Building Information Filtering Networks with Topological Constraints: Algorithms and Applications

Guido Previde Massara

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of

University College London.

Department of Computer Science

University College London

December 22, 2020

I, Guido Previde Massara, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

We propose a new methodology for learning the structure of *sparse networks* from data; in doing so we adopt a dual perspective where we consider networks both as weighted graphs and as simplicial complexes.

The proposed learning methodology belongs to the family of *preferential attachment* algorithms, where a network is extended by iteratively adding new vertices. In the conventional preferential attachment algorithm a new vertex is added to the network by adding a single edge to another existing vertex; in our approach a new vertex is added to a *set* of vertices by adding one or more new simplices to the simplicial complex. We propose the use of a score function to quantify the strength of the association between the new vertex and the attachment points. The methodology performs a greedy optimisation of the total score by selecting, at each step, the new vertex and the attachment points that maximise the gain in the score.

Sparsity is enforced by restricting the space of the feasible configurations through the imposition of topological constraints on the candidate networks; the constraint is fulfilled by allowing only topological operations that are invariant with respect to the required property. For instance, if the topological constraint requires the constructed network to be be planar, then only planarity-invariant operations are allowed; if the constraint is that the network must be a clique forest, then only simplicial vertices can be added. At each step of the algorithm, the vertex to be added and the attachment points are those that provide the maximum increase in score while maintaining the topological constraints.

As a concrete but general realisation we propose the clique forest as a possible topological structure for the representation of sparse networks, and we allow to specify further constraints such as the allowed range of clique sizes and the saturation of the attachment points. In this thesis we originally introduce the Maximally Filtered Clique

Forest (MFCF) algorithm: the MFCF builds a clique forest by repeated application of a suitably invariant operation that we call *Clique Expansion operator* and adds vertices according to a strategy that greedily maximises the gain in a local score function. The gains produced by the Clique Expansion operator can be validated in a number of ways, including statistical testing, cross-validation or value thresholding. The algorithm does not prescribe a specific form for the gain function, but allows the use of any number of gain functions as long as they are consistent with the Clique Expansion operator. We describe several examples of gain functions suited to different problems.

As a specific practical realisation we study the extraction of planar networks with the Triangulated Maximally Filtered Graph (TMFG). The TMFG, in its simplest form, is a specialised version of the MFCF, but it can be made more powerful by allowing the use of specialised planarity invariant operators that are not based on the Clique Expansion operator.

We provide applications to two well known applied problems: the Maximum Weight Planar Subgraph Problem (MWPSP) and the Covariance Selection problem. With regards to the Covariance Selection problem we compare our results to the state of the art solution (the Graphical Lasso) and we highlight the benefits of our methodology.

Finally, we study the geometry of clique trees as simplicial complexes and note how the statistics based on cliques and separators provides information equivalent to the one that can be achieved by means of homological methods, such as the analysis of Betti numbers, however with our approach being computationally more efficient and intuitively simpler. Finally, we use the geometric tools developed to provide a possible methodology for inferring the size of a dataset generated by a factor model. As an example we show that our tools provide a solution for inferring the size of a dataset generated by a factor model.

Impact Statement

Complex Networks and Big Data challenges

Complex networks are ubiquitous in complex systems modelling. The structure and geometry of a network is sometimes an explicit, immediately observable, feature of a system such as in technological and infrastructure networks, social networks, brain

connectivity, food networks. More often, networks are used as a tool to understand and model the dependencies and interactions between agents of highly connected systems as in the case of economic and financial networks (e.g. financial contagion networks, spillover networks), biological networks (e.g. gene co-expression, metabolic networks, protein-protein interactions), psychometric networks (e.g. co-occurrence of personality traits or psychopathological disorders) or in complex computational tasks (e.g. image processing, multivariate statistical models, expert systems). Many networks are characterised by a high, often growing, number of nodes and a relatively low number of observations, often exhibiting spurious associations between agents: the Big Data regime. Especially in the Big Data regime, it is necessary to be able to filter spurious dependencies and to retain the dependencies that provide insight into the system, or that allow for robust and parsimonious modelling.

Topologically Constrained Information Filtering Networks

Information Filtering Networks (IFN) are sparse networks aimed at capturing the most significant interactions by removing spurious agent-agent dependencies. Topologically Constrained Information Filtering Networks (TCIFNs) are IFNs that are built by imposing constraints on the geometry of the network (e.g. planar networks, clique forests), privileging few local interactions between selected subsets of the agents (cliques) and penalising dependencies that give rise to overly complex topological structures. The simplicity of the underlying geometry provides insight into the system and, most importantly, the possibility to develop effective descriptive and predictive models that can be tuned to real data. TCIFNs can be analysed as networks in their own right, as general multivariate probabilistic systems, or as topological objects, providing the applied researcher with a wide and ever expanding toolbox for the analysis of data.

Benefits of cross-collaboration

TCIFNs have already found interest and application in a number of applied fields, such as combinatorial optimization (Maximum Weight Planar Subgraph Problem), financial risk management (risk modelling, stress testing, systemic risk), analysis of macroeconomic policies (spillover networks) and psychometric networks (theory of personality, questionnaires), statistics (covariance selection and penalised regression). In return, the interaction with applied researchers has proven invaluable in highlighting possibilities for further developments of the methodology. The generality of the methodology makes it suitable for use in any context where the researcher uses tools from topology or multivariate statistical analysis to analyse data.

An actionable tool for analysis and supervision

The industrial and commercial applications are potentially as wide ranging;: from psychometrics networks to biological networks, from technological and infrastructure networks to social networks, and from financial risk management and portfolio management to applied econometrics, especially in the analysis of sparse networks to estimate systemic risk. We foresee this capability as of potential interest to regulators and macroprudential regulatory bodies, with enhanced analysis towards the possibility to identify weak spots in financial networks.

Acknowledgements

Undertaking a research project at an age when one is well into, if not past, the mature student age-bracket is an immensely rewarding challenge that, however, puts huge pressure on personal and working life, with the continuous need to balance personal and family commitments, job demands and research. That this project has proven possible at all is an extraordinary testament to the incredible adaptability and support of family members, friends, co-workers, collaborators, co-authors and, naturally, my supervisor Prof. Tomaso Aste. In the full knowledge that words cannot make full justice, I will nevertheless try to convey my heartfelt thanks.

Starting from work I would like to thank all my former colleagues of the Credit Portfolio Modelling Team in KPMG for providing a great working environment and for extremely helpful discussions that informed many views of mine; some are reflected in this thesis. More specifically, I would like to thank my friend and former colleague Alun Wyn-Jones for reading and giving suggestions on preliminary drafts on a paper that constitutes the main inspiration for Chapter 4, and Dr. Etienne Hofstetter for some helpful discussions on dependence modelling in risk management.

Moving to collaborations, I should start by thanking two students of my supervisor Prof. Tomaso Aste. Yuqin Long studied and implemented in Python the algorithm described in Chapter 4; Daniel Savu implemented the MFCF in Python and carried out some original and promising experiments extending the MFCF with gain functions based on kernel regression. I could hardly overstate the help of Dr. Alexander P. Christensen for taking an early interest in our work and for introducing us to the world of applications of network theory in psychology. That the TMFG and the MFCF are now known and used in the field of network science in psychology is entirely due to Alex and to his initiative in implementing both algorithms in his Network Toolbox, and to his original applications in his papers. Alex has also kindly reviewed our work on clique forests and provided many valuable insights, new problems and ideas that we hope to develop in future collaborations.

Among the co-authors I would like to start by thanking Prof. Tiziana Di Matteo for the collaboration on two papers and especially for providing practical, no-nonsense approach to data modelling. Dr. Wolfram Barfuss is the first author of the paper that provides a substantial part of the material for 5 and has been invaluable in clarifying many points of the TMFG methodology through the application to multivariate Gaussian distributions. Prof. Lorenzo Magnani sits squarely in the intersection between friends, collaborators and co-authors and I will always be grateful to him for our long past, but ever fruitful, collaboration; in my view his influence on methodological issues is still very much present and of great value to me.

My second supervisor Dr. Simone Severini has been invaluable in reviewing and improving our work in pointing out inaccuracies and also, in one case, a material mistake.

It is impossible to overstate the debt I owe to Prof. Tomaso Aste for providing the inspiration for the work in this thesis, for giving me the opportunity to talk at the UCL CS Financial Computing and Analytics Group, for being an extremely organised, challenging and effective co-author in all of our papers, for keeping me honest through my research activity and pointing out logical inconsistencies, for providing both deep theoretical insight and good practical advice but, above all, for putting up with my ever changing schedule and for his commitment and flexibility as witnessed by the highly unusual number of week-end calls to discuss papers, thesis and other research topics. In my view, that puts him in the set of friends, too.

Finally, I have to give infinite thanks to my wife Valeria for allowing me to begin and see through this project in the face of the many challenges of an adult life, four home moves in as many cities, all sorts of mishaps and for standing in my stead when I was busy with work or study. I can only promise that I will try my best to make up for those missed week-ends, holidays and all those errands that you carried out for me.

Contents

1 Information Filtering Networks **18**

1.1 Motivation: Networks and Complex Systems 19

1.2 Information Filtering Networks: the Case for Sparse Modelling 22

1.3 Network Modelling: Approaches to Filtering and Applications 25

1.3.1 Planar Information Filtering Networks 27

1.4 Objective of the Research and our Approach 30

1.4.1 Fundamental requirements for a new IFN algorithm 30

1.4.2 Approach and Direction of Research 31

1.4.3 Applications 32

2 Approaches to the Modelling of Sparse Networks **34**

2.1 Graphical models and Markov Random Fields 34

2.1.1 Definitions 34

2.1.2 Conditional Independence 35

2.1.3 Markov Properties 39

2.1.4 Clique Factorisation Property and Hammersley-Clifford Theorem 41

2.2 Review of Methodologies for building Information Filtering Networks . 43

2.2.1 Structure Learning in Graphical Models 44

2.2.1.1 Score based methods 44

2.2.1.2 Constraint based Algorithms 49

2.2.1.3 Bayesian Methods 49

2.2.2 Sparse graphical models through regularisation and covariance selection (Lasso, Ridge, Elastic Net) 50

2.2.3 Information Filtering Networks 51

2.2.4 Triangulated Maximally Filtered Graphs 52

3 Learning Clique Forests 53

3.1 Graph Theory Prerequisites 53

3.1.1 Definitions 54

3.1.2 Chordal Graphs 56

3.1.3 Perfect Elimination Order 59

3.1.4 Perfect Sequences of Cliques 60

3.1.5 Clique Forest 62

3.2 The Clique Expansion Operator 64

3.3 The MFCF algorithm 68

3.3.1 Relationship with Prim's Minimum Spanning Tree Algorithm . 74

3.3.2 Relationship with the Maximum Cardinality Search algorithm . 75

3.3.3 Gain Functions 77

3.3.3.1 Similarity Matrix 78

3.3.3.2 Gain function from log-likelihood 79

3.3.3.3 Multivariate Normal Distribution 80

3.3.3.4 Multivariate Normal Distribution statistically validated 81

3.3.3.5 Random Gain Function 82

3.3.3.6 Regression 82

3.4 Other CF-Invariant Operations on Clique Forests 82

3.4.1 The Direct Join Operator 83

3.4.1.1 Score Functions and generalisation of Kruskal algorithm 84

3.4.2 Pruning a Clique Forest 84

4 TMFG and other approximate solutions for the Maximum Weight Planar Subgraph Problem 90

4.1 The Maximum Weight Planar Subgraph Problem 91

4.1.1 Planar Maximally Filtered Graph 93

4.2 Triangulated Maximally Filtered Graph 94

4.2.1 TMFG and Deltahedron heuristic 94

4.2.2 TMFG construction 97

4.3 Planarity invariant operators: $\mathbf{T_1}$, $\mathbf{T_2}$, $\mathbf{A}$, & $\mathbf{S}$ 103
4.4 Variants of the TMFG algorithm 107
4.4.1 TMFG-T1 107
4.4.2 TMFG-S 110
4.4.3 TMFG-A 113
4.5 Additional observations 116
4.5.1 Dynamical adaptability 116
4.5.2 Parallelization and big data 117
4.5.3 Memory usage 117
4.6 Comparison between TMFG and PMFG 118
4.6.1 Comparison between the performances of the various methods . 118

5 Probabilistic modelling with TMFG / MFCF and Financial Applications 123
5.1 Modeling with TMFG: information theoretic perspective 124
5.2 Financial Applications of the TMFG 127
5.2.1 Financial applications: Stress Testing 127
5.2.2 Risk Allocation 128
5.3 Financial Applications of the MFCF 129
5.3.1 Learning with Mixed Frequency Time Series 130

6 Application to the Covariance Selection Problem 133
6.1 Covariance Selection 133
6.1.1 Penalised Likelihood Maximisation 135
6.1.2 Graphical Lasso 135
6.2 The MFCF approach to to covariance selection 136
6.2.1 Construction of the precision matrix in the multivariate Gaussian case 137
6.2.2 Shrinkage procedures 137
6.2.2.1 Step 1 138
6.2.2.2 Step 2 138
6.2.2.3 Step 3 138
6.3 Testing Methodology 138
6.3.1 Generation of Synthetic Data 139

6.3.1.1 Synthetic data: sparse decomposable precision matrix 140
6.3.1.2 Synthetic data: Full Positive Definite Matrix from package “clusterGeneration” 140
6.3.1.3 Random Factor Model with noise 140
6.3.2 Usage and Treatment of Real Data 141
6.3.3 Algorithms used in the Testing 141
6.3.4 Performances indicators 143
6.4 Results 144
6.4.1 Synthetic data: sparse decomposable precision matrix 144
6.4.2 Synthetic data: Full Positive Definite Matrix from package “clusterGeneration”. 151
6.4.3 Random Factor Model with noise 158
6.4.4 Real Data 164

7 Topological Data Analysis with the MFCF 171
7.1 Applications to Topology of Data 171
7.1.1 Abstract Simplicial Complexes 172
7.1.2 The Simplicial Complexes built by the MFCF 177
7.1.3 Examples and applications 178
7.1.3.1 Identification of cliques 178
7.1.3.2 Identification of large dimensional blocks 182
7.1.3.3 Structure of factor models 186

8 Conclusions and further research 191
8.1 Conclusions 191
8.2 Further research 193
8.2.1 Non linear interactions and fat-tailed distributions 193
8.2.2 Financial applications 194
8.2.3 Geometry of clique forests 194
8.2.4 Confirmatory factor analysis 194
8.2.5 Geometry of factor models 195

Appendices 196

9 Appendix A **196**

9.1 Computer Codes 196

9.1.1 TMFG 196

9.1.2 TMFG-T1 198

9.1.3 TMFG-S 201

9.1.4 TMFG-A 204

Bibliography **212**

List of Figures

2.1 Conditional independence relationship 37

2.2 Pairwise Markov property 40

2.3 Global Markov property 40

2.4 Local Markov property 41

3.1 Cycle with chord 54

3.2 The graph K_4 57

3.3 A non-chordal graph 58

3.4 Chordal graph with clique forest 63

3.5 Chordal graph with clique forest 64

3.6 The clique expansion operator 66

3.7 The Maximally Filtered Clique Forest Algorithm 86

3.8 The Maximum Cardinality Search Algorithm 87

3.9 Direct join of clique forests 88

3.10 The Direct Join Operator 88

3.11 The edge pruning operator 89

4.1 The T_2 move 95

4.2 TMFG: initialisation stage 98

4.3 TMFG: gain selection stage 99

4.4 TMFG: update stage 99

4.5 Leung's extension 104

4.6 T_1 move 105

4.7 Leung's extension as a combination of T_2 and T_1 105

4.8 A move 105

4.9 S move 106

4.10 TMFG: T_1 variant 108
4.11 TMFG: T_2 and Swap 111
4.12 TMFG-A variant 114
4.13 TMFG scalability compared to the PMFG 119

5.1 Illustration of Equation 5.3 126

6.1 Sparse decomposable precision matrix: log-likelihood 146
6.2 Sparse decomposable precision matrix: performance 147
6.3 Sparse decomposable precision matrix: eigenvalue distance 148
6.4 Sparse decomposable precision matrix: inverse eigenvalue distance . . . 149
6.5 Sparse decomposable precision matrix: cliques composition 150
6.6 Random positive definite matrix: log-likelihood 153
6.7 Random positive definite matrix: performance 154
6.8 Random positive definite matrix: eigenvalue distance 155
6.9 Random positive definite matrix: inverse eigenvalue distance 156
6.10 Random positive definite matrix: cliques composition 157
6.11 Random factor model: log-likelihood 159
6.12 Random factor model: performance 160
6.13 Random factor model: eigenvalue distance 161
6.14 Random factor model: inverse eigenvalue distance 162
6.15 Random factor model: cliques composition 163
6.16 Real data: log-likelihood 165
6.17 Real data: performance 166
6.18 Real data: eigenvalue distance 167
6.19 Real data: inverse eigenvalue distance 168
6.20 Real data: cliques composition 170

7.1 A simplicial complex made of two simplices 174
7.2 Block Structured Matrix 180
7.3 Convergence to real clique size 181
7.4 Convergence to real Betti numbers 182
7.5 Large connected components 183

7.6 Convergence to real clique size 184
7.7 Convergence to real clique size 185
7.8 Clique sizes and eigenvalue analysis 189
7.9 Betti numbers for factor models 190

List of Tables

4.1 Relative performance of TMFG variants compared to PMFG 122

4.2 TMFG: increase in relative performance with size 122

6.1 Sparse chordal precision matrix: penalty/shrinkage parameters 145

6.2 Sparse chordal precision matrix: number of non-zero coefficients 145

6.3 Random positive definite matrix: penalty/shrinkage parameters 151

6.4 Random positive definite matrix: number of non-zero coefficients . . . 152

6.5 Factor model: penalty/shrinkage parameters 158

6.6 Factor model: non-zero coefficients 163

6.7 Real data: penalty/shrinkage parameters 169

6.8 Real data: non-zero coefficients 169

Chapter 1

Information Filtering Networks

We begin this introductory chapter with an overview of certain types of networks that arise frequently in the study of complex systems in relation to a wide range of scientific disciplines and fields of application. The purpose of the review is not completeness, since it would take many works the size of the present one just to survey a single field of research or application[1], but rather to provide an indicative idea of the remarkably diverse range of challenges that the analysis and modelling of complex networks offers to the researcher and the modeller[2]; some of the challenges and specific issues described in this chapter have been a source of inspiration for the approach that we advocate in the present thesis. Some results of the work described in this thesis have already been published and made available to the scientific community (Massara et al., 2016; Barfuss et al., 2016; Massara and Aste, 2019) and therefore we have the benefit of partial hindsight with regards to their use and applications; for this reason we describe in slightly greater detail studies where the methodologies described in Chapters 3-6 have already found application.

