanding and to abstract data and tasks (second level of the nested model). Then, the biological analysis tasks were mapped into visualisation tasks suitable to start designing a system. Based on the previous observations, our literature review, as well as semi-structured interviews with our collaborators, we summarised the following tasks as the most crucial ones related to the understanding of Bayesian networks for modelling biological interactions. The first three are mostly concerned with answering Q2a and the rest with Q2b.

T1 - Overview large sets of networks: overview and explore topologies of hundreds of directed networks together with the distribution of their respective scores. Modellers were interested in observing the solution space, consisting of network structures and their scores.

T2 - Compare different runs: display and order multiple collections of networks produced in different runs. Heuristic search algorithms are executed multiple times and their results often vary. Modellers were interested in sorting and comparing results from different runs.

T3 - Group networks: organise and combine networks into groups and enable comparisons within and across groups. Modellers were interested in summarising results by creating network aggregates and then compare them to identify common and different edges.

T4 - Filter nodes and edges based on user-defined criteria, such as connectivity of nodes and weight of edges. Modellers wanted to identify nodes and edges that could be removed from the final consensus network.

T5 - Summarise and check the consistency of outgoing edges for selected nodes across multiple networks. This is important for exploring networks at a node level. Modellers wanted to see if nodes with the same degree would also maintain the same edges across multiple networks.

T6 - Determine consensus network: explore and construct a possible consensus network. Modellers were interested in a flexible method that would enable them to combine multiple networks to determine the final consensus network.

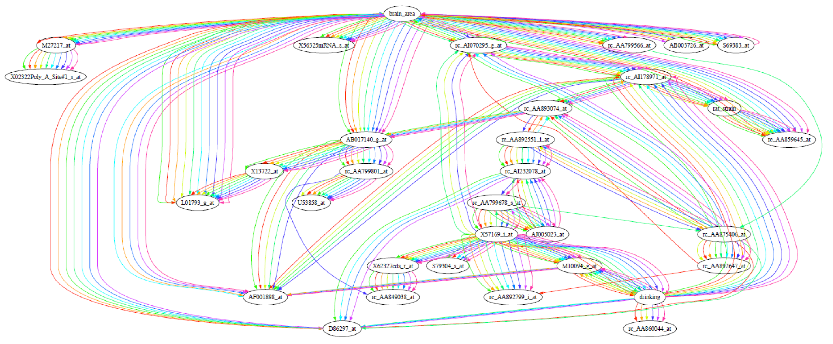
These tasks have implications for interface design; scalability to many networks, visualise differences between network topologies, show network scores, show link directions, support manual creation of the consensus network, etc. In the following Section 5.3 we describe in detail how the design of BayesPiles supports each of the identified tasks. In Section 5.4, we present an evaluation based on three case studies in which real biological data sets were analysed using BayesPiles. The chapter finishes with a summary of the contribution.

5.3 Design and Implementation

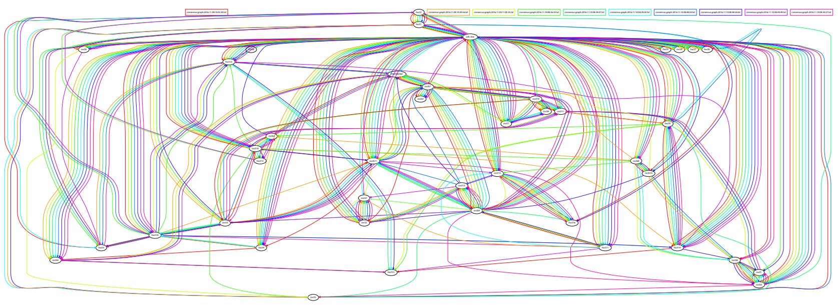
As part of the third level of the nested model, after we identified the analysis tasks, we sought a visualisation idiom which could be successfully applied to the data to support the tasks. As described in Section 3.2.2, we conducted a literature review to identify relevant visualisation tools and techniques that could be potentially extended to support our analysis tasks for inferring network models. We found that Bayesian networks (BNs) are usually represented as node-link diagrams using a hierarchical or force-directed graph layout. The example in Figure 5.1 (a) shows a set of 10 superimposed BNs of a similar structure produced with the BANJO system and visualised using the Graphviz visualisation library [65]. Superimposing larger collections of BNs, or BNs that have many structural differences, results in a dense network (Figure 5.1 (b)) because the superimposed network includes all edges and because adding different networks in the collection increases the overall number of edges. Link colour represents the links in each network and the layout is optimised to minimise edge crossings. The visual encoding is limited in several ways: it seems impossible to overlay more than ten networks (resulting in too many lines and hardly discernible colours) and many of the lines are overlapping. Aggregating adjacent edges (*i.e.* edge bundling) in directed networks can only partly alleviate the problem because it makes it hard to discern between target and source nodes, while edge crossings can still be a problem. Moreover, while node-link diagrams are effective for path following tasks, our users are mostly interested in tasks related to degree and adjacency [73].

As a scalable alternative to node-link diagrams, adjacency matrices have been proven effective in visualising dense networks [73]. Adjacency matrices represent networks in table format; each node corresponds to a row and to a column. Whenever two nodes are related (linked), the respective matrix cell is filled by some visual mark. Given an appropriate ordering of rows and columns, matrices show topological network patterns such as clusters [26]. Ghoniem and Fekete [73] found that matrix representations perform better than node-link for many network visualisation tasks such as spotting clusters, finding highly connected nodes, finding common neighbours, as well as comparing weights on links [9]. Consequently, matrices have been used to visualise and compare the architecture of software systems [3, 23], brain connectivity [9, 17, 18], or ontologies [16]. Alper *et al.* [9] demonstrated that matrices are more readable in comparing two networks than node-link diagrams such as shown in Figure 5.1.

While looking for an appropriate visualisation approach to support our tasks, we discussed existing visualisation tools and techniques with our biology collaborators. To receive initial feed-



(a)



(b)

Figure 5.1: (a) A "rainbow" consensus network of 10 networks superimposed, shown in Graphviz [65]. This is the most dense network analysts can currently handle. (b) A denser consensus network which is almost impossible to read.

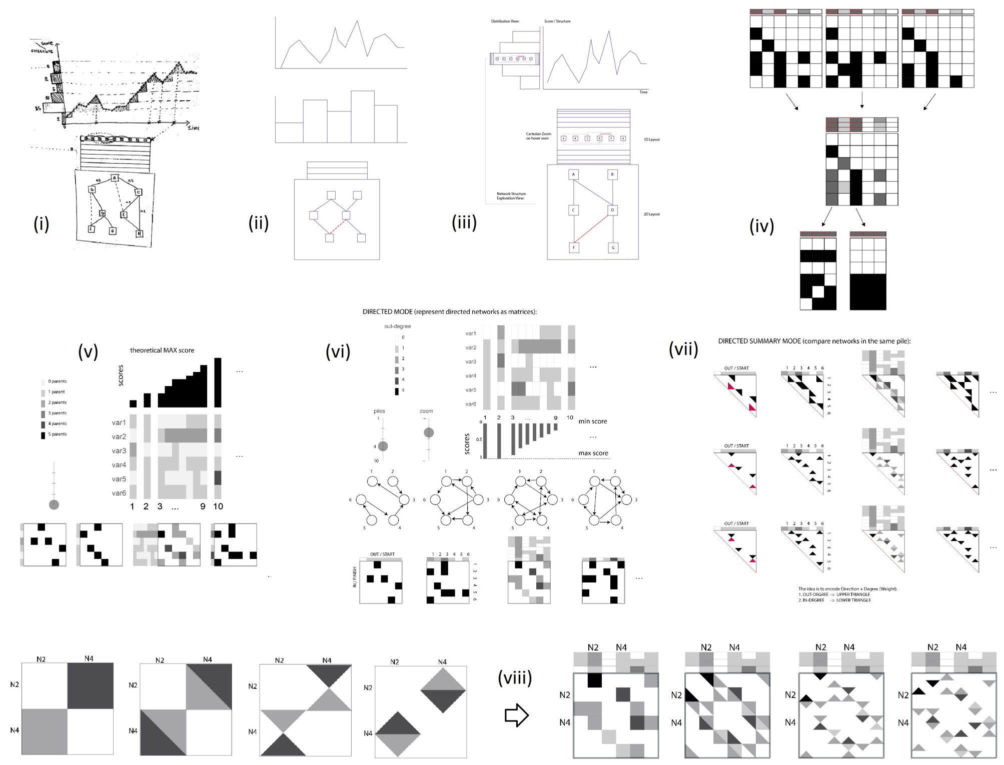


Figure 5.2: Some of the sketches drawn and discussed during our meetings with the computational biologists. The first six sketches show unrefined encoding ideas of the solution space and the last three sketches are concerned with encoding directed networks in matrices.

back regarding design options, we held two one-hour meetings with two of our collaborators. In these meetings we discussed different visualisation tools and techniques found in the literature (Section 3.2.2). They were mostly interested in tools that could represent multiple networks, such as Cerebral [21], eXamine [58], Caleydo [113] and MultiPiles [18]. Their feedback was particularly positive when matrix-based approaches were discussed; *"[t]he matrix representation allows for a lot better overview of the data set, as looking at such a large detail in the traditional network representation is too confusing when there are so many nodes and edges. It is quickly obvious in this format which nodes have many edges attached to them and which are connected to very few other nodes."* Our literature review, evaluation studies [9, 73] and feedback from our collaborators indicated that matrix-based representations were more promising compared to node-link diagrams because of their visual scalability and potential for information design. After we discussed different visualisation tools and techniques with our biology collaborators we drawn sketches (first row in Figure 5.2) and we identified MultiPiles (Figure 5.3) as the most relevant matrix-based visualisation technique for the respective type of data (large collections of scored Bayesian networks) and our tasks.

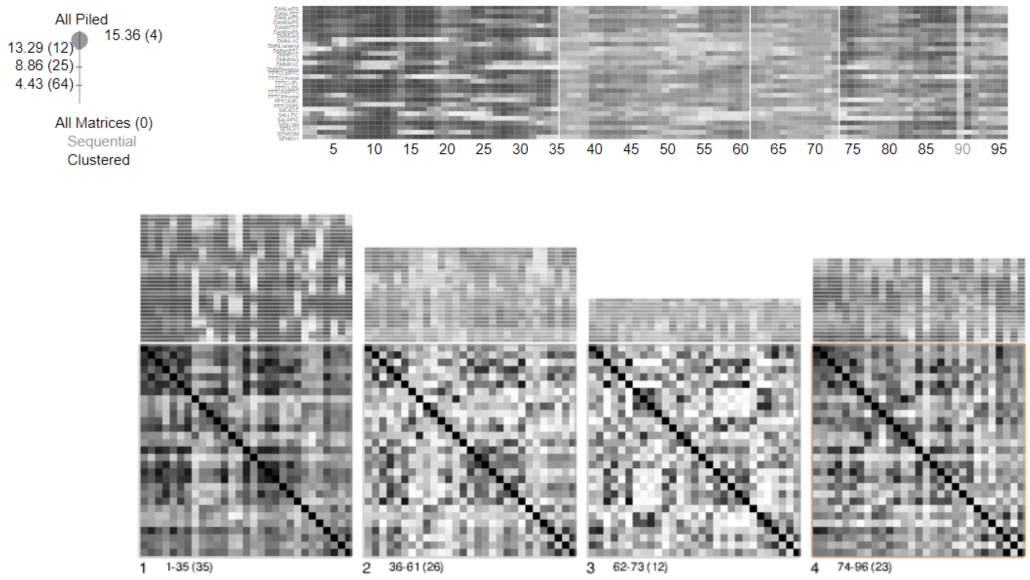


Figure 5.3: A screenshot of MultiPiles [18] showing four piles of networks represented as adjacency matrices. The interface was extended in BayesPiles to provide visualisation support for tasks related to Bayesian network inference.

MultiPiles [18], was designed for the exploration of dynamic networks where each time step in the dynamic network is represented as a thumbnail-size matrix. Matrices in MultiPiles can be juxtaposed or superimposed to visually form piles, thus scaling to many matrices (networks). Piles are visually summarised by showing the weighted mean of all edges for all networks in the pile. Users can interactively create, refine, and explore the contents of each pile through simple drag-and-drop and hover interactions. Feedback from our collaborators on MultiPiles was again very positive, stating that *"[t]he matrices were a much more concise representation of networks than [what] we had been using, and particularly the ability to 'pile' them up and see both a summary of multiple networks and the variation in the individual networks was far better than our previous 'rainbow' output."*

Although MultiPiles provided a promising technique for visualising hundreds of networks, we had to alter and extend its design to support the tasks. Inspired by the design of MultiPiles and as part of the third and fourth levels of the nested model, we developed BayesPiles to help modellers explore, combine and compare multiple Bayesian networks to infer a final consensus network. During the development of BayesPiles we followed a user-centred design approach with iterative development of sketches (Figure 5.2) and prototypes. We took design decisions not only based on known design principles for presenting relational information [119, 47, 28], but also considering anecdotal feedback from our collaborators who were testing our software prototypes to identify weaknesses in the visual encoding and the supported interactive operations. BayesPiles is a web-based tool implemented in JavaScript using the D3 [34] and WebGL libraries for graphics. The source code can be found in an online repository (Appendix A). Figure 5.4 shows the interface

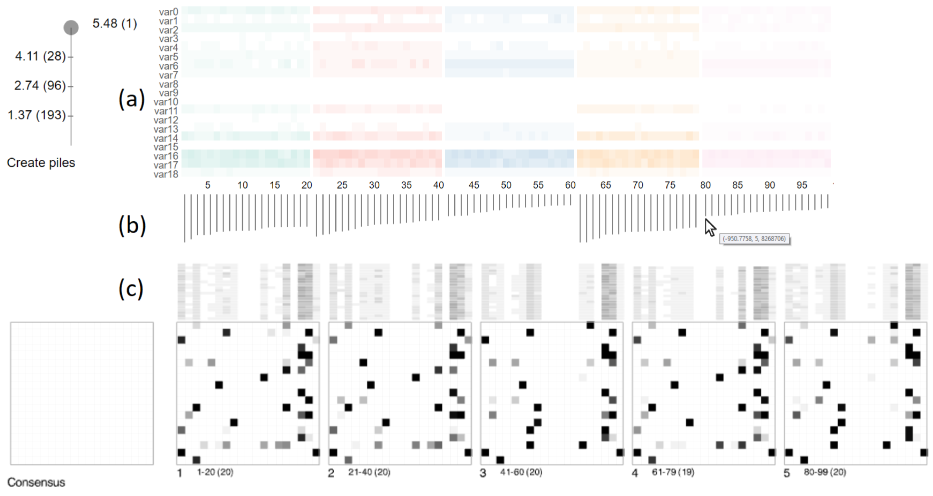


Figure 5.4: The two linked views of BayesPiles. (a) Overview of 99 networks produced in five runs and shown as summary columns. Different colours indicate different runs. (b) A histogram with the distribution of scores. By hovering over each bar, details such as the computed score value, the run ID and the iteration appear as a tooltip. (c) Initially, the consensus pile is empty. Piles 1-5 contain networks from the five different runs and shown using the top-down mode. Opacity encodes the weight of each edge (cell) in piles of superimposed networks. Opacity is also used to summarise the out-degrees which except the overview also appear at the top edge of each pile.

of BayesPiles with the following views: (a) a heat-map summary view for each network (column) and their node degree (row), (b) bar charts visualising each network's assessed score, and (c) a detail view of networks grouped in piles, alongside with a placeholder (empty matrix) for the user-created consensus network. The following details how BayesPiles inspired by the interface of MultiPiles satisfies each of the tasks. In order to support each of these tasks, changes needed to be made in the design and the implementation of MultiPiles. Each of the following subsections from 5.3.1 to 5.3.6 describes in more detail the changes to MultiPiles required for supporting the tasks.

5.3.1 Exploring Hundreds of Scored Directed Networks to Support T1

To support task T1 and enable the exploration of hundreds of scored directed networks, the design of MultiPiles had to be updated. In MultiPiles, networks have to be in a *fixed order* (time), and piles can only be created on adjacent networks. Moreover, a second limitation is that MultiPiles does not visualise edge direction but assumes every network is symmetric. Although adjacency matrices could represent directed graphs [183], there are no clear guidelines of how this can be done effectively. Thus, to support directed edges, we were inspired by other approaches that represent directionality in matrices [114], such as OntoTrix [16], to adopt a top-down matrix representation. Rows indicate incoming edges and columns indicate outgoing edges, resulting in an adjacency matrix which is not symmetric. Figure 5.5 demonstrates how the node-link diagram

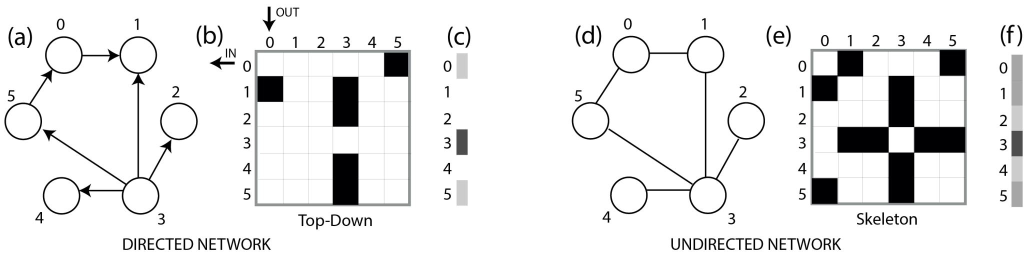


Figure 5.5: Directed versus undirected node-link and matrix representations. (a) Node-link representation of a directed network. (b) The same directed network as encoded in top-down mode. Rows encode incoming edges and columns outgoing edges resulting in an adjacency matrix that is not symmetric. (c) The out-degree of each node as encoded using opacity in a summary column. (d) Node-link representation of an undirected network. (e) The undirected network shown in skeleton-mode resulting in a symmetric adjacency matrix (MultiPiles visualisation method). (f) The degree of each node as encoded using opacity in a summary column.

in (a) is encoded as a top-down directed adjacency matrix in (b). In Figure 5.5 (b), column 3 shows the four outgoing edges from node 3, while row 3 shows that there are no incoming edges to node 3. In addition to the top-down mode, the design of BayesPiles can also support a skeleton mode which ignores the direction of the edges and shows an undirected adjacency matrix similar to MultiPiles (Figure 5.5 (e)).

The technique of MultiPiles is scalable to hundreds of networks because it enables networks to be piled using superimposition and because it provides a single-column summary overview. When multiple networks are superimposed, opacity encodes the edge weights in the resulting network. These weights provide an indication of how frequently the edge appears in networks present in the pile. Moreover, in BayesPiles, opacity is used in the column summary of the directed network (Figure 5.5 (c)) that encodes the out-degree of each node. In skeleton mode, opacity encodes node degree, as the edges are undirected (Figure 5.5 (f)).

A third limitation of MultiPiles is that it does not visualise any data specific to networks, such as a numeric quality score in our case. We hence added value bars at the bottom of each summary column. This encoding is a top-down histogram of network scores which provides an overview of their distribution in the solution space (Figure 5.4 (b)). In order to allow for comparisons between runs, we normalise the bar lengths between the highest and the lowest network score.

5.3.2 Importing and Ordering Multiple Network Collections to Support T2

In order to support T2, networks can be sorted based on their run ID, iteration, or score. Different network orderings support the exploration of the search results and support the identification of trends in the data such as sudden changes in score values. When networks are ordered by score, it gives an impression of the overall shape of the solution space. For instance, a dramatic rise in scores indicates a high-scoring network (hilltop) found in an otherwise flat solution space. These

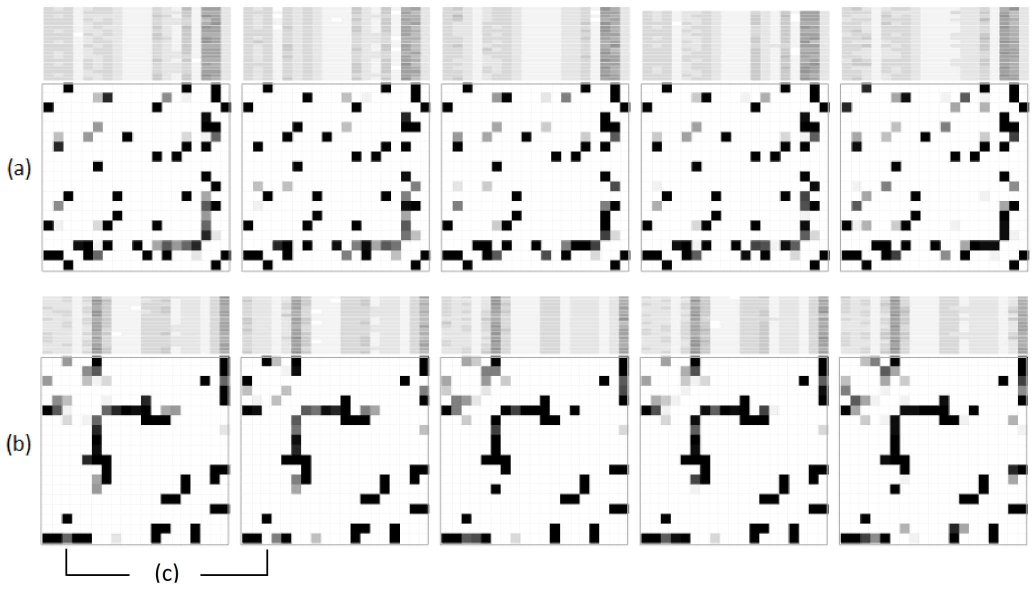


Figure 5.6: Node reordering improves network comparison and pattern recognition in matrices. (a) Five piles in skeleton mode before applying node reordering. (b) The same five piles after node reordering. (c) It is easier to spot differences such as an edge which is only missing in the second pile from the left (in last row and third column).

variations in score indicate which networks to include in a consensus network and which to omit. The actual score of the network, calculated using the BDe metric [87], is visible by hovering over its bar as shown in Figure 5.4 (b) for network 80. When networks are sorted by iteration, the analyst can find periods when the algorithm made rapid progress towards finding high-scoring networks and periods when the algorithm was stuck in low scoring areas of the search space. This information is important for guiding future heuristic searches and helps users in tuning parameters for improving search results.

Apart from changing the order of the sampled networks, users can also reorder the nodes within the matrices (Figure 5.6). In general, node reordering is important for effective matrix-based representations of networks and is similar to graph layouts in node-link diagrams [26]. In BayesPiles, node reordering is applied globally to the entire data with all matrices taking the same order. In particular, the *optimal leaf ordering* algorithm [20] is applied which uses hierarchical clustering to place the most similar rows across all matrices close to each other. Then the ordering of the rows is also applied to the columns. The similarity matrix for each network is calculated using the Manhattan distance. Node reordering enables comparisons between matrices and piles and makes it easier for the users to see differences between collections of networks. It can reveal interesting patterns in the data [114, 26] (Figure 5.6 (b)). At any point, the user can switch back to the original node ordering as provided by the data set (Figure 5.6 (a)).

BayesPiles also supports the visualisation of multiple collections of BNs, generated by different runs of the heuristic search algorithm. Comparing runs is useful for determining the shape of the solution space more reliably. For instance, if networks appear to approach the same single optimum solution across multiple runs, the analyst would have confidence that the top-scoring network is reproducible and representative of the data since no other variation is observed in the solution space. When this is the case, the gradual improvement in score is coinciding with the addition of an edge until no more improvement in score is observed. We refer to this pattern as the *hill-climbing pattern*. However, if multiple runs produce high-scoring networks of different structure, the user would need to construct a consensus network.

Comparing high-scoring networks found in different search attempts is important for discovering variation between runs. In order to facilitate comparison, scores are globally normalised and colour used to encode the results of up to 12 different algorithm runs. The palette of colours was generated using ColorBrewer [84]. Results from five runs are shown in Figure 5.4 (a). For a larger number of runs, colours are repeated, but analysts rarely explore the results from more than ten runs simultaneously in our experience. By examining the scores (Figure 5.4 (b)), it becomes clear that only a few runs found substantially high-scoring networks (runs 1, 2 and 4), while the other runs (runs 3 and 5) only found local optima in the search space. This variation in network score between runs indicates a complex solution space with multiple optima for which the construction of a consensus network is required.

5.3.3 Group and Compare Networks to Support T3

Our analysts were interested in exploring and comparing many BNs and found grouping BNs together was important. They were interested in grouping networks of a similar score to understand if they belong to the same equivalence class. Networks of the same equivalence class are mathematically equivalent representations, differing in only the directionality of some (or all) edges. Networks of the same equivalence class have the same score. If too many of them are added to the consensus, then it could be a biased result based on the size of the equivalence class rather than the probability of the solution. The histogram can help identify networks of similar scores, but it cannot be used alone to check if the networks belong to the same equivalence class. When there are multiple hills in the data set, networks of the same score may have a very different structure. Therefore, in order to find equivalence classes, it is important to inspect and compare edges and their directionality within groups of networks that have the same score.

The user needs to group BNs of the same score and then needs to identify networks of the same equivalence class within that group. In BayesPiles, the top-down histogram sorted by score can help analysts identify networks of a similar score to pile (i.e. group) them together. Piling networks could be automatic by using the dynamic slider provided by MultiPiles, or by manually clicking on the summary view (Figure 5.4 (a)). White vertical lines indicate separate piles. Automatic piling based on score difference is sometimes possible, but identifying a distinctive drop in score is often a question of individual judgement, especially when the solution space has multiple hills. Skeleton mode can help to identify networks that belong to the same equivalence class as these networks have nodes with the same degree. Therefore, their column summaries look the same.

The cover matrix of each pile summarises the networks it contains and enables the interactive comparison of edges within these piles by hovering over their summaries. Edges that exist in all networks appear black in the cover matrix, while lighter shades of grey indicate a lower frequency of the edge (Figure 5.4 (c)). In skeleton mode, edges in the cover matrix that belong to the same equivalence class will appear very dark. If many equivalence classes appear in the results, it could mean that the directionality of the edges is not informative. In these cases, it would be preferable to represent these networks ignoring directionality by using skeleton mode.

In the top-down mode, the two directions of an edge are on opposite sides of the diagonal of the matrix, making it difficult to compare opacity levels. Inspecting only one of the edges does not reveal much information about the opacity of the edge that points to the opposite direction because opacities depend on the frequency of the edges in the pile and not on the number of edges that point in the opposite direction.

In order to further satisfy task T3 and enable comparison of edge direction, we developed a third matrix mode, diamond mode (Figure 5.7). This design is inspired by other approaches [192, 16, 9] for representing directionality in adjacency matrices, and it was developed as an evolution of the top-down mode. Depending on the task, analysts can easily switch between the three network representations.

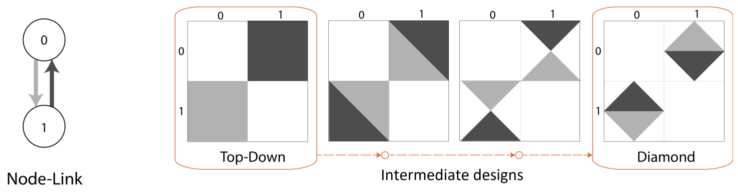


Figure 5.7: The evolution of the design for comparing edges of opposite directions. Intermediate design options created ambiguities and visual artefacts because of adjacent neighbouring edges. In the final design (diamond mode), only the (top and bottom) triangles that encode the opposite directions (in and out respectively) appear adjacent.

Our analysts found the diamond design easier to read, compared to alternative encodings shown in Figure 5.7. The selected encoding avoids visual artefacts which appear due to neighbouring triangles. The users also found intuitive the way directionality is encoded, as the upper and lower triangles could be easily interpreted as arrowheads pointing inwards and outwards the node indicated by each column.

Juxtaposing matrices and piles, as in MultiPiles, can help users to make these comparisons. However, this task becomes more and more difficult as the size or the number of matrices increases. The difficulty is that the user must make a judgement on the opacity of two edges which could be separated in the visualisation.