Next, we introduce the concept of Information Filtering Network (IFN) (Tumminello et al., 2005; Barfuss et al., 2016) as a tool to extract useful and relevant information from complex networks. Real networks are often densely populated and characterised by a huge number of interactions that can be spurious, redundant or simply accidental. The objective of IFN is to eliminate noise and spurious interactions from complex networks, revealing the overall structure of the underlying system and drawing

[1]A search for "complex networks" on Google Scholar returned 527,000 results on June the 29th 2019.

[2]The positive, bi-directional, interplay between practical approaches and theoretical constructs in the process of scientific discovery is described in several works in the fields of cognitive science and epistemology, see for instance Magnani (2004) and references therein.

the attention to a few essential and defining characteristics. We examine three possible approaches to IFNs that originate from the disciplines of Econophysics, Statistics and Topological Data Analysis and which, in our opinion, provide unique and different insights into complex systems and complement each other, providing a general and rich toolset for the study of complex systems. We provide a slightly more detailed description of the extraction of planar graphs as a way to produce IFNs, as this is the branch from which stems the research exposed in this thesis.

Finally, we set out our direction of research where we try to incorporate the three approaches in a coherent methodology for the construction of IFN and we use it to produce a set of tools that can be brought to bear in various applied problems. Throughout, we will keep constant attention to the issues of practical implementation, adequate performance and scalability that are still a prerequisite for solving real world problems, even in these days of ever increasing computing power.

1.1 Motivation: Networks and Complex Systems

Network science has experienced an explosive pace of development since the beginning of the century, eventually occupying a central role in the study of complex systems (Strogatz, 2001; Barabási, 2011). Networks seems to be a natural modelling approach in all the disciplines where it is important to model the interactions among numerous agents; the nodes represent agents[3] and edges represent the interactions amongst them. However, besides this simple and initially reassuring unifying concept, there are many questions and complexities that enter into play and that may be addressed, emphasized, approximated or ignored according to the domain of application (Strogatz, 2001).

- *Structural complexity*. The network can be more or less populated, more or less sparse. This aspect is central to IFNs: how many of the interactions within agents are inherently significant to understand the behaviour of the system? How many are effectively redundant given the information that we have about a subset of agents and their interactions? How many edges is it necessary to retain to characterise a network faithfully or, at least, effectively?

[3]The meaning of "agent" obviously depends on the system being modelled. It could be economic entities, assets, proteins, hardware resources, biological species, people, geographical places, and so on (Newman, 2010).

- *Network evolution.* Does the wiring change over time (as it is the case with the Internet) or is it stable over relatively long times (for instance, in gene expression networks)? Does the structure change over time, and how?

- *Connection diversity.* Edges between nodes could have different weights (as it is often the case with networks representing similarity, affinity, correlation) or a preferred direction (in causal networks, in network that model the supply of resources, in the modelling of knowledge).

- *Dynamical complexity.* Do nodes represent quantities that vary in time maybe in a highly non-linear way (for instance metabolic networks, gene expression networks)?

- *Node diversity.* Sometimes nodes represent similar entities, for instance pixels in a picture or particles in a lattice, in other times nodes can represent completely different entities, for instance corporations in a financial network or genes in a gene expression network.

- *Meta-complication.* Where one level of complexity influences the other; for instance when two neurons fire together (dynamic complexity) their connection is strengthened (connection diversity).

In **biology** (Ideker and Nussinov, 2017; Mohammad, 2018; Barzel et al., 2013) the set of networks traditionally studied (cell connectivity, immune system, food webs and ecosystems) has recently and dramatically expanded with the inclusion of an increasing set of experimental data produced by high-throughput techniques (including, but not limited to, gene transcription, protein-protein interaction, metabolic and gene expression networks). A problem that is particularly relevant to networks arising from high-throughput measurements is that the number of experiments or samples can be much lower than the number of variables measured (such as proteins or metabolites)[4] posing new challenges in model selection (Kirpich et al., 2018; Wu et al., 2019) that is often addressed with the use of regularisation techniques (Hoerl and Kennard, 1970; Tibshirani, 1996; Zou and Hastie, 2005; Friedman et al., 2008). In bioinformatics,

[4]The “small n, large p” problem.

graphical models[5] are increasingly adopted and are now a tool used for describing and measuring the outcomes of experiments and measurements, as well as a modelling tool and a language for formulating hypothesis about the interactions of the agents. The study of brain connectivity has always been one of the motivations, and indeed one of the inspirations, for the study of and modelling of networks (Bishop et al., 1995). Recently the study of brain connectivity has seen very interesting developments in the study of higher order topological structures in neuroscience with a significant focus on homological methods (Petri et al., 2014; Giusti et al., 2015; Sizemore et al., 2018).

In **psychology** network science is now also used alongside, or instead of, the traditional statistical technique of factor analysis in several fields; for instance in personality theory (Christensen et al., 2018b) or in psychopathology (symptoms interaction networks, see Borsboom (2017); Christensen et al. (2018c)). In these field the ability to recover sparse networks is crucial to understanding discrete facets of psychological traits or to isolate meaningful interactions between clinical symptoms. In particular Schmittmann et al. (2013); Kruis and Maris (2016) develop a network-based interpretation in psychological measurements. In this novel interpretation the object measured is the set of causal relationships observed among the observations: psychological attributes are seen as the network underlying the observations. In Golino and Epskamp (2017) network sparsification methods based on the Graphical Least Absolute Shrinkage and Selection Operator (or GLASSO, see the seminal paper Friedman et al. (2008)) are used to formulate hypotheses about the dimensionality of psychological data.

Technological networks (Newman, 2010; Boutaba et al., 2018; Chen et al., 2013) such as the Internet, transport, delivery and distribution networks, power grids, have been among the first to motivate the study of networks, from the early attempts to optimise electric networks by Otakar Borůvka (Boruvka, 1926) to the last studies on the technological infrastructure underneath social networks (Chen et al., 2013; Boutaba et al., 2018). In particular the problem of extracting planar subnetworks has a long tradition of study in operations research and facility layout (Foulds and Robinson, 1978) and VLSI design (Gogoi and Kalita, 2012).

The main driver for the development and use of network algorithms in the last

[5]As we will see in the next chapter, graphical models are networks whose nodes are random variables and whose edges represent probabilistic dependence, or more accurately, where missing edges represent a relationship of conditional independence (Lauritzen, 1996).

decade has been of course the expanding catalogue of **social networks**, including the all-too-known web social networks, but also social and economic agent networks (Easley and Kleinberg, 2010; Jackson, 2011; Livan et al., 2017), along with the similar but distinct group of information networks (Newman, 2010), including the World Wide Web (Barabási, 2002), citation networks (Li et al., 2018, 2019), patents and innovation networks (Ozman, 2009), and recommender networks (Bobadilla et al., 2013) largely used in media distribution and in virtually every aspect of retail marketing and sales.

In **economics and finance** networks are widely used in the study of systemic risk, contagion and macro-prudential regulation (Hser, 2015; Pozzi et al., 2013; Chinazzi and Fagiolo, 2015; Gai and Kapadia, 2010), in the study of the dependency structure of financial assets (Bonanno et al., 2004; Fiedor, 2014; Tumminello et al., 2007; Musmeci et al., 2015b), in portfolio management and stress testing (Rebonato and Denev, 2014; Rebonato, 2010b), in credit risk modelling (Filiz et al., 2012), in the study of regime changes in financial portfolios (Procacci and Aste, 2018), in the study of corporate networks (de Jeude et al., 2019) and many other applications (Souma et al., 2003). A field in economics where the methodology exposed in this thesis has been applied is the field of policy evaluation using the tool of "spillover networks" (Castañeda Ramos and Guerrero, 2018a; Castañeda Ramos et al., 2018; Castañeda Ramos and Guerrero, 2019, 2018b). In spillover networks every node represents a policy issue, and the edge going from one node to the other represents a spillover effect of one policy on the other.

1.2 Information Filtering Networks: the Case for Sparse Modelling

We have mentioned before that one of the challenges in analysing networks is the structural complexity. A large collection of edges could hide important information on the structure of the network at large, hence it makes sense to devise a set of procedures to filter out unnecessary or spurious links. In these first examples we could define a sparse representation of a network as a representation where the edges are of the same order of magnitude as the nodes (for instance, as in Jackson (2010)), and network filtering a procedure that produces a sparse network starting from a dense network. While there are different approaches to this problem, there is consensus on the idea that a sparse representation of a network would provide several benefits (Hastie et al., 2015):

- **Reduce the "curse of dimensionality"**. The number of parameters required to model a network is at least as large as the number of edges. For example we could identify the correlation matrix of a set of p variables with a network of p nodes with weighted undirected edges equal to the sample correlation between n realisations of the corresponding variables. This network contains $p(p-1)/2$ parameters. When $p \gg n$ the sample correlation matrix is almost certainly not positive definite and therefore it cannot be used for any significant analysis, but *sparse* correlation matrices need much fewer data points to be almost certainly positive definite, and the number of data points required depends on the geometry of the network (Gross et al., 2018)[6]. This is an example of a topological constraint, where imposing a requirement on the output network, such as the size of the largest clique[7], has a direct consequence with regards to the properties of the model. It also shows that any filtering procedure must be designed bearing in mind the type of the problem, and should warn against applying, without careful consideration, some intuitively appealing, but ultimately inadequate, procedures such as filtering based on value thresholding.

- **Estimation error**. Network parameters are affected by estimation error. Using again the example of a correlation matrix, it is known that some correlation coefficients in a correlation matrix estimated from a noisy sample could even have the wrong sign, and this is particularly serious in portfolio management as it could cause incorrect asset allocation and hedging (Engle, 2009; Pafka and Kondor, 2004) or induce noise in the spectral properties of correlation matrices (Laloux et al., 1999). This case is particularly insidious, because it would induce systematic errors in the use of the model. The more the parameters in the network, the higher the chance that some model parameters are actually fitting noise, rather than true dependency relationships. Besides, even if the spurious edges do not introduce bias into the model, they can still affect the overall uncertainty by unnecessarily inflating the volatility and further complicating the analysis of

[6]Interestingly for the topic of this thesis, it is the size of the *treewidth*, or the size of the largest clique in the *minimal chordal coverage* of the network.

[7]A *clique* is a set of fully connected nodes.

scenarios.

- **Better understanding**. By removing non-significant details, it is possible to make sense of the overall structure of the data. This applies at the purely graphical level, where subject matter experts could examine visually complex dependencies without being distracted by spurious dependencies. For instance the sparse junction tree representation of a graphical statistical model allows an intuitive knowledge representation in expert systems (Lauritzen and Spiegelhalter, 1988). In other cases a sparse representation allows to have an appreciation for the overall structure of the data being analysed. For instance, the use of a sparse representation allows using tools from Topological Data Analysis in Neuroscience (Giusti et al., 2015). Additionally, a sparse representation could be easily modified by the researcher in order to formulate an hypothesis about the dependence structure of a particular experiment and to perform confirmatory factor analysis.

- **Modelling**. Sparsity allows meaningful representations: for example decomposable graphical models have very useful explicit representations of joint distributions (Massara et al., 2016; Barfuss et al., 2016; Giudici and Green, 1999) and closed form expressions for maximum likelihood estimates, and do not require the calculation of the partition function, which is a distinct advantage in computations. Again, not any sparse model will provide the same benefits. When sparsity is associated with the concept of conditional independence and this concept is linked to the geometry of the network a number of very powerful result will be available. This aspect is strongly emphasized in the present thesis.

- **Efficiency and potential parallelism** in the use of computing resources. Sparse representations have a smaller memory footprint, and many sparse algorithms have lower computational resources, as showcased by decades of research, for instance in sparse numerical linear algebra (see Bunch and Rose (2014)). With regards to efficiency, we will see how some topological structures such as trees and clique trees are much more expedient to efficient calculations, especially because they encode the idea of *local* interactions, and because they are in general more consistent with respect to calculations. In a decomposable graphical model[8] the local consistency of the marginal distributions implies global consistency. In non-decomposable models the presence of *loops* or *hyperloops* leads generally to more complex and less stable algorithms, and always to approximate solutions. Moreover, in many cases the geometry of the network suggests orderings of the variables or of the cliques or clusters that are well suited to exact inference (Lauritzen and Spiegelhalter, 1988).

1.3 Network Modelling: Approaches to Filtering and Applications

We now look at how the problem of providing a sparse representation of a complex network has been approached in the disciplines of Econophysics, Statistics and Topology. While the problem is essentially stated in similar terms, the three approaches are different in nature and provide unique insights into the properties of the network[9]. In Section 1.3.1 we provide a more detailed overview of the subject of planarization of dense networks, since this is the initial approach from which the main thrust of this thesis has originated; besides, some early published results of our research have found application in different fields and constitute, therefore, a good example of the interplay between theory and application.

- **Echonophysics: Information Filtering Networks**. One approach to filtering is achieved by imposing a topological constraint on the modelled network (Lecoutre, 2013; Hleap and Blouin, 2014; Mantegna, 1999; Tumminello et al., 2005). Mantegna (1999) models the correlation structure of a portfolio of traded stocks by creating a tree that spans the components of the portfolio. The tree induces a hierarchical order on the portfolio, which is used to analyse the structure of common factors and clusters of related stocks. Tumminello et al. (2005) take the next logical step by imposing a constraint on the topological genus of the graph and analyse in depth the *planar* case (genus 0); the planar graph has a

[8]We will see in 3 that a decomposable graphical model is a graphical model where the underlying dependency structure is chordal.

[9]These approaches are those that have influenced the most our research, but they are by no means exhaustive of the variety of methodologies adopted (Fan et al., 2011; Huang and Aviyente, 2007; Deng et al., 2013; Carmi et al., 2014)

richer structure than a tree but is still sparse. The analysis of the closed curves on the planar graph allows to define a nested hierarchy of subgraphs ("bubbles") that is useful in the study of communities and their relationship. In further developments Aste et al. (2005a) show the importance of embeddings in hyperbolic surfaces with increasing topological genus. The availability of a topological structure allows analysing the overall structure of the network: degree distribution, scaling properties, hubs, hierarchical structure, clustering, small worldness (Newman, 2010). The road of looking at embeddings of increasing topological genus is a way to generalise the concept of a tree. In the present thesis we explore another generalisation of the concept of a tree: if we look at a tree as a graph with no cycles we might regards the next step as a graph with the smallest possible cycles and this leads to the concept of a *chordal graph*, which is a graph where the cycles are the smallest possible (Galinier et al., 1995), and where every cycle longer than four must have a chord.

- **Statistics: Graphical Models**. A second way to look at the filtering problem is to understand it as the task of "learning" the structure of the underlying graphical model (Lauritzen, 1996; Koller and Friedman, 2009; Barber, 2012; Wainwright and Jordan, 2008)[10]. An (undirected) graphical model (or Markov Random Field) is a collection of random variables $X_1, X_2, \ldots, X_p$ associated to the nodes of a graph. The edges represent a direct association between the variables (Giudici and Spelta, 2016). A very effective and expanding field of research in statistics is the *regularisation method*[11]: this method works by optimising a score function (usually likelihood) penalised using some convex norm of the size of the model parameters: the penalisation forces the model parameters to be small or zero, depending on the norm used. In the field of graphical models the most important regularisation method is the *graphical lasso* (Friedman et al., 2008). The ability to associate a statistical model to a network widens considerably the range of analysis that can be performed, especially in the case of financial networks: the many inference tools available in graphical models allow the calculation of

[10]As Lauritzen (1996) puts it, conditional independence can be described as formal theory of irrelevance.

[11]The idea of regularisation has been brought to bear in the mathematical sciences by Tikhonov (Tikhonov, 1943).

marginal and conditional probabilities. opening the door to scenario analysis, stress testing and portfolio management.

- **Topology: Computational Topology**. When we look at some biological, brain-structural and collaboration networks (Barzel et al., 2013; Courtney and Bianconi, 2016, 2017) we observe that some characteristics, such as clustering and scaling, suggest the hypothesis that the underlying system may be governed by interactions between more than two nodes, involving cliques (triangles, tetrahedra and higher dimensional simplexes). If there is a need to emphasize this aspect of a network, a standard and convenient representation is a *simplicial complex* (Munkres, 1984). In the Topological Data Analysis community there are several methodologies, mostly based on homology theory, to build simplicial complexes from networks of potentially noisy data (Zomorodian and Carlsson, 2005; Carlsson, 2009; Otter et al., 2017). The approach pursued is to study the topological features of the simplicial complex while the homology scale parameter varies in a range, so that it is possible to isolate persistent features from the transient ones. Persistent features are good indicators of higher order structures in the data. This approach is used for instance in neural data analysis (Giusti et al., 2015) and in the use of correlation networks to study critical transitions in financial markets (Gidea, 2017). In some cases previous experiments might suggest the existence of "topological motifs" that should be taken as given and incorporated *a priori* in the network being analysed and this requirement too, can be formulated as a constraint on the network topology (Fiori et al., 2012; Milo et al., 2002). Finally, a richer topological and geometrical structure is amenable also to tools from discrete differential geometry and discrete Morse Theory (Wang and Wei, 2016) and allows for instance a local analysis on the basis of intuitive geometric concepts such as curvature (Knill, 2010, 2012) and critical points. Sandhu et al. (2016) studies the properties of the discrete Ricci curvature in financial networks as an indicator of market fragility and systemic risk.

1.3.1 Planar Information Filtering Networks

In the field of Econophysics a particularly important topic is the generation of IFNs by planarization of dense networks. The motivation for building and using planar networks reaches across different disciplines:

- Analysis of financial data, where nodes generally represent financial assets (such as stock prices, spreads, liabilities, risk or liquidity indicators etc.) and the edges represent correlations or other measures of dependence between them (Aste et al., 2005b; Tumminello et al., 2005; Pozzi et al., 2013; Fiedor, 2014);

- Facilities layout, where nodes represent the facilities and the edges the affinities between them (Foulds and Robinson, 1976, 1978; Liebers, 2001);

- Integrated circuit design, where nodes are the electrical elements and connections are the physical connections (Lengauer, 1990);

- Systems biology, where nodes can represent proteins and edges protein interactions in a metabolic network (Song et al., 2007);

- Social systems, where nodes represent social agents (e.g. individuals, companies, groups) and edges represent social interaction (see Easley and Kleinberg (2010) and Jackson (2010) for a detailed overview).

In Chapter 4 we describe the TMFG methodology, originally developed with this thesis, which is an alternative development to the Planar Maximally Filtered Graph (PMFG) described in Tumminello et al. (2005) and that has been applied in a number of different fields, such as:

1. Modelling of multivariate gaussian distributions with applications in Risk management (see Barfuss et al. (2016) and Section 5.2);

2. Studies on the structure of correlation in financial networks (Musmeci et al., 2014a, 2015a,b, 2017);

3. Parallel algorithms for the approximate solution to the Maximum Weight Planar Subgraph Problem (MWPSP) (Coelho et al., 2016);

4. Psychometric and network science in psychology, (Golino et al., 2018; Christensen et al., 2018c; Pelowski et al., 2019; Christensen et al., 2018b,a);

5. Macroeconomics and policy assessment using spillover networks (Castañeda Ramos and Guerrero, 2018a; Castañeda Ramos et al., 2018; Guerrero and Castañeda Ramos, 2018; Castañeda Ramos and Guerrero, 2019, 2018b);

6. Spatial networks methods in the study of fluid turbulence analysis (Iacobello et al., 2017).

In some domains, such as facility layout or integrated circuit design, the constraint of planarity is a direct consequence of the two-dimensional geometry of the problem, while in other domains, network planarity is a way to constraint the complexity of the graph reducing the degree of interwovenness (Aste et al., 2005b).

Planar filtered graphs are powerful tools to study the geometric structure of complex datasets. It has been shown in Song et al. (2012a) that by making use of the 3-clique structure of the PMFG a clustering structure can be extracted, allowing dimensionality reduction that keeps both local information and global hierarchy in a deterministic manner without the use of any prior information. Applications of planar filtered graphs to financial datasets can meaningfully identify industrial sectors and structural market changes (Musmeci et al., 2014b, 2015b). Planar filtered graphs can be used to diversify financial risk by building a well-diversified portfolio that effectively reduces risk by investing in stocks that occupy peripheral regions of the graph (Pozzi et al., 2013).

There are other areas of application where the planarity of a graph can be exploited to obtain better performance or outcomes. These algorithms have many applications for instance in scheduling or optimization (see Nishizeki and Chiba (1988)). Planarity ensures easy visualization of the network with the possibility to draw the network without edge-crossing, thus allowing for exploratory data analysis of networks. Another appealing advantage of planar filtered networks concerns graphical modelling where planarity (which limits the treewidth of the filtered graph) grants that some exact inference algorithms can be performed in an efficient fashion. Jaakkola (2007) describes specific inference algorithms for graphical models based on planar graphs, but we argue that the MFCF, described in detail in Chapter 4, is a better choice for limiting the treewidth of a graph and offers a more direct connection to inference algorithms.

1.4 Objective of the Research and our Approach

The objective of our research is the design of effective and efficient algorithms for information filtering networks which subsume the three approaches described in Section 1.3, enabling therefore a researcher to build networks that can be analysed using a large and complementary set of tools. We summarize in the next paragraphs the detailed requirements for an IFN algorithm, we set out our approach to the research problem and finally point to actual and possible applications that are described in the following chapters.

1.4.1 Fundamental requirements for a new IFN algorithm

On the basis of the research problems and approaches described in the previous section we assume a broad set of requirements for the IFN algorithm:

1. The information network filtering procedure should be specified in terms of topological constraints. Since we want to be able to enforce sparsity, the methodology should allow to control the density of the network.

2. The information filtering network produced must allow the use of efficient inference algorithms, in the sense of graphical models, whenever a multivariate probability distribution can be defined for the network data; we want the network to be such that the inference and simulation tasks can be performed efficiently.

3. The networks produced should be tractable with the tools of Topological Data Analysis. In particular the output should be a simplicial complex, and the complexity of the complex should be controlled by the topological constraints.

4. The algorithms used to produce and use the IFNs can be applied to large networks with realistic performances, be parsimonious in the use of computing resources and parallelizable.

5. The algorithms should be general and easily customized to different types of data and networks. In particular, it should grant the researcher the possibility to specify how to define the association between variables.

6. The algorithms should be able to incorporate hypothesis on the data structure and expert judgement by including topological motifs.

These requirements are in our view a deep generalisation of the characteristics of the TMFG algorithm (Massara et al., 2016) which has been developed in the first part of our research and that is described in full in Chapter 4.

1.4.2 Approach and Direction of Research

The general idea is to build networks by repeated addition of vertices using only operations that are invariant with respect to some topological property (planarity in the case of the TMFG, chordality in the case of the MFCF). We have identified in the Clique Forest[12] the data structure that is most promising in addressing the research problem due to the many expedient properties of its geometry. In particular:

- The maximum size of the cliques allowed in the forest is a natural topological penalizer and the *treewidth* (= size of largest clique - 1) is related to the complexity of probabilistic inference (Bach and Jordan, 2001) that can be performed on the forest, as well as the minimum number of samples required to obtain positive definite estimated of correlation matrices in the important case of multivariate Gaussian distributions.

- The graphs underlying the clique forests are *chordal*, or *triangulated* graphs and this means that:

  1. Chordal graphs are a case of *perfect graphs* and they have polynomial time solutions for a number of combinatorial problems that would otherwise be intractable (for example: graph coloring, maximum clique, maximum independent set), see Golumbic (2004).

  2. Graphical models with underlying chordal graphs are *decomposable*, have a number of convenient properties, such as a convenient factorisation of the probability density in terms of the clique marginal, and do not require the calculation of the partition function (Lauritzen, 1996).

  3. Graphical models represented by clique forest are amenable to exact inference with the *junction tree algorithm*.

[12]The Clique Forest is better known in the Literature as the Clique Tree (Blair and Peyton, 1993), or the *junction tree* (Lauritzen, 1996).

4. Gaussian decomposable models are particularly suited to sampling from large systems (Jones et al., 2005).

- The clique tree is a particular case of a simplicial complex, the *clique complex* of a graph (Bandelt and Chepoi, 2008), which opens the possibility to analysis of data based on tools from computational topology.

In Chapter 3 we develop an algorithm for learning the structure of clique forests based on the idea of *topological penalisation*. Specifically, the algorithm allows to effectively control the l_0 semi-norm of the model, a goal which is impossible to achieve with regularisation methods based on the l_1 and l_2 norms. As a consequence it is possible to cleanly separate structure learning and parameters learning, while the other regularisation approaches achieve sparsity by shrinking the model parameters too. The algorithm is based on the geometry of a clique tree, the space of the allowed solutions is explored by iterative application of one or more topological operators (e.g. *clique expansion operator*). Any topological constraint (such as planarity, chordality) must be an *invariant* of the top ological operators used. The algorithm is formulated in a way that generalises the preferential attachment network construction method (Albert and Barabási, 2002) whereby vertices are added one by one to a growing network. The choice of the next vertex and the next attachment point is driven by a gain function, which represents the gain in score resulting from the application of the clique expansion operator. The algorithm can be seeded with an initial clique tree and therefore topological motifs can be easily incorporated. We have made an effort to keep the distinction between the geometry of the network and the analytical forms of the scoring functions used by using a *generic programming* approach, so that the algorithm can be used, without change, with any scoring function compatible with the definition of the clique expansion operator.

1.4.3 Applications

We provide four applications. Firstly, in Chapter 4 we provide an approximate solution to the Maximum Weight Planar Subgraph Problem (MWPSP) (Castonguay et al., 2017); secondly, in Chapter 5 we show some applications to the risk management of financial portfolio (Barfuss et al., 2016); thirdly, in Chapter 6 we use the algorithm to find solutions to the Covariance Selection problem (Dempster, 1972); finally, in Chapter 8 we discuss some applications in Topological Data Analysis.

Chapter 2

Approaches to the Modelling of Sparse Networks

In this chapter we provide, initially, some background information on Undirected Graphical Models, with a specific focus on the relationship between the conditional independence of the random variables as specified by the Markov properties and the factorisation of the underlying probability density. We also introduce the important class of decomposable models.

Next, we proceed to define the problem of structure learning and we review the main approaches in the fields of structure learning in graphical models and information filtering networks. More technical details on decomposable models required by our methodology will be provided in Chapter 3.

2.1 Graphical models and Markov Random Fields

2.1.1 Definitions

Definition 1 (Graphs and edges)**.** A graph $G = G(V,E)$ is defined by a collection of p vertices (or nodes) $V = \left\{v_1,\ldots,v_p\right\}$ and a collection of m edges $E \subset V \times V$. A directed edge is an ordered pair of vertices (v_i, v_j), and we say that the edge goes form v_i to v_j; an undirected edge is an unordered pair of vertices, so that (v_i, v_j) is the same as (v_j, v_i). If $(v_j, v_i) \in E$ we say that v_i and v_j are *adjacent*. A graph that contains only undirected (resp. directed) edges is called an *undirected* (resp. *directed*) graph. In this thesis by graph we will mean, unless otherwise specified, an undirected graph.

Definition 2 (Neighbours and closure)**.** The *neighbours* of a vertex $v_i \in V$ in an undirected graph $G(V,E)$ are the vertices adjacent to v_i, $\mathrm{neib}(v_i) = \{v_j \in V \mid (v_i,v_j) \in E\}$. If A is a subset of V then the neighbours of A are defined as $\mathrm{neib}(A) = \cup_{v\in A} \mathrm{neib}(v) \setminus A$. The *closure* of a set of vertices $a \subset V$ is defined as $\mathrm{cl}(A) = A \cup \mathrm{neib}(A)$.

Definition 3 (Cliques)**.** A *clique* is a subset $C \subset V$ of vertices that are fully connected or *complete*, that is $\forall v_i, v_j \subset C, (v_i,v_j) \in E$, and C is not a proper subset of any other complete set. In some cases cliques are defined as the complete subsets of a graph, relaxing the requirement of being maximal subsets. In this case what we call cliques are called "maximal cliques". For us it is convenient to use the work cliques for maximal complete subgraphs. We will denote with $\mathscr{C}(G)$ the set of the cliques of the graph G.

Definition 4 (Undirected graphical models)**.** An undirected graphical model ("UGM"), or Markov Random Field ("MRF") (Lauritzen, 1996; Wainwright and Jordan, 2008; Koller and Friedman, 2009; Drton et al., 2018) $\mathscr{H}$ is composed of an undirected graph $G(V,E)$ and a collection $\mathbf{X} = \{X_1,\ldots,X_p\}$ of random variables. Each vertex v_i is associated with a random variable X_i; an edge between two vertices v_i and v_j represents a probabilistic relationship between the two associated variables X_i and X_j. We indicate the space where the variable X_i takes values as $\mathscr{X}_i$. For instance $\mathscr{X}_i$ could be the real number system $\mathbb{R}$ or a discrete set $\{x_1,\ldots,x_n\}$. Specific values in $\mathscr{X}_i$ are designated using the corresponding lower case variable x_i. The expression $\{X_i = x_i\}$ means the event that the variable X_i assumes the value x_i. If C is a subset of V we indicate with $\mathbf{X}_C \in \mathbf{X}$ the sub-vector of the variables indexed by C, $\mathbf{X}_C = (X_c)_{c\in C}$, and with $\mathscr{X}_C$ the subspace where $\mathbf{X}_C$ takes values.

Definition 5 (Directed graphical models: Bayesian Networks)**.** A Bayesian Network is an example of a *directed graphical model.* In this example the edges are oriented from a *parent* X_p to a *child* vertex X_c and it is only possible to condition the child upon the parent $(X_c \mid X_p)$. The edge represents therefore the conditional dependence relation.

2.1.2 Conditional Independence

Definition 6 (Conditional independence for random variables with positive density)**.** If X, Y, and Z are random variables with a joint density $f_{XYZ}(x,y,z)$ [1] with respect to a product measure μ we say that X and Y are independent conditionally on Z, and

[1] And associated marginal densities $f_X(x)$, $f_{XY}(x,y)$, and so on.

we write $X \perp\!\!\!\perp Y \mid Z$, when Equation 2.1 holds almost surely or, if the densities are continuous, for all z such that $f_Z(z) > 0$.

$$X \perp\!\!\!\perp Y \mid Z \iff f_{XYZ}(x,y,z) f_Z(z) = f_{XZ}(x,z) f_{YZ}(y,z) \tag{2.1}$$

The intuition behind conditional independence is that, knowing Z, Y does not provide any further information about X. This happens for example when both X and Y are influenced by Z but do not interact directly with each other, as shown in Example 1. An intuitive example is the case where X and Y describe the temperature measured by two thermometers in the same room and Z is the temperature in the room; knowing Z we can predict the values of X and Y without having to measure the correlation between X and Y.