In order to support the comparison of piles or individual matrices, the user can select to display the differences between a selected matrix or pile with all other matrices or piles in the data set. Differences are illustrated using a red to blue colour scale. Missing edges or edges with lower

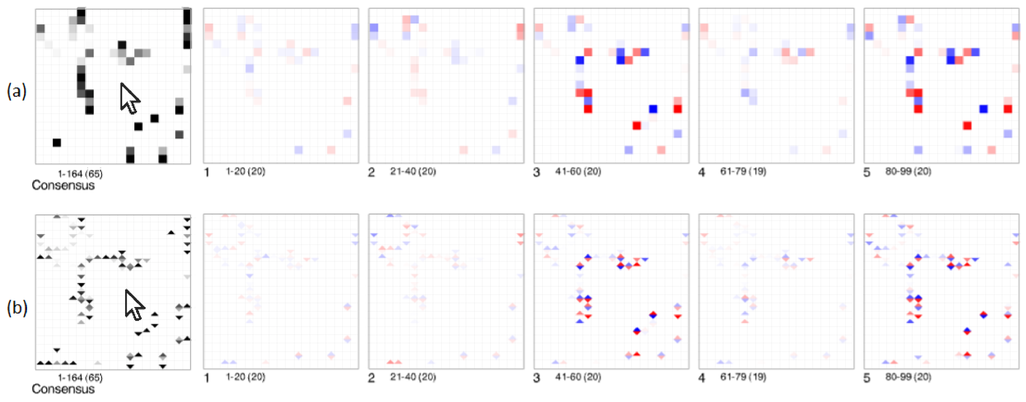


Figure 5.8: Interactively comparing piles. (a) Top-down mode highlights differences in addition and removal of edges. (b) Diamond mode highlights change in direction of edges between piles. In both modes, it becomes evident that the hovered consensus pile is more similar to piles 1, 2 and 4 and less similar to piles 3 and 5.

weight will appear as red edges, while new edges or edges with a higher weight will appear as blue edges. The opacity levels of red and blue encode the degree of difference between the network or pile (Figure 5.8).

5.3.4 Graph Filtering of Nodes and Edges to Support T4

As part of task T4, analysts wanted to focus on a small number of nodes within a network. Even though MultiPiles supports subnetwork selection by dragging the mouse over node labels, our analysts required filtering based on graph connectivity across all matrices and the ability to exclude nodes that are disconnected. Our analysts also wanted to filter edges in the cover matrix based on the percentage of networks in the pile that contain the edge.

To satisfy task T4, we introduced two dynamic sliders [6] that provide edge and node filtering capabilities. Disconnected nodes are automatically filtered out and appear in a list. Node filtering is applied globally based on the overall connectivity of nodes in all networks. Edge filtering, however, is applied locally at a pile level. For instance, when the edge filtering level is set to 50%, all edges that appear in less than half of the networks in each pile are removed from the cover matrix. Setting this slider to 100% will place only those edges that are present in all networks of the pile in the cover matrix, resulting in a logical AND operation. On the contrary, setting this slider to 0% will include all edges in the cover matrix, resulting in a logical OR operation. Depending on the circumstances experts may choose to present a denser network with weaker edges or only show the strong edges for which they are more confident about. Filtering edges provides less information about the results (i.e. content) but also increases confidence about the consistency of the reported edges. Our analysts found this feature particularly useful when deciding which edges should be included in the consensus network because this is a task that requires integration of domain knowledge. According to our users, the final decision for setting the slider depends

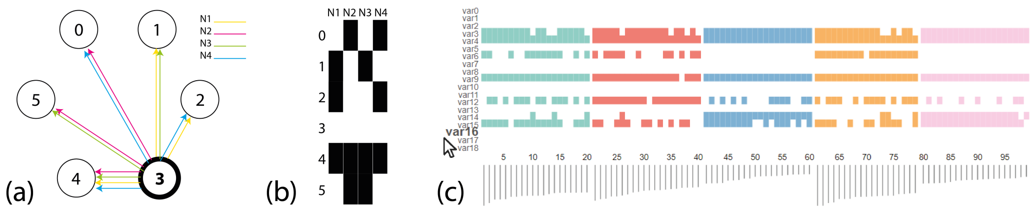


Figure 5.9: Comparing representations that show the outgoing edges from a selected node across a collection of networks. (a) Showing outgoing edges from node 3 in 4 networks using a node-link diagram in which networks are superimposed and encoded using different colours. The resulting visualisation is already hard to read. (b) For the same task, a matrix-based representation, similar to a heatmap, is much more scalable. Opaque rectangles indicate the existence of an edge in a network and blank rectangles indicate its absence. (c) Users can hover over the label of a node (here var16) and all outgoing edges will appear in the column summaries across all networks of multiple runs (here there are 99 networks in total). Interesting patterns may appear. For instance, the analyst can observe that the edge from var16 to var6 does not appear in any of the networks found by runs 3 (blue) and 5 (pink).

equally on what is shown on the display and their background knowledge about the system under study. One of our users reported: *"if I'm working with a biologist who is planning to do very expensive experiments, one per edge and could only do a handful of experiments, I'd want to give them only 100% edges if I could! Basically, the most confident I could be. Alternatively, if we're more interested in a holistic view of 'network structure' including more edges with less confidence would make sense"*.

5.3.5 Viewing Outgoing Edges of Nodes in Multiple Networks to Support T5

The overview of piled node summaries enables analysts to identify nodes with high out-degree. To better support T5, we extend this feature as analysts were interested in seeing if the same outgoing edges appear in multiple networks for a selected node, as the out-degree may be similar between networks, but the actual edges differ from network to network. Analysts were interested in stable blocks of edges as network scores change. Consistency and variation in edge appearance are important, but these patterns cannot be easily explored in collections of hundreds of networks. Thus, we modified a feature of MultiPiles, which shows all the connections of a node, to present a summary of the outgoing edges for a selected node. By hovering over a node label, all the outgoing edges from that node appear for every network and collection in the data set. An opaque rectangle indicates the existence of an edge from the selected node to another node in the data set for every network, while blank spaces indicate that no edge is present (Figure 5.9 (b) and (c)).

5.3.6 Manual Consensus Network Construction to Support T6

Probabilistic search methods, such as simulated annealing, often produce different results between runs due to the size and the complexity of the solution space. In these cases, constructing an average consensus network, which combines the results from different runs, is useful. In order to satisfy T6, BayesPiles enables interactive network construction by allowing users to add/remove networks from a consensus pile, located at the top-left side of the piles' canvas (Figure 5.16 and Figure 5.4 (c)). A single network can be added to the consensus pile by pressing the shift key and clicking on its summary (or score bar). Its score bar will turn red and a red dot will appear next to its summary in the piles canvas (bottom linked view), indicating that it has been added to the consensus network. The selected network will be copied to the consensus pile and can only be added once. Repeating the same interaction on the same network will remove it from the consensus pile. Piles of networks can be also added/removed from the consensus by clicking on a pile at the bottom view.

5.4 Evaluation

We evaluated BayesPiles in three case studies with three computational biologists visualising real biological data. The data was explored by our analysts without our presence. In the first two cases, the analysts examined data from experiments conducted and published in previous studies. The purpose of repeating previous experiments was to verify findings, detect inconsistencies, and gain new insights regarding the decision-making process followed during the analysis. This test was the first time that BayesPiles was used under real conditions. In the third case study, network inference had not been previously undertaken by the domain scientist and BayesPiles was used as part of their exploration strategy. In particular, our collaborators wanted to extend their analysis to infer networks. This case study provided an opportunity to assess how BayesPiles can improve the analyst's workflow in an ongoing experiment.

Two of our analysts were already familiar with BayesPiles as they provided feedback when developing BayesPiles. After a very short training period, they were using it with increased confidence. They performed the analysis of their own data individually and were interviewed afterwards. From these interviews, we collected anecdotal evidence about the effectiveness of our approach and its applicability to computational biology research. A third researcher used BayesPiles to analyse a fourth real data set of neuroimaging scans, guided by one of our analysts who had experience with BayesPiles. The analysis revealed a hill-climbing pattern, similar to our first case study. Below, we present three use cases from our collaborators.

5.4.1 Brain Regions on Songbird

The first data set consisted of electrophysiological recordings from the brains of female songbirds listening to auditory stimuli. Each bird had eight electrodes placed in her auditory regions. Networks were produced representing the flow of neural information among these regions [167]. In the original analysis, a greedy search was run and the single top-scoring network presented as the solution. Repeated runs of the algorithm resulted in the same top-scoring network.

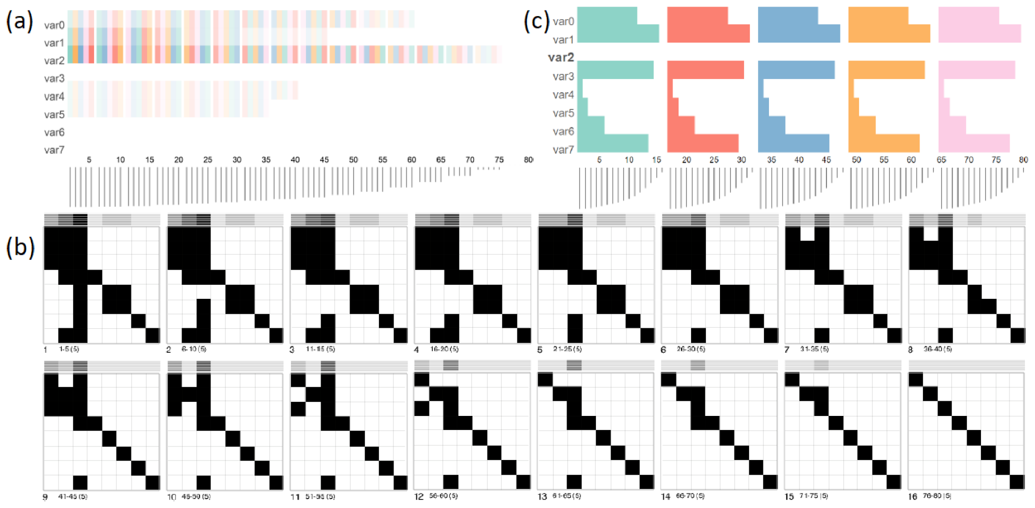


Figure 5.10: The hill-climbing pattern consistently appearing in five repetitions of the search. (a) Summary of networks when ordered by score. Networks of the same score are piled. (b) Piles shown in the top-down mode. Fully opaque edges show that all five runs produced identical results. (c) The outgoing edges for var2 look the same across all runs when networks are ordered based on their run ID. A smooth asymptotic curve appears in the histogram of their scores, indicating a hill-climbing pattern.

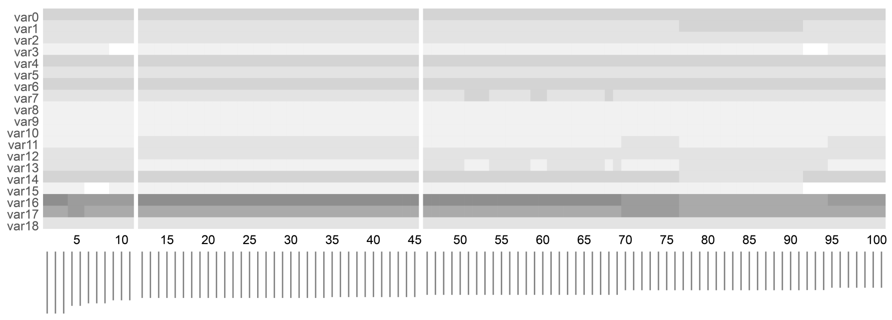
Our analyst repeated the original search on one bird's data and the networks were visualised in BayesPiles. The resulting visualisation showed a smooth, asymptotic increase across all scores, suggesting a single hill-climb. By piling all networks together and scanning the mouse down the pile, or by juxtaposing them as small multiples, the analyst could confirm the hill-climbing pattern. Our analyst made one further check, repeating the search five times and importing them all into BayesPiles. All searches revealed the same asymptotic curve in scores (Figure 5.10 (c)) and the same summary matrix representation, suggesting each search had climbed the same hill. This was confirmed by ordering the networks by score and piling identical scores (Figure 5.10 (a)). Each pile showed identical links visible via fully opaque squares in the cover matrices (Figure 5.10 (b)). This finding was also confirmed by hovering on the labels of each node showing that for each run networks had the exact same outgoing edges (Figure 5.10 (c)).

The analyst not only reproduced the results of a previous experiment but also was more confident that selecting the top-scoring network from a greedy search was the correct choice for this data set. BayesPiles provided a concrete visualisation of the search's hill-climb, enabling a decision made not solely on reproducibility but instead on a visualisation of the sampled search space.

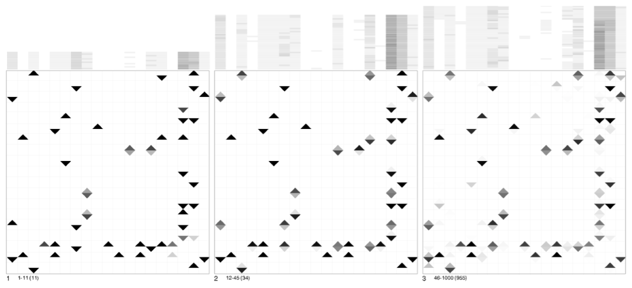
5.4.2 Genes and Brain Regions on Rats

The second data set consisted of gene expression analysed from the brains of rats bred for alcohol dependence. Gene expression was measured for rats in alcohol and alcohol-free environments, looking for candidate genes implicated in alcoholism [122]. The original analysis, which was done before the existence of BayesPiles, started with increasing lengths and networks combined in greedy searches, then moved through the same in simulated annealing. The final procedure used simulated annealing to search 200 million networks, computed a consensus across 1000, and used only the links in common from ten such searches. This was somewhat unsatisfactory as almost half the links across all consensus networks were ignored, but reproducibility was considered paramount.

The analyst repeated this experiment in a similar way by beginning with a greedy search but moved immediately onto simulated annealing when BayesPiles revealed many hills in the solution space and vastly different links present in each high-scoring network. Although the analyst decided to follow the same analysis scenario as before, this time, BayesPiles provided the opportunity to explore the solution space of 1000 networks. While only 200 could appear at the same time on screen, the analyst could use the mouse wheel to scroll through all of them. After sorting them by score, one of the first things that the analyst spotted was that the solution space appeared flat in many areas. However, after a closer look at the networks of the same score, this was revealed to not be the case. Skeleton mode revealed that many networks belonged to the same equivalence class (Figure 5.11 (a)), and diamond mode showed that there was high variance in edge direction within and across equivalence classes (Figure 5.11 (b)).



(a) Flat areas in the solution space



(b) Diamond mode showing inconsistent directionality of edges

Figure 5.11: (a) The even length of score bars together with the solid opacity of column summaries in skeleton mode suggest that networks 12-45 belong to the same equivalence class. (b) However, using the diamond mode reveals that there is a lot of variation in the directionality of edges within and across piles.

These findings led our analyst to prune networks of the same equivalence class in the parameters and to switch to skeleton mode, ignoring edge directionality. This decision reduced the number of networks considerably, from 1000 to 100. By analysing the solution space of ten runs and using new parameters with a reduced number of networks, our analyst realised that there was still a lot of variation between the high-scoring networks and that scores dropped consistently after the top 20 networks. The analyst decided to ignore lower scoring networks and considered only the top 20 networks from each run. A manual consensus network was constructed from the top-scoring networks using the consensus pile (Figure 5.16).

The analyst could easily try out different networks from the final selection of approximately 200 networks in total by adding them in the consensus pile. The analyst would apply node reordering during the process of exploring the piles in order to improve comparisons when selecting representative networks (Figure 5.6). After reordering the nodes and by hovering over the results of the first run (Figure 5.8), it became evident that while runs 1, 2 and 4 produced networks with consistently similar edges, some other runs (3, 5, 6 and 8 in particular) found networks with edges that did not appear consistently either across runs, or within the same run. In addition, the analyst could see in the histogram of scores that this structural variation seemed to have caused a drop in the score values. For these reasons, the analyst decided to include all the networks from the runs that showed consistency, ignore networks from runs with many inconsistent edges and selectively pick representative networks with edges which were consistent across runs as shown in Figure 5.16 for networks 120 and 160-164.

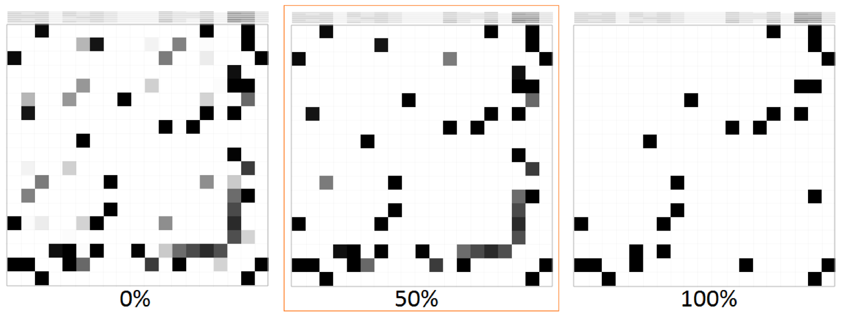


Figure 5.12: Flexible edge filtering. By moving a slider, users can interactively filter out edges from the consensus network and all other piles. Filtering out edges that appear in fewer networks contributes to the construction of a more reliable and reproducible consensus network. In other words, users are enabled to identify and control which edges to include based on how consistently they appear in high-scoring networks.

Finally, edge filtering was used to refine edge selection in the consensus network. The analyst observed that a rate of just over 50% would remove many of the edges from the view. Instead, the analyst decided to include those and only filter out edges that appeared in less than half of the networks (Figure 5.12). Thus, the analyst achieved to present a consensus network that contained a rich number of edges which were also consistent across runs. In the presentation of the final network, the consistency of each edge is reflected in its opacity.

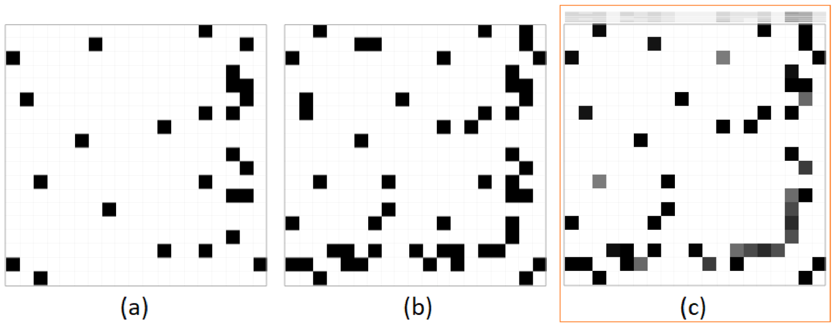


Figure 5.13: Comparison between the final BN model found by BANJO without BayesPiles with the one constructed after using BayesPiles. (a) The top-scoring network found by BANJO. (b) The same network as shown in skeleton mode. (c) The consensus network constructed manually by the analyst using BayesPiles. Users not only can gain control over the process of consensus network construction but also, they can visualise uncertainties about edges (shown in lower opacity).

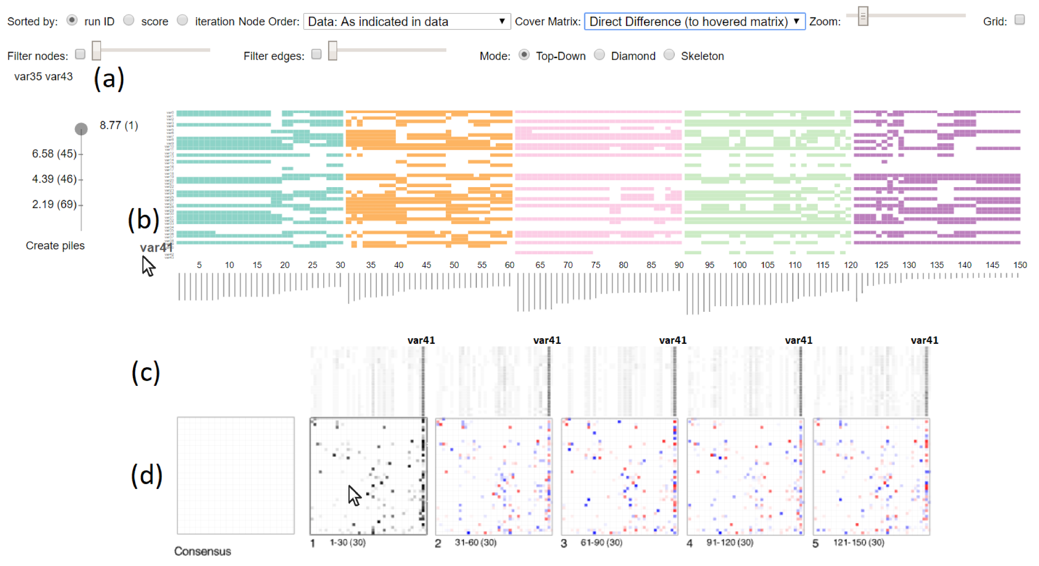
This case study showed that before BayesPiles, modellers could not explore the network structures produced by multiple runs and instead, they chose the top-scoring network found by the heuristic search algorithm (shown in Figure 5.13 (a)). This approach produced a very different result compared to the consensus network construction using BayesPiles (shown in Figure 5.13 (c)). Figure 5.13 (b) shows the top-scoring network Figure 5.13 (a) when edge directionality is ignored. Comparing (b) with (c) demonstrates that the top-scoring network and the one found using BayesPiles not only differ in the directionality of their edges but also their structure.

Note that the fact that the user-created consensus network was different does not necessarily mean that it was also better in describing the biological process. The expert user made conscious choices of constructing a different model than the one found before. This was an informed choice that was not possible using purely automatic tools. Due to the lack of ground truth knowledge in this real data set, it is hard to quantify the benefit of using the tool to estimate how much it improved modellers' decision making.

5.4.3 Gene Clusters on Ovarian Cancer Cells

The third data set consists of expression data of recognised differentially expressed genes in ovarian cancer, in two cell types - one which is responsive to the standard medication regimen, and one which is resistant.

In this ongoing experiment, our analyst initially decided to run a greedy search but switched to simulated annealing after inspecting the shape of the solution space. It was found that there was a lot of variation between the high-scoring networks in both runs and that simulated annealing found more networks that had higher scores. Then the analyst checked the consistency of results between multiple runs and found high variety, too. She then selected the 30 highest scoring networks from each run and visualised them in BayesPiles (Figure 5.14, one colour per run). When the networks


Sorted by: run ID score iteration Node Order: Data: As indicated in data Cover Matrix: Direct Difference (to hovered matrix) Zoom: Grid:

Filter nodes: Filter edges: Mode: Top-Down Diamond Skeleton

Figure 5.14: Results from five search attempts finding Bayesian networks in gene clusters of ovarian cancer cells. (a) User interface controls. (b) A summary of outgoing edges for var41 in five collections of thirty networks each. (c) Networks grouped in five piles and juxtaposed. The column that corresponds to var41 in each pile (manually labelled) appears darker indicating a high out-degree for var41. (d) Differences between the first and the other four runs are shown in the blue and red cells which correspond to edge additions and removals.

were grouped in separate piles based on their run ID (Figure 5.14 (d)), the analyst noticed that several columns (nodes) appeared very dark, suggesting consistency. Choosing one of such a highly connected node (here *var41*) (Figure 5.14 (c)) and hovering over its label in Figure 5.14 (b), the analyst sees a summary of all its outgoing edges in Figure 5.14 (b). From the rather diverse row patterns within each of the coloured columns in Figure 5.14 (b), the analyst found that the particular edges of this node are not consistent; both within the same pile (same colour, same matrix pile) and across runs (different colours, different matrix piles). The lack of consistency across runs was confirmed by hovering over the cover matrix of each pile which showed many differences in red and blue (Figure 5.14 (d)). In other words, the high node degree suggested by the dark colours in the matrices was resulting from combining the individual networks, which however did not show consistency in their edges. Consequently, the solution space is highly diverse, with many inconsistent edges, which is not clear from the score within BANJO. Given this level of uncertainty, the user was not confident enough to report a consensus network before modifying BANJO parameters and repeating the experiment.

It was also clear from the tags under the filter nodes slider (Figure 5.14 (a)) that neither var35 nor var43 have any connections to other nodes and that they have been filtered. The analyst commented: *"Looking at these initial results would definitely lead to me running BANJO on the set, minus variable 43 to see if it has any effect on the overall network. I would also probably, having looked at these results, changed my run to search for a shorter time, or for the consensus* *graphs to consist of fewer high scoring graphs. Overall, I think this would make the process more efficient."*

5.4.4 Subjective Feedback

Throughout the design and development of BayesPiles, we consulted our analyst collaborators (denoted [A1-3]) on its functionality and usability, while visualising our collaborator's data sets (every 2 weeks, around 20 in total plus minor ad-hoc sessions). During the scenarios, described in the last section, we obtained more formal feedback on the overall usability and its impact on the analysts work. All collaborators found that BayesPiles greatly extended their capabilities to explore the data. Below we report on the most prominent remarks. While some of the highlighted aspects refer to features already present in MultiPiles, we see them as evidence of our general approach of adopting MultiPiles to Bayesian network exploration.

Perhaps most important, A3 reported that *"You can see the **shape of the search space"*** which was one of our main goals with BayesPiles. **Scalability** with respect to the number of individual networks was noted: *"I think this method will be of particular use to larger data sets, as it allows for a more instinctive overview of the data and identifies areas or nodes of interest very easily upon first look."* [A1]. On the other hand, interactivity sometimes fell below real-time for data sets around 1000 networks with 50 nodes each. While 1000 is a common size for the data our analysts are dealing with, many other data sets are in the range of several hundred. We attribute this issue to the fact that BayesPiles (as well as MultiPiles) are currently implemented in WebGL and the relatively prototypical nature of our implementation. Future optimisations could possibly increase scalability further while keeping the browser-memory limits in mind.

Besides its use of reducing the number of visually present matrices and thus coping with many networks, **piling** has been found to allow for*"a quick preview of what a consensus graph would look like and which edges would be prominent, which will hopefully improve the efficiency with which I optimise my BANJO run and therefore save me lots of time"* [A1]. The interactive comparison of piles was found *"easy and [happening] in a visual manner"* [A1]. It has been found particularly useful for *"understand[ing] which nodes are reliable across runs."* [A1] Another analyst found **interactive comparison** *"useful for finding patterns between piles, such as swapping directions."* [A2]. The **manual creation and selection of consensus network** construction has been reported as *"an extremely valuable new feature [which] allows for a lot more interactive and fluid consensus matrix and will allow removal of any networks that one decides do not better the consensus network. The function is easy to use and understand, especially due to the inclusion of information about which networks are being included and how many. It would be excellent if there was the option to export this manually curated consensus network so it could be used in further study."* [A1]. Export as well as other common extensions are straightforward and discussed as future work.

Though already part of MultiPiles, **reordering nodes and columns** has been regarded an important feature *"[aiding] detection of patterns as it allows the main nodes of variation to be* *focused into one area of the graph. [This] makes it easier to focus on the most important patterns. This new feature definitely highlights for me the benefits of using a visual based method for sorting the networks."* [A1]. On the other hand, in the third case study, the user preferred the original ordering because they knew that the last set of variables (*i.e.* nodes) were combinations of treatments and thus belonged to a different class than the rest of the variables which were gene clusters. Encodings for different classes of variables or an ordering mechanism that distinguishes these classes of variables should be supported in a future version of BayesPiles.

As for the overall assessment of BayesPiles for the exploration of biological networks in the described contexts, A2 reported that she found in BayesPiles a way of **assessing and refining her previous feature-selection** steps during which the most important variables (features) were selected to be included as nodes in the network [180]. Since BayesPiles was used in an ongoing experiment, the selection of the variables was not yet fixed and could change based on visual evidence found using BayesPiles. Moreover, statistical dependencies between candidate variables were explored by comparing networks from multiple runs. Common structural properties between networks were detected and new hypotheses could be formed. BayesPiles has been highlighted for its **use in presentation**: *"[BayesPiles] makes explanation of results to a wider audience a lot easier, as the summary of many networks can be done easily and cleanly, retaining a lot of information within one image."* [A1]. On the other hand, after not using the system for a period *"[it] was hard to remember how the top-down and diamond modes were read* [A3]. This could indicate that the glyph design is not intuitive enough, however, the user had only used the system once and it was the first time they could visualise data of this complexity, therefore we believe that regular use will overcome this issue.

Moreover, identifying cycles and other patterns that require following paths is not supported effectively in BayesPiles: *"It may be that a biologist may care about cycles in the network, but BayesPiles is to help me to find what to present to the biologist; the visualisation of the final network I present for biological interpretation may be (most likely should be!) done using a different method (e.g., node-link)"* [A2]. We designed BayesPiles to support the tasks of the computational biologist in finding a consensus network. It is clear now that extending the system to support the export of the consensus network for presentation to biologists, would be beneficial. Though we had not designed BayesPiles for presentation scenarios, we believe there is a lot of potential in creating clean visualisations and interactive demos. Along similar lines, one of the analysts commented that *"[v]isualisation can help machine learning (ML) people and users of ML to do a better job in deciding how to guide my search—stay/leave the search area. It helps to learn how the heuristic search works and opens up possibilities in studying heuristic search"* [A2]. We take this as evidence for the general potential of creating visualisation interfaces for biological models.