There is a large literature regarding the definition of conditional independence in very general settings using probabilistic, algebraic and combinatorial tools (see Lauritzen (1996, Par 3.1), Studeny (2006), and Drton et al. (2018, Ch. 1) and the literature quoted therein), but since in this thesis we will always assume the existence of a positive joint probability density we will use Definition 6 in terms of marginal densities, which has the benefit of being simple and intuitive.

Definition 6 can be specialised to the variables in a graphical model as in Definition 7

Definition 7 (Conditional Independence)**.** Given three mutually disjoint subsets $A \subset V, B \subset V, C \subset V$ of the vertices of a graphical model with associated variables X_A, X_B and X_C and associated densities f_A, f_B and f_C, Equation 2.1 specialises as:

$$X_A \perp\!\!\!\perp X_B \mid X_C \iff f_{ABC}(x_A, x_B, x_C) f_C(x_C) = f_{AC}(x_A, x_C) f_{BC}(x_B, x_C) \tag{2.2}$$

Provided that the marginal density f_C is positive the joint density can be rewritten in Equation 2.3 in a way that formally generalises Bayes' formula[2].

$$f_{ABC}(x_A, x_B, x_C) = \frac{f_{AC}(x_A, x_C) f_{BC}(x_B, x_C)}{f_C(x_C)} \tag{2.3}$$

[2]The original formula from Bayes involves only two random variables and the conditioning is performed on one of the two random variables.

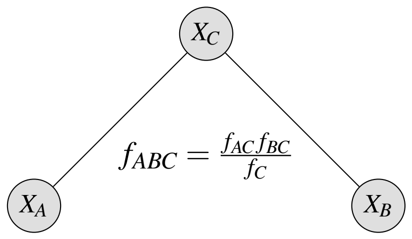


Figure 2.1: Graphical illustration of the conditional independence relationship $X_A \perp\!\!\!\perp X_B \mid X_C$

or, using for instance the *ansatz* $\phi_{AB} = f_{AC}(x_A, x_C)$ and $\phi_{BC} = \frac{f_{BC}(x_B, x_C)}{f_C(x_C)}$:

$$f_{ABC}(x_A, x_B, x_C) = \phi_{AC}(x_A, x_C)\phi_{BC}(x_B, x_C) \tag{2.4}$$

Figure 2.1 illustrates, in the simple case where A, B, and C contain exactly one element each, how the relationship of conditional independence is reflected in the graph structure: since X_A and X_B are independent conditionally on X_C, the graph does not include the edge (X_A, X_B). There could be, in fact, significant correlation between X_A and X_B, but it would be wholly explained by the effects of X_C. We will see that in Gaussian Markov Random Fields the measure linked to the presence or absence of an edge is the *partial correlation coefficient* (Anderson, 1962).

Example 1 (Conditional independence of multivariate Gaussian variables)**.** Let us assume that X_A, X_B, X_C are three random variables with a joint multivariate normal distribution and where X_C has unit variance and $X_A = \alpha X_C + \sqrt{1-\alpha^2}\varepsilon_A$ and $X_B = \beta X_C + \sqrt{1-\beta^2}\varepsilon_B$, with ε_A and ε_B two univariate normal distributions with zero mean and unit variance independent from each other and from X_A, X_B, X_C. The correlation matrix of the joint distribution is:

$$C = \begin{array}{c|ccc|} & X_A & X_B & X_C \\ X_A & 1 & \alpha\beta & \alpha \\ X_B & \alpha\beta & 1 & \beta \\ X_C & \alpha & \beta & 1 \end{array}$$

while the inverse of the correlation matrix is:

$$J = \begin{array}{c} \\ X_A \\ X_B \\ X_C \end{array} \begin{array}{c} \begin{array}{ccc} X_A & X_B & X_C \end{array} \\ \begin{bmatrix} \frac{1}{1-\alpha^2} & 0 & \frac{\alpha}{\alpha^2-1} \\ 0 & \frac{1}{1-\beta^2} & \frac{\beta}{\beta^2-1} \\ \frac{\alpha}{\alpha^2-1} & \frac{\beta}{\beta^2-1} & \frac{1-\alpha^2\beta^2}{(\alpha^2-1)(\beta^2-1)} \end{bmatrix} \end{array}$$

If α and β are close to 1 in absolute value then ρ_{AB}, the correlation between X_A and X_B, will also be close to 1. The partial correlation $\rho_{AB.C}$ is instead equal to 0 because it is the correlation of ε_A, the residuals of the regression of X_A against X_C, and ε_B, the residuals of the correlation of X_B against X_C. The partial correlation $\rho_{AB.C}$ is zero since ε_A and ε_B are independent by construction. In general for a multivariate Gaussian the partial correlation is linked to the elements of the inverse covariance matrix J: $\rho_{AB.C} = -\frac{J_{AB}}{\sqrt{J_{AA}J_{BB}}}$. For more details about the idea of partial correlation see Anderson (1962); Baba et al. (2004).

This elementary case is very important for the understanding of the rôle played by conditional independence, partial correlation and precision matrix in Gaussian graphical models. A network model based on the correlation matrix, maybe with some thresholding, would probably have produced a system of (X_A, X_B, X_C) with a full network, instead of the representation in Figure 2.1. This would have not only "wasted" one parameter, given that the correlation ρ_{AB} is fully determined by ρ_{AC} and ρ_{BC}, but also would have introduced unnecessary noise in the system by trying to parametrise the essentially spurious correlation between the residues ε_A and ε_B[3].

Example 2 (Conditional independence of multivariate Gaussian variables in relation to the zeros of the precision matrix)**.** When the variables X_v follow a multivariate Gaussian distribution the conditional independence of two variables X_a and X_b is directly reflected in the inverse of the covariance matrix (also called *precision* or concentration matrix) J: $X_a \perp\!\!\!\perp X_b \mid (X_r, r \in V \setminus \{a,b\}) \Longleftrightarrow J_{ab} = 0$. The pattern of missing edges corresponds to conditionally independent variables and it is exactly the same as the zero elements in the precision matrix. The problem of estimating a precision matrix is called "Covariance selection" (Dempster (1972)) and the associated graphical models are called Gaussian Graphical Models (GGM) or Gaussian Markov Random Fields

[3]See also Giudici and Spelta (2016) for additional considerations on the effectiveness of graphical models in providing parsimonious representations.

(GMRF).

2.1.3 Markov Properties

We have seen in Example 1 how, in a simple case of three random variables, the structure of a Markov Random Field can encode a relationship of conditional independence. In more complex cases there is a need to specify precisely how to read the conditional independence statements encoded in the graph. Given a Markov Random Field $\mathscr{H}$ with underlying graph G there are three possible choices of the set of variables that can be used as a conditioning set, and each of the three choices leads to three different definitions of conditional independence: the "Markov properties".

- *Global Markov property (GM)*: Any two set of variables are conditionally independent given a separating subset:

  $$X_U \perp\!\!\!\perp X_V \mid X_S$$

  where every path from an element of U to any element of V passes through S.

- *Pairwise Markov property (PM)*: Any two variables X_i and X_j are conditionally independent given all the others:

  $$X_i \perp\!\!\!\perp X_j \mid X_{V \setminus \{i,j\}} \text{ if } \{i,j\} \notin V$$

- *Local Markov property (LM)*: Any variable is conditionally independent from all the other given its neighbours:

  $$X_i \perp\!\!\!\perp X_{V \setminus \mathbf{cl}(V_i)} \mid X_{\mathbf{neib}(V_i)} \text{ where } \mathbf{cl}(V_i) = V_i \cup \mathbf{neib}(V_i)$$

In general (Lauritzen, 1996, Prop. 3.4) the global Markov property implies the local Markov property which in turn implies the pairwise Markov property.

$$\text{Global Markov property} \Rightarrow \text{Local Markov property} \Rightarrow \text{Pairwise Markov property} \tag{2.5}$$

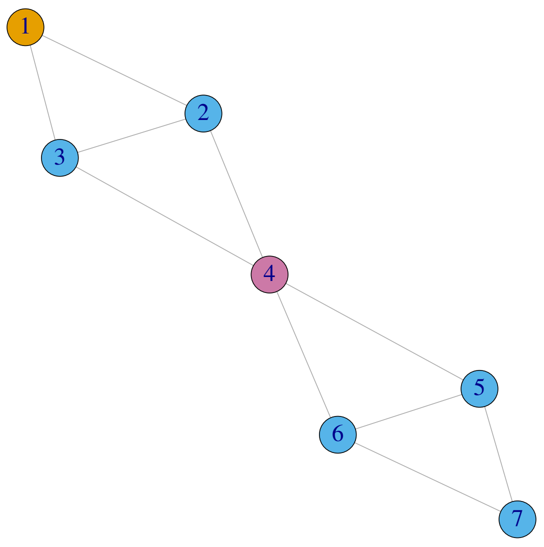


Figure 2.2: Pairwise Markov property. X_1 and X_4 are independent conditionally on $\{X_2, X_3, X_5, X_6, X_7\}$

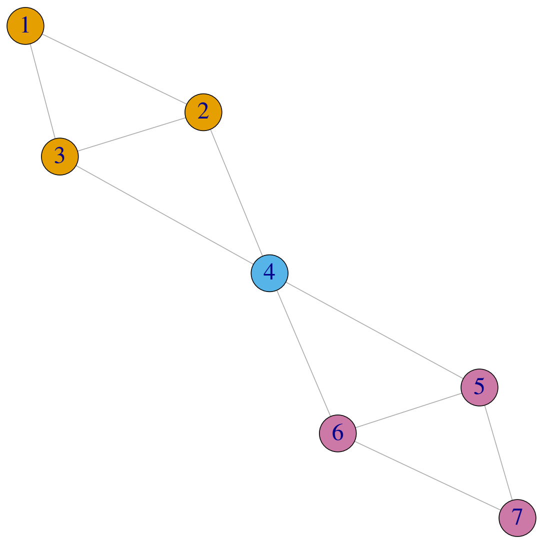


Figure 2.3: Global Markov property. Any of $\{X_1, X_2, X_3\}$ is independent from any of $\{X_5, X_6, X_7\}$ conditionally on X_4.

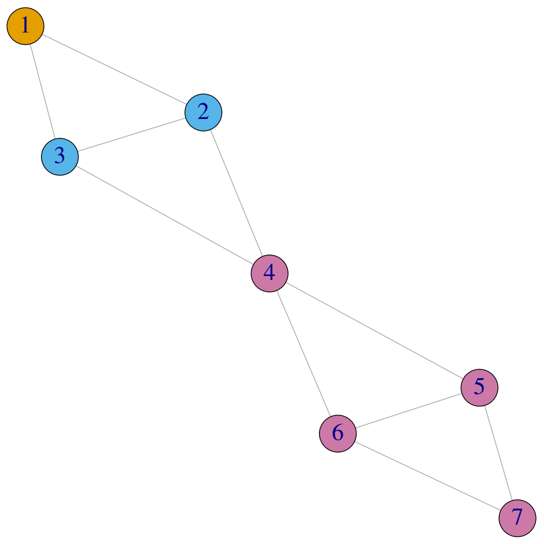


Figure 2.4: Local Markov property. X_1 is independent from $\{X_4, X_5, X_6, X_7\}$ given $\{X_2, X_3\}$

There are examples of discrete probabilities where the reverse implications do not hold (Lauritzen, 1996, Example 3.5-3.6). However, under some additional assumptions, for instance when the probability has a continuous and positive density in the product space, the opposite implications hold as well (Lauritzen, 1996, Th. 3.7).

$$\text{Global Markov property} \Leftrightarrow \text{Local Markov property} \Leftrightarrow \text{Pairwise Markov property} \tag{2.6}$$

2.1.4 Clique Factorisation Property and Hammersley-Clifford Theorem

An MRF models the joint probability distribution of the random variables X_i. A *factor*[4] is a function from a subset of the node variables to $\mathbb{R}^+$. For instance a factor of the three variables X_1, X_2 and X_3 is a function $\phi_{123}(X_1, X_2, X_3)$ with values in $\mathbb{R}^+$.

We say that a distribution *factorizes* over $\mathscr{H}$ if there are factors $\phi_i(C_i)$ over the cliques of $C_i \in \mathscr{C}(G)$ such that

$$P(X = x) = \frac{1}{Z} \prod_{C_i \in \mathscr{C}(\mathscr{H})} \phi_i(x_{C_i}) \tag{2.7}$$

[4]Factors are also called *potentials* or *compatibility functions* in the literature.

where we have

$$Z = \sum_{X_1=x_1,\dots,X_n=x_n} \prod_{C_i \in \mathscr{C}(\mathscr{H})} \phi_i(x_{C_i}) \quad \text{discrete case} \tag{2.8}$$

$$Z = \int_{x_1,\dots,x_n} \prod_{C_i \in \mathscr{C}(\mathscr{H})} \phi_i(x_{C_i}) d\mathbf{x} \quad \text{continuous case} \tag{2.9}$$

Z is called the *partition function* and is such that the function $P(X)$ adds up to 1. Please see Kindermann et al. (1980) for more details.

The Hammersley-Clifford theorem (Lauritzen, 1996) states that Equation 2.7 can be fulfilled if the density $P(x)$ is positive. In this case Equation 2.7 can be expressed as a *Gibbs measure*:

$$P(X = x) = \frac{1}{Z} exp(-U(x)) \tag{2.10}$$

where

$$U(x) = \sum_{C_i \in \mathscr{C}(\mathscr{H})} \log \phi_i(x_{C_i}) \tag{2.11}$$

When the probability distribution P has a positive density in the product space the three Markov properties and the factorisation property are equivalent.

$$\begin{gathered} \text{Global Markov property} \Leftrightarrow \text{Local Markov property} \Leftrightarrow \\ \text{Pairwise Markov property} \Leftrightarrow \text{Factorisation property} \end{gathered} \tag{2.12}$$

Definition 8 (Graph decomposition)**.** Let $A \subset V$, $B \subset V$, $C \subset V$ be three mutually disjoint subsets of vertices of a graph G. For $a \in A$, $b \in B$ we say that C is an (a,b)-separator if every path connecting a and b intersects C; C is a *minimal* (a,b)-separator if it is the smallest subset with such a property. Similarly we say that C separates A and B if C is an (a,b)-separator for every $a \in A$ and $b \in B$. We say that the triple (A,B,C) decomposes $G(V,E)$ if $V = A \cup B \cup C$ and the following conditions hold: 1. C separates A and B and 2. C is a complete graph. As a side note, the requirement of completeness for C might seem redundant, but it turns out to be fundamentally linked to the property of *chordality* of a graph and to desirable properties in the variable elimination process.

Let us assume that the random variable in a MRF $\mathscr{H}$ belong to two non-necessarily disjoint subsets $A \subset V$ and $B \subset V$ such that $V = A \cup B$. Then $(A \setminus A \cap B, B \setminus A \cap B, A \cap B)$ is a partition of V. If $\mathscr{H}$ enjoys the Global Markov property then from Equation 2.2 follows that the probability density function factorises:

$$f(\mathbf{X}) = \frac{f(\mathbf{X}_A) f(\mathbf{X}_B)}{f(\mathbf{X}_{A \cap B})} \tag{2.13}$$

The decomposition of V can be associated with a tree of subsets of V that has two vertices A and B joined by the edge $C = A \cap B$.

In case A and B are in turn decomposable (or fully connected cliques) it can be shown (Lauritzen, 1996, Chap. 3) that the joint probability distribution can be further recursively factored into finer decomposable components until we get to the clique set ($\mathscr{C}$) and separator set ($\mathscr{S}$) of G:

$$P(\mathbf{X}) = \frac{\prod_{c \in \mathscr{C}} P(\mathbf{X}_c)}{\prod_{s \in \mathscr{S}} P(\mathbf{X}_s)} \tag{2.14}$$

This property will be further explored in Chapter 3 in relation to the MFCF algorithm.

Remark 9 (Empty set as a separator)**.** Note that in case two sets of variables are disjoint (unconditionally independent) the corresponding separator is the empty set and the tree structure is a forest.

Remark 10 (Multiplicity of separators)**.** It is possible that a separator appears more than once in a clique forest, for example when it separates more than two cliques. In such a case in our notation the separator set reports the separator more than once, so that the separator multiplicity is automatically taken into account.

2.2 Review of Methodologies for building Information Filtering Networks

We structure our review of methodologies to build information filtering networks from data around four principal approaches: (i) Structure learning algorithms in graphical models, (ii) Sparse graphical models through regularisation and covariance selection, (iii) Information filtering networks, (iv) The Triangulated Maximally Filtered Graph algorithm, which we will describe fully in Chapter 4 as a particular case of the MFCF methodology.