5.5 Summary of Contribution

In this chapter, we presented a visual analytics system, *BayesPiles*, designed to support computational biologists (*i.e.* modellers) in exploring and combining Bayesian networks interactively.

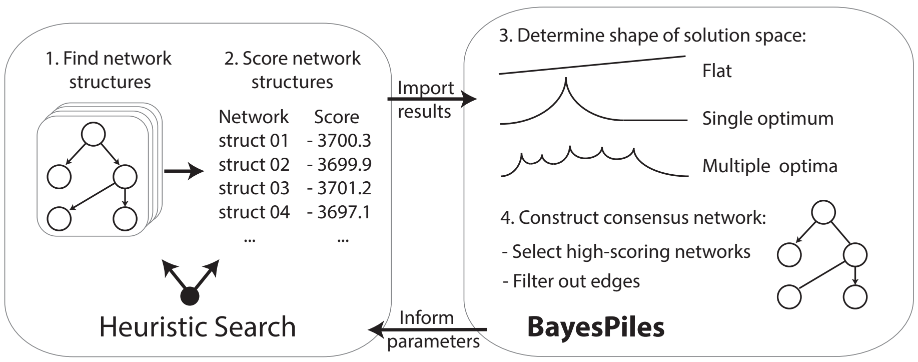
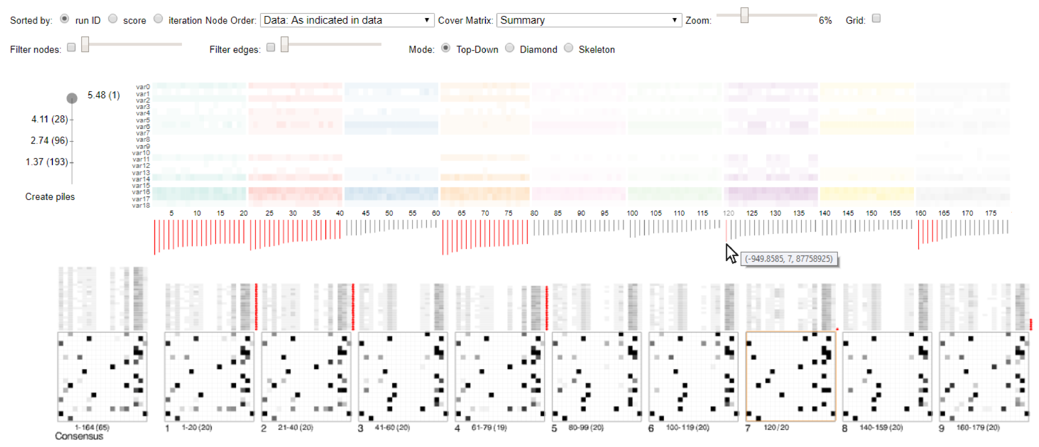


Figure 5.15: An overview of the workflow. BayesPiles visualises the results of heuristic search algorithms, informs their parameter settings and enables the construction of a consensus network structure.

Figure 5.15 shows a typical analysis process in systems biology; 1) find and generate network structures, 2) score networks, 3) find optimal networks, and 4) select a final network, the *consensus network*. Rather than relying on a single (automatically selected) network for consensus, BayesPiles encourages the active involvement of the analyst in constructing the consensus network by supporting the visual exploration of the most interesting (*i.e.* high-scoring) networks in the search space and the manual creation of a *consensus network* by combining selected networks from the entire set, thereby accounting for multiple optimum solutions in the search space. The learning process of the final network involves a combination of automated steps (performed by heuristic search algorithms) for generating networks and manual steps (performed by a "human-in-the-loop") for understanding the solution space and for constructing the consensus network. BayesPiles visualises intermediate search results in order to help with the adjustment of parameters that control the next round of a heuristic search, generating a novel set of networks. This is repeated until a good sampling of the search space is achieved and the results represent the most interesting regions of the search space (*i.e.* the solution space). Therefore, BayesPiles is part of the wider learning process that involves multiple executions of search algorithms, each followed by the interactive exploration of the results by experts.

BayesPiles (Figure 5.16) is inspired by an existing visualisation interface, called MultiPiles [18], which has been proven successful for exploring temporal states in dynamic (temporal) networks with large numbers of timesteps. MultiPiles uses a sequence of matrix representations, one for every timestep, and which can be automatically or interactively grouped into *piles* (or stacks) of matrices. BayesPiles provides features specifically designed to support domain-specific requirements for exploring, comparing and combining multiple BNs: a) visualising network quality scores; b) ordering and grouping networks by experimental run ID, score, and iteration number; c) additional visual encodings (e.g. for directed edges); d) extended filtering capabilities; and e) manual construction of a consensus network through combining networks. BayesPiles was developed in close collaboration with computational biologists and was tested on three different


Sorted by: [✓ run ID] score iteration Node Order: Data: As indicated in data Cover Matrix: Summary Zoom: 6% Grid:

Filter nodes: Filter edges: Mode: [✓ Top-Down] Diamond Skeleton

Figure 5.16: A snapshot of BayesPiles taken during the interactive construction of an average consensus network.

of the biologists' data sets. In these case studies, BNs were used to find relationships between brain regions and the singing behaviour of birds; genes and brain regions of rats; and relationships between genes, cell lines and treatments on ovarian cancer cells. Subjective feedback from the biologists revealed that our tool provided new insights and more confidence in achieving not only more reproducible but also more representative results.

In the next chapter, we investigate support for the representation and exploration of dynamic Bayesian networks, which result in the comparison of hundreds of multivariate networks. Despite their name, dynamic Bayesian networks are static networks with multiple types of edges which can describe dynamic systems. The specific challenges that arise in representing and exploring dynamic Bayesian networks are discussed in more detail in the following chapter.

6 Visual Encodings for Edges in Matrices

The previous chapter describes how BayesPiles handles collections of many scored directed networks and enables the exploration, combination and comparison of results from different heuristic search runs. The case studies showed how biologists, based on visual feedback from BayesPiles, effectively represent and analyse Bayesian networks. Besides Bayesian networks, computational biologists are also interested in other types of networks which are more complicated and difficult to represent visually. As discussed in Section 3.3.1, examples of such networks are the dynamic Bayesian networks (DBNs), which contain different types of edges. In this chapter, we focus on the problem of visualising DBNs and in general, networks with multiple edge types between the same pair of nodes. Such networks are also known as multivariate, multilayer or multi-modal networks and are very common for representing relationships in complex systems in disciplines such as biology, ecology, social networks and software engineering [55, 74].

We focus on answering the third research question: **Q3** ***"How to effectively represent information related to the dynamics of biological systems, encoded in the edges of inferred networks?"*** We present how we applied the nested model methodology [132] to understand the tasks related to DBN inference and propose effective visual encodings for networks with different edge types. In Section 6.1, we present how the first level of the nested model was applied to characterise the domain problem. In Section 6.2, as part of the second level of the nested model, we present a list of tasks related to the visual analysis of DBNs, while in Section 6.3, we explore the design space and discuss the rationale followed for designing the proposed encodings (third level). Then, in Section 6.4, we present the design of a formal user study with which we evaluated our proposed visual encodings. Finally, in Section 6.5, we present the results of this study followed by a section that describes how the encodings were integrated into BayesPiles (fourth level) and how that extension was evaluated by a group of three computational biologists (Section 6.6). The paper for this evaluation study was presented in the *VIS 2019 Workshop on the Visualization of Multilayer Networks* [182].

6.1 Domain Problem Characterisation

As part of the first level of the nested model (Section 2.4), we collaborated with three computational biologists (*i.e.* modellers) who wanted to use visualisation to help them infer DBNs. Our first collaborator was specialised in computational biology using Bayesian methods. The second was a biology graduate who analysed DBNs as part of a research project. Our third collaborator was a PhD candidate in bioinformatics who was interested in inferring DBNs from neuroimaging data.

In order to understand the domain problem, we performed a literature review (Section 3.3) and we held four one-hour meetings with the most experienced of our collaborators and two one-hour meetings with each of the research students (eight meetings in total). We found that Dynamic Bayesian networks extend Bayesian networks and they can model dynamic systems. A DBN does not change over time but rather it can describe the interactions of a system that evolves over time [135]. Although the structure of DBNs is static, representing these networks visually is challenging because their edges are associated with multivariate data. In particular, DBNs contain different types of edges, each corresponding to the number of time-slices skipped before observing the interaction between a pair of nodes.

Modellers are interested in inferring DBNs to understand the dynamics of biological processes within living organisms. Important events and behaviours could be explained by changes that occur in different parts of the system over a period of time. DBNs can be used to make predictions, identify causes and provide reasonable explanations about the ways variables interact over time [32]. Identifying connections that integrate information about their underlying dynamics and their perturbations can help biologists to understand the state of a dynamic system, to design experiments for collecting data and to reason about causal relationships between variables that explain the evolution of interesting natural phenomena. For instance, the singing behaviour of a songbird can be related to the activation of certain regions in the brain of the bird and this neural flow network can be described as a DBN [167]. In medicine, inferring DBNs can be helpful for identifying and describing profiles of cells based on their response to certain conditions measured over time [106].

While reviewing the literature related to network visualisation, we were surprised *not* to find any indication about how to best visualise multiple edges and their types in a matrix format (Section 3.3.2). In particular, our collaborators use DBNs to model probabilistic dependencies in complex biological processes. These processes range from gene regulation [195] to brain connectivity [167] and ecological networks [4]. Heuristics are used to find the most probable relationships in these large search spaces. Visualisation and comparison of these networks require adjacency matrix visualisations as shown in the previous chapter, which require modellers to find and compare edge types, compare edges across networks, as well as find edges of specific types. While edges are shown as matrix cells, the visual design space for both i) multiple edges, and ii) edge types is potentially very large, including simple designs such as bar charts [63, 61, 92], Gestaltlines [36], contrast [18], and glyphs [187]. However, to the best of our knowledge, no study or design paper exists that investigates the respective effectiveness of these design decisions.

This chapter presents several visual designs (encodings) which were developed to represent multivariate networks as matrices. Characteristics of DBNs were explained and current approaches for visualising multivariate networks and design options were discussed in Section 3.3. In the following Section 6.2, we present a list of common tasks related to DBN analysis.

6.2 Tasks

Modellers analyse sequential time-series data, to infer the structure of a DBN that describes the dynamics of a system. Similarly to BN inference, described in the previous chapter, biologists use heuristic search algorithms that scan the space of all possible networks. The results of each execution of the algorithm (*i.e.* run) is a collection of high-scoring DBNs. As with BN inference, the role of the modeller is to guide the heuristic search and decide on a method that determines the structure of the final consensus network. This process requires the visual representation and analysis of multiple DBNs. However, representing DBNs visually, using BayesPiles or other tools, is a challenging task mainly because these networks are multivariate.

To create effective visualisation tools for analysing DBNs and other multivariate networks of similar characteristics, we identified a list of DBN analysis tasks. After four sessions of one hour each, in the form of informal interviews with two of our collaborators, we identified four analysis tasks where visualisation could help. Abstracting the data from the tasks was part of the second level of the nested model.

T1: Identify the interaction with the highest number of different edge types (MLs) in a DBN. Modellers are interested in finding information about the dynamics of the system incorporated in the edges of the DBN. When an interaction appears in multiple MLs, then there is uncertainty about the actual dynamics of that interaction in the system. Thus, it is important to identify edges with multiple MLs.

T2: Identify which edge type appears more often in a DBN. Time granularity is related (and often coincides) to the sampling rate picked for collecting expensive experimental measurements from the system. Identifying the most common ML in a DBN informs tuning the frequency of time-slices (or the sampling rate) accordingly so that more detail about the dynamics of the system is revealed in the inferred networks. For example, if most of the edges in a network are of type 3, time granularity is too fine; most edges being of type 0 implies the opposite.

T3: Identify which combination of two edge types appears more often in a pile of networks. Multiple DBNs could be combined to form an aggregated network (a *pile* in BayesPiles [181]). Following the previous task (T2) modellers are also interested in finding interactions that combine two edge types in the same pile of networks (e.g. ML0 *and* ML3) between a pair of nodes. This could indicate a relationship between two edge types and could help to reason about the effect of the interactions of those edge types over time. For instance, it could indicate that there may be

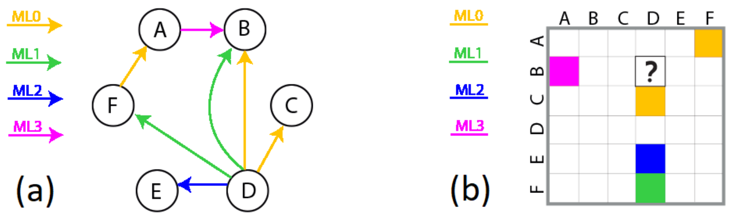


Figure 6.1: (a) A DBN visualised as a multivariate node-link diagram and in (b) as a multivariate matrix.

two different biological processes captured in the same network, which have different dynamics but affect the same pairs of nodes.

T4: Identify which DBN contains the highest number of a particular edge type in a collection of DBNs. Because modellers often have to analyse collections of multiple DBNs, we need to compare features between two DBNs and identify differences. For example, modellers are interested in finding networks that contain many edges of a particular type, e.g., when networks are assessed based on their structure to be included in the consensus (aggregated) network. For instance, modellers are interested in creating consensus DBNs which mostly contain edges of a particular type. They could choose to include networks with many persistence edges (ML1) and exclude networks with a lot of edge type variation in their structure or networks with many intra-time-slice edges (ML0).

The above analysis tasks show how important it is for the modellers (*i.e.* computational biologists) to identify different edge types in visual representations of DBNs. All of the above tasks are related with the ability to distinguish between edge types, identify instances and combinations of them, and to count them and compare them. After discussing with modellers, to identify their analysis tasks, we discovered that for the visual analysis of DBNs it was important to create effective encodings that support multiple edge types. Based on an evaluation of the encodings, the design of BayesPiles could be informed so that it could also support the analysis of multivariate networks, such as DBNs.

6.3 Encoding Multiple Edges in Matrices

For the traditional node-link representation of a DBN found in the literature [135, 105, 170], all transitions from one time-slice to the next are required and the nodes are repeated in every time-slice, as shown in Figure 3.13. The resulting network is read from left to right following the sequence of time. DBNs can become very long when they extend to many time-slices and are therefore hard to visualise. A more compact representation requires additional visual encoding to help distinguish between the different edge types. Figure 6.1(a) shows how the network becomes much smaller when the nodes are not repeated and edge types are encoded using colour. However, this approach creates visual clutter for more dense networks because of edge crossings [9].

Figure 6.1(b) shows how the same DBN can be represented as a directed adjacency matrix, again using colour to encode edge types. Edges are represented by adjacent cells in the matrix read by column and row (*i.e.* top-down). This representation is less cluttered than node-link diagrams when the networks are dense [9, 75], but when there are multiple edge types for the same pair of nodes, the visual encoding becomes challenging. For instance, in Figure 6.1(b) it is not clear how to encode two edge types coexisting in the one cell that corresponds to the pair of nodes D and B. One possibility is to create a glyph by splitting the cell into as many parts as edge types and use a different colour for each part. In the following section, we explore the design space for encoding networks with multiple edge types as matrices.

6.3.1 Matrix Cell Designs

In Section 3.3.2 we discussed different approaches to the problem of multivariate network visualisation. We found that integrated approaches are more efficient than multiple and coordinated views [96, 77]. However, embedding additional multivariate data into standard network representations that use node-link diagrams or adjacency matrices, is not intuitive and it often results in visual clutter. Because we were interested in extending BayesPiles to also support the analysis of DBNs, we only considered integrated approaches for matrix-based representations. Node-link diagrams were not considered, not only because they are not supported in BayesPiles, but also because there are limitations in embedding additional multivariate data in edges represented as arrows. Moreover, node-link diagrams often suffer from visual clutter caused by multiple edge crossings even for networks that are not multivariate [9, 73]. On the other hand, matrices provide more opportunity for encoding multivariate data in their edges. The available plane space in each matrix cell could be used to encode not only the existence of an edge but also its type. Encoding multiple types in the same cell could be done either by splitting the cell or by using icons and glyphs.

Criteria: Based on the task identification, modellers' feedback and a literature review, we developed several potential visual encodings for representing DBNs through matrices (Figure 6.2). The goal of our design was to find a representation that fulfils the following criteria per edge: C1) multiple edge types and C2) in a given order. In matters of scalability, we expected that a successful encoding would be able to represent clearly at least four different edge types in a single cell. For our empirical tests and prototypes, we considered matrices of up to 50 nodes in size, which were either displayed within BayesPiles or in images that could fit in a standard laptop monitor. Given the limited amount of space in each matrix cell, to avoid unnecessary visual complexity and clutter, we tried to maximise the *data-to-ink ratio* [174] utilising the maximum number of pixels within the cell. In other words, we allowed a minimum number of pixels to remain unused, those that were necessary for distinguishing between the visual marks that encoded the different edge types. This approach produced more salient marks to encode edge types in the matrix representation. Also, we based the design of those encodings on primary visual variables, such as colour, opacity, size, shape, texture, orientation, position and their combinations.

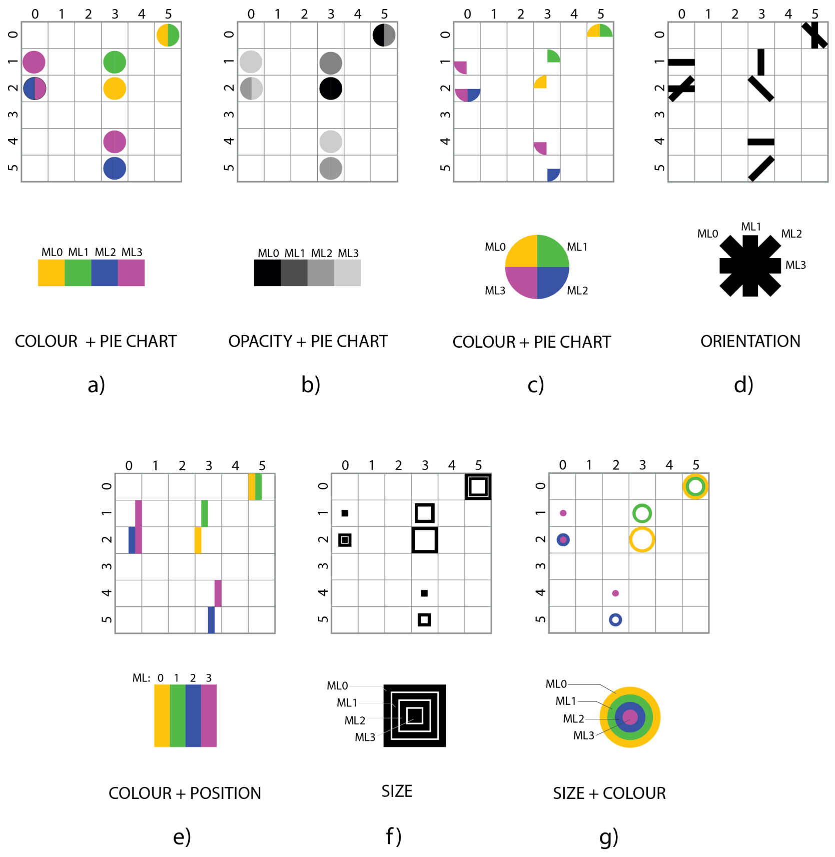


Figure 6.2: Examples of visual designs considered for encoding multiple types of edges in matrices. The top row shows an example of a single matrix and the bottom row shows the encoding for each ML type. The encodings use one or more visual variables to represent multiple edges: a) uses a coloured pie chart, b) uses opacity in a pie chart, c) uses a segmented and coloured pie chart d) uses orientation, e) combines position and colour, f) uses size and g) combines size and colour to create a glyph.

Design Space: Figure 6.2 shows some of our initial designs, created through iterations and discussions with the modellers. The upper row in this figure shows the encodings for a single matrix where we need to visualise potentially multiple edge types (C1) and their order (C2). For example, design (a) uses colour and design (b) uses shades of grey to differentiate types of edges. Design (c) is a variation of design (a), using equally-spaced and coloured segments of a pie chart to indicate the presence of a lag (one segment per lag type). Design (d) uses orientation (angle) to differentiate edge types and encode their order. Design (e) uses position and colour within each cell, resulting in a striped cell design. Design (f) uses size and design (g) is a variation of design (f) which uses both size and colour to encode category and order, but instead of superimposed squares, it uses rings.

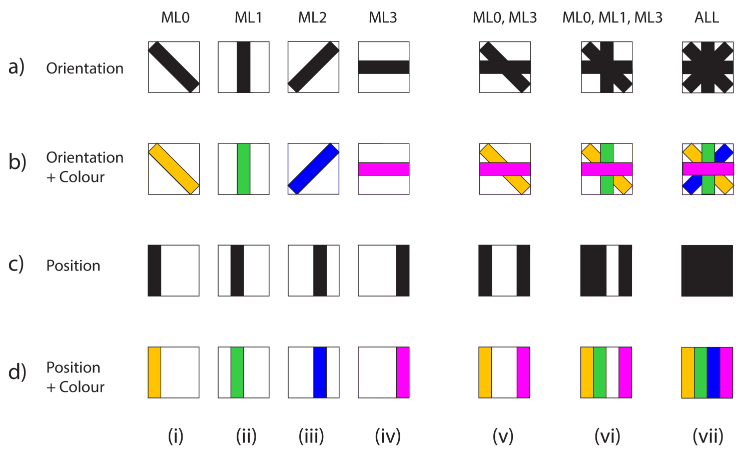


Figure 6.3: Proposed visual encodings for the different types of edges (MLs) tested in the user study: a) orientation without colour (`ORI`), b) orientation with colour (`ORI+COL`), c) position without colour (`POS`) and d) position with colour (`POS+COL`). Columns (i), (ii), (iii) and (iv) show the encoding of edge types ML0, ML1, ML2 and ML3 accordingly. Columns (v), (vii) and (vii) show how the combination of two, three and four edge types look in the different encodings.

Final Designs After we identified criteria for potentially successful encodings and explored the design space of possible combinations, we discussed potential solutions in three one-hour sessions with a team of four visualisation experts and two domain experts (*i.e.* computational biologists), to select the four most promising encodings which we decided to compare in our final study (Figure 6.3). We rejected designs that would easily get cluttered when they were combining multiple edge types in the same cell and those that do not make it easy to distinguish between the different types of edges. For instance, we found that encodings that used texture were easily getting cluttered and they were hard to discern in the limited space of a matrix cell. Our selected encodings use at least one of the following visual variables: *position*, *orientation* and *colour hue* as they can not only easily encode the four types of edges found in DBNs, but also they can scale to encode up to a dozen different types. Orientation and position can stand alone or in combination with colour, however, colour can only be used in combination with another visual variable.

When colour was used as double encoding in addition to position or orientation, several colouring schemes were tested. However, we found that a colour scheme using ColorBrewer [84] was more appropriate since the colours looked more balanced and distinct. Examples of the final encodings used on the same multivariate network are shown in Figure 6.4. All four selected encodings could sufficiently represent networks with multiple types of edges, such as DBNs. However, we wanted to compare them and find which was the most effective for supporting the identified analysis tasks that modellers most commonly perform when they infer DBNs. Therefore, we ran a quantitative evaluation study to assess the effectiveness of the final encodings. The study involved participants from the general public who were asked to perform simple visual analysis tasks on matrices that use those encodings.

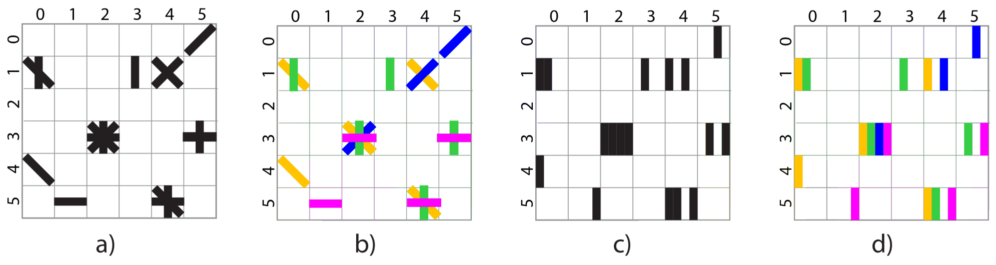


Figure 6.4: An example of the final encodings showing the same multivariate network: a) orientation without colour (`ORI`), b) orientation with colour (`ORI+COL`), c) position without colour (`POS`) and d) position with colour (`POS+COL`).

6.4 User Study

To test the effectiveness of the encodings in multivariate networks represented as matrices, we performed four controlled user studies through Amazon Mechanical Turk (AMT). The visualisation tasks that participants were asked to complete in the study derived from the biological analysis tasks described in Section 6.2. Each experiment evaluated the encodings with a different task, with each participant performing only one task at a time. Participants could perform each task (experiment) only once, but if they wanted they could participate in all four of them. We present the results of the four experiments individually and then discuss the results.

6.4.1 Data

It was important to find a realistic data set that could be used across the four experiments. In addition, we wanted to have control over the characteristics of the data set to draw useful results about the tested encodings so that the tasks were not too easy or too hard. Therefore, we generated data that resembled the network structures in our real-world data. We used the following process for generating the multivariate networks in the experiments and refined the parameters in two pilot studies; one lab-study with 5 visualisation experts which included three experienced academic researchers and two PhD students in the area of information visualisation, as well as through an AMT study with 20 participants for each task/experiment.

1. We generated Watts-Strogatz networks using NetworkX [83], to target a wide range of real-world networks which often exhibit small-world properties. Such networks are neither regular ($p = 0$) nor random ($p = 1$), but they are located in an intermediate point ($0 < p < 1$) [190]. We tried different settings but our final base-networks were directed and of two sizes (30 and 50 nodes) because DBNs in biology are usually small and rarely exceed this number of nodes [121]. The probability of a random edge (p) was set to 0.25 and to avoid disconnected networks, every node was initially connected to its 4 neighbouring nodes ($k = 4$). The resulting base-network had many resemblances with real DBNs in matters of density, structure and size.

2. For each trial, we created a version of the base-network by randomly removing a number of its connections. Each time, we randomly removed 15% of the total connections in the base-network, resulting in a collection of networks with similar structure and the same number of connections.

3. The networks were transformed into multivariate networks with four types of edges. For each edge type (ML0-ML3), we added one link between each node pair that was connected in the network. As a result, every connected node pair included all four types of edges.

4. Finally, for each trial, we specified the percentages that each edge type or possible combination of two or three edge types (15 in total) appeared. All edge types and their combinations appeared in the same frequency counterbalancing each other except the correct answer (type or combination) which appeared more times in the network. The difficulty of finding the correct answer was based on how much higher the percentage of the correct answer was in relation to the other (wrong) answers. The higher the percentage, the easier the task. Each of those percentages could be adjusted easily through an interface using filters.

We tried different combinations of network sizes and percentages for each experiment until we found settings that resulted in tasks that were neither too easy nor too hard for participants to complete. In our pilot studies, we tested networks of two sizes (of 30 and 50 nodes) and four difficulty levels ("very easy", "easy", "hard" and "very hard"). The percentages in which wrong and correct answers appeared in the generated matrix are summarised in Table 6.1. For task T1 there was always only one correct answer (*i.e.* a cell that contained all 4 MLs). For task T3, the remaining percentage of cells contained only one mark with all four types (MLs) equally distributed in each matrix e.g. for the "very easy" setting, cells with a single ML would occupy 40% of the edges (10% for each ML). This network generation process ensured precision and control over the properties of the networks so that each data set would consist of different networks with similar characteristics, such as structure, size, regularity, connectivity and difficulty. Every participant in the pilot study would perform one task in all 4 difficulty levels. We found that participants were responding very slowly when the data sets were "hard" or "very hard". Slow response times indicated that the tasks were becoming too difficult for the participants. Therefore, to minimise fatigue in our experiments, we only used "very easy" or "easy" networks of 30 nodes each.

6.4.2 Questions to Participants

For each task described in Section 6.2, we performed a crowdsourcing user study. For the needs of each study, the tasks were rephrased and presented as generic visualisation tasks to the participants, who were not required to have any background knowledge in biological modelling.