2.2.1 Structure Learning in Graphical Models

Definition 11 (Structure learning)**.** Structure learning in graphical models is the task of inferring from realisations or experiments what is the structure of the conditional independence relations of the vertices, which is represented by the graph underlying the graphical model. This means ascertaining which edges are present in the graph and how well they represent the dependencies in the data set.

The problem of finding the dependency structure underneath a data set is hard in general since an undirected graph on p variables has $a(n) = 2^{(p\cdot(p-1)/2)}$ possible configurations[5]. Structure learning is very difficult also when we restrict the problem to more tractable cases: Karger and Srebro (2001) analyse the problem of learning maximum-likelihoood graphical models in the particular case that the underlying structure is a clique tree of fixed size and show that it is equivalent to the NP-hard problem of finding a maximum weight subgraph of bounded treewidth. Bogdanov et al. (2008) provide NP-hardness results for the problem of reconstructing Random Markov Fields with bounded degree trees and hidden nodes. Chickering (1996) and Chickering et al. (1994) prove that learning bayesian networks (introduced with Definition 5) is NP-complete[6]. It is therefore understandable that a large effort is gone into the development of heuristic and approximate methods.

General approaches to structure learning in Graphical Models can be classified into three main categories (Drton and Maathuis, 2017; Koski and Noble, 2012; Zhou, 2011; Lauritzen, 2012; Scutari and Strimmer, 2011; Koller and Friedman, 2009): (i) *score based*, (ii) *constraint based*, and (iii) *Bayesian methods*. Let us here briefly introduce and comment them one by one.

2.2.1.1 Score based methods

Score based algorithms perform structure learning by detecting edges or other structures that optimize some global function such as likelihood, Kullback-Leibler divergence (Kullback and Leibler, 1951), Bayesian Information Criterion (BIC) (Schwarz

[5]The number of graphs on p labeled nodes (Harary and Palmer, 2014)

[6]Bayesian networks are, however, *directed* graphical models and structure learning is further complicated by the check for acyclicity in the resulting network.

et al., 1978), Minimum Description Length (Rissanen, 1978) or the likelihood ratio test statistics (Petitjean and Webb, 2015). In general, the identification of the structure that optimises the score function results in a difficult combinatorial optimization problem (Koller and Friedman, 2009, Ch. 20) and some sort of greedy approach should be implemented to produce a sequence of steps that optimize a limited space of solutions.

Chow and Liu Trees. One of the leading methods in the score based sparse representation of joint probability distributions is based on *Chow-Liu* trees (CLT). In the original paper, Chow and Liu (1968) proposed a factorisation of a probability distribution where a p-order discrete distribution is approximated by the product of a number of second-order distributions.

There are $\frac{p\cdot(p-1)}{2}$ second order marginal distributions available. The authors propose to approximate the joint probability density P with the product Q of $n-1$ bivariate densities such that the underlying graph $G(V,E)$ is an oriented tree. In this approach there are *in nuce* several ideas that we will develop in this thesis: one idea is the approximation of a complex probability distribution with the product of simpler marginal distributions; a second idea is to force the sparsity of the representation by enforcing a constraint of a topological nature. The particularly simple topology of the proposed approximate solution and the formulation of the problem in terms of a rooted and directed graph allows to deal with the dependency issues making use of simple conditional probability, without the need for the full toolset of the theory of graphical models. The probability P to be approximated is in general unknown and therefore the marginal bivariate probabilities need to be estimated from the observations (the authors use the empirical observed marginal distributions $\hat{P}$).

$$Q(\mathbf{X}) = \hat{P}_r \prod_{v\to w\in E} \hat{P}(X_w|X_v) \tag{2.15}$$

where $\hat{P}_r(X_r)$ is the probability distribution of the root of the tree, and E is the edge set of a tree spanning the nodes associated to the p variables $\mathbf{X}_V$.

Equation 2.15 is similar to how the probability density was written in the original paper, but a different representation in terms of marginal densities of dimension 1 and 2 is more adequate for our purpose:

$$Q(\mathbf{X}) = \prod_{(v,w)\in E} \frac{\hat{P}_{vw}(X_v,X_w)}{\hat{P}_v(X_v)\hat{P}_w(X_w)} \prod_{v\in V} \hat{P}_v(x_v) \tag{2.16}$$

Equation 2.16 is mathematically equivalent to Equation 2.15 in that it is represented by the same density function, but we observe that in the new formulation there is not a “preferred” direction in the underlying graph and all the vertices play a similar role, without the unwarranted need for a precise ordering, nor do we need to store and manage the ordering of the vertices.

The paper proposes to find the approximate distribution $Q(\mathbf{X})$, among all the possible distributions on the trees on the p variables, that minimises the Kullback-Leibler divergence of Q relative to the P distribution $D_{KL}(P \parallel Q) = \sum_{x\in\mathscr{X}} P(x) \log \frac{P(x)}{Q(x)}$

$$\begin{aligned}
D_{KL}(P \parallel Q) &= \sum_{x\in\mathscr{X}} P(x) \log \frac{P(x)}{Q(x)} \\
&= -\sum_{x\in\mathscr{X}} P(x) \log Q(x) + \sum_{x\in\mathscr{X}} P(x) \log P(x) \\
&= -\sum_{x\in\mathscr{X}} P(x) \left(\sum_{(v,w)\in E} \log \frac{P_{vw}(x_v,x_w)}{P_v(x_v)P_w(x_w)} \right) \\
&- \sum_{x\in\mathscr{X}} P(x) \log P_v(x_v) + \sum_{x\in\mathscr{X}} P(x) \log P(x) \\
&= -\sum_{(v,w)\in E} P_{vw}(x_v,x_w) \log \frac{P_{vw}(x_v,x_w)}{P_v(x_v)P_w(x_w)} \\
&- \sum_{v\in V} P_v(x_v) \log(P_v(x_v)) + \sum_{x\in\mathscr{X}} P(x) \log(P(x)) \\
&= -\sum_{(v,w)\in E} D_{KL}(P_{vw} \parallel P_v \cdot P_w) - \sum_{v\in V} H(P_v) + H(P)
\end{aligned} \tag{2.17}$$

The last two addends in Equation 2.17 are independent of the tree structure and therefore if we want to minimise the distance $D_{KL}(P \parallel Q)$ we have to find the tree for which $\sum_{(v,w)\in E} D_{KL}(P_{vw} \parallel P_v \cdot P_w)$ is maximum[7]. It can also be shown (Drton and Maathuis, 2017) that maximising the total empirical mutual information is equivalent to maximising the empirical likelihood. This problem can be solved exactly and efficiently using an algorithm that solves the Maximum Spanning Tree problem for a weighted graph whose edge weight is the Kullback-Leibler distance (or mutual infor-

[7]The Kullback-leibler distance is positive.

mation) of the variables associated with the vertices. Tarjan (1983, Chapter 6) provides an excellent exposition of the classical algorithms available for solving the Minimum Spanning Tree problem. At the time of writing the fastest non-randomized algorithm available for solving the problem is due to Chazelle (2000).

Remark 12 (Extension of CLT to normally distributed variables)**.** The paper by Chow and Liu is limited to the approximation of discrete variables, but the theory holds for a more general class of distributions, for which it is possible to calculate the Kullback-Leibler distance. For instance when the marginals can be taken as bivariate normal, then the Kullback-Leibler distance between two variables is $D_{KL}(P_{vw} \parallel P_v \cdot P_w) = -\frac{1}{2}\log(1 - r_{vw}^2)$, with r_{vw} the empirical correlation of the two variables. Mantegna (1999) uses a very similar distance function to build the Minimum Spanning Tree to analyse a portfolio of stocks. Note also that the Kullback-Leibler distance in this simple case is proportional to the logarithms of the determinant of the correlation matrix or, using a different interpretation of the minus sign, it is proportional to the logarithm of the determinant of the *inverse* correlation matrix.

Remark 13 (Computational efficiency for Chow-Liu Trees)**.** A natural development of the CLT methodology is to consider the product of marginal probabilities of dimension greater than two. In this case the already mentioned result of Karger and Srebro (2001) shows that there is no tractable algorithm to solve the problem exactly. The case of dimension 2 stands out for its tractability. The many further developments described below are therefore based on heuristics.

Ku and Kullback (1969) extended the CLT for discrete probability distributions by allowing the use of marginal probabilities of order greater than two. They used the Kullback-Leibler divergence as a scoring function. They propose to approximate the distribution by means of an iterative procedure of "adjusting the marginals" over a given set of cliques. The Chow-Liu algorithm is recovered as one of the first approximations when only some marginals are available. In this approach the topological constraint described by Chow and Liu is discarded and the consequence is that there is no longer a closed form for the approximate density.

Huang et al. (2002) propose an improvement of the CLT where the initial tree is an approximation of the distribution and where frequent itemsets are successively joined in a single node in the dependence tree structure (and the edges are adjusted accordingly). The result is called Large Node Chow-Liu Tree (LNCLT), and includes the idea of structuring closely associated nodes into a larger clique structure.

In further applications the Chow-Liu theory has been further developed in different directions, including discrete time-series (Kirshner et al., 2004), clustering (Chan and Liu, 2015), generalisation to non-discrete random variables using Minimum Description Length (Suzuki, 2010), feature selection (Schaffernicht et al., 2007), and kernel density estimation (Liu et al., 2011). The statistical learning theory of Chow-Liu trees is presented in detail in Koski (2010).

Decomposable graphical models. As already mentioned, decomposable graphical models have desirable properties and therefore there are several score based algorithms dedicated to learning their structure. In this case the methodology requires to to examine only the chodal configurations: in this area there are a number of methods that efficiently explore the graphical structure (directed, in the case of Bayesian networks, or undirected, in the case of log-linear or multivariate Gaussian models) with the help of suitable graph algorithms based on the manipulation of data structures representing junction trees or clique graphs (Giudici and Green, 1999; Deshpande et al., 2001; Petitjean and Webb, 2015). The structure of these algorithms is usually based on an operation of edge addition: every candidate edge is assessed with regards to the increase in score and to the compatibility with the chordal structure. To our knowledge there are no methodologies for the learning of decomposable models that work as a preferential attachment model, which is very common in other areas of network science (Barabási and Albert, 1999; Dorogovtsev, 2010).

The theory about which edges can be removed from a decomposable model without loosing the decomposability is well established (Lauritzen, 1996, Lemma 2.19) and relatively easy and allows the design of *backward selection* algorithms (Mezzini and Moscarini, 2010; Malvestuto, 2012). Along the same lines Kovács and Szántai (2013) describe a “pruning” approach for multivariate discrete distributions which removes links iteratively refining a junction-tree, optimising the total correlation (“information content”).

Szántai and Kovács (2013) developed an algorithm specialised for a particular clique tree (the “t-cherry” junction tree) and used it to approximate a multivariate discrete distribution. A t-cherry junction tree is a clique tree with a regular structure where all the cliques are of the same dimension.

To the best of our knowledge all structure learning methods[8] for decomposable models deal with the chordality constraint on an edge-by-edge basis and, differently from the approach proposed in this thesis, do not model the clique forest as an aggregation of cliques.

2.2.1.2 Constraint based Algorithms

Constraint based algorithms often start from a complete model and adopt a *backward selection* approach by testing the independence of vertices conditioned on subsets of the remaining vertices (e.g. in the Spirtes-Glymour-Scheines (SGS) and Peter-Clark (PC) (Spirtes et al., 2000; Zhou, 2011) algorithms) and removing edges associated to vertices that are conditionally independent; the algorithm stops when some criteria are met—e.g. every vertex has less than a given number of neighbours. Conversely *forward selection* algorithms start from a sparse model and add edges associated to vertices that are discovered to be conditionally dependent. An hybrid model is the Grow-Shrinkage (GS) algorithm where a number of candidate edges is added to the model (the "grow" step) in a forward selection phase and subsequently reduced using a backward selection step (the "shrinkage" step) (Margaritis and Thrun, 2000; Zhou, 2011). The complexity of checking a large number of conditional independence statements makes these methods unsuitable for graphs with a large number of vertices. Furthermore, aside from the complexity of measuring conditional independence, these methods do not generally optimize a global function, such as likelihood or the Akaike Information Criterion (Akaike, 1974, 1998) but they rather try to exhaustively test all the conditional independence properties of a set of data and therefore are difficult to use in a probabilistic framework.

2.2.1.3 Bayesian Methods

Bayesian methods consider the presence or absence of an edge in the inference network structure as a random variable. More precisely (Madigan et al., 1995) the likelihood of a model result from the product of a discrete probability distribution over the space of all graphs ($P(G)$ and a continuous distribution of the random variables

[8]With the exception of Szántai and Kovács (2013).

conditional on the graph structure $P(X,G) = P(G)P(X|G)$. Usually the probability over the graph structure is taken as uniform, meaning that each graph is equally probable. Applying Bayes rule the posterior probability of a graph can be taken as $P(G,X) \propto P(X|G)P(G)$. If the probaility on all graphs is the same optimising the probability of the graph is equivalent to optimising the marginal likelihood $P(X|G)$ (see for instance Madigan et al. (1995); Eaton and Murphy (2012) and references therein).

2.2.2 Sparse graphical models through regularisation and covariance selection (Lasso, Ridge, Elastic Net)

Regularisation approaches tend to optimise the *penalised likelihood* of a model. The penalty term is the product of a regularisation parameter and a norm of the coefficients. The idea of regularisation can be traced back to the seminal paper from Tikhonov (Tikhonov, 1943). Specifically ridge regression uses a ℓ_2-norm penalty; instead the *lasso* method (Tibshirani, 1996) uses an ℓ_1-norm penalty and the elastic-net approach uses a convex combination of ℓ_2 and ℓ_1 penalties (Zou and Hastie, 2005). These approaches are among the best performing regularization methodologies presently available. The ℓ_1-norm penalty term favours solutions with parameters with zero value leading to models with sparse parameters.

In the field of Gaussian Graphical Models the problem of learning the structure exploits the link between edges and zero-elements of the precision matrix; the general idea is to maximise the likelihood of the multivariate normal distribution (which can be expressed in terms of the sparse inverse covariance matrix) penalised by a non-decreasing function of the number and weight of the non-zero elements in the precision matrix.

Given S, the sample covariance matrix of a number of observations, and J, an estimate of the precision matrix the likelihood of the distribution over the data is given by

$$\mathscr{L}(X|J) = \log\det(J) + \mathrm{Tr}(SJ) + \lambda \parallel J \parallel_p \tag{2.18}$$

Sparsity is controlled by regularization parameter $\lambda > 0$[9]; the larger the value of

[9]The graphical lasso methodology allows to have a specific regularisation parameter for every entry in the precision matrix λ_{ij}, but in most practical uses the same parameter is used for all the matrix

the parameter the more sparse the solution becomes.

The most successfull approach to the optimisation of penalised likelihood is the popular *graphical lasso* (Glasso) (Friedman et al., 2008). This approach is extremely popular and it has developed from a large body of literature with several novel algorithmic techniques that are continuously advancing this method (Meinshausen and Bühlmann, 2006; Banerjee et al., 2006, 2008; d'Aspremont et al., 2008; Ravikumar et al., 2011; Hsieh et al., 2011; Oztoprak et al., 2012).

Remark 14 (Practical issues in model selection with the graphical lasso)**.** In practical applications there are two issues with the graphical lasso. There is not a statistical tests that helps setting the value of the lasso penalty to a specific value. Usually this value is set using cross-validation. If a cross-validation sample is not available the link between the penalty parameter and the sparseness of the graph is not motivated by the data and must be set by the researcher. A second aspect is that the method performs structure and parameters learning at the same time, since the parameters are "shrunk" up to the point where some reach zero. In many circumstances it would be more helpful to be able to decouple the tasks of learning the structure of the graph and the weights of the edges.

2.2.3 Information Filtering Networks

With Information Filtering Networks we refer to a set of approaches aimed at cleaning correlation matrices by simplifying the structure retaining a sparse network with the most relevant correlations only. This methodology originated in the Econophysics community where the interest in modelling dependence stems from studies on the spectral properties of the correlation matrix of financial portfolios (Laloux et al., 1999), focusing on cleaning methodologies inspired by Random Matrix Theory (RMT) (Bun et al., 2017). An alternative approach has been to use tools from topology to investigate the structure of financial markets. One seminal idea (Mantegna, 1999) was to use the Minimum Spanning Tree algorithm to build a hierarchical tree structure that retains the largest correlations. In further developments other topological constraints have been investigated, notably imposing the planarity of the filtered network (Tumminello et al., 2005) and studying hyperbolic embeddings (Aste et al., 2005a; Tumminello et al.,

elements.