Task	very easy	easy	hard	very hard
T1	2x95%, 3x5%	2x75%, 3x25%	2x50%x, 3x50%	3x100%
T2	3x20%, 1x40%	3x21%, 1x37%	3x22%, 1x34%	3x23%, 1x31%
T3	5x8%, 1x40%	5x10%, 1x30%	5x11%, 1x25%	5x12%, 1x20%
T4	3x20%, 1x40%	3x21%, 1x37%	3x22%, 1x34%	3x23%, 1x31%

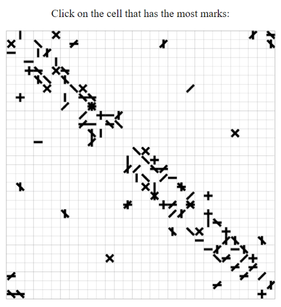
Table 6.1: Parameter settings as percentages of marks in each matrix used for generating networks of the same density but of different difficulty levels.

Q_T1 For each trial in task T1, participants were shown a matrix and they were asked to *"Click on the cell that has the most marks"* with each mark encoding a different edge type. In each trial, the correct answer either contained all 4 edge types or 3 out of 4 edge types. The rest of the connected cells in the matrix (*i.e.* node pairs) either contained a single edge type or a combination of edge types. In particular, there was one instance for every possible combination of edge types that contained one edge type less than the correct answer. For example, if the correct answer included all 4 edge types (ML0, ML1, ML2, ML3), then in the matrix there was one instance for each of the following combinations: (ML0, ML1, ML2), (ML0, ML1, ML3), (ML0, ML2, ML3) and (ML1, ML2, ML3). An example of a trial that uses the orientation encoding is shown in Figure 6.5(a).

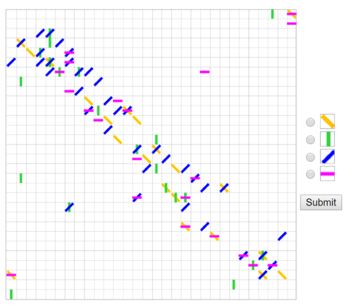
Q_T2 For each trial in task T2, participants were shown a matrix and were asked to select *"Which mark appears more often in the image?"* from a list which displayed all four edge types as icons. In each trial, the correct answer had a double number of instances compared to any other type and 25% of those instances appeared alone and the rest in combination with a second edge type distributed in equal numbers. An example of a trial that uses orientation with colour is shown in Figure 6.5(b).

Q_T3 For each trial in task T3, participants were asked to select *"Which combination of two marks appears more often in the image?"* In each trial, the correct answer had a double number of instances compared to any other combination of two edge types. Also, while there were cells with a single edge type or combinations of more than two edge types, they appeared in the same proportion. An example of the interface that uses position is shown in Figure 6.5(c).

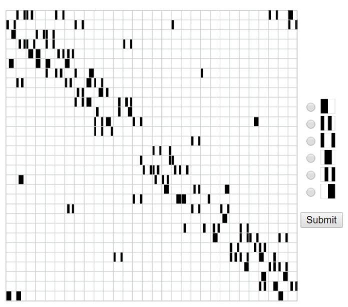
Q_T4 For each trial in task T4, participants were shown two matrices and they were asked to select *"Which matrix has more cells"* of a particular edge type. Alternative choices were: *"Both the same"* and *"I don't know"*, but those options were never the correct answer in the test. In each trial, the wrong matrix had the same percentage of either edge type. However, the correct matrix had an increased percentage of the requested edge type, while the rest of the edge types shared an equal percentage of instances. Thus, both matrices maintained the same density (*i.e.* the total number of edges). An example of the interface that uses position



(a) Example from `ORI`

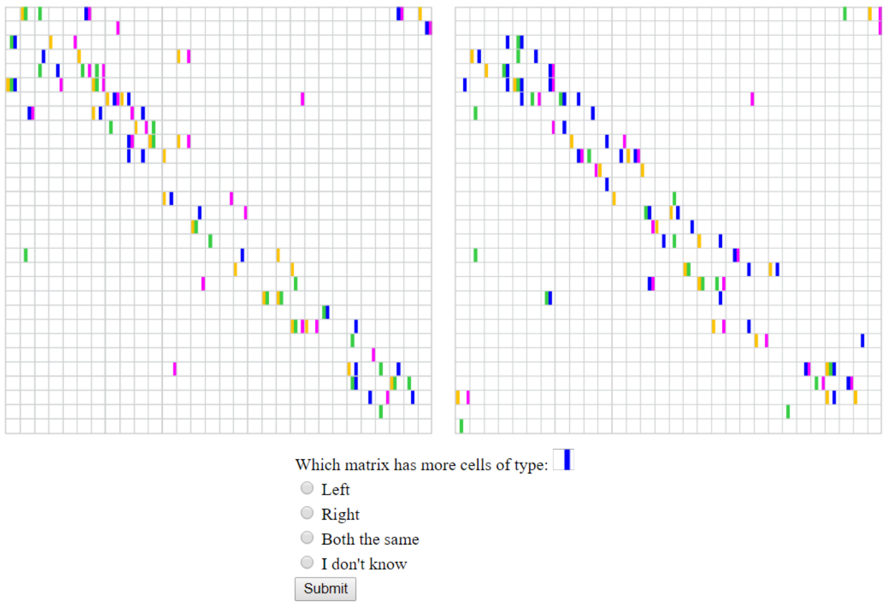


(b) Example from `ORI+COL`



(c) Example from `POS`

Figure 6.5: a) An example of the interface used in the first task. This trial uses the encoding orientation without colour (`ORI`). b) An example of the interface used for the second task which uses orientation and colour (`ORI+COL`). c) An example of the interface used for the third task which uses position without colour (`POS`).



Which matrix has more cells of type:

- Left
- Right
- Both the same
- I don't know

Submit

Figure 6.6: An example of the interface used for the fourth task which uses position with colour (`POS+COL`).

with colour is shown in Figure 6.6.

6.4.3 Hypotheses

T1: The first task asked to *find the most effective visual encoding for multiple edge types(marks) combined in a single cell of a matrix.* The task required participants to (a) identify the densest cell in the matrix and (b) compare it to other dense cells in the same matrix to verify their choice.

- **H1(a)**: `ORI` will outperform `POS`. When multiple edge types are combined in the same cell, `ORI` looks like a star whereas `POS` is a square which is less clear. The shapes produced by `ORI` are more perceptually salient.

- **H1(b)**: Colour will improve the performance. We believe the redundant encoding would make it easier to compare the differences between cells.

T2: The second task asked to *find the most effective visual encoding for assessing the frequency of a mark in a matrix.* Participants had to: (a) distinguish between the four edge types and (b) compare their frequency of appearance in the matrix.

- **H2(a)**: Again, `ORI` will outperform `POS` as the shapes it forms are more visually salient.

- **H2(b)**: We believe that colour will improve performance as it will help in estimating the number of marks of a particular type in the matrix.

T3: The third task asked to *find the most effective visual encoding for finding the most frequent combination of two edge types in the matrix.* Participants had to: (a) distinguish between the 6 possible combinations of two edge types and (b) compare their frequency of appearance in the matrix. In many ways, this task is similar to T2 but instead of looking for single edge types, we are interested in combinations of two edge types and their resulting visual encoding.

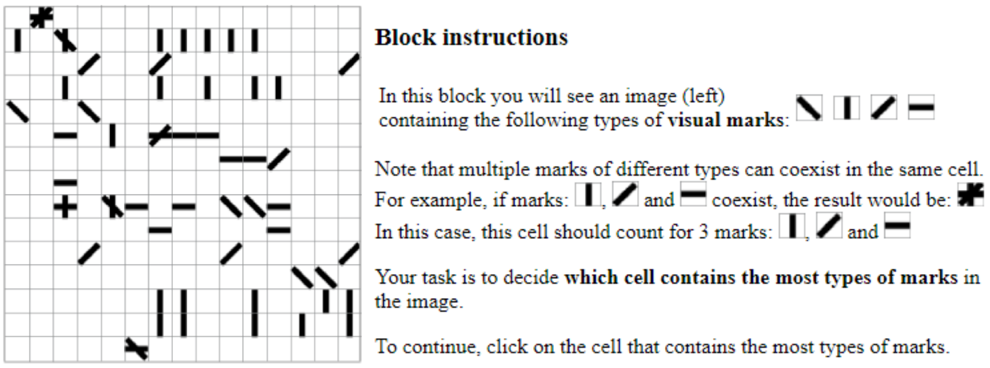
- **H3(a)**: `ORI` will outperform `POS`. Pairs of marks will have more distinctive shapes than those produced by `POS`(compare (a) with (c) in Figure 6.3(v)).

- **H3(b)**: We believe that colour will improve performance as it would help make the patterns of pairs of marks more distinctive.

T4: The fourth task involves *selecting the matrix with the most frequent edge of a particular type.* Participants had to: (a) distinguish between the 4 edge types to find the one requested and (b) compare the frequency of its appearance between two matrices. This task is similar to T2 but involves the comparison of two matrices.

- **H4(a)**: We believe that `ORI` would outperform `POS` as the marks are more distinctive.

- **H4(b)**: We also believe that colour would improve performance as it would make it easier to assess the number of edges of a particular type.



Block instructions

In this block you will see an image (left) containing the following types of **visual marks**: \ | / —

Note that multiple marks of different types can coexist in the same cell.
For example, if marks: |, / and — coexist, the result would be: ✱
In this case, this cell should count for 3 marks: |, / and —

Your task is to decide **which cell contains the most types of marks** in the image.

To continue, click on the cell that contains the most types of marks.

Figure 6.7: Instructions were shown before every block of trials. This is an example of how instructions were shown before the block that was testing orientation in the first task (T1).

6.4.4 Experimental Procedure

Each of the four user studies tested one task in a within-subject 2 × 2 design (× 2 techniques: orientation/position × 2 colour schemes: with/without colour), testing the following 4 experimental conditions (*i.e.* encodings in Figure 6.3): a) orientation without colour, b) orientation with colour, c) position without colour and d) position with colour.

The first page of each study contained information about the study and a list of terms which participants were asked to read carefully. Then participants were asked demographic questions such as: age, gender, familiarity using data visualisations, and the device they were using for the study (laptop, desktop, tablet or phone). In the case they selected phone, the website showed a message that participation was not possible due to the small available screen size. Throughout each study, the entire information was shown in one full-screen window without a need to zoom or pan.

Each study had 4 *blocks* (parts) of trials, one for each encoding. Before each block, there was a page with instructions that explained the task and demonstrated how to use the interface to submit answers (Figure 6.7). Participants were asked to first complete the task on a demo trial before proceeding to the main study, to familiarise themselves with the interface. Following feedback explained why their response was correct or incorrect. Except for the demo trial at the beginning of each block, participants did not receive feedback on the correctness of their answers in the recorded trials.

After the demo trial, each block contained 3 training trials, followed by 12 recorded trials for tasks T1 and T3 and 8 recorded trials for tasks T2 and T4. Fewer trials were used for tasks that participants spent more time to complete in our pilot studies. Also for tasks T2, T3 and T4, attention trials (gold standard) were placed randomly within the recorded trials showing a very simple situation where an attentive participant would not be expected to make an error. Attention questions were similar to the recording trials with the only difference that the correct answer was

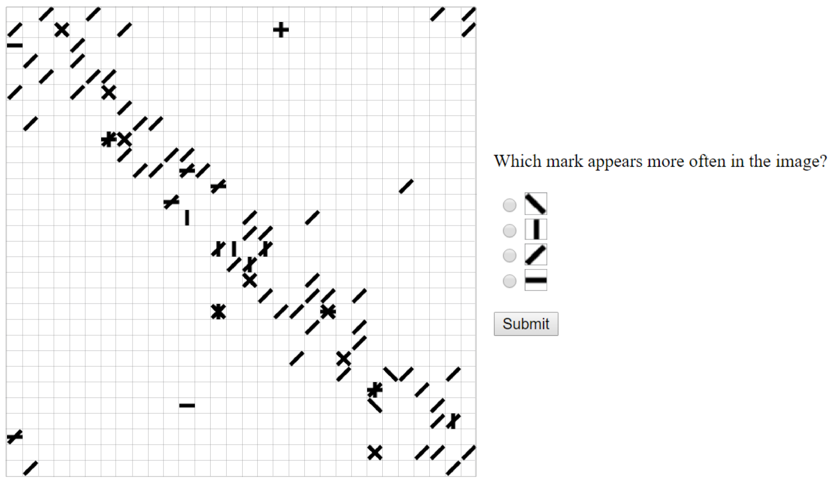


Figure 6.8: An example of a gold standard question for task T2.

very easy to find. For example, for T2 the correct answer would appear in a very high percentage compared to all other answers (Figure 6.8). Those trials were later used to identify participants who were not invested in the experiment or who did not understand the task. In task T1, there was no gold standard and the participant attention to the task was evaluated by comparing the mean error rate with an error rate close to chance. In between trials, a neutral screen would appear for two seconds to help participants focus their attention to the next image (Figure 6.9).

The blocks of the experiment were counterbalanced to alleviate the effects of presentation order (learning and fatigue effects). Visual encodings were very different, precluding randomising experimental trials. For each of the four experiments, approximately half of all the participants started the experiment orientation first. Within each encoding (orientation and position), approximately half performed the task with colour first. Except for the order of blocks, the recorded trials were also shuffled within each block for each participant. Between the 4 blocks, participants were encouraged to take breaks.

The effectiveness of each encoding was evaluated through two dependent variables: (a) participants' *response time* (in milliseconds), that measured efficiency and their (b) *error rate*, that measured accuracy. For each trial, there was only one correct answer, so the average error rate of each block was ranging between 0 and 1. However, regarding participants' response times, there was a danger of including outliers by taking the average, since we did not have control over the study environment [146]. Therefore, for each encoding, the average response time was calculated as the 25% truncated (trimmed) mean of response times within each block of recorded trials. This excluded the minimum and maximum response times of each block from calculating the mean. Thus, from the 8-12 trials in each block, we only considered 6-10.

Each study was run on Amazon Mechanical Turk (AMT), linking to our own JavaScript website. All participants were located in the US and paid $3 for completing the test which lasted

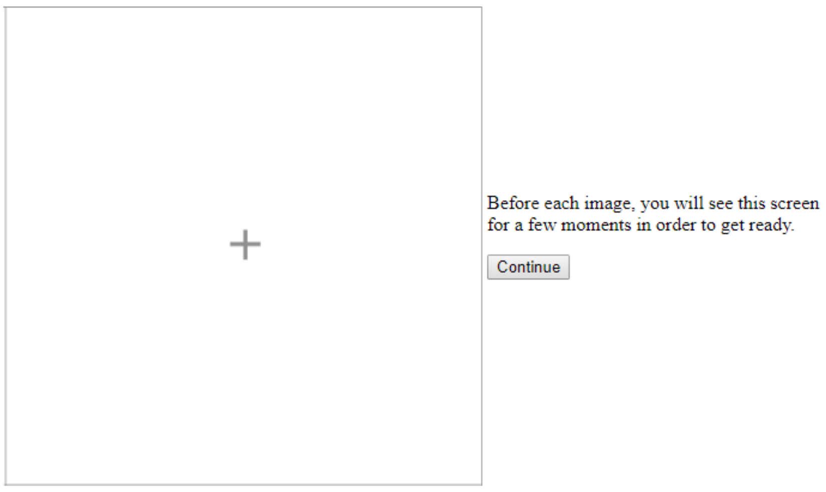

Before each image, you will see this screen for a few moments in order to get ready.

Figure 6.9: This image appeared between trials to help participants concentrate.

approximately 15 minutes. Participants were told to answer as fast and accurate as possible. There was no time limit to complete the study. Multiple participation in the same study was prohibited, but participants could participate in all four studies. At the end of the test, an authentication code would appear on screen which participants could use to claim payment through AMT for completing the task.

6.5 Results

We analyse the results considering each task as a separate experiment as set out in our experimental design. Our statistical protocol was set out in advance and applied separately to the response time and error rate of the four combinations of factors: `ORI`, `ORI+COL`, `POS`, and `POS+COL`. Before analysis, we applied data quality checks. Participants that answered with an accuracy close to chance or that failed more than half of the four gold standard questions (available in T2-T4) were removed from the analysis. Participants that did not complete the full set of experimental trials were also removed (within-subject design). For the first experiment, Figure 6.10 (left) shows the distribution of all valid participants' average response times for each experimental condition (block). Figure 6.10 (right) shows the distribution of all participants' average error rates for the first experiment. For the second, third and fourth experiments, the corresponding distributions are shown in Figures 6.11, 6.12 and 6.13, respectively.

After the quality checks, the response time and error rate data were analysed separately. For each of the four distributions, we used a Shapiro-Wilk test with a significance level of $\alpha = 0.05$ to determine if the data were normally distributed. In all cases for response time and error, at least one of the distributions was found to not be normally distributed. Therefore, we ran a Friedman test with a significance level of $\alpha = 0.05$ to determine if there was a significant difference between

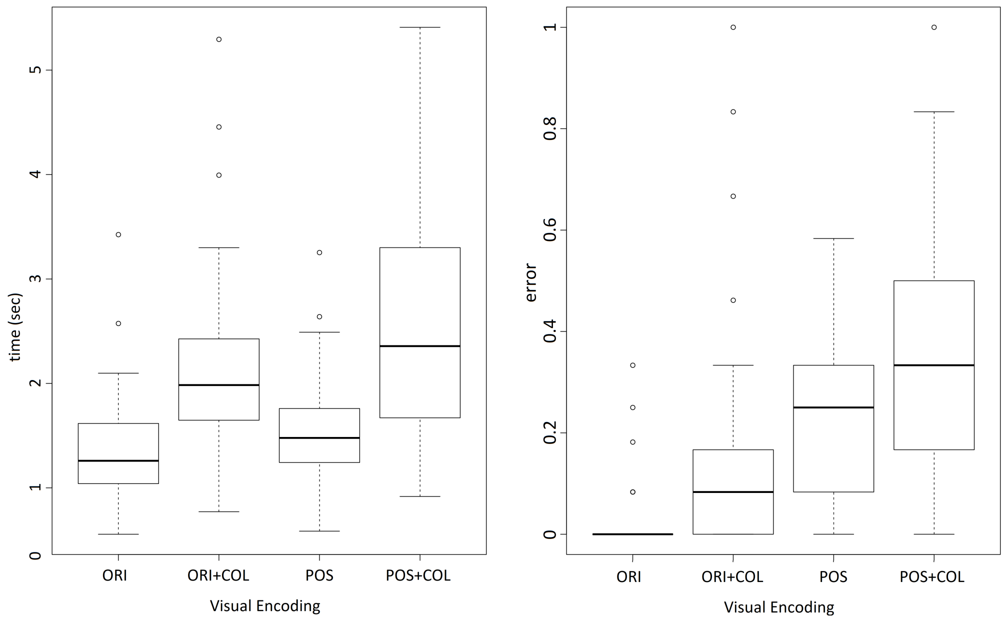


Figure 6.10: Distributions of response times (left) and error rates (right) for the first experiment.

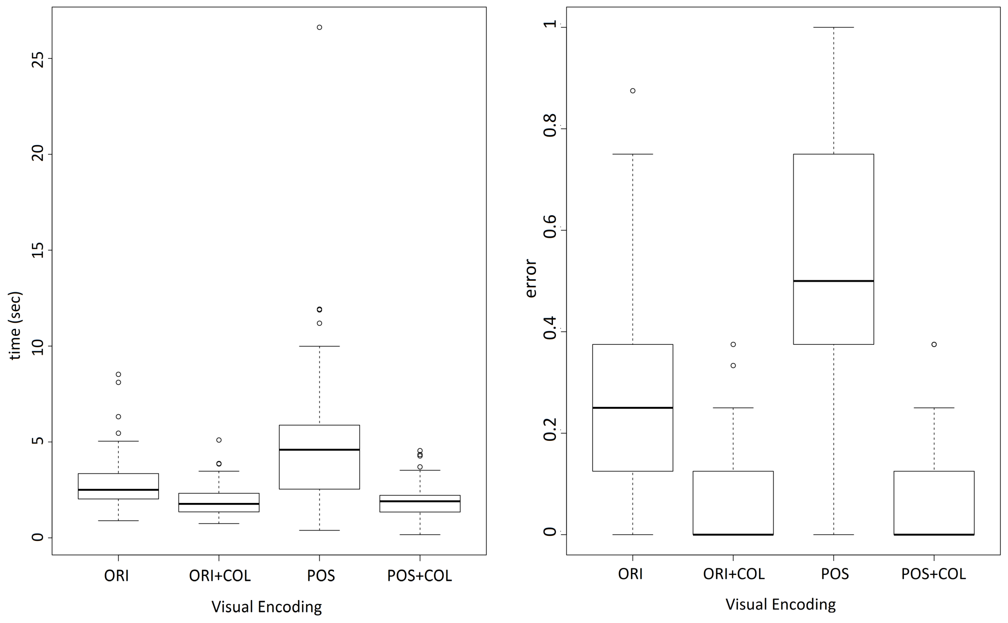


Figure 6.11: Distributions of response times (left) and error rates (right) for the second experiment.

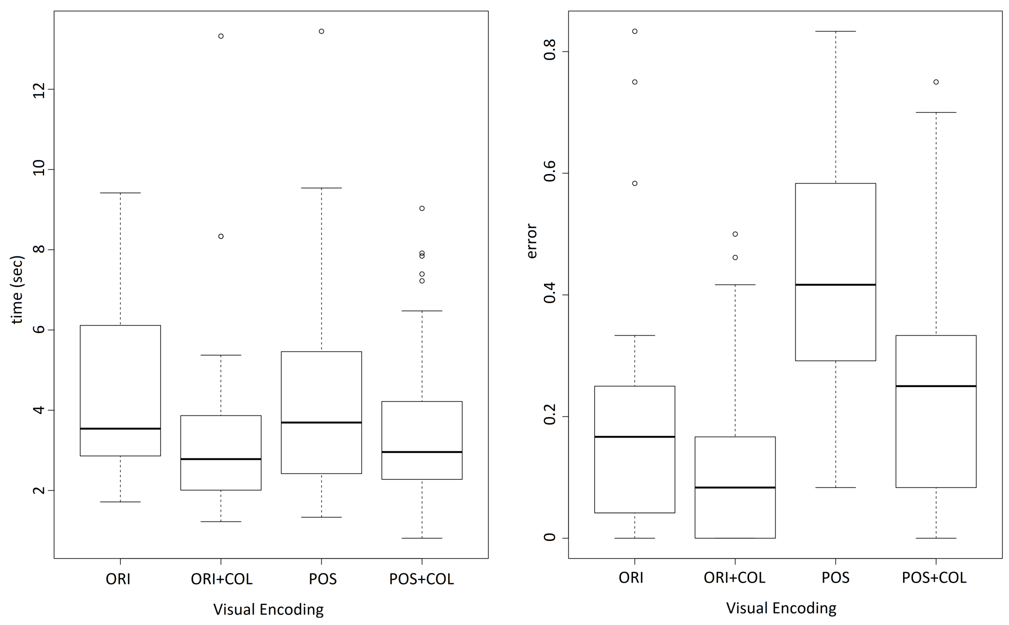


Figure 6.12: Distributions of response times (left) and error rates (right) for the third experiment.

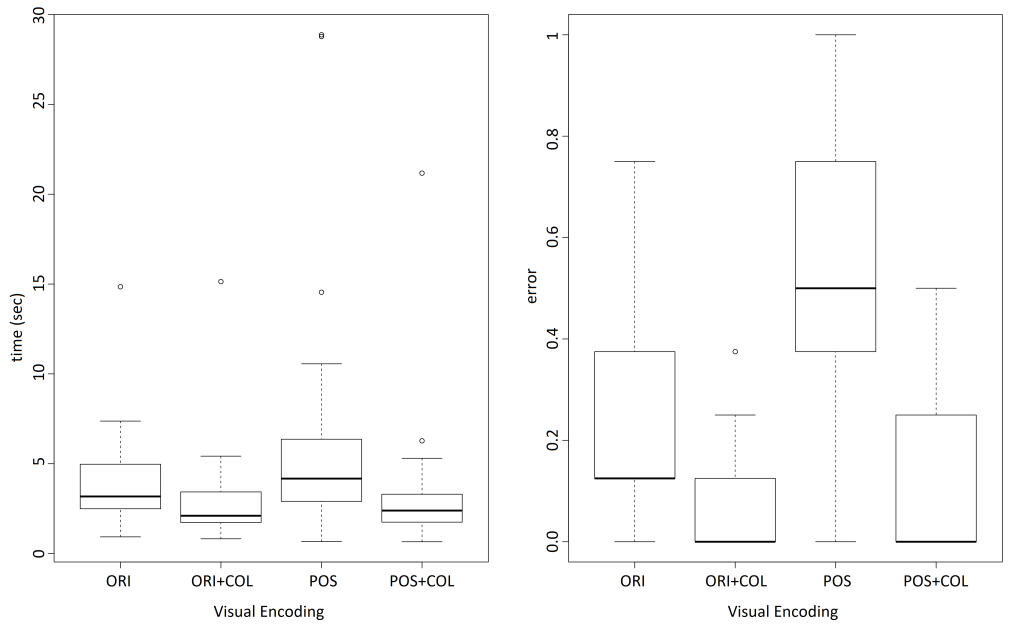


Figure 6.13: Distributions of response times (left) and error rates (right) for the fourth experiment.

the four distributions. We used a post-hoc approximative (Monte Carlo) Nemenyi-Damico-Wolfe-Dunn test to determine the pairwise significant differences. This post-hoc test includes a correction to compensate for the number of tests performed to reduce the likelihood of false positive results.

6.5.1 Task 1 - Cell with Most Marks

Our first experiment asked: *Click on the cell that has the most marks.* We collected the data from 42 participants in this experiment, but one participant was excluded as they did not complete the full experiment.

Demographics: The remaining 41 participants had an average age of 36.5 years (21 to 71 years). 39% of participants had no experience in using data visualisations, 15% had little experience (use visualisations once a year), 17% had some experience (once a month), 17% were experienced (once a week), and 12% were experts (use visualisations daily). To perform the experiment, 29% of the participants used a laptop and 71% used a desktop computer. None of the participants gave answers with an average error rate close to chance (error rate of 98%).

Counterbalancing: There was a good counterbalancing concerning the order of the techniques in the blocks, with 21 participants starting with orientation and 20 starting with position. Also the counterbalancing of colour was good with 20 participants starting the experiment with a block that had colour and 21 with a block that did not have colour, while 22 participants finished the experiment with a block that had colour and 19 with a block that did not have colour. Finally, 11 participants started both techniques with colour while 13 started both techniques without colour.

Results: Our results are shown in Figure 6.14. We find significant results for response time for this task ($\chi^2 = 75.117$, $df = 3$, $p < 0.05$). Our post-hoc analysis reveals that `ORI` was significantly faster than both `ORI+COL` and `POS+COL`. `POS` was significantly faster than `ORI+COL` and `POS+COL`. Also, we find significant results for error rate for this task ($\chi^2 = 71.032$, $df = 3$, $p < 0.05$). Our post-hoc analysis reveals that `ORI` produced significantly fewer errors than `POS`, `ORI+COL`, and `POS+COL`. `ORI+COL` produced significantly fewer errors than `POS` and `POS+COL`.

As `POS` was faster than `ORI+COL` but produced more errors, we performed a correlation analysis between response time and error. The analysis showed negative correlation between response time and error for both `POS` and `ORI+COL` distributions. This means that participants who answered faster tended to answer incorrectly. However, the error rate for `POS` was still low (near 20%).

Discussion: Our results are surprising as they provide evidence that colour significantly hurts the efficiency (response times) of participants completing the task in all cases (**reject H1(b)**). For this task, users were not required to differentiate between types of marks across cells. Thus, colour seems to have added distracting complexity to the representation. Orientation consistently

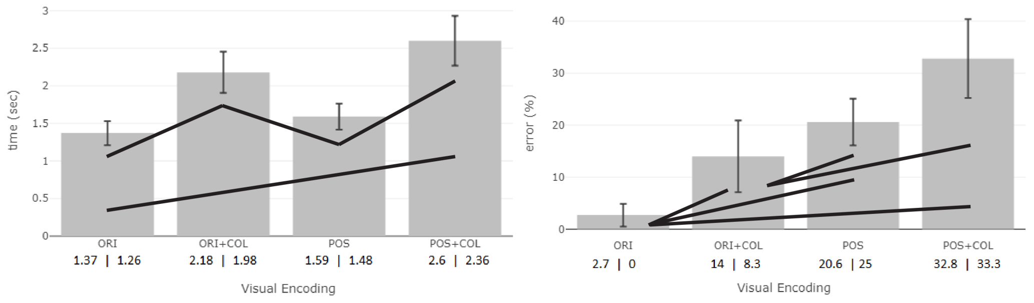


Figure 6.14: Average response times (top) and error rates (bottom) for experiment 1. The margin of error for 95% confidence intervals is shown in each box while the black lines between boxes indicate significance between visual encodings. Mean and median values are indicated below each bar.

produces significantly fewer errors than position with no significant difference in response time (**accept H1(a)**). It could be that the density and star-like patterns of the orientation support the task.

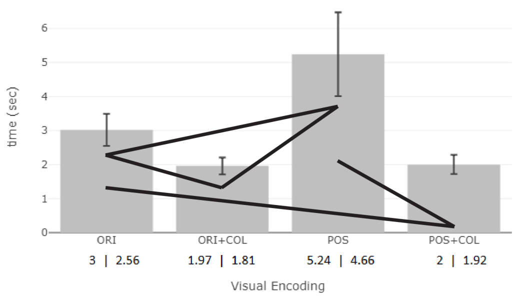
6.5.2 Task 2 - Most Frequent Mark

Our second experiment asked: *Which mark appears most often in the image?* In total, 50 participants performed experiment 2. From those 4 were excluded because of incomplete data and 1 because they used a phone. No participant had an average error rate close to chance (75%).

Demographics: In total, 45 participants were included in the analysis, with an average age of 35 years (24 to 54 years). In terms of experience, 47% of participants had no experience in using data visualisations, 4% had little experience (use visualisations once a year), 33% had some experience (once a month), 9% were experienced (once a week), and 7% were experts (use visualisations daily). For the devices used, 40% of the participants used a laptop and 60% used a desktop computer.

Counterbalancing: There was a good counterbalancing concerning the order of the techniques in the blocks, with 24 participants starting with orientation and 21 starting with position. Also the counterbalancing of colour was good with 24 participants starting the experiment with a block that had colour and 21 with a block that did not have colour, while 20 participants finished the experiment with a block that had colour and 25 with a block that did not have colour. Finally, 17 participants started both techniques with colour while 9 started both techniques without colour.

Results: Our results are shown in Figure 6.15. We find significant results for response time for this task ($\chi^2 = 77.948$, $df = 3$, $p < 0.05$). Our post-hoc analysis reveals that `ORI+COL` is significantly faster than `ORI` and `POS`. Similarly, `POS+COL` is significantly faster than `ORI` and `POS`.



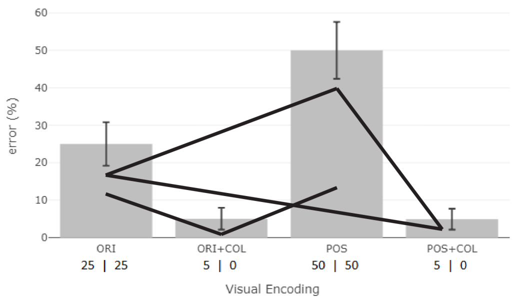


Figure 6.15: Bar charts of average response times (top) and error rates (bottom) for experiment 2. The margin of error for 95% confidence intervals is shown in each box while the black lines between boxes indicate significance between visual encodings. Mean and median values are indicated below each bar.

We also found that `ORI` is significantly faster than `POS`. We find significant results for error rate for this task ($\chi^2 = 96.746$, $df = 3$, $p < 0.05$). The pairs of results are exactly the same as those found in response time. Our post-hoc analysis reveals that `ORI+COL` is significantly more accurate than `ORI` and `POS`. Similarly, `POS+COL` is significantly more accurate than `ORI` and `POS`. We also found that `ORI` is significantly more accurate than `POS`.

Discussion: In general, the results show that for this task it is preferable to use colour for encoding the different types of edges (**accept H2(b)**). For this task, it could be that colour provides a method for quickly gauging the number of marks of each type at a glance, allowing for the quick identification of a more prevalent one. When colour is not used, orientation seems to perform better than position (**accept H2(b)**). It could be that since the marks for orientation have a more unique appearance, it is easier to judge the number of such marks in the matrix visualisation.

6.5.3 Task 3 - Most Frequent Mark Pair

Our third experiment asked: *Which combination of two marks appears more often in the image?* In total, 45 participants took part in experiment 3. When checking the data quality, 4 participants were excluded because of incomplete data and 1 did not answer at least half of the four gold standard questions. None of the remaining participants had an average error close to chance (84%).

Demographics: We analysed the data of the remaining 40 participants with an average age of 36.5 years (24 to 71 years). In terms of experience, 40% of participants had no experience in using data visualisations, 10% had little experience (use visualisations once a year), 7.5% had some experience (once a month), 37.5% were experienced (once a week), and 5% were experts (used visualisations daily). To perform the experiment, 50% of the participants used a laptop and 50% used a desktop computer.

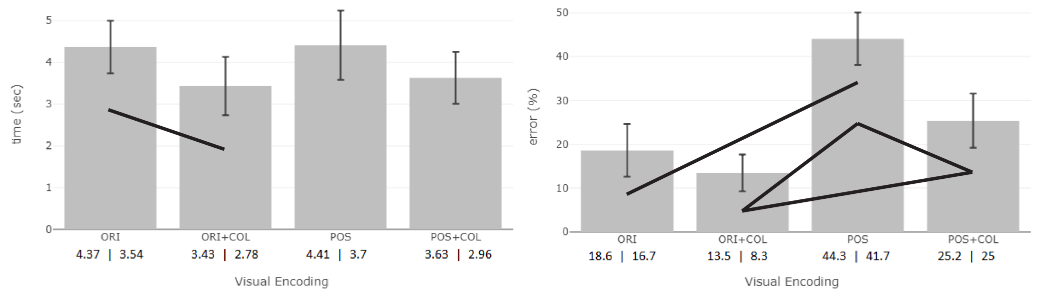


Figure 6.16: Bar charts of average response times (top) and error rates (bottom) for experiment 3. The margin of error for 95% confidence intervals is shown in each box while the black lines between boxes indicate significance between visual encodings. Mean and median values are indicated below each bar.

Counterbalancing: There was a good counterbalancing concerning the order of the techniques in the blocks, with 22 participants starting with orientation and 18 starting with position. Also the counterbalancing of colour was good with 21 participants starting the experiment with a block that had colour and 19 with a block that did not have colour, while 20 participants finished the experiment with a block that had colour and 20 with a block that did not have colour. Finally, 11 participants started both techniques with colour while 10 started both techniques without colour.

Results: Our results are shown in Figure 6.16. We find significant results for response time for this task ($\chi^2 = 14.91$, $df = 3$, $p < 0.05$). Our post-hoc analysis reveals only one significant difference with `ORI+COL` being significantly faster than `ORI`. In terms of error rate, we find significant differences for this task ($\chi^2 = 45.608$, $df = 3$, $p < 0.05$). `ORI+COL` has significantly fewer errors when compared to `POS+COL` and `POS`. `POS+COL` has significantly fewer errors than `POS`. Also, `ORI` has significantly fewer errors than `POS`.

Discussion: The error rates indicate that orientation has significantly fewer errors than position in many cases (**accept H3(a)**). When orientation is used, it either reduces the error rate with no difference in response times or it reduces response times with no difference in error rate (**accept H3(b)**).

In this task, we asked users to select the pair of marks that occurred most frequently together. Colour may have helped the participant judge the frequency of each mark type at a global level and select the pair that occurred more frequently. As the orientation marks are more unique in appearance, it could have helped the participants make this judgement.

6.5.4 Task 4 - Matrix with Most Cells of a Mark Type

Our fourth experiment asked: *Which matrix had the most cells of a particular mark type*. In total, 40 participants performed the experiment. From these participants, 3 failed more than half of the gold standard questions, 2 had incomplete data, and 2 had a variance of answers equal to zero – meaning they clicked on the same answer for the duration of the experiment. All these 7 participants were excluded. From the remaining 33 participants, no one had an average error rate close to chance (75%).

Demographics: We analysed the data from the remaining 33 participants with an average age of 39.5 (24 to 72 years). In terms of expertise, 37% of participants had no experience in using data visualisations, 21% had little experience (use visualisations once a year), 24% had some experience (once a month), 18% were experienced (once a week), and none was expert (daily). To perform the experiment, 58% of the participants used a laptop and 42% used a desktop computer.

Counterbalancing: There was a good counterbalancing concerning the order of the techniques in the blocks, with 15 participants starting with orientation and 18 starting with position. Also the counterbalancing of colour was good with 19 participants starting the experiment with a block that had colour and 14 with a block that did not have colour, while 18 participants finished the experiment with a block that had colour and 15 with a block that did not have colour. Finally, 10 participants started both techniques with colour while 9 started both techniques without colour.

Results: Figure 6.17 presents our results. We find significant results for response time for this task ($\chi^2 = 30.018$, $df = 3$, $p < 0.05$). Our post-hoc analysis reveals that `ORI+COL` was significantly faster than `ORI` and `POS`. We also find that `POS+COL` is significantly faster than `POS`. In terms of error rate, we find significant differences for this task ($\chi^2 = 71.181$, $df = 3$, $p < 0.05$). `ORI+COL` has significantly fewer errors than `ORI` and `POS`. `POS+COL` has significantly fewer errors than `ORI` and `POS`. Finally, `ORI` has significantly fewer errors than `POS`.

Discussion: From the results, it is clear that colour provides the most benefit and helps participants perform this task (**accept H4(b)**). When colour is not used, then orientation produces significantly lower error rates than position (**accept H4(b)**). As the task requires the participant to make a global assessment of the presence of a particular mark in the matrix, colour could have provided a means to quickly gauge the number of marks. If no colour is present, the uniqueness of the marks in the orientation condition might have helped participants make this judgement.

Considering *all* four experiments, `ORI` outperformed `POS` and colour was usually of benefit. Thus, we were able to accept all hypotheses except H1(b). In the following section, we discuss how we integrated the results of our experiments to extend the design of BayesPiles to support the exploration of dynamic Bayesian networks. Then we report on the feedback we received from three domain experts who qualitatively evaluated the tool using realistic data sets that contained multiple networks.

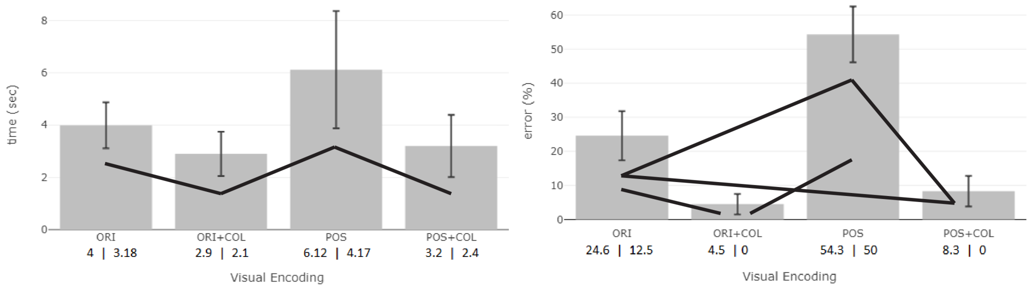
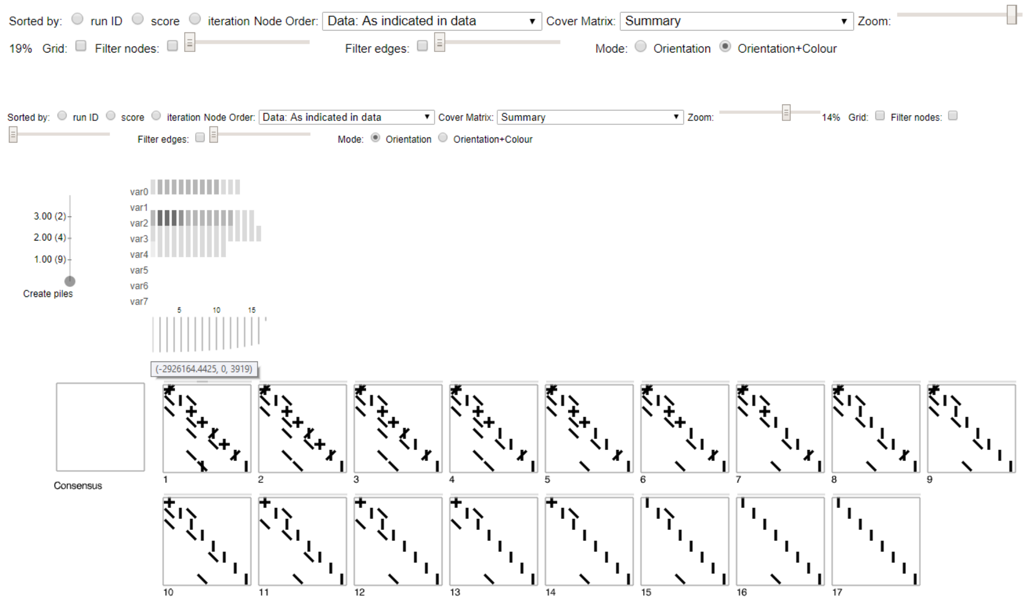


Figure 6.17: Bar charts of average response times (top) and error rates (bottom) for experiment 4. The margin of error for 95% confidence intervals is shown in each box while the black lines between boxes indicate significance between visual encodings. Mean and median values are indicated below each bar.


Sorted by: ○ run ID ○ score ○ iteration Node Order: Data: As indicated in data Cover Matrix: Summary Zoom: 19% Grid: ☐ Filter nodes: ☐ Filter edges: ☐ Mode: ○ Orientation ● Orientation+Colour

Sorted by: ○ run ID ○ score ○ iteration Node Order: Data: As indicated in data Cover Matrix: Summary Zoom: 14% Grid: ☐ Filter nodes: ☐ Filter edges: ☐ Mode: ● Orientation ○ Orientation+Colour

Figure 6.18: A collection of 17 dynamic Bayesian networks shown in BayesPiles.

6.6 Integration into BayesPiles and User-Centred Evaluation

Our experiments showed that orientation is the best visual encoding for our tasks. Colour was mostly beneficial but it could also harm the performance. Based on those results, we extended the design of BayesPiles to support the visualisation of DBNs using orientation (Figure 6.18). Users can also switch between a greyscale and a colour mode (`ORI` and `ORI+COL`) to explore their data and perform comparison tasks.

To receive feedback regarding this new extension of our tool and to also understand better how this approach could be improved, we invited three computational biologists to test our tool using a realistic data set. In our crowdsourcing experiments, we were interested in finding the best visual encoding for multivariate edges in matrices by participants drawn from the general public. In this second evaluation study, we were interested in collecting qualitative feedback from domain experts who could potentially use BayesPiles for inferring DBNs as part of their research. The first participant was an experienced computational biologist who had used BANJO and BayesPiles to infer Bayesian networks in the past. The second participant was a biology graduate who was interested in using BayesPiles to compare dynamic Bayesian networks inferred from electrophysiological recordings of zebra finch brain during playback of white noise and song stimuli. The third participant was a PhD student in neuroscience who was interested in learning how to use BayesPiles to visualise networks constructed from brain scans, using live imaging data of neural activity.

Instead of single networks or pairs of networks as used in the crowdsourcing experiment, BayesPiles can support the visualisation of hundreds of networks. In addition, with BayesPiles, users can interact with the interface and create summaries of networks as piles. Creating piles results in summary matrices with edges which are not only multivariate but also weighted. Every edge weight is calculated based on the average number of matrices that contain this edge in the pile. Using a combination of orientation and colour to encode edge types and opacity to encode weights in the summary matrix, was a design aspect that increased the complexity of the representation which was not tested in our previous experiments. It was interesting to see how this new variable, edge weight, would affect participants' performance. Also, it was interesting to see how challenging it would be for the domain experts to perform the same set of tasks, as those described in Section 6.2, but in the context of multiple networks and piles of networks. Those low-level tasks in the context of multiple networks are important for high-level tasks such as the exploration of the solution space and the construction of a consensus network.

6.6.1 Data

To get as close as we could to a real-world scenario and to effectively test our software extension in challenging settings, we first needed to generate realistic collections of networks, similar to the ones found in real biological data sets. However, we wanted control over the patterns found in those data sets, so that we could evaluate performance based on some ground truth. To create a realistic collection of DBNs we used a set of real songbird data and we ran BANJO to generate 17 DBNs. The results followed the "hill-climbing" pattern in which edges were added gradually (read from bottom-up and from right-to-left) improving the score (Figure 6.18). However, the resulting networks were relatively simple and with little diversity in edge types. To make the data set more challenging, we manually edited this network collection adding interactions that contained combinations of edge types (*i.e.* MLs) and single edges as distractors. We also inserted networks with edges appearing in known combinations of MLs. Those new networks were permutations of the top-scoring network and of the same density and size. In one of the networks, we introduced the most dense edge type in the collection that contained all 4 MLs (full star edge). Also, we edited the shape of the solution space creating three score levels and we introduced more edges of a known type in the networks that had the lowest score. The result of those interventions was a curated but realistic collection of 30 DBNs which incorporated known patterns within the data set (Figure 6.19). This data set would help us evaluate in a controlled way the implemented visual encodings when domain experts were performing tasks using the interface. In summary, we inserted the following known patterns in the data set:

1. A matrix with the densest cell in the collection that included all 4 MLs (useful for task T1).
2. A matrix in which a particular ML would appear more times than the rest of MLs (useful for task T2).
3. Networks of three distinct levels of score which could easily form three piles.
4. In one of the piles, we inserted matrices with cells that combined two MLs more frequently than other combinations (useful for task T3).
5. In the networks of one of the piles, we inserted edges of a particular ML type making it appear more often than in other piles (useful for task T4).

In the following, we describe how we collected feedback from our domain experts and we discuss the results.

6.6.2 Evaluation

First, we wanted to make sure that all participants knew how to use the interface of BayesPiles. Therefore, we provided a quick demonstration of the software to each of them. One of the participants had already used BayesPiles in the past but the other two did not know how to use it. Second, we created and distributed an evaluation form to all three participants asking them to load the data set of the 30 networks in BayesPiles and then perform the tasks using both orientation encodings (with and without colour). For each task, we already knew the correct answer so we could evaluate their performance in matters of accuracy. In the form, we included an index that showed which visual mark corresponds to which type of edge (*i.e.* ML). We also provided clarification about the tasks we wanted them to perform in the form of general guidelines. Finally, we asked participants to provide feedback regarding difficulties they might have encountered in performing those tasks as well as suggestions for further improvements.

1. For the first task, participants were asked to inspect all 30 networks and try to find which matrix contained the cell with the most MLs. They were asked to report on the matrix and the cell they found. Instead of looking in a single matrix, this time participants had to explore a collection of 30 networks to find the densest edge.

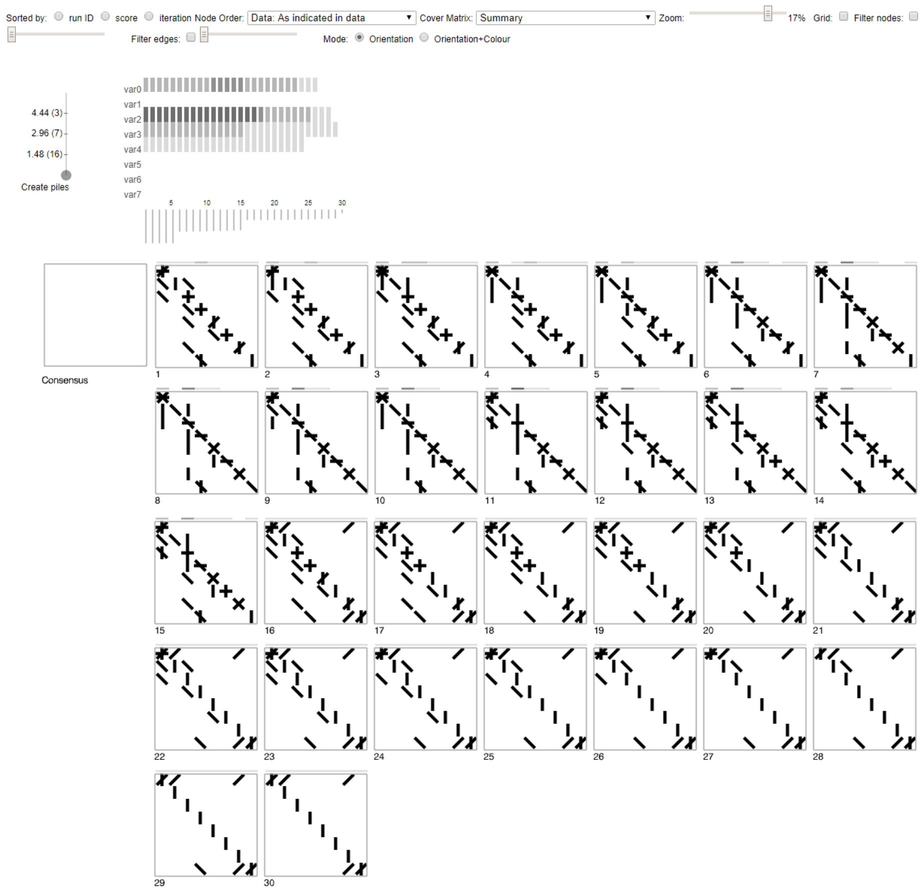
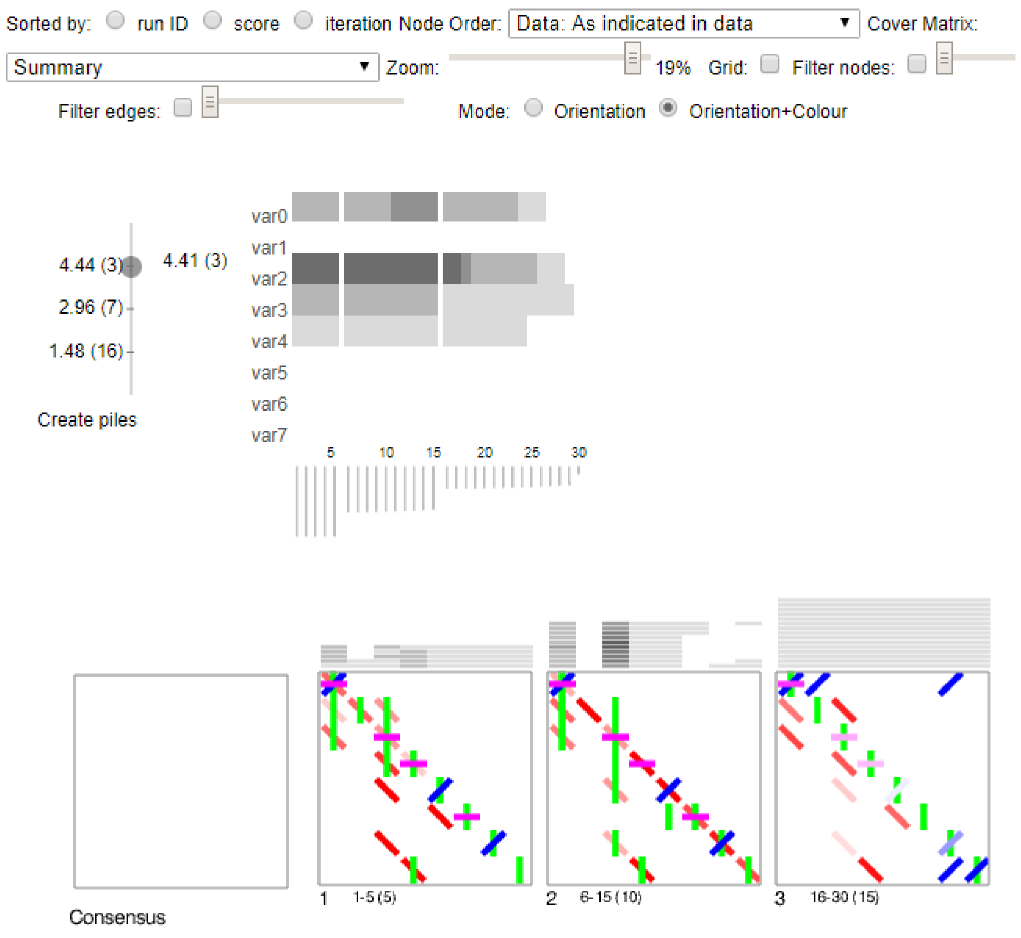

Node Order: Data: As indicated in data
Cover Matrix: Summary
Zoom: 17%
Mode: Orientation

Figure 6.19: A collection of 30 dynamic Bayesian networks shown in BayesPiles.

2. For the second task, we asked participants to find a specific matrix in the collection and then identify the ML that appeared more often. This task was very similar to T2 in the crowdsourcing study.

3. As part of the third task, participants were asked to use the slider and create 3 piles based on the different score levels (Figure 6.20). Then they were asked to inspect the first pile and find which combination of two MLs was the most common.

4. For the fourth task, participants were asked to identify which of the three piles contained the highest concentration of a particular ML. This was similar to task T4 in the crowdsourcing study but this time instead of a pair of single networks, participants had to compare edge type across three piles of weighted networks.


Sorted by: run ID score iteration Node Order: Data: As indicated in data Cover Matrix:

Summary Zoom: 19% Grid: Filter nodes:

Filter edges: Mode: Orientation [✓ Orientation+Colour]

Figure 6.20: Three piles of 30 networks in total shown using orientation and colour. To increasing the salience of the representation, we used red instead of yellow for encoding ML0.

6.6.3 Results

Although all domain experts found this new extension to the tool potentially useful for their research, we received mixed results regarding their performance in completing the tasks as shown in Table 6.2. Only one participant (*P1*) completed all tasks correctly. A second participant (*P2*) answered correctly in three of the tasks but gave the wrong answer for the third task (T3). The third participant (*P3*) gave three wrong answers, only performing task T2 correctly. Regarding colour, the feedback of most participants confirmed the results of our crowdsourcing experiments in which we found that colour is helpful for all tasks except task T1.

Participant (*P1*) found that orientation without colour was the preferred encoding for task T1, while for the rest of the tasks they preferred orientation with colour. Participant (*P2*) found that orientation without colour was the preferred encoding for tasks T1 and T3, while for tasks T2 and

Task	T1	T2	T3	T4
P1	CORRECT	CORRECT	CORRECT	CORRECT
P2	CORRECT	CORRECT	WRONG	CORRECT
P3	WRONG	CORRECT	WRONG	WRONG

Table 6.2: A summary of domain experts' results.

T4 they preferred orientation with colour. Finally, the third participant (*P3*) found that colour was the best encoding for all tasks. All participants answered correctly in task T2 and they all found colour particularly helpful, confirming the results of the crowdsourcing experiment for task T2. The task that most participants found difficult to perform and provided mostly wrong answers, except *P1* who answered correctly, was task T3. For this task, each of the three participants provided a different answer. Also while participants *P1* and *P3* agreed that colour helped them to perform the task, participant *P2* had a different opinion and preferred not to use colour. Regarding the last task T4, participants *P1* and *P2* answered correctly, while participant *P3* gave a wrong answer. They all agreed that colour helped them to perform the task. Participants provided anecdotal feedback regarding difficulties in performing the tasks. Their comments are discussed in Section 7.3, together with an interpretation of the results.

6.6.4 Qualitative Feedback

Participants were not only asked to comment on difficulties they have encountered performing the tasks but also to provide qualitative feedback about their overall experience and expectations using the tool. Participant *P1* commented that *"the tool is useful for combining multiple networks into one to find the most common edges and the way networks are displayed on BayesPiles facilitates comparison of different networks."* Participant *P2* commented: *"I think it will be useful to visually pull out which Markov lag is most common, by making piles of similar networks and inspecting colours. I think, in fact, it may be more useful at a slightly later analysis stage than the previous version of BayesPiles: I would want to work with instead of networks from a given search, a selection of "answers" and then look for commonalities regarding the system about them. However, an analysis at the search stage could still also be useful to identify what Markov lags to concentrate on"*.