2007). These methodologies have enabled the study of several properties of financial portfolios with applications to portfolio diversification (Pozzi et al., 2013; Musmeci et al., 2015b), clustering (Musmeci et al., 2015a; Song et al., 2012a) and dynamics of correlation in markets (Aste et al., 2010b).

2.2.4 Triangulated Maximally Filtered Graphs

In Massara et al. (2016) we proposed a greedy algorithm that builds a Triangulated Maximally Filtered Graph by recursively adding vertices to a k-width tree while minimising a given score function (which in a particular probabilistic application is the Kullbak-Leibler divergence). In Barfuss et al. (2016) this general algorithm was applied to the approximation of multivariate normal distributions by using the multivariate normal Kullback-Leibler divergence as a scoring function; in the same paper some basic results on Gaussian Markov random fields are used to provide applications to financial portfolio modelling. The TMFG produces planar and chordal networks by restricting the size of the cliques and clique-intersections and by constraining the topology of the clique tree. Christensen et al. (2018c) carry out a comparison of the graphical lasso and information filtering networks based on TMFG from the point of view of psychometric networks showing that TMFG have better interpretability. The work in the present paper is a radical generalisation of the TMFG algorithm where the size of the clique is no longer a constraint but an adjustable parameter that can be tuned to the data, and the size and use of separators is driven by the gain in score.

Chapter 3

Learning Clique Forests

We begin this chapter by outlining some fundamental results regarding chordal graphs and clique forests. The topic is extensive and the applications are numerous and encompass various fields, therefore we introduce only the concepts required for a sound understanding of the functioning and the outputs of the MFCF algorithm, directing to the literature for more detailed and more general expositions.

After the exposition of the fundamental results we introduce the class of the *CF-invariant* operations, the operations on graphs that preserve the property of having a clique forest. A CF-invariant operation is a chordality-preserving operation, but with this definition we stress the fact that we want to describe the transformation in terms of changes to the structure of the clique forest and the consequent gain or loss defined by changes to a score function.

Next, we define the building blocks of the MFCF algorithm by introducing a CF-invariant operator, the *clique expansion operator*, and the concept of *gain function*. We proceed to describe the MFCF algorithm and we prove the correctness of the algorithm. We highlight certain analogies to the Minimum Spanning Tree algorithm of Prim (Prim, 1957a) and to the Maximum Cardinality Search algorithm of Tarjan (Tarjan, 1976).

Finally we explore two additional CF-invariant operators, the *bridge* operator and the *pruning* operator, discussing possible applications to structure learning.

3.1 Graph Theory Prerequisites

In this chapter $G = G(V,E)$ is, unless otherwise stated, an undirected graph as described in Section 2.1.1. If A is a subset of V we will denote with G_A the graph $G(A, E_A)$ *induced* by A, where $E_A \subset E$ is the subset of edges with both endpoints in A.

3.1.1 Definitions

Definition 15 (Path)**.** A *path* of length n is a sequence of vertices of $v_0, \dots, v_n$ such that $\forall i \in \{1, \dots, n\}, (v_{i-1}, v_i) \in E$. If all the vertices are distinct, that is $v_i \neq v_j$ whenever $i \neq j$, the path is called *simple*. When the graph is *directed* every edge must be considered oriented from v_{i-1} to v_i.

Definition 16 (Cycle)**.** If we have a path $v_0, \dots, v_n$ and we add an edge joining back v_n to v_0 we obtain a *cycle* $v_0, \dots, v_n, v_0$. A cycle is *simple* if it is obtained by joining the first and last vertices of a simple path.

Definition 17 (Directed Acyclic Graph)**.** If a graph G is *directed* and it does not contain any (directed) cycle, it is called a *Directed Acyclic Graph* (DAG). A *topological ordering* of the vertices in a DAG is an ordering of the vertices $v_1 < \cdots < v_i < \cdots < v_p$ such that if $(v_i, v_j) \in E$ (that is there is an edge oriented from v_i to v_j), then $v_i < v_j$.

Definition 18 (Chord)**.** A *chord* of a cycle is any edge joining two non consecutive vertices. See Figure 3.1

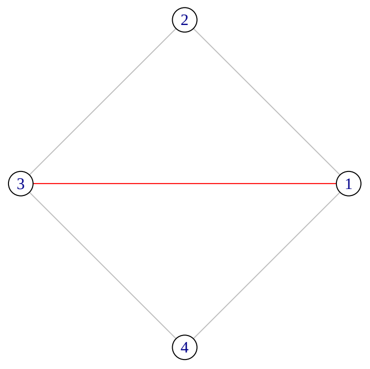


Figure 3.1: The cycle (v_1, v_2, v_3, v_4) and its chord (v_1, v_3) (higlighted in red).

Definition 19 ((v_a, v_b)-separator)**.** A set $S \subset V$ is called a (v_a, v_b)-separator if every path from v_a to v_b intersects S. If, in addition, S is the minimal set with that property (no proper subset of S separates v_a and v_b) then it is called a *minimal* (v_a, v_b)-separator. With reference to Figure 3.1, the set $\{v_1, v_3\}$ is a minimal (v_2, v_4)-separator. Note that $\{v_2, v_4\}$ is *not* a (v_1, v_3)-separator.

Definition 20 (Graph decomposition)**.** A *decomposition* of a graph $G = G(V, E)$ is a triple (A, B, S) of pairwise disjoint subsets of V with $V \subset A \cup B \cup S$ such that:

1. S separates A and B: any path between any $v_a \in A$ and any $v_b \in B$ must include at least one element of S. We will refer to S as the *separator* of the decomposition.

2. S is a complete subset in G.

If both A and B are non empty the decomposition is called *proper*.

Definition 21 (Decomposable graph)**.** A graph $G = G(V,E)$ is *decomposable* if it is complete, or if there is a proper decomposition (A,B,S) such that the graphs $G_{A\cup S}$ and $G_{B\cup S}$ are decomposable. We can also say that (A,B,S) *decomposes* G.

Intuitively, a decomposable graph is a graph that can be iteratively broken down into smaller components until the components left are complete (cliques). This definition gives also a first insight into the definition of a clique forest: the cliques are the nodes of the forest, and the edges are the sets that decompose the graph. When a graph has at least two connected components with vertex set A and B, then it naturally decomposes according to the decomposition $(A,B,\varnothing)$. Allowing the possibility of the empty set as a separator allows to talk about decomposition of clique forests, as opposed to the decomposition of clique trees, as is more common in the literature.

In graphical models a decomposition of the underlying graph has a very important relationship to the factorisation of probability distributions as shown in Theorem 22 (Lauritzen, 1996, Prop. 3.16).

Theorem 22 (Factorisation of the probability distribution of a decomposable graphical model)**.** If (A,B,S) decomposes G, then a probability distribution P factorises with respect to G if and only if both its marginal distributions $P_{A\cup S}$ and $P_{B\cup S}$ factorise with respect to $G_{A\cup S}$ and $G_{B\cup S}$ respectively and the densities satisfy:

$$f(x)f(x_S) = f_{A\cup S}(x_{A\cup S})f_{B\cup S}(x_{B\cup S}) \tag{3.1}$$

Theorem 22 is of fundamental importance because, when the graph is decomposable, the repeated application of Equation 3.1 can lead to the expression of the joint probability density as a simple function of the marginal probabilities of the cliques and the separators. This argument will be made completely rigorous once we define perfect sequences of cliques in Section 3.1.4. Obviously, when the separator S is the empty set the factorisation of the probability density in Equation 3.1 implies that the variables in the sets A and B are (unconditionally) independent.

3.1.2 Chordal Graphs

The description of a decomposable graph given in Definition 20 is based on a top-down characterisation of decomposability that requires taking into account the whole graph structure. Fortunately, there are alternative definitions that are local in nature and that therefore lend themselves better to the implementation of iterative algorithms. In this section we describe chordal graphs and we come to a construction of decomposable systems through the sequential attachment of simplicial vertices.

Definition 23 (Chordal graphs)**.** A graph is *chordal* (or *triangulated*, or *rigid circuit* graph) when every cycle of length ≥ 4 has a chord.

Remark 24 (Chordal graphs as generalisation of topologically constrained networks)**.** There are some aspects to the definition of a chordal graph that makes it a feasible generalisation of previous approaches used in IFNs.

1. All the cycles of minimal length have length of three. This is consistent with maximal planar graphs where all the faces are delimited by a triangle, but it should be noted that not all maximal planar graphs are chordal. We will discuss a number of such cases in Chapter 4. In this respect chordal graphs can be regarded as a generalisation of the PMFG (Tumminello et al., 2005).

2. Chordality is a topological constraint requiring that the length of any cycle is kept to a minimum; as such it can be seen as a generalisation of the tree structure where no cycle is allowed. In this respect chordality can be seen as a generalisation of a tree structure (Chow and Liu, 1968; Mantegna, 1999).

The following theorem (Lauritzen, 1996, Prop. 2.5) establishes an initial characterisation of chordal graphs[1].

Theorem 25. The following statements are equivalent for an undirected graph $G = G(V,E)$:

[1]Lauritzen (1996) develops a comprehensive theory of decomposability that covers also *marked* graphs. In his textbook the term *weakly decomposable* is equivalent, for non-marked graphs, to our definition of decomposable

1. G is decomposable.

2. G is chordal.

3. Every minimal (v_a, v_b)-separator is complete.

Graphical models whose underlying graph is chordal are *decomposable models* (Lauritzen, 1996, chap. 2). It is easily established that every clique, being a complete graph, is also a chordal graph (Fig. 3.2). It is also an easy consequence of the definition that any induced graph G_A of a chordal graph is chordal.

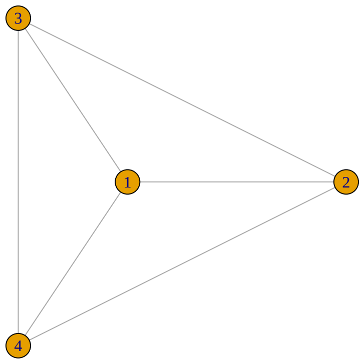


Figure 3.2: The graph K_4 is an example of a chordal graph.

The next example shows the simplest non-chordal graph and it is a useful "counterexample" to use when discussing the properties of chordal graphs. It shows that, even in this simple case, the calculation of marginal and conditional probabilities introduces spurious edges that were not present from the outset and that can lead to a denser network (the so called *fill in* problem). This also provides an heuristic explanation as to why it is necessary, differently from chordal graphs, to have the partition function in the probability density, as it is needed to somehow account for the extra interactions that cannot be read immediately from the factorised probability distribution.

Example 3 (Non-chordal graphs: the four cycle graph)**.** The simplest example of a non-chordal graph is the *four-cycle* (Fig. 3.3). According to the clique factorization property, described in Section 2.1.4, a probability density over the four variables $X_1,\ldots,X_4$ would factorise as:

$$f_{1234}(x_1,x_2,x_3,x_4) = \frac{1}{Z}\phi_{12}(x_1,x_2)\phi_{23}(x_2,x_3)\phi_{34}(x_3,x_4)\phi_{14}(x_1,x_4) \tag{3.2}$$

Having defined a probability, it is now possible to perform some of the most common inference tasks on a graphical model (Koller and Friedman, 2009, Ch. 9), such as the calculation of the marginal probability density $f_{123}(x_1,x_2,x_3)$.

In order to calculate the marginal distribution of X_1,X_2,X_3 we need to "eliminate" X_4. Assuming for simplicity that the underlying probability distribution is discrete:

$$
\begin{aligned}
f(x_1,x_2,x_3) &= \frac{1}{Z}\phi_{12}(x_1,x_2)\phi_{23}(x_2,x_3)\sum_{x_4\in\mathscr{X}_4}\phi_{34}(x_3,x_4)\phi_{14}(x_1,x_4) && (3.3)\\
&= \frac{1}{Z}\phi_{12}(x_1,x_2)\phi_{23}(x_2,x_3)\overline{\phi}_{1,3}(x_1,x_3) && (3.4)
\end{aligned}
$$

The variable elimination process has introduced a new factor $\overline{\phi}_{1,3}(x_1,x_3)$ corresponding to a chord of the graph that did not exist before (see Figure 3.1). It is easy to see that, if $\text{neib}(x_4)$ had been a complete set, no new chord would have been introduced. Vertices of a graph that can be eliminated without introducing spurious edges are called *simplicial*. A similar calculation shows that we would introduce a new factor also if we were to condition $f_{1234}(x_1,x_2,x_3,x_4)$ upon X_4. This fact motivates the abstract Definition 26 of variable elimination on graphs.

Definition 26 (Abstract variable elimination on graphs)**.** The operation of *elimination* of a variable v from the graph G is performed by joining all the vertices in $\text{neib}(v)$ and then removing v and all the incident edges (Drton et al., 2018, Def. 4.2.3).

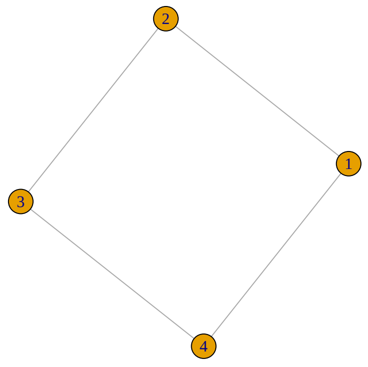


Figure 3.3: The four cycle is the simplest non-chordal graph

A simplicial vertex is a particular case of a simplicial set:

Definition 27 (Simplicial set)**.** Let A be a subset of the vertex set of $G(V,E)$. We say that the subset A is simplicial if $\mathrm{neib}(A)$ is complete. In particular a vertex $v \in V$ is *simplicial* if its neighbours in $G(V,E)$ constitute a complete set.

In a clique any vertex is simplicial. On the contrary in the *four-cycle* graph no vertex is simplicial, as it can be shown working out all the possible cases similarly to the case in Example 3. In general for chordal graphs the following theorem of Dirac holds (see for instance Blair and Peyton (1993, Lemma 1) or Lauritzen (1996, Lemma 2.9)).

Theorem 28 (Dirac (1961))**.** Every chordal graph G has at least a simplicial vertex. If G is not complete, then it has at least two nonadjacent simplicial vertexes.

Remark 29. Theorem 28 is used often in the description of recursive algorithms on chordal graphs and in proofs based on the mathematical induction principle because it readily provides the first case in an inductive proof and an initial state for recursive algorithms to start the initial decomposition of a graph.

3.1.3 Perfect Elimination Order

We introduce in this Section the important concept of Perfect Elimination Order (PEO). A graph is chordal if and only if it has a PEO (Theorem 31) and the MFCF methodology that we propose in Section 3.3 can be interpreted as a network growth model where vertices are added in a reverse PEO.

Let $G(V,E)$ be an undirected graph. Let $\mathrm{neib}(v_i, G)$ be the vertices that are adjacent to v_i in G. Let $\sigma = [v_1, v_2, \ldots, v_n]$ be an ordering of the vertices of $G(V,E)$. We define $V_{[i,n]}$ as the set of vertices $\{v_i, v_{i+1}, \ldots, v_n\}$ and G_i as the graph induced by $V_{[i,n]}$.

It helps the intuition to think of G_i as a sequence of graphs of decreasing order where the index is linked to time and vertices are successively eliminated from the graph until the graph is empty. If we reverse the direction of time the image is one of a series of graphs that grow as new vertices are added.

Definition 30 (Perfect Elimination Order (PEO))**.** We say that the ordering σ is a *perfect elimination order* (PEO) if, for all i, $\mathrm{neib}(v_i) \cap G_{i+1}$ is a clique in G_{i+1}.

This is equivalent to saying that v_i is a *simplicial vertex* in G_i. If G has a perfect elimination order, v_1 (the first vertex in the PEO) is simplicial in $G = G_1$ and can be eliminated without introducing any new edge, then v_2 is simplicial in G_2 (which is G after the elimination of v_1) and can be eliminated without the need to add a new edge and so on until all of the vertices have been eliminated. This is a fundamental property in recursive algorithms where variables are eliminated one at a time and allows to maintain the sparsity of the graph. This property of limiting the fill-in is relevant in many algorithms in numerical linear algebra, see for instance Golumbic (2004, Ch. 12) for applications to Gaussian Elimination.

Chordal graphs are characterised by the property of having a perfect elimination ordering (Blair and Peyton, 1993, Th. 2.2).

Theorem 31. A graph G is chordal if and only if G has a perfect elimination ordering.

The intuition behind Theorem 31 is that it is possible to eliminate one by one all of the vertices from a chordal graph without introducing any new edge. One very useful application of this fact is that is is possible to perform the Choleski decomposition of a matrix that has a chordal underlying graph without introducing any fill-in, as long as the rows and columns are permuted according to a perfect elimination ordering.