Participants also shared their ideas for future improvements. Participant *P1* found it difficult to remember which visual mark corresponded to which ML and wrote *"it would have been easier to perform the tasks if there was a legend for Markov Lag value representations in the network."* Participant *P2* commented: *"I like the interface, and I particularly like the ability to switch back and forth between greyscale and colour. This enables doing different things with the different options when one is easier than the other"*. Participant *P3* commented the following: *"I wasn't sure whether I had clicked a network or not. I didn't really realise that hovering over the grey boxes above each network actually did something either. It would be useful to have tooltips on* *each of the controls to say a bit more about what they do. Lastly, maybe having a summary box with some basic stats about the networks and piles would help. Perhaps also when clicking on a network, a network specific summary box can appear.”*.

6.7 Summary of Contribution

In computational biology, modellers use heuristics that sample the search space of all possible networks to find those that best represent their data. They then either choose the best network or combine networks to construct a final consensus network of several multivariate networks. Visualisation plays an important role in network selection and consensus construction. For this step of the workflow, we collaborated with computational biologists to create encodings for dynamic Bayesian networks. Our encodings apply to multivariate networks represented as matrices when the multivariate data is associated with the edges. Our main contribution is a formal evaluation of encodings that can inform the design of visualisation tools for this domain and similar ones. We ran four crowdsourced experiments to evaluate these encodings. The performance of our visual encodings was task-dependent. For more local tasks, we found that colour hindered performance, but for all other tasks, it helped. For most tasks, orientation outperformed position.

Based on those results, we extended BayesPiles to also support dynamic Bayesian networks. MLs were encoded using orientation, and colour could be enabled and disabled in the interface on demand. We tested our implementation with a small group of 3 domain experts on a realistic data set of 30 networks. Although the results were encouraging, we identified that using opacity to encode edge weights in piles of networks that also use colour can be challenging. Therefore, there is a need for more formal evaluation studies that would test the performance of visual encodings that combine more than two visual variables. A discussion and interpretation of the results, as well as future work regarding all three challenges addressed in this thesis, are discussed in the following final chapter which also provides an overview of the three main contributions presented in this thesis.

7 Conclusions and Future Work

This thesis presented research motivated by the question: *"How can visualisation support modellers in their workflow to infer networks from biological data?"* First, to further investigate this question, it was important to understand the steps that modellers commonly follow when they analyse biological data to infer networks. To understand modellers' requirements, we collaborated closely with computational biologists who were interested in using visual analysis to improve their workflow when they were inferring Bayesian networks from biological data. We found that visualisation can play an important role in supporting modellers in three steps of this workflow. In Chapter 1, we laid out the research questions that emerged from each step and motivated our thesis:

- **Q1** *"How to provide visual support for the effective hierarchical clustering of many multi-dimensional variables?"*
- **Q2** *"How to support the visual analysis of heuristic search results, to infer representative models for biological systems?"*
- **Q3** *"How to effectively represent information related to the dynamics of biological systems, encoded in the edges of inferred networks?"*

To address each of these questions, we used the nested model methodology to understand the underlying domain problems, identify tasks and design effective visual encodings and interaction techniques. Two novel visualisation tools were designed and implemented to support the identified tasks in practice: MLCut and BayesPiles. Making those tools available to our biology collaborators, we managed to receive constructive feedback from case studies that involved the analysis of real biological data. To address Q3, we also conducted a formal crowd-sourcing study to identify perceptually effective visual encodings for multivariate edges in matrices. The results informed an extension to the design of BayesPiles. We presented our contributions for each of the three workflow steps, in separate chapters. In the following, we summarise our work and reflect on how our contributions answer the research questions derived from the three workflow steps. Also, we discuss limitations and future work for each of those contributions, providing guidelines for further research.

7.1 Contribution 1: Hierarchical Clustering with MLCut

As part of the first step in modellers' workflow, hierarchical clustering algorithms organise multidimensional biological data into groups, generating a dendrogram. The research question was: *Q1 "how to provide visual support for the effective hierarchical clustering of many multidimensional variables?"* To understand the challenges involved in answering this question, we held regular meetings with modellers and we performed a card sorting session, through which we characterised the domain problem (Section 4.1) and identified requirements (Section 4.2). We found that the operations that modellers perform to identify which branches correspond to clusters, require the concurrent exploration of the dendrogram in coordination with the original multidimensional data. Also, we found that it was important to enable modellers to cut the dendrogram at multiple levels. Our literature review (Section 3.1) showed that existing tools for hierarchical clustering only provided support for cutting the tree at a single level.

To help answer Q1, we developed MLCut (Figure 7.1), a novel visualisation tool which can represent potentially large multidimensional data sets as parallel coordinates and provides a second coordinated view in which the resulting dendrogram is displayed in a space-efficient radial layout. The modeller can interact with the dendrogram to select clusters by cutting the tree at different levels. In Chapter 4 (Section 4.3), we presented how the design of MLCut evolved through the development of two versions of a prototype. In an earlier version of the tool, the main focus was the design of an effective dendrogram representation. The tree unfolds in a radial layout, utilising the two-dimensional space and providing opportunities for interactive exploration aided by two dynamic sliders. To further support the hierarchical clustering of many multidimensional variables (Q1), in the final prototype of MLCut, we introduced a coordinated view of the original data linked to the dendrogram. Selections of multiple branches enable the comparison between clusters of records in the original multidimensional data. MLCut was evaluated by modellers who used it to detect clusters of variables in large sets of time-series gene expression data. The tool helped them identify distinct temporal patterns in those data sets and also enabled the comparison between those patterns. Overall, MLCut helped modellers select variables from complex data sets to become nodes in the inferred network model, successfully answering Q1.

MLCut has potential applications in many clustering challenges in high-dimensional molecular biology. It can be applied to any type of hierarchical clustering algorithms and could be extended to support hybrid methods for hierarchical clustering [45]. Therefore, one of the future challenges would be to identify requirements and prioritise technical specifications which would guide the further development of this approach to a more generic tool for hierarchical clustering. Also, it would be interesting to provide support for other unsupervised learning methods, such as density based clustering [107] and k-means [100]. Investigating consensus clustering approaches [110], would be another interesting extension. For example, results from different clustering algorithms could be combined and compared visually. Building consensus could improve variable selection through the mutual allocation of variables into clusters.

A complement to the hierarchical clustering step for variable selection in biological research

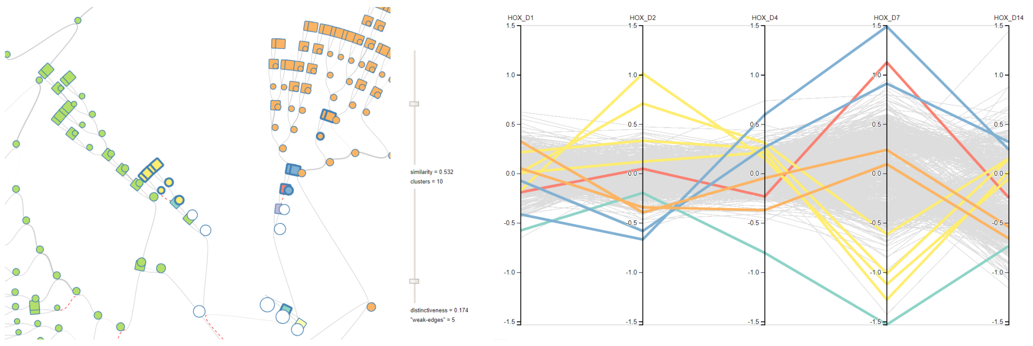


Figure 7.1: Semantic zooming in a large dendrogram for low-level exploration of individual variables and small clusters, using MLCut.

would be to help researchers evaluate their selected groups of variables by identifying them in overrepresented pathways found in previous studies. This is part of an analysis step called: over-representation, or enrichment analysis. In order to evaluate and further refine the selection of variables, it is important to identify overrepresented pathways, in which most (or all) of those variables are present. Some of the common tools for enrichment analysis include: DAVID [90], KeyPathwayMiner [8] and MeSH [173]. However, those tools mostly focus on the systematic integrative analysis of different data sets, which uses algorithms for detecting related pathways. Less attention is given to the use of visualisation for showing relationships between biomarkers and already known pathways [7]. Therefore, a future research objective could be to investigate opportunities for visualising the results of enrichment analysis and how those results are linked to the original data. As a future direction, it would be interesting to link the interface of MLCut with curated pathway repositories such as the Kyoto Encyclopedia of Genes and Genomes (KEGG) [99], the Pathway Commons [40], PANTHER [128] and Reactome [51].

Although the dendrogram representation of MLCut can scale well for thousands of variables, there are limitations in scalability regarding the number of clusters that it can visualise at the same time. This is mainly because clusters in the dendrogram and in the original data view are encoded using colour. Although the visual variable of colour is intuitive and can effectively represent up to approximately 15 clusters [133], when the number of clusters is larger then additional colours are similar to those already used. Therefore, as future work, it would be interesting to explore ways in which the tool could support the analysis of more clusters without repeating colours. For instance, branches already identified as clusters could be either aggregated or filtered from the display and branches of interest could be expanded. Alternatively, visual variables other than colour could be explored for encoding the different clusters in dendrograms. For example, different shapes or glyphs could be used to encode the leaves of the tree to help distinguish between clusters of variables that would otherwise get the same colour. Then, colour could be used as double encoding to provide a more salient representation. Combining visual variables generates a huge number of possible design alternatives. Therefore, it is important to formally investigate and evaluate combinations of visual variables that result in the best encodings for a number of common analysis tasks. Designers who deal with the visualisation of complex data sets need clear guidelines to avoid ineffective visual representations for their data.

The representation of the original data as parallel coordinates also has its limitations. A future challenge would be to deal with scalability issues for supporting data sets with more dimensions or time-points than those tested in the case studies (Section 4.4). Currently, MLCut carries the inherent limitations of the parallel coordinates technique which can represent approximately 15 dimensions in a satisfactory way. Representing hundreds of dimensions while limiting information loss is a common problem in data visualisation [116]. Moreover, when the dimensions in the data set do not follow a natural order, then the axes of the parallel coordinates need to be reordered appropriately to reveal interesting patterns. However, determining the optimal order of the axes is a difficult computational problem for which several automated methods have been proposed. Enabling automated axes reordering would be a useful extension to the tool. Edge bundling could be used to reduce clutter and occlusion in the parallel coordinates view when the data sets involve thousands of records. For such data sets, the edges of each cluster could be aggregated forming bundles. However, aggregation inevitably leads to information loss, and it becomes difficult to identify separate variables of interest within clusters. Therefore, our recommendation for user interface designers would be to provide a switch that could enable and disable edge bundling on demand.

Hierarchical clustering is a useful unsupervised method for analysing biological data. For large and complex data sets, there are ambiguities in the way that data records form clusters, which cannot be resolved automatically using a heuristic algorithm. Therefore, to answer Q1, we followed a synergistic approach for exploring clustering scenarios in the context of the original data records, the output of the hierarchical clustering algorithm and users' tacit knowledge. We developed MLCut, an interactive software tool that enables a visual exploration approach for performing hierarchical clustering analysis. Human intervention is not sufficient when it is limited in choosing a similarity threshold for cutting the dendrogram at a single level. With MLCut, the user can cut the dendrogram at multiple levels and see the effect in the original data. Selected clusters can be exported and used in other steps of the network inference workflow. There is still room for improvement, particularly regarding its interoperability with statistical packages for hierarchical clustering and its scalability in matters of the number of clusters and the number of dimensions that it can handle. However, our research has shown that this method provides more transparency and confidence in the process of assigning data records to clusters and can be applied successfully to multidimensional biological data.

7.2 Contribution 2: Inferring Networks with BayesPiles

As part of the second step in the modellers' workflow, heuristic search algorithms sample the search space of all possible networks based on a score that encapsulates their statistical fit to the data. The research question was: *Q2 "how to support the visual analysis of heuristic search results, to infer representative models for biological systems?"* To understand the challenges involved in answering this question, we applied the nested model methodology which helped to characterise the domain problem (Section 5.1) and derive common analysis tasks (Section 5.2). We found that the operations that modellers often perform involve the exploration, combination and comparison of potentially large collections of candidate networks produced by heuristic search algorithms. The purpose is to guide the heuristic search and decide on a method that determines a final consensus network. Thus, to answer Q2, we had to overcome the challenge of representing and comparing collections of potentially hundreds of networks.

Our literature review (Section 3.2.2) showed that exploring and comparing multiple networks is a challenging task for visualisation, especially when hundreds of networks are involved. We found that most of the existing visualisation tools can only handle a small number of networks. Moreover, previous research has shown that for some of our identified tasks, matrix representations perform better than node-link diagrams [9, 73]. Therefore, we decided to use matrices for representing our collections of Bayesian networks. Our design process led to the development of BayesPiles, which is inspired by MultiPiles – a dynamic network visualisation tool. In MultiPiles, networks are represented as juxtaposed matrices which can be organised in groups to form piles of superimposed networks.

BayesPiles was designed to answer Q2 supporting a list of tasks that modellers commonly perform when they infer networks from biological data. BayesPiles (Figure 7.2) supports the overview of all network scores providing a view to the shape of the solution space, two modes for representing directed networks as matrices, the ordering of networks within collections, the ordering of nodes in each network, the filtering of nodes/edges based on connectivity criteria, the exploration of outgoing edges for a particular node, a mechanism for the manual construction of a consensus network etc. In other words, BayesPiles provides extended functionality for supporting the visual analysis of heuristic search results, to infer Bayesian networks (Q2).

To assess how well the design of BayesPiles can support modellers in their workflow answering Q2, we evaluated the tool in three case studies that involved real biological data sets (Section 5.4). For each of those data sets, a computational biologist used BayesPiles to visually explore the heuristic search results, to take decisions about parameter settings of the heuristic search algorithms and to construct a final consensus network. They found that BayesPiles helped them gain new insights from their data and construct more representative networks, successfully answering Q2. We also received subjective feedback, which helped us identify opportunities for improvement.

There are several avenues for future work, including both technical extensions as well as conceptual features and questions. Future directions for further extending the capabilities of BayesPiles include support for a wider range of tasks (e.g. finding clusters), support for large networks (e.g. gene regulatory networks that may involve hundreds of nodes), as well as the implementation of a more efficient algorithm for improving performance and interactivity. Although

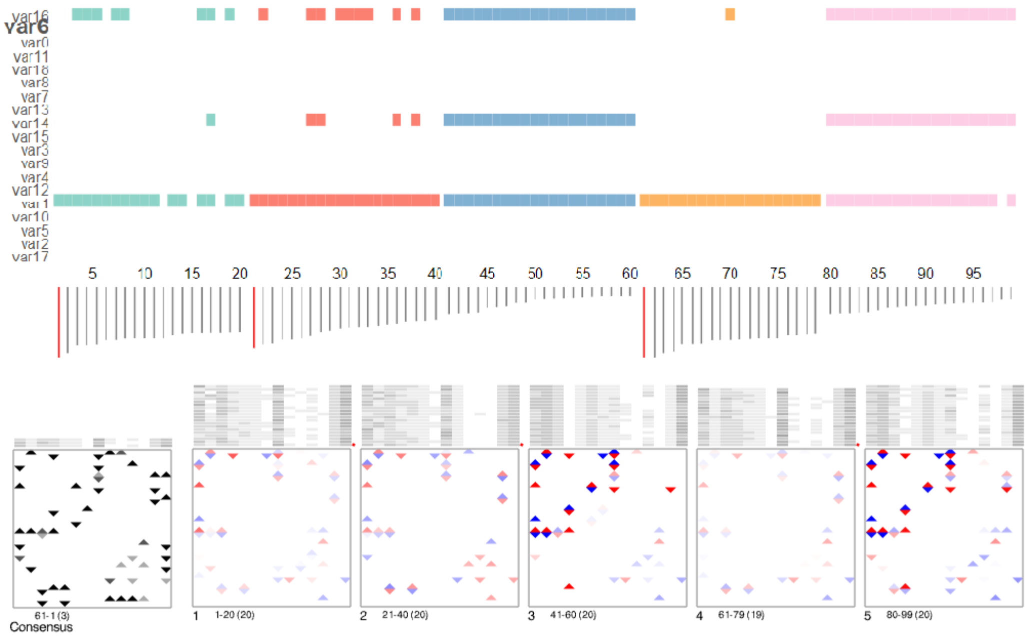


Figure 7.2: Exploring a data set of reordered matrices from five runs. At the top, an overview of all outgoing edges for node var6. At the bottom, the difference between the consensus and five piles of networks in diamond mode.

our decision to represent networks as matrices improved the scalability of the tool regarding the number of networks that it can handle, we found that it was not particularly effective when networks with more than 50 nodes were represented. In these larger networks, it was difficult to see details in the cells and zooming in was required. For example, it was hard to observe differences between networks in the diamond mode because of the small size of the triangles. Node filtering or clustering could be used to decrease the size of the networks, but this would lead to information loss.

Currently, BayesPiles only supports the search algorithms and the scoring metric found in BANJO. It would be useful to extend its scope by integrating other network inference methods (such as evolutionary algorithms) to make them more accessible to a wider community of analysts. A feature that users asked for but is not supported in BayesPiles is the ability to automatically create one pile for each run. Exporting the consensus network would be a small but useful feature. Currently, sorting the order of the networks in the collection is only based on three aspects (score, run ID, iteration). Networks of similar structure also have similar scores. Sorting by score and then piling networks of similar scores, enables the identification and grouping of similar structures even for collections that have more than 1000 graphs in total. However, in complex solution spaces, different networks may also have similar scores making the groups appear structurally heterogeneous. Therefore, it would be interesting to investigate unsupervised methods (clustering of networks by structure) in addition to the scoring from BANJO.

BayesPiles could also be extended to support community-based methods to infer consensus networks. Although the value of community-based methods to build consensus has been noted in the area of gene network inference [121], there is a lack of visualisation tools that can help modellers integrate results from different network inference methods. As future work, it would be interesting to develop visualisation methods that could enable modellers to explore, combine and compare results produced from different methods, to help achieve consensus. One of the bottlenecks in this process is that there is no unified strategy that modellers follow to normalise heterogeneous network results. For instance, some methods tend to favour sparse networks and penalise more dense ones to avoid overfitting. Also, comparing between scores calculated using different metrics is another normalisation problem. Overcoming these analysis obstacles will open the way for BayesPiles to also support community-based methods for network inference.

Adjacency matrices utilise the fact that networks can be represented by their connectivity matrix, with patterns corresponding to interesting subgraphs, such as cliques, bicliques and clusters, appearing along the diagonal of an appropriately ordered adjacency matrix [123]. However, there are many different ordering methods which can help identify interesting patterns in adjacency matrices [25]. To optimise node reordering for Bayesian network inference, first, it is important to identify interesting patterns which could be targeted by the reordering algorithm. For instance, after experimenting with different data sets, our modellers identified two interesting patterns in directed adjacency matrices, the *cycle* and the *combo* patterns. Every Bayesian network is a directed acyclic graph and therefore cycles are not permitted in single Bayesian networks. However, when many Bayesian networks are superimposed creating a pile, then a cycle could appear in the cover matrix. A cycle corresponds to a feedback loop, which is an important feature that most modellers would be interested to notice. Cycles would appear in BayesPiles as shown in Figure 7.3 (a). However, they are difficult to spot visually, especially when the nodes in the matrix are not ordered appropriately.

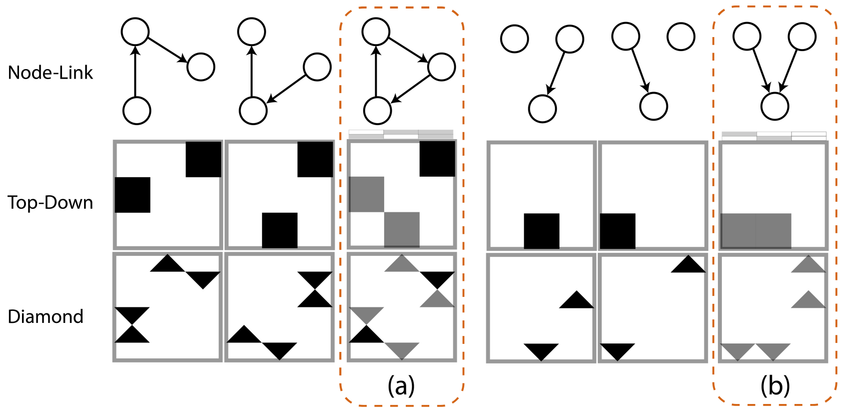


Figure 7.3: Two patterns that appear after combining Bayesian networks in BayesPiles: (a) the *cycle* pattern and (b) the *combo* pattern.

The second interesting pattern describes a combinatorial relationship (*i.e.* a *combo* pattern). Although networks with high node in-degrees are usually restricted by parameters in the heuristic search algorithm, nodes of unexpectedly high in-degrees can occur in the cover matrix when networks are piled. The combo pattern appears in our encodings as shown in Figure 7.3 (b). Both of those patterns can only appear in the summary matrix of a pile. Therefore, such patterns would most likely appear in a lower opacity and therefore, they will be harder to spot visually. A node reordering algorithm that could help identify such patterns would be useful for exploring Bayesian networks represented as matrices. For a more flexible node reordering mechanism, distinguishing between the different classes of nodes and edges could be supported. Also, there is a huge space for investigating novel visual encodings in matrices for the different classes of nodes and edges and for annotating specific network structures within the matrices, such as cycles.

Inferring Bayesian networks from biological data using heuristic search algorithms plays a pivotal role in understanding how important mechanisms in biological systems work. To help answer Q2, we developed BayesPiles, a visual analytics system inspired by MultiPiles and redesigned for Bayesian networks. BayesPiles allowed modellers to understand the shape of the solution space, to explore, compare and combine network structures, and to construct consensus networks. As Bayesian methods produce hundreds of scored directed networks, they are hard to explore without the aid of visualisation. Helping to answer Q2, BayesPiles enables the exploration, organisation and comparison of hundreds of scored directed networks from multiple heuristic search runs. It features two matrix-based representation modes for directed networks (top-down and diamond), a normalised histogram that shows the distribution of scores in the solution space, flexible network ordering based on run ID, iteration or score, node reordering, interactive comparison of networks across groups, support for the manual construction of a consensus network, interactive graph filtering mechanisms and a summary view of all outgoing edges for selected nodes.

Our work suggests that there is still room for improvement and that more visualisation approaches are required for tackling optimisation problems, in understanding the solution space, and how heuristic search works. However, our case studies have shown that we can successfully answer Q2 as our tool enables the visual analysis of heuristic search results and helps to infer representative Bayesian network models for biological systems.

7.3 Contribution 3: Matrix Representations for Networks with Multiple Edge Types

As part of the third step in modellers' workflow, heuristic search algorithms not only infer information about the structure of the network but also about its dynamics. The research question was: *Q3 "how to effectively represent information related to the dynamics of biological systems, encoded in the edges of inferred networks?"* We used the nested model methodology to characterise the domain problem (Section 6.1) and identify tasks (Section 6.2) that modellers perform when they explore the dynamics of their inferred networks. We discovered how important it is for modellers to be able to explore the heuristic search results and understand information about the dynamics of the interactions encoded in the edges of the inferred networks. We found that the tasks that modellers often perform include type lookup, type frequency, and comparison within the same as well as across networks. In particular, our modellers were interested in performing those tasks on collections of dynamic Bayesian networks. Such networks are similar to Bayesian networks shown in BayesPiles but with up to four different types of edges. To be able to answer Q3, we were interested in extending BayesPiles to provide support for dynamic Bayesian networks. However, to achieve that, we had to find out how to encode different edge types in networks represented as matrices.

Our literature review (Section 3.3.2) showed that representing multivariate networks with different types of edges is a challenging visualisation problem. Moreover, we were surprised not to find any formal study on visual encodings for multiple edge types in matrices. In our effort to identify potentially effective encodings for matrix cells with up to 4 different edge types, we explored the design space generating a series of sketches. Based on feedback from 5 visualisation experts, we identified four potentially effective visual encodings for representing multivariate networks as matrices. These encodings derived from combinations of the visual variables: orientation, position and colour. To test which encoding performs better for modellers' tasks, we conducted a formal user study using the crowd-sourcing platform of Amazon Mechanical Turk. We presented the results of our study in Chapter 6 (Section 6.5). In summary, we found that the encodings performed differently depending on the task. However, colour was found to help in all tasks except lookup tasks. Orientation outperformed position in all of our tasks. Based on our findings, we extended the design of BayesPiles to support the representation of dynamic Bayesian networks using orientation with or without colour. If the set of precise tasks is unknown, we would suggest an orientation encoding with colour (Figure 7.4).

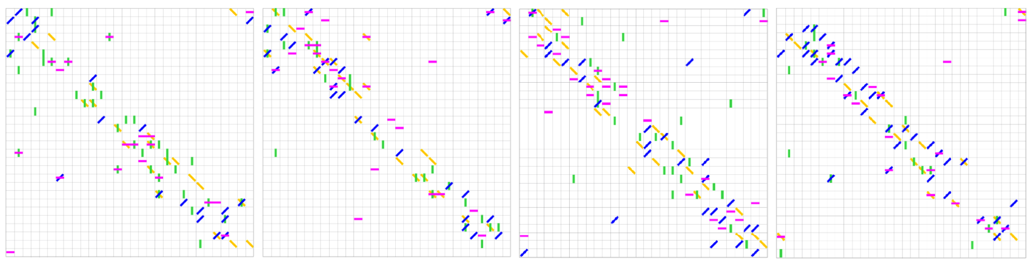


Figure 7.4: Orientation and colour used to encode multiple edge types in four dynamic Bayesian networks.

After we extended the design of BayesPiles to support dynamic Bayesian networks, we invited three computational biologists (*i.e.* modellers) to test it and provide feedback. We wanted to evaluate our progress in answering Q3 in realistic settings by testing how our encodings could help domain experts in the third step of their network inference workflow. We created a challenging data set of 30 networks arranged in three piles, and we asked the modellers to perform analysis tasks in the context of multiple dynamic Bayesian networks.

The tasks that domain experts performed were similar to those tested in our previous crowd-sourcing experiments. However, in our evaluation with domain experts, the context was wider and more demanding since it involved multiple networks and piles of networks with weighted edges. Also, participants were required to know how to use the interface, to interactively scroll through the networks and move sliders to zoom or form piles. The results showed that for task T1 most participants could visually inspect all 30 networks of the data set when they were juxtaposed as small multiples and find the denser cell in the whole collection. Participants that answered correctly preferred not to use colour for this task. *P2* commented that *"it was easier to see a dense block of lines without colour. With colour, the "star" of all blended into the vertical ML1 below it, and didn't look so different from networks 4-8 which didn't have the ML1 in the "star" but did in the cell directly below it."*

For task T2, the results were particularly encouraging as all participants could identify which ML was the most common in the matrix. Although they answered correctly, they did not find this an easy task. Participant *P1* commented that *"it would be helpful if there was some information about the number of edges at different Markov Lags, as it is easy to make mistakes if the edges have to be detected by eye, especially in larger networks"*. For piles of networks that demand an estimate of all ML types based on edge weights, shown using opacity, the task becomes particularly demanding. Participant *P2* commented that *"it was easier to compare different opacity levels in the grey scale, but the trade-off was that it was harder to distinguish between MLs when colour was not present.*

In task T3, which involved finding the most common combination of two MLs in a pile of networks, the answers varied a lot between participants. This was probably because the edges in the pile were weighted and participants had to estimate the overall weight of not only one but of a combination of two MLs, as those appeared in the whole matrix. The task was particularly challenging for our data set as the correct answer had many instances of the combination but in relatively low weights. Thus, most instances of the correct combination appeared "faint" in the pile. However, when added up they would result in the highest number of co-occurrences in the pile. On the other hand, other combinations would appear in the matrix with a higher opacity but in fewer cells. It was challenging for the participants to correctly estimate the overall weight of each combination of two MLs appearing in the matrix and then compare those estimations effectively.

Regarding the last task, most participants could identify correctly the pile that contained the highest concentration of a particular edge type. However, this task becomes much harder when participants are required to estimate the overall weight of an edge type (*i.e.* ML) in a pile of heterogeneous networks. When opacity is used with colour it tends to affect the distinctiveness of the colour and for edges that have a lower weight it often becomes difficult to recognise their colour. When colour is not used, it becomes even more challenging to both identify all instances of a particular type and also estimate their overall summary. Participant *P2* commented that *"I found this the hardest task. I'm not positive I'm right! It was easier to identify ML2 with colour, but then I was unsure if perhaps the shading might make there be less overall ML2, even though there was clearly more *kinds* of ML2."*.

The tasks where colour was beneficial (T2, T3, T4) asked participants to judge the prevalence of a mark globally in the visualisation. This result agrees with similar results on node-link diagrams [12]. One possible explanation for the negative impact of colour in T1 is that this task requires retrieving detailed information within a cell but marks do not need to be distinguished. In this case, the shape of the glyph (star-like shape in orientation) was the predominant factor while colour added unnecessary visual complexity.

Design guidelines: Given these results, our best encoding is `ORI+COL`. However, we suggest an option to turn colour on and off to support a wider range of tasks. Naturally, we suggest colours being discernible and of similar value as colours too light or too dark can easily blend with the white or black background respectively. Also, we recommend encodings that result in unified (connected) glyph designs, even when attributes are missing. Fragmented glyphs become harder to perceive as one, especially when they are located next to other glyphs and occupy a small space in the display. In addition, we recommend leaving margins between neighbouring glyphs or to use a grid, because it is easy to perceive adjacent glyphs as part of the same entity. We believe that `ORI` is scalable to around 12 values while remaining more performant than `POS` for both, coloured and non-coloured versions. We believe `POS` to not be very scalable, nor do we think that any of the other designs in Figure 6.2 is as scalable as `ORI`. For larger networks, we conjecture our results would be similar but the visual search would increase the difficulty of the task, but the relative differences between our techniques would stay the same.

Limitations and future work: The qualitative evaluation study with domain experts confirmed most of the results of our crowd-sourcing study and that these results were applicable to the context that involved multiple networks. However, our evaluation with domain experts had certain limitations. First, the sample size of 3 participants is too small to generalise the results, but the fact that all 3 participants were domain experts adds more value to the study. It would be preferable to evaluate the tool with a higher number of domain experts, but this is practically difficult to achieve for such a specialised study. Second, the data set used in this study was realistic because it originated from real data in which many distractors and ground truth were added manually. However, it would be more useful if domain experts could evaluate the tool using real data they owned, as was done previously for BayesPiles in the three case studies presented in Section 5.4. A third limitation of this evaluation is the number and the quality of tasks that domain experts performed to provide feedback about the tool. The list of tasks was constrained compared to the number and relevance of tasks that modellers perform in real case scenarios. Perhaps, it would be more interesting if modellers could analyse their data and provide feedback about their experience using the tool based on tasks interesting for their research. It would be interesting to characterise the space of possible tasks on multivariate networks from a variety of application domains. Eventually, a taxonomy that would map tasks with visual encodings would be extremely useful for designers of visualisation tools and interfaces.

Although the implementation of our encodings in BayesPiles received positive feedback, we identified several avenues for further improvement. For instance, while modellers' feedback mostly confirmed the results of the crowd-sourcing study, there were cases in which finding the correct answer in the context of piled networks was difficult. Participants that had less experience using the interface were more likely to get overwhelmed and make mistakes. As the requirement for visualising more complex data increases, there is a danger that visual interfaces will become too complicated for analysts to use. While addressing this danger, it was interesting to identify trade-offs between visual encodings when we presented users with more complex data. For instance, we found that while on the one hand, using colour helped participants to distinguish between edge types, on the other hand, it was making it harder to estimate and compare weights across different edge types. Using opacity and colour to encode weight and type in the same encoding seemed to be problematic. Therefore, in future work, it would be interesting to further explore the design space of encodings that combine multiple visual variables. As networks become increasingly multivariate, there is a need for more formal user studies which evaluate combinations of visual variables which form glyphs. Thus, understanding the principles for composing effective glyphs for multivariate data in the context of matrices, is one of the future challenges in the area of multivariate network visualisation. Also, more work is needed for understanding and evaluating the performance of approaches that combine two or more visual variables in the same encoding.

Also, it is important to test the scalability of orientation and colour when networks contain more than four edge types. We can assume that most users would be able to distinguish between 12 possible edge types (as many as the hours in the traditional clock). However, this encoding would require to visualise only half of the bar currently used (top half for twelve o' clock or bottom half for six o' clock). Another alternative would be to keep the current full length of the bar and only change the angle. Theoretically, this approach can scale more, but it is unrealistic to believe the users will be able to recognise the difference between a five-degree (5°) and a ten-degree (10°) angle. Also, colour cannot encode effectively more than 15 edge types at the same time [133]. It would be interesting to test the scalability of those approaches through formal evaluation studies similar to the one performed for comparing the position and orientation with/without colour encodings. Finally, it would be interesting to improve the interoperability of visualisation tools for multiple networks, such as BayesPiles, with graph database management systems. NoSQL databases such as *Neo4j* become increasingly popular because they can provide infrastructure for scalable, fast and easy access to voluminous, interconnected resources. Therefore, one of the future challenges is to facilitate communication between visualisation interfaces and resources stored in graph databases.

When the collected biological data contain multiple measurements over time, then heuristic search algorithms can infer information about the dynamics of the interactions. In those cases, the results of the search algorithms contain collections of networks with multiple types of edges, such as dynamic Bayesian networks. To help answer Q3, we were interested in extending BayesPiles to support dynamic Bayesian networks, but we realised that representing networks with multiple edge types is a challenging visualisation problem. Therefore, we conducted a formal experiment on visual encodings of multivariate networks when the multivariate data is associated with the edges. The tested encodings used the visual variables of orientation, position and colour. The results of the experiment informed the design of BayesPiles, and the extension was evaluated by three domain experts who used the tool in the context of multiple networks. Their feedback provided evidence that our encodings were helpful for the tested tasks. However, answering Q3 in the context of multiple networks requires more work in understanding how to combine three or more visual variables (e.g. orientation, colour and opacity) in the same encoding. Also, we found that the scalability of the proposed encodings requires further investigation and evaluation. Still, our work made significant progress in answering Q3 providing useful guidelines for designers who want to encode up to four edge types in matrix representations for networks.

7.4 Contribution Summary

In this thesis, we presented three main contributions each of which was undertaken and can be applied independently. However, in the wider scope of this thesis, all three contributions are complementary to each other, providing visual support to modellers who are interested in inferring networks from biological data. First, we presented a novel visualisation tool which can help modellers to explore their data and cut dendrograms in multiple levels. Second, we presented a novel visual analytics tool for exploring the solution space of heuristic search algorithms and inferring a consensus Bayesian network. Third, we presented a quantitative evaluation of effective visual encodings for multivariate networks represented as matrices, in which the multivariate data is associated with the edges. Our findings informed the design of one of our tools, supporting the analysis of dynamic Bayesian networks. This thesis demonstrates how our tools can enable human-machine intelligence to help modellers infer networks by effectively integrating their expertise and tacit knowledge with computer algorithms and interactive visualisations.

References

[1] A. Abdul-Rahman, J. Lein, K. Coles, E. Maguire, M. Meyer, M. Wynne, C. R. Johnson, A. Trefethen, and M. Chen. Rule-based visual mappings – with a case study on poetry visualization. *Computer Graphics Forum*, 32(3pt4):381–390, 2013. doi: 10.1111/cgf.12125.

[2] J. Abello, F. van Ham, and N. Krishnan. Ask-graphview: A large scale graph visualization system. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):669–676, 2006. doi: 10.1109/tvcg.2006.120.

[3] A. Abuthawabeh and D. Zeckzer. Immv: An interactive multi-matrix visualization for program comprehension. In *2013 First IEEE Working Conference on Software Visualization (VISSOFT)*, pages 1–4, Sept 2013. doi: 10.1109/VISSOFT.2013.6650549.

[4] A. Aderhold, V. A. Smith, and D. Husmeier. *Biological Network Inference at Multiple Scales: From Gene Regulation to Species Interactions*, pages 525–554. Wiley Online Library, 2015. ISBN 9781119078845. doi: 10.1002/9781119078845.ch27.

[5] F. Agostini, D. Cirillo, R. D. Ponti, and G. G. Tartaglia. Seamote: a method for high-throughput motif discovery in nucleic acid sequences. *BMC Genomics*, 15(1):925, Oct 2014. ISSN 1471-2164. doi: 10.1186/1471-2164-15-925.

[6] C. Ahlberg and B. Shneiderman. Visual information seeking: Tight coupling of dynamic query filters with starfield displays. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 313–317. ACM, 1994. doi: 10.1145/191666.191775.

[7] M. Albrecht, A. Kerren, K. Klein, O. Kohlbacher, P. Mutzel, P. Wolfgang, F. Schreiber, and M. Wybrow. *On Open Problems in Biological Network Visualization*, pages 256–267. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. ISBN 978-3-642-11805-0. doi: 10.1007/978-3-642-11805-0_25.

[8] N. Alcaraz, J. Pauling, R. Batra, E. Barbosa, A. Junge, A. G. L. Christensen, V. Azevedo, H. J Ditzel, and J. Baumbach. Keypathwayminer 4.0: Condition-specific pathway analysis

by combining multiple omics studies and networks with cytoscape. *BMC systems biology*, 8(1):99, 2014. doi: 10.1186/s12918-014-0099-x.

[9] B. Alper, B. Bach, N. H. Riche, T. Isenberg, and J. D. Fekete. Weighted graph comparison techniques for brain connectivity analysis. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 483–492, 2013. ISBN 978-1-4503-1899-0. doi: 10.1145/2470654.2470724.

[10] K. Andrews, M. Wohlfahrt, and G. Wurzinger. Visual graph comparison. In *2009 13th International Conference Information Visualisation*, pages 62–67, July 2009. doi: 10.1109/IV.2009.108.

[11] D. Archambault. Structural differences between two graphs through hierarchies. In *Proceedings of Graphics Interface 2009*, GI '09, pages 87–94, Toronto, Ont., Canada, Canada, 2009. Canadian Information Processing Society. ISBN 978-1-56881-470-4.

[12] D. Archambault, H. C. Purchase, and B. Pinaud. The readability of path-preserving clusterings of graphs. *Computer Graphics Forum*, 29(3):1173–1182. doi: 10.1111/j.1467-8659.2009.01683.x.

[13] D. Archambault, T. Munzner, and D. Auber. Grouseflocks: Steerable exploration of graph hierarchy space. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):900–913, 2008. doi: 10.1109/tvcg.2008.34.

[14] D. Archambault, H. C. Purchase, and B. Pinaud. *Difference Map Readability for Dynamic Graphs*, pages 50–61. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[15] J. Aurisano, K. Reda, A. Johnson, and J. Leigh. Bacterial gene neighborhood investigation environment: A large-scale genome visualization for big displays. In *2014 IEEE 4th Symposium on Large Data Analysis and Visualization (LDAV)*, pages 103–104, Nov 2014. doi: 10.1109/LDAV.2014.7013210.

[16] B. Bach, E. Pietriga, I. Liccardi, and G. Legostaev. Ontotrix: A hybrid visualization for populated ontologies. In *Proceedings of the 20th International Conference Companion on World Wide Web*, WWW '11, pages 177–180, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0637-9. doi: 10.1145/1963192.1963283.

[17] B. Bach, E. Pietriga, and J. D. Fekete. Visualizing dynamic networks with matrix cubes. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, pages 877–886, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2473-1. doi: 10.1145/2556288.2557010.

[18] B. Bach, N. H. Riche, T. Dwyer, T. Madhyastha, J. D. Fekete, and T. Grabowski. Small multipiles: Piling time to explore temporal patterns in dynamic networks. *Computer Graphics Forum*, 34(3):31–40, 2015. ISSN 1467-8659. doi: 10.1111/cgf.12615.

[19] B. Bach, P. Dragicevic, D. Archambault, C. Hurter, and S. Carpendale. A descriptive framework for temporal data visualizations based on generalized space-time cubes. *Computer Graphics Forum*, 36(6):36–61, 2017. doi: 10.1111/cgf.12804.

[20] Z. Bar-Joseph, D. K. Gifford, and T. S. Jaakkola. Fast optimal leaf ordering for hierarchical clustering. *Bioinformatics*, 17:S22–S29, 2001. doi: 10.1093/bioinformatics/17.suppl_1.S22.

[21] A. Barsky, T. Munzner, J. Gardy, and R. Kincaid. Cerebral: Visualizing multiple experimental conditions on a graph with biological context. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1253–1260, Nov 2008. ISSN 1077-2626. doi: 10.1109/TVCG.2008.117.

[22] F. Battke, S. Symons, and K. Nieselt. Mayday - integrative analytics for expression data. *BMC Bioinformatics*, 11(1):121, 2010. doi: 10.1186/1471-2105-11-121.

[23] F. Beck and S. Diehl. Visual comparison of software architectures. *Information Visualization*, 12(2):178–199, 2013. doi: 10.1177/1473871612455983.

[24] F. Beck, M. Burch, S. Diehl, and D. Weiskopf. A taxonomy and survey of dynamic graph visualization. *Computer Graphics Forum*, 2016. ISSN 1467-8659. doi: 10.1111/cgf.12791.

[25] M. Behrisch, B. Bach, M. Hund, M. Delz, L. von Ruden, J. D. Fekete, and T. Schreck. Magnostics: Image-based search of interesting matrix views for guided network exploration. *IEEE Transactions on Visualization and Computer Graphics*, PP(99):1–1, 2016. ISSN 1077-2626. doi: 10.1109/TVCG.2016.2598467.

[26] M. Behrisch, B. Bach, N. H. Riche, T. Schreck, and J. D. Fekete. Matrix Reordering Methods for Table and Network Visualization. *Computer Graphics Forum*, 2016. ISSN 1467-8659. doi: 10.1111/cgf.12935.

[27] R. Bellman and S. Dreyfus. Functional approximations and dynamic programming. *Mathematical Tables and Other Aids to Computation*, pages 247–251, 1959.

[28] J. Bertin. *Semiology of Graphics*. University of Wisconsin Press, 1983. ISBN 0299090604.

[29] J. Bertin. Matrix theory ofgraphics. *Information Design Journal*, 10(1):5–19, 2000.

[30] J. Beyer, M. Hadwiger, and H. Pfister. State-of-the-art in gpu-based large-scale volume visualization. *Computer Graphics Forum*, 34(8):13–37, 2015. doi: 10.1111/cgf.12605.

[31] A. Bezerianos, F. Chevalier, P. Dragicevic, N. Elmqvist, and J. D. Fekete. Graphdice: A system for exploring multivariate social networks. *Computer Graphics Forum*, 29(3):863–872. doi: 10.1111/j.1467-8659.2009.01687.x.

[32] C. Bielza and P. Larranaga. Bayesian networks in neuroscience: a survey. *Frontiers in Computational Neuroscience*, 8:131, 2014. ISSN 1662-5188. doi: 10.3389/fncom.2014.00131.

[33] R. Borgo, J. Kehrer, D. H. S. Chung, E. Maguire, R. S. Laramee, H. Hauser, M. Ward, and M. Chen. Glyph-based Visualization: Foundations, Design Guidelines, Techniques and Applications. In M. Sbert and L. Szirmay-Kalos, editors, *Eurographics 2013 - State of the Art Reports*. The Eurographics Association, 2013. doi: 10.2312/conf/EG2013/stars/039-063.

[34] M. Bostock, V. Ogievetsky, and J. Heer. D3 data-driven documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309, 2011. doi: 10.1109/TVCG.2011.185.

[35] U. Brandes and B. Nick. Asymmetric relations in longitudinal social networks. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2283–2290, Dec 2011. ISSN 1077-2626. doi: 10.1109/TVCG.2011.169.

[36] U. Brandes, B. Nick, B. Rockstroh, and A. Steffen. Gestaltlines. *Computer Graphics Forum*, 32(3pt2):171–180. doi: 10.1111/cgf.12104.

[37] M. Burch, N. Konevtsova, J. Heinrich, M. Hoeferlin, and D. Weiskopf. Evaluation of traditional, orthogonal, and radial tree diagrams by an eye tracking study. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2440–2448, 2011. doi: 10.1109/TVCG.2011.193.

[38] N. Cao, L. Lu, Y. R. Lin, F. Wang, and Z. Wen. Socialhelix: Visual analysis of sentiment divergence in social media. *Journal of Visualization*, 18(2):221–235, May 2015. ISSN 1343-8875. doi: 10.1007/s12650-014-0246-x.

[39] R. Cava, C. M. Dal Sasso Freitas, and M. Winckler. Clustervis: Visualizing nodes attributes in multivariate graphs. In *Proceedings of the Symposium on Applied Computing*, SAC '17, pages 174–179, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4486-9. doi: 10.1145/3019612.3019684.

[40] E. G. Cerami, B. E. Gross, E. Demir, I. Rodchenkov, O. Babur, N. Anwar, N. Schultz, G. D. Bader, and C. Sander. Pathway commons, a web resource for biological pathway data. *Nucleic Acids Research*, 39(SUPPL. 1), 1 2011. ISSN 0305-1048. doi: 10.1093/nar/gkq1039.

[41] S. C. Chan, L. Zhang, H. C. Wu, and K. M. Tsui. A maximum a posteriori probability and time-varying approach for inferring gene regulatory networks from time course gene microarray data. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 12(1):123–135, Jan 2015. ISSN 1545-5963. doi: 10.1109/TCBB.2014.2343951.

[42] C. Chang, B. Bach, T. Dwyer, and K. Marriott. Evaluating perceptually complementary views for network exploration tasks. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, pages 1397–1407, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4655-9. doi: 10.1145/3025453.3026024.

[43] J. Chen, A. M. MacEachren, and D. J. Peuquet. Constructing overview+ detail dendrogram-matrix views. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):889–896, 2009. doi: 10.1109/tvcg.2009.130.

[44] D. M. Chickering. *Learning Bayesian Networks is NP-Complete*, pages 121–130. Springer New York, New York, NY, 1996. ISBN 978-1-4612-2404-4. doi: 10.1007/978-1-4612-2404-4_12.

[45] H. Chipman and R. Tibshirani. Hybrid Hierarchical Clustering with Applications to Microarray Data. *Biostatistics*, 7(2):286–301, 2006. ISSN 1465-4644. doi: 10.1093/biostatistics/kxj007.

[46] S. S. Choi, S. H. Cha, and C. C. Tappert. A survey of binary similarity and distance measures. *Journal of Systemics, Cybernetics and Informatics*, 8(1):43–48, 2010. ISSN 1690-4532.

[47] W. S. Cleveland and R. McGill. Graphical perception and graphical methods for analyzing scientific data. *Science*, 229(4716):828–833, 1985. ISSN 0036-8075. doi: 10.1126/science.229.4716.828.

[48] G. F. Cooper. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial Intelligence*, 42(2):393 – 405, 1990. ISSN 0004-3702. doi: 10.1016/0004-3702(90)90060-D.

[49] M. Cossalter, O. J. Mengshoel, and T. Selker. Visualizing and understanding large-scale bayesian networks. In *Proceedings of the 17th AAAI Conference on Scalable Integration of Analytics and Visualization*, AAAIWS'11-17, pages 12–21. AAAI Press, 2011.

[50] P. Craig, A. Cannon, R. Kukla, and J. Kennedy. Matse: The microarray time-series explorer. In *Symposium on Biological Data Visualization (BioVis)*, pages 41–48. IEEE, 2012. doi: 10.1109/biovis.2012.6378591.

[51] D. Croft, G. O'Kelly, G. Wu, R. Haw, M. Gillespie, L. Matthews, M. Caudy, P. Garapati, G. Gopinath, B. Jassal, et al. Reactome: a database of reactions, pathways and biological processes. *Nucleic Acids Research*, 39(suppl_1):D691–D697, 11 2010. ISSN 0305-1048. doi: 10.1093/nar/gkq1018.

[52] I. W. Davis, C. Benninger, P. N. Benfey, T. Elich, and J. L. Heazlewood. Powrs: Position-sensitive motif discovery. *PloS one*, 7(7):e40373, 2012.

[53] D. De Maeyer, B. Weytjens, J. Renkens, L. De Raedt, and K. Marchal. Phenetic: Network-based interpretation of molecular profiling data. *Nucleic Acids Research*, 43(W1):W244–W250, 04 2015. ISSN 0305-1048. doi: 10.1093/nar/gkv347.

[54] P. D'haeseleer, S. Liang, and R. Somogyi. Genetic network inference: from co-expression clustering to reverse engineering. *Bioinformatics*, 16(8):707–726, 2000. doi: 10.1093/bioinformatics/16.8.707.

[55] S. Diehl and A. C. Telea. Multivariate networks in software engineering. In *Multivariate Network Visualization*, pages 13–36. Springer, 2014. ISBN 978-3-319-06792-6. doi: 10.1007/978-3-319-06793-3.

[56] E. S. Dimitrova, M. P. V. Licona, J. McGee, and R. Laubenbacher. Discretization of time series data. *Journal of Computational Biology*, 17(6):853–868, 2010. doi: 10.1089/cmb.2008.0023.

[57] C. Ding and H. Peng. Minimum redundancy feature selection from microarray gene expression data. *Journal of Bioinformatics and Computational Biology*, 03(02):185–205, 2005. doi: 10.1142/S0219720005001004.

[58] K. Dinkla, M. El-Kebir, C. I. Bucur, M. Siderius, M. J. Smit, M. A. Westenberg, and G. W. Klau. examine: Exploring annotated modules in networks. *BMC Bioinformatics*, 15(1): 201, 2014. ISSN 1471-2105. doi: 10.1186/1471-2105-15-201.

[59] S. I. O. Donoghue, A. Gavin, N. Gehlenborg, D. S. Goodsell, J. Hériché, C. B. Nielsen, C. North, A. J. Olson, J. B. Procter, D. W. Shattuck, T. Walter, and B. Wong. Visualizing Biological Data — Now and in the Future. *Nature Publishing Group*, 7(3s):S2–S4, 2010. ISSN 1548-7091. doi: 10.1038/nmeth0310-S2.

[60] D. Dotan-Cohen, A. Melkman, and S. Kasif. Hierarchical Tree Snipping: Clustering Guided by Prior Knowledge. *Bioinformatics*, 23(24):3335–3342, 2007. ISSN 13674803. doi: 10.1093/bioinformatics/btm526.

[61] S. H. C. Du Toit, A. G. W. Steyn, and R. H. Stumpf. *Graphical Exploratory Data Analysis*. Springer-Verlag, Berlin, Heidelberg, 1986. ISBN 0-387-96313-8.

[62] C. Dunne and B. Shneiderman. Motif simplification: Improving network visualization readability with fan, connector, and clique glyphs. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 3247–3256, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1899-0. doi: 10.1145/2470654.2466444.

[63] C. Dunne, N. H. Riche, B. Lee, R. Metoyer, and G. Robertson. Graphtrail: Analyzing large multivariate, heterogeneous networks while supporting exploration history. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 1663–1672, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1015-4. doi: 10.1145/2207676.2208293.

[64] R. Edgar, M. Domrachev, and A. E. Lash. Gene Expression Omnibus: NCBI Gene Expression and Hybridization Array Data Repository. *Nucleic Acids Research*, 30(1):207–210, 2002. doi: 10.1093/nar/30.1.207.

[65] J. Ellson, E. Gansner, L. Koutsofios, S. C. North, and G. Woodhull. *Graphviz— Open Source Graph Drawing Tools*, pages 483–484. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002. ISBN 978-3-540-45848-7. doi: 10.1007/3-540-45848-4_57.

[66] J. Ernst and Z. Bar-Joseph. Stem: a tool for the analysis of short time series gene expression data. *BMC Bioinformatics*, 7(1):1, 2006. doi: 10.1186/1471-2105-7-191.

[67] J. A. Ferstay, C. B. Nielsen, and T. Munzner. Variant view: Visualizing sequence variants in their gene context. *IEEE Transactions on Visualization and Computer Graphics*, 19(12): 2546–2555, Dec 2013. ISSN 1077-2626. doi: 10.1109/TVCG.2013.214.

[68] M. Freire, C. Plaisant, B. Shneiderman, and J. Golbeck. Manynets: An interface for multiple network analysis and visualization. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 213–222, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-929-9. doi: 10.1145/1753326.1753358.

[69] N. Friedman and title = Discretizing Continuous Attributes While Learning Bayesian Networks booktitle = Proceedings of the Thirteenth International Conference on International

Conference on Machine Learning series = ICML'96 year = 1996 isbn = 1-55860-419-7 location = Bari, Italy pages = 157–165 numpages = 9 acmid = 3091716 publisher = Morgan Kaufmann Publishers Inc. address = San Francisco, CA, USA Goldszmidt, M.

[70] J. Fuchs, F. Fischer, F. Mansmann, E. Bertini, and P. Isenberg. Evaluation of alternative glyph designs for time series data in a small multiple setting. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 3237–3246, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1899-0. doi: 10.1145/2470654.2466443.

[71] L. I. Furlong. Human diseases through the lens of network biology. *Trends in Genetics*, 29 (3):150 – 159, 2013. ISSN 0168-9525. doi: 10.1016/j.tig.2012.11.004.

[72] N. Gehlenborg, S. I O'Donoghue, N. S. Baliga, A. Goesmann, M. A. Hibbs, H. Kitano, O. Kohlbacher, H. Neuweger, R. Schneider, D. Tenenbaum, et al. Visualization of omics data for systems biology. *Nature methods*, 7:S56–S68, 2010. doi: 10.1038/nmeth.1436.

[73] M. Ghoniem, J. D. Fekete, and P. Castagliola. A comparison of the readability of graphs using node-link and matrix-based representations. In *IEEE Symposium on Information Visualization*, pages 17–24, 2004. doi: 10.1109/INFVIS.2004.1.

[74] M. Ghoniem, F. Mcgee, G. Melançon, B. Otjacques, and B. Pinaud. The state of the art in multilayer network visualization. *arXiv preprint arXiv:1902.06815*, 2019.

[75] M. Gleicher, D. Albers, R. Walker, I. Jusufi, C. D. Hansen, and J. C. Roberts. Visual comparison for information visualization. *Information Visualization*, 10(4):289–309, October 2011. ISSN 1473-8716. doi: 10.1177/1473871611416549.

[76] M. Glueck, M. P. Naeini, F. Doshi-Velez, F. Chevalier, A. Khan, D. Wigdor, and M. Brudno. Phenolines: Phenotype comparison visualizations for disease subtyping via topic models. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):371–381, Jan 2018. ISSN 1077-2626. doi: 10.1109/TVCG.2017.2745118.

[77] C. Görg, M. Pohl, E. Qeli, and K. Xu. Visual representations. In *Human-Centered Visualization Environments: GI-Dagstuhl Research Seminar, Dagstuhl Castle, Germany, March 5-8, 2006, Revised Lectures*, pages 163–230. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. ISBN 978-3-540-71949-6. doi: 10.1007/978-3-540-71949-6_4.

[78] M. Graham, J. Kennedy, and D. Benyon. Towards a methodology for developing visualizations. *International Journal of Human-Computer Studies*, 53(5):789–807, November 2000. ISSN 1071-5819. doi: 10.1006/ijhc.2000.0415.

[79] S. Gratzl, A. Lex, N. Gehlenborg, H. Pfister, and M. Streit. Lineup: Visual analysis of multi-attribute rankings. *IEEE Transactions on Visualization and Computer Graphics*, 19 (12):2277–2286, Dec 2013. ISSN 1077-2626. doi: 10.1109/TVCG.2013.173.

[80] O. Guitart-Pla, M. Kustagi, F. Rügheimer, A. Califano, and B. Schwikowski. The cyni framework for network inference in cytoscape. *Bioinformatics*, 31(9):1499–1501, 2015. doi: 10.1093/bioinformatics/btu812.

[81] C. A. Hackett, K. McLean, and G. J. Bryan. Linkage Analysis and QTL Mapping Using SNP Dosage Data in a Tetraploid Potato Mapping Population. *PLoS ONE*, 8(5):1–21, 2013. doi: 10.1371/journal.pone.0063939.

[82] C. A. Hackett, B. Boskamp, A. Vogogias, I. Milne, and K. F. Preedy. TetraploidSNPMap: Software for Linkage Analysis and QTL Mapping in Autotetraploid Populations Using SNP Dosage Data. *Journal of Heredity*, 108(4):438–442, 03 2017. ISSN 0022-1503. doi: 10.1093/jhered/esx022.

[83] A. Hagberg, P. Swart, and D. S Chult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[84] M. Harrower and C. A. Brewer. Colorbrewer. org: an online tool for selecting colour schemes for maps. *The Cartographic Journal*, 40(1):27–37, 2003. doi: 10.1179/000870403235002042.

[85] M. Hascoët and P. Dragicevic. Interactive graph matching and visual comparison of graphs and clustered graphs. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, AVI '12, pages 522–529, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1287-5. doi: 10.1145/2254556.2254654.