3.1.4 Perfect Sequences of Cliques

Definition 32. An *ordering* of the cliques of a graph is a bijective application σ from the first m natural numbers (where m is the cardinality of $\mathscr{C}$, the number of maximal cliques) into $\mathscr{C}$, $\sigma : \{1,\dots,m\} \to \mathscr{C}$. The cliques of the graph $C_l \in \mathscr{C}$ can be ordered by the relation $<_\sigma$ induced by σ where we say that $C_a <_\sigma C_b$ if $\sigma(a) < \sigma(b)$. Therefore we can think the maximal cliques to be ordered according to $<_\sigma$ as $C_{\sigma(1)} <_\sigma \cdots <_\sigma C_{\sigma(m)}$. As a shorthand notation we will also write $\sigma = [C_1,\dots,C_m]$ meaning that the cliques are ordered according to σ.

The following will be shown to be an alternative characterisation of chordal graphs, with the added benefit that it links the chordality to the structure of the cliques.

Definition 33 (Running Intersection Property (RIP))**.** Let $G(V,E)$ be a graph, $\mathscr{C}$ the set of cliques of G and $\sigma = [C_1,C_2,\dots,C_m]$ an ordering of $\mathscr{C}$. We say that σ has the *running intersection property* if for every clique C_i with $2 \leq i \leq m$ there is a clique C_j, with $j < i$ such that:

$$C_i \cap (C_1 \cup C_2 \cup \ldots C_{i-1}) \subset C_j \tag{3.5}$$

It is entirely possible that there is more than one clique that satisfies Equation 3.5. If we make the decision to always choose the one with the lowest index, then there is only one clique C_j that precedes C_i and we can call this unique predecessor the *parent* of C_i.

Definition 34 (Perfect sequence of cliques)**.** Let $G(V,E)$ be a graph, $\mathscr{C}$ the set of cliques of G and $\sigma = [C_1, C_2, \ldots, C_m]$ an ordering of $\mathscr{C}$.

Let us define the following sets:

1. $H_i = (C_1 \cup C_2 \cup \ldots C_i)$ (the *histories*),
2. $S_i = C_i \cap (C_1 \cup C_2 \cup \ldots C_{i-1})$ (the *separators*),
3. $R_i = C_i \setminus H_{i-1}$ (the *residuals*).

We say that $C_1, C_2, \ldots, C_m$ is a perfect sequence of cliques if

1. σ has the RIP, and
2. the separators are complete.

Remark 35. Definition 34 is not the most general definition of a perfect sequence of sets in a decomposable graph but it is adequate for the description of the MFCF. Lauritzen (1996, Par. 2.1.3) develops a general treatment of perfect sequences of sets, as opposed to cliques, where the perfect sequences of cliques can be extracted by "thinning".

Remark 36. The RIP seems at first a very complicated definition but the geometric meaning is in reality quite straightforward. The RIP means in practice that, as we add cliques such as C_i following the ordering of a perfect sequence of subsets, the intersection with the previous cliques is always contained in a clique (it is complete) and therefore it cannot introduce chordless cycles longer than three. This fact prevents the new clique to build a "bridge", or close a loop, between two previous cliques. This gives an heuristic illustration to the following result, formally proved in (Blair and

Peyton, 1993, Th. 3.4): any connected graph has a clique tree if and only if the cliques have the running intersection property[2].

Remark 37. A second important consequence of the running intersection property is that (H_{i-1}, C_i, S_i) is a decomposition of H_i. If we imagine to add the cliques in a perfect sequence of sets one by one we have that the repeated application of Eq. 3.1 gives a complete characterisation of the joint density as a function of the marginal probabilities of the cliques and separators as in Equation 3.6.

$$f(x) = \frac{\prod_{c_i \in \mathscr{C}} \phi_i(x_i)}{\prod_{s_i \in \mathscr{S}} \phi_i(x_i)} \tag{3.6}$$

We note that in Equation 3.3 the multivariate density function is fully specified in terms of marginal densities, without the need to be normalised through the partition function.

3.1.5 Clique Forest

We now formally describe a clique forest. Given a graph $G(V,E)$ with set of cliques $\mathscr{C} = \{C_1, C_2, \dots, C_m\}$ of G, we say that there is an edge between C_i and C_j if $S_{ij} = C_i \cap C_j$ is not empty: note that then S_{ij} is necessarily complete. We denote the set of edges with $\mathscr{S} = \cup_{C_i, C_j \in \mathscr{C}} S_{ij}$, with the understanding that when $C_i \cap C_j = \varnothing$ then S_{ij} is missing.

Definition 38 (Intersection graph)**.** The graph $\mathscr{K}(\mathscr{C}, \mathscr{S})$ where the vertices are the cliques of G and the edges are the not-empty intersections of the cliques is called the *clique graph or intersection graph* of $G(V,E)$.

Definition 39 (Clique intersection property and clique forest)**.** A *clique forest* for $G(V,E)$ is a clique graph with no cycles that includes all the cliques of G and that additionally fulfils the *clique-intersection* property (Blair and Peyton, 1993):

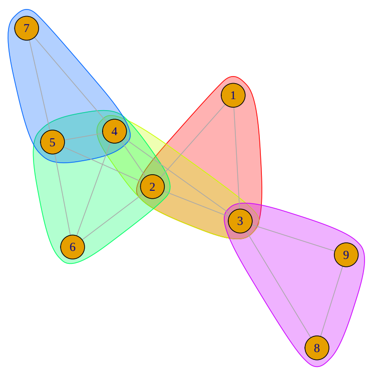
For any two cliques $C_i, C_j \in \mathscr{C}$ the set $C_i \cap C_j$ is contained in every clique on the path between C_i and C_j in the tree. We will denote a clique forest with $\mathscr{F}(\mathscr{C}, \mathscr{S})$.

When a graph $G = G(V,E)$ has a clique forest we say that the graph has the *CF-property*.

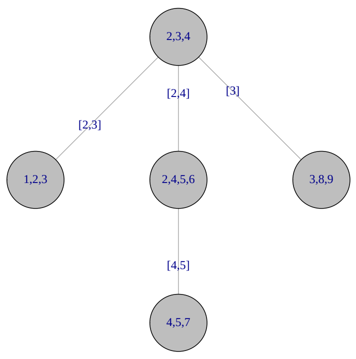
[2]For non connected graphs the same result applies to the connected components

Example 4 (Clique forest)**.** Figure 3.4 shows an example of a chordal graph with its associated representation as a clique forest.

Example 5 (The four-cycle does not have the CF-property)**.** The four cycle graph, shown in Figure 3.5 is a negative example. Any tree spanning the nodes of the clique graph does not enjoy the running intersection property.



(a) A graph with the maximal cliques highlighted.



(b) The same graph represented as a clique tree. The edges of the tree are labelled with the elements of the intersection.

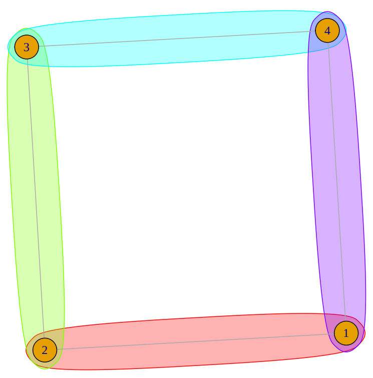
Figure 3.4: Illustration of the relationship between a chordal graph and the associated clique forest.

The following theorem is of fundamental importance in establishing the link between chordal graphs and clique forests.

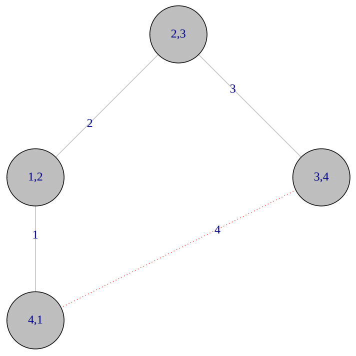
Theorem 40. A graph G has the CF-property if and only if it is chordal.

Proof. See Blair and Peyton (1993, Th. 3.1) or Koller and Friedman (2009, Th. 4.12) for a proof. □

Given Theorem 40 ensures the equivalence between chordal graphs and clique forests, it is reasonable to ask whether there is a procedure that, given a chordal graph in input can produce a clique forest. The answer is yes, as the Maximum Cardinality Search (MCS) algorithm (Tarjan, 1976) is an algorithm used to test the chordality of a graph and that can be adapted to extract the cliques and separators of a clique tree structure. We will discuss the MCS later in Section 3.3.2 and we will highlight the similarities to the MFCF.



(a) The four cycle graph with the cliques highlightes



(b) The four cycle as a clique tree (grey edges only). The edges of the tree are labelled with the elements of the intersection. This tree does not have the clique intersection property: the intersection between $\{4,1\}$ and $\{3,4\}$ (4, highlighted with the dotted red edge) is not included in the path between the two nodes along the tree.

Figure 3.5: Illustration of the relationship between a chordal graph and the associated clique forest.

3.2 The Clique Expansion Operator

In this section we describe the main tool for building clique forests originally introduced in Massara and Aste (2019), but we will introduce first the concept of *CF-invariance*.

Definition 41 (CF-invariant operator)**.** Let $G = (V,E)$ be a graph and let $G' = T(G)$ be the result of a transformation of the graph G through an operator T (for example the addition or removal of certain combinations of vertices or edges, the swapping or re-labelling of two or more vertices, etc.). We say that the operator T is *CF-invariant* if, whenever G has the CF-property, so does G'. This definition is not different from the definition of a chordality-invariant operator, but we want introduce a new definition to stress the fact that the description of a CT-invariant operator should include the effect on the cliques and edges of the clique forest.

Definition 42 (The clique expansion operator)**.** The *clique expansion* operator takes as input a clique C_a and a vertex v disjoint from C_a and produces a new clique C_b and a separator S such that $S = C_b \cap C_a \subset C_a$ and $v \in C_b$. The intuitive idea is that $S \subset C_a$

contains the vertices that have the strongest relationship with v and the new clique is $C_b = S \cup v$.

Figure 3.6 describes the *clique expansion* operation. The inputs of the operation are: the clique $C_1 = \{1,2,3,4\}$ and the isolated vertex $\{5\}$. Figure 3.6b shows the output of the clique expansion in the general case: two cliques C_1 and $C_2 = \{1,2,5\}$ and the separator $S = C_1 \cap C_2 = \{1,2\}$. There are two special cases.

1. If none of the elements of C_a have a strong relationship with v, then vertex v is not attached and $C_b = \{v\}$ and $S = \varnothing$ (see Figure 3.6d).
2. If, on the other side, all the elements in C_a have a strong relationship with the isolated vertex then C_a is replaced by $C_b \leftarrow C_a \cup v$ and the separator S is empty (see Figure 3.6f).

We note that S, being a subset of a complete graph, is complete, and also that it separates $C_a \setminus S$ and $C_b \setminus S$.

Remark 43 (The clique expansion operator adds simplicial vertices)**.** It is worth noting that the vertex v added by the clique expansion operator is simplicial by construction, since it applies to a subset of a clique. However the vertex is simplicial only in relation to the graph the operator is applied to; any subsequent modification to the graph will change the underlying graph and could in general cause the vertex v to no longer be simplicial.

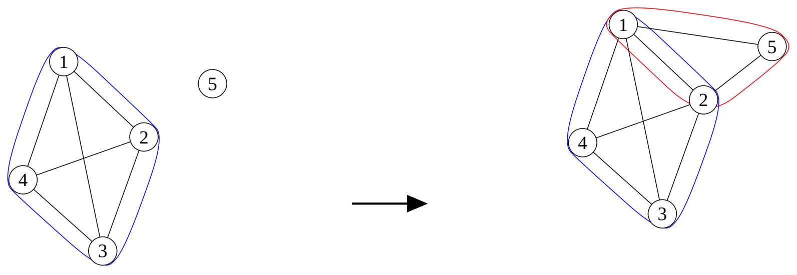
The clique expansion operator is the major building block of the MFCF algorithm, so we prove now some important properties that will be used in proving the correctness of the algorithm.

Theorem 44 shows that the clique expansion operator is CF-invariant.

Theorem 44. Let $G(V,E)$ be a chordal graph with $|V| \geq 2$ and at least an isolated vertex v. Let G' be the graph transformed by expanding a maximal clique C of G with v. If G has the CF-property then also G' has the CF-property.

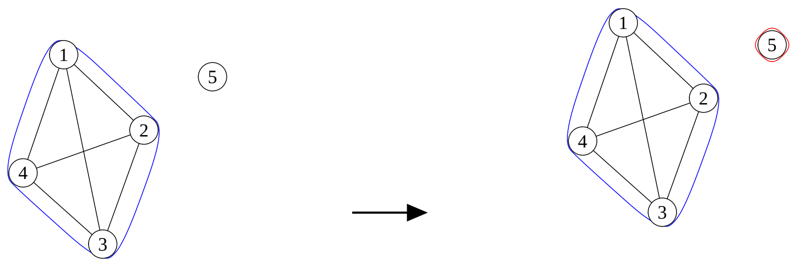
Proof. There are three cases.

Case 1. The vertex v is not added to any clique of $\mathscr{F}$ as in Figure 3.6d. Then v on its own becomes a new clique and $\mathscr{F} \cup \{v_i\}$ is a clique forest. No new edge is introduced and the clique intersection property is trivially satisfied.



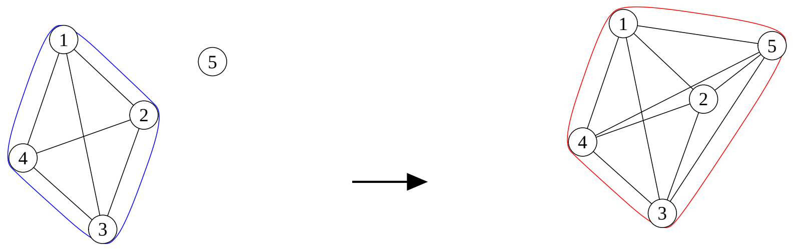
(a) Before clique expansion (general case)
$P(X = x \mid G_a) = \phi_{1234}(X_1, X_2, X_3, X_4)\phi_5(X_5)$

(b) After clique expansion (general case),
$P(X = x \mid G_b) = \frac{\phi_{1234}(X_1,X_2,X_3,X_4)\phi_{125}(X_1,X_2,X_5)}{\phi_{12}(X_1,X_2)}$
$S = \{1, 2\}$



(c) Before clique expansion (isolated vertex case)
$P(X = x \mid G_a) = \phi_{1234}(X_1, X_2, X_3, X_4)\phi_5(X_5)$

(d) After clique expansion (isolated vertex case),
$P(X = x \mid G_b) = \phi_{1234}(X_1, X_2, X_3, X_4)\phi_5(X_5)$
$S = \varnothing$



(e) Before clique expansion (full expansion)
$P(X = x \mid G_a) = \phi_{1234}(X_1, X_2, X_3, X_4)\phi_5(X_5)$

(f) After clique expansion (full expansion),
$P(X = x \mid G_b) = \phi_{12345}(X_1, X_2, X_3, X_4, X_5)$
$S = \varnothing$

Figure 3.6: Illustration of the clique expansion operator.
If S is a proper subset of C_1 the operation produces two cliques and a separator; if $S = \varnothing$ the result produces two disconnected cliques C_1 and $C_2 = \{5\}$ and the separator is the empty set if $S = C_1$ the operation purely expands the original clique with the new vertex and does not introduce a new clique or separator.

Case 2. The vertex v is attached to a maximal clique $C \in \mathscr{F}$ and all the vertices in C are connected to v as in Figure 3.6f. Then we replace the clique C with $C_E = C \cup v$ in $\mathscr{F}$. v is not contained in any other clique other than C_E and the clique intersection property is satisfied trivially also in this case.

Case 3. The vertex v is attached to a maximal clique $C \in \mathscr{F}$ but only a subset S of the vertices in C are connected to v as in Figure 3.6b. In this case we introduce a new clique $C_E = S \cup v_i$, we say that C is the parent of C_E (or, if we do not wish to stress the direction of $\mathscr{F}$ we add the edge $S_{ij} = (C, C_E)$ to $\mathscr{S}$) and we note that v does not belong to any other clique than C_E and any intersection of C_E with another clique must contain only elements of $S \subset C$, and therefore the clique intersection property is satisfied as well, since it holds for C. □

Given Theorem 40 we would be allowed to conclude that G' in Theorem 44 is also chordal, but it is easy to provide a direct proof with the next Theorem 45.

Theorem 45. Let $G(V,E)$ be a chordal graph with $|V| \geq 2$ and at least an isolated vertex v. Then expanding one clique of G with v does not introduce a chordless cycle of length ≥ 4.