[86] M. Hecker, S. Lambeck, S. Toepfer, E. van Someren, and R. Guthke. Gene regulatory network inference: Data integration in dynamic models - a review. *Biosystems*, 96(1):86 – 103, 2009. ISSN 0303-2647. doi: 10.1016/j.biosystems.2008.12.004.

[87] D. Heckerman, D. Geiger, and D. M. Chickering. Learning bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20(3):197–243, 1995. ISSN 1573-0565. doi: 10.1007/BF00994016.

[88] S. M. Hill, L. M. Heiser, T. Cokelaer, M. Unger, N. K. Nesser, D. E. Carlin, Y. Zhang, A. Sokolov, E. O. Paull, C. K. Wong, et al. Inferring causal molecular networks: Empirical assessment through a community-based effort. *Nature methods*, 13(4):310–318, 2016. doi: doi:10.1038/nmeth.3773.

[89] H. Hochheiser and B. Shneiderman. Interactive exploration of time series data. In *The Craft of Information Visualization*, Interactive Technologies, pages 313 – 315. Morgan Kaufmann, San Francisco, 2003. ISBN 978-1-55860-915-0. doi: 10.1016/B978-155860915-0/50039-1.

[90] D. W. Huang, B. T. Sherman, and R. A. Lempicki. Systematic and integrative analysis of large gene lists using david bioinformatics resources. *Nature protocols*, 4(1):44–57, 2008. doi: 10.1038/nprot.2008.211.

[91] D. Husmeier. Sensitivity and Specificity of Inferring Genetic Regulatory Interactions from Microarray Experiments with Dynamic Bayesian Networks. *Bioinformatics*, 19(17):2271–2282, 2003. ISSN 13674803. doi: 10.1093/bioinformatics/btg313.

[92] J. F. Im, M. J. McGuffin, and R. Leung. Gplom: The generalized plot matrix for visualizing multidimensional multivariate data. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2606–2614, Dec 2013. ISSN 1077-2626. doi: 10.1109/TVCG.2013.160.

[93] A. Inselberg and B. Dimsdale. Parallel coordinates. In *Human-Machine Interactive Systems*, pages 199–233. Springer, 1991. doi: 10.1007/978-1-4684-5883-1_9.

[94] G. Jäger, F. Battke, and K. Nieselt. Tiala time series alignment analysis. In *2011 IEEE Symposium on Biological Data Visualization (BioVis).*, pages 55–61, Oct 2011. doi: 10.1109/BioVis.2011.6094048.

[95] C. V. Jones. *Visualization and Optimization*, volume 6. Springer Science & Business Media, 2013. ISBN 978-1-4613-6848-9. doi: 10.1007/978-1-4615-4121-9.

[96] I. Jusufi. *Multivariate Networks: Visualization and Interaction Techniques*. PhD thesis, 2013.

[97] N. Kadaba, P. Irani, and J. Leboe. Visualizing causal semantics using animations. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1254–1261, Nov 2007. ISSN 1077-2626. doi: 10.1109/TVCG.2007.70528.

[98] M. A. Kallio, J. T. Tuimala, T. Hupponen, P. Klemelä, M. Gentile, I. Scheinin, M. Koski, J. Käki, and E. I. Korpelainen. Chipster: User-friendly analysis software for microarray and other high-throughput data. *BMC Genomics*, 12(1):1–14, 2011. ISSN 1471-2164. doi: 10.1186/1471-2164-12-507.

[99] M. Kanehisa and S. Goto. KEGG: Kyoto Encyclopedia of Genes and Genomes. *Nucleic Acids Research*, 28(1):27–30, 01 2000. ISSN 0305-1048. doi: 10.1093/nar/28.1.27.

[100] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(7):881–892, July 2002. ISSN 0162-8828. doi: 10.1109/TPAMI.2002.1017616.

[101] G. Karypis and V. Kumar. Chameleon: Hierarchical Clustering Using Dynamic Modeling. *Computer*, 32(8):68–75, 1999. ISSN 00189162. doi: 10.1109/2.781637.

[102] D. Keim, G. Andrienko, J. D. Fekete, C. Görg, J. Kohlhammer, and G. Melançon. *Visual Analytics: Definition, Process, and Challenges*. Springer-Verlag, Berlin, Heidelberg, 2008. ISBN 978-3-540-70955-8. doi: 10.1007/978-3-540-70956-5_7.

[103] J. B. Kennedy, K. J. Mitchell, and P. J. Barclay. A framework for information visualisation. *SIGMOD Rec.*, 25(4):30–34, December 1996. ISSN 0163-5808. doi: 10.1145/245882.245895.

[104] S. Y. Kim, S. Imoto, and S. Miyano. Inferring gene networks from time series microarray data using dynamic Bayesian networks. *Briefings in Bioinformatics*, 4(3):228–235, 09 2003. ISSN 1477-4054. doi: 10.1093/bib/4.3.228.

[105] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press, 2009. ISBN 0262013193, 9780262013192.

[106] A. Koussounadis, S. P. Langdon, D. J. Harrison, and V. A. Smith. Chemotherapy-induced dynamic gene expression changes in vivo are prognostic in ovarian cancer. *British journal of cancer*, 110(12):2975–2984, 2014. doi: 10.1038/bjc.2014.258.

[107] H. P. Kriegel and M. Pfeifle. Density-based clustering of uncertain data. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, KDD '05, pages 672–677, New York, NY, USA, 2005. ACM. ISBN 1-59593-135-X. doi: 10.1145/1081870.1081955.

[108] B. K. Kuntal, A. Dutta, and S. S. Mande. CompNet: a GUI based tool for comparison of multiple biological interaction networks. *BMC Bioinformatics*, 17(1):185, 2016. ISSN 1471-2105. doi: 10.1186/s12859-016-1013-x.

[109] C. Lacave, M. Luque, and F. J. Diez. Explanation of bayesian networks and influence diagrams in elvira. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(4):952–965, Aug 2007. ISSN 1083-4419. doi: 10.1109/TSMCB.2007.896018.

[110] A. Lancichinetti and S. Fortunato. Consensus clustering in complex networks. *Scientific Reports*, 2, 2012. doi: 10.1038/srep00336.

[111] P. Langfelder and S. Horvath. Wgcna: an r package for weighted correlation network analysis. *BMC Bioinformatics*, 9(1):559, 2008. ISSN 1471-2105. doi: 10.1186/1471-2105-9-559.

[112] P. Langfelder, B. Zhang, and S. Horvath. Defining clusters from a hierarchical cluster tree: the dynamic tree cut package for r. *Bioinformatics*, 24(5):719–720, 2008. doi: 10.1093/bioinformatics/btm563.

[113] A. Lex, M. Streit, E. Kruijff, and D. Schmalstieg. Caleydo: Design and evaluation of a visual analysis framework for gene expression data in its biological context. In *2010 IEEE Pacific Visualization Symposium (PacificVis)*, pages 57–64, March 2010. doi: 10.1109/PACIFICVIS.2010.5429609.

[114] I. Liiv. Seriation and matrix reordering methods: An historical overview. *Statistical Analysis and Data Mining*, 3(2):70–91, 2010. ISSN 1932-1872. doi: 10.1002/sam.10071.

[115] J. Linde, S. Schulze, S. G. Henkel, and R. Guthke. Data-and knowledge-based modeling of gene regulatory networks: an update. *EXCLI Journal*, 14:346, 2015. doi: 10.17179/excli2015-168.

[116] S. Liu, D. Maljovec, B. Wang, P. Bremer, and V. Pascucci. Visualizing high-dimensional data: Advances in the past decade. *IEEE Transactions on Visualization and Computer Graphics*, 23(3):1249–1268, March 2017. ISSN 1077-2626. doi: 10.1109/TVCG.2016.2640960.

[117] D. J. Lunn, A. Thomas, N. Best, and D. Spiegelhalter. Winbugs - a bayesian modelling framework: Concepts, structure, and extensibility. *Statistics and Computing*, 10(4):325–337, Oct 2000. ISSN 1573-1375. doi: 10.1023/A:1008929526011.

[118] Z. W. Luo, C. A. Hackett, J. E. Bradshaw, J. W. McNicol, and D. Milbourne. Construction of a genetic linkage map in tetraploid species using molecular markers. *Genetics*, 157(3): 1369–1385, 2001. ISSN 0016-6731.

[119] J. Mackinlay. Automating the design of graphical presentations of relational information. *ACM Trans. Graph.*, 5(2):110–141, 1986. ISSN 0730-0301. doi: 10.1145/22949.22950.

[120] P. Mahanta, H. A. Ahmed, D. K. Bhattacharyya, and J. K. Kalita. Triclustering in gene expression data analysis: a selected survey. In *2nd National Conference on Emerging*

Trends and Applications in Computer Science (NCETACS), pages 1–6. IEEE, 2011. doi: 10.1109/ncetacs.2011.5751409.

[121] D. Marbach, J. C. Costello, R. Küffner, N. M. Vega, R. J. Prill, D. M. Camacho, K. R. Allison, M. Kellis, J. J. Collins, G. Stolovitzky, et al. Wisdom of crowds for robust gene network inference. *Nature Methods*, 9(8):796–804, 2012. doi: doi:10.1038/nmeth.2016.

[122] F. Matthäus, V. A. Smith, A. Fogtman, W. H. Sommer, F. Leonardi-Essmann, A. Lourdusamy, M. A. Reimers, R. Spanagel, and P. J. Gebicke-Haerter. Interactive molecular networks obtained by computer-aided conversion of microarray data from brains of alcohol-drinking rats. *Pharmacopsychiatry*, 42(S 01):S118–S128, 2009. doi: 10.1055/s-0029-1216348.

[123] M. J. McGuffin. Simple algorithms for network visualization: A tutorial. *Tsinghua Science and Technology*, 17(4):383–398, 2012. doi: 10.1109/TST.2012.6297585.

[124] M. J. McGuffin and J. M. Robert. Quantifying the space-efficiency of 2d graphical representations of trees. *Information Visualization*, 9(2):115–140, 2010. doi: 10.1057/ivs.2009.4.

[125] M. L. Metzker. Sequencing technologies—the next generation. *Nature reviews genetics*, 11 (1):31, 2010. doi: 10.1038/nrg2626.

[126] M. Meyer, T. Munzner, and H. Pfister. Mizbee: a multiscale synteny browser. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):897–904, 2009. doi: 10.1109/TVCG.2009.167.

[127] M. Meyer, B. Wong, M. Styczynski, T. Munzner, and H. Pfister. Pathline: A tool for comparative functional genomics. *Computer Graphics Forum*, 29(3):1043–1052, 2010. doi: 10.1111/j.1467-8659.2009.01710.x.

[128] H. Mi, A. Muruganujan, J. T. Casagrande, and P. D. Thomas. Large-scale gene function analysis with the panther classification system. *Nature Protocols*, 8(8):1551–1566, 2013. doi: 10.1038/nprot.2013.092.

[129] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: Simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002. ISSN 0036-8075. doi: 10.1126/science.298.5594.824.

[130] B. Mirel and C. Görg. Scientists' sense making when hypothesizing about disease mechanisms from expression data and their needs for visualization support. *BMC Bioinformatics*, 15(1):117, 2014. ISSN 1471-2105. doi: 10.1186/1471-2105-15-117.

[131] P. Montero and J. Vilar. Tsclust: An r package for time series clustering. *Journal of Statistical Software*, 62(1):1–43, 2014. ISSN 1548-7660. doi: 10.18637/jss.v062.i01.

[132] T. Munzner. A nested model for visualization design and validation. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):921–928, 2009. doi: 10.1109/TVCG.2009.111.

[133] T. Munzner. *Visualization Analysis and Design*. A K Peters/CRC Press, New York, NY, USA, 2014. doi: 10.1201/b17511.

[134] T. Munzner, F. Guimbretière, S. Tasiran, L. Zhang, and Y. Zhou. Treejuxtaposer: Scalable tree comparison using focus+context with guaranteed visibility. In *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, pages 453–462, New York, NY, USA, 2003. ACM. ISBN 1-58113-709-5. doi: 10.1145/1201775.882291.

[135] K. P. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, 2002. AAI3082340.

[136] S. Navlakha, R. Rastogi, and N. Shrivastava. Graph summarization with bounded error. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 419–432, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-102-6. doi: 10.1145/1376616.1376661.

[137] S. Navlakha, J. White, N. Nagarajan, M. Pop, and C. Kingsford. *Finding Biologically Accurate Clusterings in Hierarchical Tree Decompositions Using the Variation of Information*, pages 400–417. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-02008-7. doi: 10.1007/978-3-642-02008-7_29.

[138] M. Newman. *Networks: An Introduction*. Oxford University Press, Inc., New York, NY, USA, 2010. ISBN 0199206651, 9780199206650.

[139] C. B. Nielsen, H. Younesy, H. O'Geen, X Xu, A. R. Jackson, A. Milosavljevic, T. Wang, J. F. Costello, M. Hirst, P. J. Farnham, et al. Spark: a navigational paradigm for genomic data exploration. *Genome Research*, 22(11):2262–2269, 2012. doi: 10.1101/gr.140665.112.

[140] A. Obulkasim, G. A. Meijer, and M. van de Wiel. Semi-supervised Adaptive-height Snipping of the Hierarchical Clustering Tree. *BMC Bioinformatics*, pages 1–11, 2015. doi: 10.1186/s12859-014-0448-1.

[141] G. A. Pavlopoulos, A. L. Wegener, and R. Schneider. A survey of visualization tools for biological network analysis. *BioData Mining*, 1(1):12, Nov 2008. ISSN 1756-0381. doi: 10.1186/1756-0381-1-12.

[142] G. A. Pavlopoulos, D. Malliarakis, N. Papanikolaou, T. Theodosiou, A. J. Enright, and I. Iliopoulos. Visualizing genome and systems biology: Technologies, tools, implementation techniques and trends, past, present and future. *GigaScience*, 4(1):1–27, 2015. doi: 10.1186/s13742-015-0077-2.

[143] C. Perin, F. Vernier, and J. D. Fekete. Interactive horizon graphs: Improving the compact visualization of multiple time series. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 3217–3226, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1899-0. doi: 10.1145/2470654.2466441.

[144] J. Poco, A. Dasgupta, Y. Wei, W. Hargrove, C. Schwalm, R. Cook, E. Bertini, and C. Silva. Similarityexplorer: A visual inter-comparison tool for multifaceted climate data. *Computer Graphics Forum*, 33(3):341–350, 2014. doi: 10.1111/cgf.12390.

[145] A. J. Pretorius and J. J. Van Wijk. Visual inspection of multivariate graphs. *Computer Graphics Forum*, 27(3):967–974. doi: 10.1111/j.1467-8659.2008.01231.x.

[146] H. C. Purchase. *Experimental Human-Computer Interaction: A Practical Guide with Visual Examples*. Cambridge University Press, New York, NY, USA, 1st edition, 2012. ISBN 0521279542, 9780521279543.

[147] J. Quackenbush. Microarray data normalization and transformation. *Nature genetics*, 32 (4s):496, 2002. doi: 10.1038/ng1032.

[148] B. Renoust, G. Melançon, and T. Munzner. Detangler: Visual analytics for multiplex networks. *Computer Graphics Forum*, 34(3):321–330, 2015. ISSN 1467-8659. doi: 10.1111/cgf.12644.

[149] J. A. Reuter, D. V. Spacek, and M. P. Snyder. High-throughput sequencing technologies. *Molecular Cell*, 58(4):586 – 597, 2015. ISSN 1097-2765. doi: 10.1016/j.molcel.2015.05.004.

[150] N. H. Riche, T. Dwyer, B. Lee, and S. Carpendale. Exploring the design space of interactive link curvature in network diagrams. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, AVI '12, pages 506–513, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1287-5. doi: 10.1145/2254556.2254652.

[151] J. C. Roberts, C. Headleand, and P. D. Ritsos. Sketching designs using the five design-sheet methodology. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):419–428, Jan 2016. ISSN 1077-2626. doi: 10.1109/TVCG.2015.2467271.

[152] M. Roux. A comparative study of divisive and agglomerative hierarchical clustering algorithms. *Journal of Classification*, 35(2):345–366, Jul 2018. ISSN 1432-1343. doi: 10.1007/s00357-018-9259-9.

[153] R. Sadana, T. Major, A. Dove, and J. Stasko. Onset: A visualization technique for large-scale binary set data. *IEEE Transactions on Visualization and Computer Graphics*, 20(12): 1993–2002, Dec 2014. ISSN 1077-2626. doi: 10.1109/TVCG.2014.2346249.

[154] R. Saito, M. E. Smoot, K. Ono, J. Ruscheinski, P. L. Wang, S. Lotia, A. R. Pico, G. D. Bader, and T. Ideker. A travel guide to cytoscape plugins. *Nature methods*, 9(11):1069–1076, 2012. doi: 10.1038/nmeth.2212.

[155] R. Sakai and J. Aerts. Card Sorting Techniques for Domain Characterization in Problem-driven Visualization Research. In E. Bertini, J. Kennedy, and E. Puppo, editors, *Eurographics Conference on Visualization (EuroVis) - Short Papers*. The Eurographics Association, 2015. doi: 10.2312/eurovisshort.20151136.

[156] P. Saraiya, C. North, and K. Duca. An evaluation of microarray visualization tools for biological insight. In *Symposium on Information Visualization*, pages 1–8. IEEE, 2004. doi: 10.1109/INFVIS.2004.5.

[157] T. Schaffter, D. Marbach, and D. Floreano. Genenetweaver: in silico benchmark generation and performance profiling of network inference methods. *Bioinformatics*, 27(16):2263–2270, 2011. doi: 10.1093/bioinformatics/btr373.

[158] F. Schreiber and H. Schwöbbermeyer. MAVisto: a Tool for the Exploration of Network Motifs. *Bioinformatics*, 21(17):3572–3574, 07 2005. ISSN 1367-4803. doi: 10.1093/bioinformatics/bti556.

[159] M. Sedlmair, M. Meyer, and T. Munzner. Design study methodology: Reflections from the trenches and the stacks. *IEEE Transactions on Visualization and Computer Graphics*, 18 (12):2431–2440, Dec 2012. ISSN 1077-2626. doi: 10.1109/TVCG.2012.213.

[160] M. Sedlmair, C. Heinzl, S. Bruckner, H. Piringer, and T. Möller. Visual parameter space analysis: A conceptual framework. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2161–2170, Dec 2014. ISSN 1077-2626. doi: 10.1109/TVCG.2014.2346321.

[161] J. Seo and B. Shneiderman. Interactively exploring hierarchical clustering results [gene identification]. *Computer*, 35(7):80–86, 2002. doi: 10.1109/mc.2002.1016905.

[162] A. Shamir and A. Stolpnik. Interactive visual queries for multivariate graphs exploration. *Computers and Graphics*, 36(4):257 – 264, 2012. ISSN 0097-8493. doi: 10.1016/j.cag.2012.02.006. Applications of Geometry Processing.

[163] P. D. Shaw, M. Graham, J. Kennedy, I. Milne, and D. F. Marshall. Helium: visualization of large scale plant pedigrees. *BMC bioinformatics*, 15(1):259, 2014. doi: 10.1186/1471-2105-15-259.

[164] C. Shi, W. Cui, S. Liu, P. Xu, W. Chen, and H. Qu. Rankexplorer: Visualization of ranking changes in large time series data. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2669–2678, Dec 2012. ISSN 1077-2626. doi: 10.1109/TVCG.2012.253.

[165] A. Sinha and M. Markatou. A platform for processing expression of short time series (pests). *BMC Bioinformatics*, 12(1):1, 2011. doi: 10.1186/1471-2105-12-13.

[166] V. A. Smith, E. D. Jarvis, and A. J. Hartemink. Evaluating functional network inference using simulations of complex biological systems. *Bioinformatics*, 18(suppl 1):S216–S224, 2002. doi: 10.1093/bioinformatics/18.suppl_1.S216.

[167] V. A. Smith, J. Yu, T. V Smulders, A. J Hartemink, and E. D. Jarvis. Computational inference of neural information flow networks. *PLOS Computational Biology*, 2(11):1–14, 11 2006. doi: 10.1371/journal.pcbi.0020161.

[168] T. Song and H. Gu. Discriminative motif discovery via simulated evolution and random under-sampling. *PLOS ONE*, 9(2):1–10, 02 2014. doi: 10.1371/journal.pone.0087670.

[169] J. Sorger, K. Bühler, F. Schulze, T. Liu, and B. Dickson. neuromap — interactive graph-visualization of the fruit fly's neural circuit. In *2013 IEEE Symposium on Biological Data Visualization (BioVis)*, pages 73–80, Oct 2013. doi: 10.1109/BioVis.2013.6664349.

[170] L. E. Sucar. Probabilistic graphical models. *Advances in Computer Vision and Pattern Recognition.*, 10:978–1, 2015. doi: 10.1007/978-1-4471-6699-3.

[171] M. Suderman and M. Hallett. Tools for visually exploring biological networks. *Bioinformatics*, 23(20):2651–2659, September 2007. ISSN 1367-4803. doi: 10.1093/bioinformatics/btm401.

[172] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996. doi: 10.1111/j.2517-6161.1996.tb02080.x.

[173] K. Tsuyuzaki, G. Morota, M. Ishii, T. Nakazato, S. Miyazaki, and I. Nikaido. Mesh ora framework: R/bioconductor packages to support mesh over-representation analysis. *BMC Bioinformatics*, 16(1):45, Feb 2015. ISSN 1471-2105. doi: 10.1186/s12859-015-0453-z.

[174] E. R. Tufte and P. Graves-Morris. The visual display of quantitative information, 1983.

[175] E. R. Tufte, N. H. Goeler, and R. Benson. *Envisioning Information*, volume 126. Graphics press Cheshire, CT, 1990.

[176] J. W. Tukey. Exploratory data analysis. 1977.

[177] S. van den Elzen and J. J. van Wijk. Multivariate network exploration and presentation: From detail to overview via selections and aggregations. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2310–2319, Dec 2014. ISSN 1077-2626. doi: 10.1109/TVCG.2014.2346441.

[178] J. J. van Wijk. The value of visualization. In *VIS 05. IEEE Visualization, 2005.*, pages 79–86, Oct 2005. doi: 10.1109/VISUAL.2005.1532781.

[179] A. Vogogias, J. Kennedy, and D. Archambault. Hierarchical Clustering with Multiple-Height Branch-Cut Applied to Short Time-Series Gene Expression Data. In Tobias Isenberg and Filip Sadlo, editors, *EuroVis 2016 - Posters*. The Eurographics Association, 2016. ISBN 978-3-03868-015-4. doi: 10.2312/eurp.20161127.

[180] A. Vogogias, J. Kennedy, D. Archambault, V. A. Smith, and H. Currant. MLCut: Exploring Multi-Level Cuts in Dendrograms for Biological Data. In Cagatay Turkay and Tao Ruan Wan, editors, *Computer Graphics and Visual Computing (CGVC)*. The Eurographics Association, 2016. ISBN 978-3-03868-022-2. doi: 10.2312/cgvc.20161288.

[181] A. Vogogias, J. Kennedy, D. Archambault, B. Bach, V. A. Smith, and H. Currant. Bayespiles: Visualisation support for bayesian network structure learning. *ACM Transactions on Intelligent Systems and Technology (ACM TIST)*, 10(1):5:1–5:23, November 2018. ISSN 2157-6904. doi: 10.1145/3230623.

[182] A. Vogogias, D. Archambault, B. Bach, and J. Kennedy. A study of matrix representations for networks with multiple edge types. In *Multilayer Network Visualization Workshop (MLNVIS)*. IEEE VIS, 2019.

[183] T. von Landesberger, A. Kuijper, T. Schreck, J. Kohlhammer, J. J. van Wijk, J. D. Fekete, and D. W. Fellner. Visual analysis of large graphs: State-of-the-art and future research challenges. *Computer Graphics Forum*, 30(6):1719–1749, 2011. ISSN 1467-8659. doi: 10.1111/j.1467-8659.2011.01898.x.

[184] J. Wang and K. Mueller. The visual causality analyst: An interactive interface for causal reasoning. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):230–239, Jan 2016. ISSN 1077-2626. doi: 10.1109/TVCG.2015.2467931.

[185] X. Wang, M. Wu, Z. Li, and C. Chan. Short time-series microarray analysis: Methods and challenges. *BMC Systems Biology*, 2(1):58, 2008. doi: 10.1186/1752-0509-2-58.

[186] M. O. Ward. A taxonomy of glyph placement strategies for multidimensional data visualization. *Information Visualization*, 1(3-4):194–210, 2002. doi: 10.1057/PALGRAVE.IVS.9500025.

[187] M. O. Ward. Multivariate data glyphs: Principles and practice. In *Handbook of Data Visualization*, pages 179–198. Springer, London, 2008. doi: 10.1007/978-3-540-33037-0_8.

[188] C. Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann Publishers Inc., San Francisco, 2nd edition, 2004. doi: 10.1016/B978-155860819-1/50001-7.

[189] M. Wattenberg. Visual exploration of multivariate graphs. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, pages 811–819, New York, NY, USA, 2006. ACM. ISBN 1-59593-372-7. doi: 10.1145/1124772.1124891.

[190] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393 (6684):440, 1998. doi: 10.1038/30918.

[191] S. Wernicke and F. Rasche. Fanmod: a tool for fast network motif detection. *Bioinformatics*, 22(9):1152–1153, 02 2006. ISSN 1367-4803. doi: 10.1093/bioinformatics/btl038.

[192] Y. Yang, T. Dwyer, S. Goodwin, and K. Marriott. Many-to-many geographically-embedded flow visualisation: An evaluation. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):411–420, Jan 2017. ISSN 1077-2626. doi: 10.1109/TVCG.2016.2598885.

[193] O. Yim and K. T. Ramdeen. Hierarchical cluster analysis: Comparison of three linkage measures and application to psychological data. *Tutorials in Quantitative Methods for Psychology*, 11(1):8–21, 2015. ISSN 1913-4126.

[194] W. C. Young, A. E. Raftery, and K. Y. Yeung. Fast bayesian inference for gene regulatory networks using scanbma. *BMC Systems Biology*, 8(1):47, 2014. ISSN 1752-0509. doi: 10.1186/1752-0509-8-47.

[195] J. Yu, V. A. Smith, P. P. Wang, A. J. Hartemink, and E. D. Jarvis. Using bayesian network inference algorithms to recover molecular genetic regulatory networks. *International Conference on Systems Biology (ICSB02)*, 2002.

[196] J. D. Zapata-Rivera, E. Neufeld, and J. E. Greer. Visualization of bayesian belief networks. In *IEEE Visualization 1999 Late Breaking Hot Topics Proceedings*, pages 85–88. Press, 1999.

A GitHub Repositories

The source code for MLCut and BayesPiles can be found online in the following GitHub repositories:

- https://github.com/than8/MLCut
- https://github.com/than8/BayesPiles

B List of Terms

Useful terms listed in order of appearance in the text.

- **Network:** a mathematical model that models entities (nodes) and the relationships between them (edges).
- **Model:** in the context of this thesis, it refers to a network model of the phenomenon being studied.
- **Modeller:** a domain scientist or expert who analyses data to infer networks. In the context of this thesis the modellers are computational biologists (bioinformaticians).
- **Network inference:** the process followed by modellers for deriving networks from the data.
- **Consensus network:** the presented final network that modellers infer from the data.
- **Candidate networks:** the network solutions that modellers consider when they infer a consensus network.
- **Visualisation:** the scientific discipline that aims at helping humans to gain a better understanding of data, through the sense of sight, using visual means.
- **Exploratory:** visualisation approach focusing on the exploration of the data to form new hypotheses.
- **Explanatory:** visualisation approach focusing on the comprehensive presentation of a concept or idea.
- **Confirmatory:** visualisation approach focusing on enabling the verification of a fact, based on evidence found in the data.

- **Dendrogram:** a tree structure generated by a hierarchical clustering algorithm in which the leaves correspond to entities and the intermediate nodes to similarity levels at which entities are merged to form clusters.

- **Cut:** in the context of this thesis it is the similarity threshold (level) that defines the height at which branches are separated from the rest of the dendrogram and correspond to clusters.

- **Distinctiveness:** the length (or height) of an edge in the dendrogram that indicates the similarity between two variables or clusters of variables (branches).

- **Search space:** the set of all possible networks from which candidate networks are sampled.

- **Solution space:** the set of highest scoring networks as generated and assessed by a network inference algorithm.

- **Markov Lag (ML):** the delay of an interaction measured by the number of time slices skipped before observing its effect.