Proof. Let us call C_a any clique of G. We choose any subset $S \subset C_a$ as a separator of the clique expansion. If S is not empty it S does not have any chordless cycle of length ≥ 4 because S is a complete induced subgraph of C_a, and the clique expansion adds all the edges between v and any vertex of S, resulting in a clique $C_b = S \cup v_i$ which is complete and therefore free from chordless cycles of length ≥ 4. If $S = G_a$ the expansion generates a larger clique $C_b = C_a \cup v_i$ which is complete. If S is empty then the expansion trivially does not add any chordless cycle of length ≥ 4. □

The clique expansion operator has also an interesting property with respect to DAGs.

Theorem 46. Let $G(V,E)$ be a chordal graph with $|V| \geq 2$ and at least an isolated vertex v_i. Assuming that G can be oriented so that it is a DAG, then it is possible to expand one clique of G with v and orient the edges so that the resulting graph is a DAG. Additionally, if $[v_1, v_2, \ldots, v_{i-1}]$ is a topological ordering for G, then $[v_1, v_2, \ldots, v_{i-1}, v]$ is a topological ordering for the expanded graph.

Proof. It is sufficient to orient all the new edges from the vertex in the separator towards the new vertex. Any new cycle introduced will not be oriented because all the edges point into v. Because v cannot reach any other vertex, it is the maximum vertex in the topological ordering. □

Remark 47. Theorem 46 allows in principle to use the clique expansion operator in all those contests where directionality is important, for instance in econometric models with lagged variables, causal reasoning, modelling of depeendencies and scheduling, modelling of causal sets.

3.3 The MFCF algorithm

We introduce the Maximally Filtered Clique Forest (MFCF) algorithm and explain the main ideas behind it before describing the algorithm more formally. Next we will demonstrate some useful qualities of the algorithm and finally we will show in which way it could be seen as a generalisation of two famous algorithms, namely Prim's minimum spanning tree algorithm (Prim, 1957b) and Tarjan's Maximum Cardinality Search (MCS) (Tarjan, 1976). The relationship with the Minimum Spanning Tree and the Maximum Cardinality Search is explained in sections 3.3.1 and 3.3.2.

Probably the easiest way to understand the algorithm is to see it as a recursive procedure that adds simplicial vertices to a clique forest by means of the clique expansion operator.

- The definition of the clique expansion operator ensures that the new vertex added is always simplicial with respect to the forest built up to that point. Because all the vertices are simplicial at the point of being added, it is an easy consequence that the vertices are added in reverse PEO. Because the algorithm is guaranteed to produce a PEO, the underlying graph is automatically chordal according to Theorem 31[3].

- The clique expansion operator is CF-invariant, therefore the MFCF does not require an onerous tests of chordality to be performed for every edge, unlike all the methods that we are aware of.

[3]Or it could be seen by induction using Theorem 45

- Because of the same reason, it is also easy to prove that new cliques are formed in such a way that their intersection with the graph is wholly contained in a clique at least, that is the clique that achieves the maximum gain. Should there be more than one such clique, ties are broken according to the order in which the cliques have been introduced. It is possible to think of the clique for which the maximum gain is achieved as the parent clique and the new clique created by the clique expansion operator as the child clique. However, this ordering is meaningful only to visualise the execution of the MFCF, because it is certainly possible to reorient the forest in a different way by arbitrarily fixing the roots of the subtrees. This is advisable for instance when calculating the conditional probability given a clique, the clique becomes the root of the forest and the conditional probabilities are updated from the root to the leaves. The cliques enjoy the running intersection property and therefore constitute a perfect sequence of sets.

- With the meaning of parent given by the MFCF, the parent clique is always produced by the algorithm before the child cliques. The cliques are produced in *topological order*. This characteristic is extremely useful in applications, since is allows to easily update the probability distribution in case of new evidence and to limit the update only to the cliques that are effectively impacted by the update.

- The MFCF selects the next vertex to be added to the network by maximising a gain function: that is by picking up the vertex that has the maximum affinity to a clique in the forest. One way to look at this is as if the MFCF operates a sort of generalised *preferential attachment* scheme: a vertex is attracted to the clique with the highest affinity. The difference with respect to a preferential attachment scheme (Albert and Barabási, 2002) is that the preference is expressed by the gain function, and it is related to a clique, rather than a high-degree vertex.

- For performance reasons the gains between cliques and vertices should be calculated only once and then kept available in a gain table using a process of *memoization*. The MFCF keeps the bookkeeping logic establishing when a gain becomes invalid and needs to be discarded or updated.

- Since we specify the algorithm in a recursive way, it is conceivable, and in some cases advisable, to seed the algorithm with an initial clique forest, to be completed by the MFCF with the remaining vertices.

All of the characteristics and the benefits of the MFCF listed above are a pure consequence of adopting a purely geometric approach exploiting the characteristic of chordal graphs and to specify the expected behaviour of the gain function, but abstracting from a specific form. This approach is similar to the approach used in “generic programming” (Musser and Stepanov, 1988), where the algorithms are specified using a template that can be filled with different blocks, as long as the blocks fit into the template. In the MFCF we abstract from the specific implementation of a gain function, as long as the implementation fulfils the “contract” that defines the interaction with the algorithm.

The complexity of the network built by the MFCF is controlled by means of three mechanisms:

1. The **minimum clique size** and the **maximum clique size** are specified for a given gain function and control how many edges can be introduced by the clique expansion operator. The use of a maximum clique size is an example of **topological regularisation**, as it limits the number of edges in the network.

2. Some gain functions may have a *validation* mechanism built in. The separator is expanded with elements from the clique as long as the marginal gain passes a test. The test can be a small sample or asymptotic statistical test, it could be non-parametric (cross-validation), or it could be based on a configurable threshold. This test potentially limits the dimension of the cliques created by the MFCF and produces a **validated** network.

3. The MFCF has a control on the *saturation* of the separators. A separator could be reused, or it could be considered saturated and never reused. This is a very effective way of controlling the topology of the network. For instance not reusing separators and fixing the maximum clique size to 4 it is equivalent to imposing the planarity constraint on the resulting network, as we will see in the next chapter on the Weighted Maximum Planar Subgraph Problem.

The user of the algorithm can change any of the parameters that drive the behaviour and the sparsity of the algorithm. This feature allows for multiscale analysis. For example one possibility would be to study networks of increasing size by progressively growing the size of the maximum allowed clique, or to keep the clique size fixed but studying the filtered networks as the significance level of the statistical test of the gain function increases. The cliques produced by the MFCF constitute a simplicial complex, the *clique complex* of the graph. With the clique complex it is possible to use the tools made available by the discipline of Topological Data Analysis to study the characteristics of he network at different scales. It should be noted though that the mapping between the MFCF parameters – such as clique size and statistical significance – and the set of edges in the filtered graph is not necessarily monotone; for instance an edge could disappear as a vertex moves from a clique to another. This is different from most of the simplicial complexes based on the thresholding of the weights of an affinity matrix, which provide a topological filtration.

Algorithm 1. MFCF: Builds a clique forest with given clique size range.

Description: Builds a clique forest by applying the clique expansion operator repeatedly until there are no more outstanding vertices. For performance reasons, the algorithm maintains a gain table that holds the possible scores for any combination of cliques already added to the forest and the outstanding vertices.

Input:

> $\boldsymbol{W}$ [mandatory]: Either a data matrix with n rows of p-variate observations (e.g. time series of stock market returns) or a p-by-p similarity matrix (e.g. correlation matrix of returns).
> ***gain_function*** [mandatory]: a function that calculates the gain of a clique expansion based on $\boldsymbol{W}$.
> ***max_cl_size*** [optional]: size of maximal clique (default value: 4, range $= [2, p]$).
> ***min_cl_size*** [optional]: size of minimal clique (default value: 4, range $= [1, max\_cl\_size]$).
> ***reuse_separators*** [optional]: whether to use separators more than once (default value: TRUE).
> $\boldsymbol{C_I}$ [optional]: Initial list of cliques
> $\boldsymbol{S_I}$ [optional]: List of separators associated with C_I

Output:

> ***cliques***: list of cliques of the clique forest, ordered as a perfect sequence of sets.
> ***separators***: list of separators of the clique forest.
> ***tree***: topological description of the clique forest.

Algorithm:

```
S1. [Initialize]. $p \leftarrow$ number of variables. $cliques \leftarrow \varnothing$. $separators \leftarrow \varnothing$. $outstanding\_vertices \leftarrow \{1,\dots,p\}$
S2. [Initialize list of cliques]
    - If $C_I$ is empty
      - $C_1 \leftarrow FirstClique()$
      - $cliques \leftarrow C_1$
      - $outstanding\_vertices \leftarrow outstanding\_vertices \setminus \{v, v \in C_1\}$.
    - Else
      - $cliques \leftarrow C_I$,
      - $separators \leftarrow S_I$,
      - $outstanding\_vertices \leftarrow outstanding\_vertices \setminus \{v, v \in C_I\}$.
S3. [Init Gain Table]. For every $v \in oustanding\_vertices$ and every $C \in cliques$, calculate score and optimal separator for $C$ and $v$ and add to gain table.
S4. [Check for termination]. If $outstanding\_vertices = \varnothing$ then return $cliques, separators, tree$.
S5. [Get best possible expansion]. Select from gain table the clique $C_a \in cliques$, separator $S \subset C_a$ and vertex $v \in outstanding\_vertices$ corresponding to the entry with the highest score.
S6. [Create new clique / separator].
    - If $S$ is a proper subset of $C_a$ then
      $C_b \leftarrow S \cup v$
      $cliques \leftarrow cliques \cup C_b$
      $separators \leftarrow separators \cup S$.
    - If $S = C_a$ (extension without new separators) then
      $C_a \leftarrow C_a \cup v$.
    - If $S = \varnothing$ (disconnected cliques) then
      $C_b \leftarrow v$
      $cliques \leftarrow cliques \cup C_b$.
S7. [Update outstanding vertices, tree]. $outstanding\_vertices = outstanding\_vertices \setminus v$.
    Set the edge between $C_a$ and $C_b$ to be the separator: $tree(C_a, C_b) \leftarrow S$
S8. [Update gain table]. Delete from gain table all entries where the vertex is $v$.
    Add to gain table entries with gains for $C_b$.
    If $reuse\_separators$ is false, delete from the gain table where the separator is $S$.
    Update gains for $C_a$.
```

```
S9. [Close loop]. Return to [S4.].
```

Remark 48. The function $FirstClique()$ provides an estimate of the best first clique. It can be obtained by starting with a clique made of the two vertices with the strongest association and growing it using the clique expansion operator until it reaches the minimum size required. It could also be used to initialise the algorithm with a list of "known" cliques C_I.

Example 6. Figure 3.7 exhibits an execution of the MFCF on a small sample network.

The MFCF algorithm is a radical extension of the TMFG algorithm proposed in Massara et al. (2016) which build a planar graph applying a special case of clique expansion operator (T_2 in that paper) with the following constraints: (a) the maximum clique size is 4, (b) the minimum clique size is also 4, and (c) the separators can be used only once.

Theorem 49. The MFCF algorithm produces cliques in a perfect order, provided that the initial cliques C_I are arranged in a perfect order.

Proof. The demonstration can be performed by induction on the number of vertices added. Let's assume that the algorithm has added $m-1$ vertices, and by definition there are cliques $C_1, \dots, C_j$ that are perfectly ordered. When adding the next v_m vertex there are three possibilities:

a) The algorithm selects a clique $C_i, 1 \leq i < j$ with a non empty separator S_i and therefore a new clique $C_k = S_i \cup v_m$ is created. The separator is clearly complete and by construction we have that $S_i \subset C_i$.

b) The algorithm selects a clique $C_i, 1 \leq i < j$ and the separator $S_i = C_i$ (extension of clique C_i). By hypothesis there is a clique C_h with $1 \leq h < i$ such that $S_i \subset C_h$ and S_i is complete. Since v_m was disconnected from all the cliques it was in particular disconnected from C_h and therefore it does not change the intersection $C_h \cup S_i$, and therefore C_h still fulfils the requirements that $S_i \subset C_h$.

c) The algorithm does not select a clique and adds a new clique made only of the vertex v_m. The intersection with any clique is the empty set and the result follows trivially.

□

Now we can use Theorem 49 to show that the MFCF builds a clique forest.

Corollary 50. The spanning forest build by the MFCF is a clique forest.

Proof. Note that in the MFCF algorithm the choice of the parent clique in the tree is based on on the running intersection property, that is we set the parent of clique C_k as the clique C_i that contains its separator. Since the intersection of cliques is a complete set, the set of cliques enjoys the clique intersection property. □

Theorem 51. The MFCF algorithm adds vertices in reverse perfect elimination order, provided that the vertices in the initial cliques C_I are arranged in a reverse perfect elimination order.

Proof. By induction on the number of vertices, let us assume that the total number of vertices is k and that j have been added by the MFCF and that they are ordered in reverse perfect order $\{v_k, v_{k-1}, \dots, v_j\}$. When adding the next vertex v_i it is by construction added to a separator S_i which is complete, and therefore $adj(i) \cap G_j = S_i$ is trivially complete. □

We also have, in the next theorem, a result related to the case when the edges are directed.

Theorem 52. If the initial cliques C_I can be oriented so that they constitute a DAG, the the MFCF can add the vertices so that the final graph is a DAG, and additionally, the vertices are added in topological order.

Proof. If the initial cliques C_I can be ordered so that the graph is a DAG, the clique expansion operator can be applied as in Theorem 46 to that the expanded graph is a DAG. Additionally, if the initial cliques C_I are a DAG the vertices have a topological order, and the MFCF adds vertices at the end of the topological order. □

3.3.1 Relationship with Prim's Minimum Spanning Tree Algorithm

The MFCF algorithm that we introduce in this paper is a generalisation Prim's minimum spanning tree algorithm (Prim (1957b)); Prim's algorithm constructs the Minimum Spanning Tree tree starting with an arbitrary vertex and adds the closest (i.e. with minimal edge weight) unconnected vertex[4]. In case the graph is not connected, for instance when some subsets are at infinite distance, the algorithm can be applied to the distinct connected components to produce a minimum spanning forest.

The MFCF generalises Prim's algorithm in the fact that it joins disconnected vertices to a growing clique tree and adds hyper-edges, rather than edges. Besides, the edge weight is replaced by the gain of the clique expansion operator. If we were to constrain the size of the allowed cliques to exactly two and we were to use the negative distance as the scoring function, the MFCF would behave exactly as Prim's algorithm. (Blair and Peyton, 1993, Ch. 4) illustrates the connection between Prim's algorithm and the maximum cardinality search algorithm (MCS, Tarjan (1976)), used to test graph chordality.

In our generalisation the vertices are replaced by cliques and we optimise a scoring function, rather than an edge weight; depending on the function we might look for the minimum (e.g. minimum cost) or the maximum (e.g. maximum gain). The algorithm starts by selecting one or more cliques and at each stage one of the unconnected vertices is added to the clique forest by performing an edge expansion. The vertex is chosen so as to optimise the scoring function. The initial clique(s) can be chosen with an heuristic (as in the variant of the algorithm presented here) or they could be assumed as given from previous knowledge or expert judgement. For instance in genetic regulatory networks there is interest in incorporating certain topological *motifs* that are known to appear frequently in this kind of networks (Fiori et al., 2012). In this case the cliques provided must be a clique forest.

3.3.2 Relationship with the Maximum Cardinality Search algorithm

The Maximum Cardinality Search (MCS) algorithm in its simplest form is an algorithm to calculate the PEO of an undirected graph. Since by Theorem 31 a graph is chordal if and only if it has a PEO, the algorithm is also naturally used to recognise chordal graphs. Let us suppose the graph G has k vertices. The idea is to initialise the algorithm with any node in the graph and to add the node to the end of the PEO, that is we assign

[4]To stay faithful to the original geometric meaning of the algorithm we keep referring to it as the *minimum* spanning tree, while the MFCF usually optimises a gain, but it is easy to see that a sign inversion would turn the problem into the minimisation of a cost or loss function.

the label *k* to the vertex. We then proceed inductively by choosing at any step the vertex that has the most neighbours among the vertices that have already been labelled, with ties broken arbitrarily. If the new vertex is simplicial we label the vertex and proceed with the next one, if the new vertex is not simplicial, that is his neighbours already labelled are not a complete set, the algorithm fails and the graph is not chordal. In some variants of the algorithm used to calculate triangulations, the missing edges are added so that the vertex is simplicial.

The version of the MCS shown in Algorithm 1 is taken from Blair and Peyton (1993, Figure 4.2) and is adapted to extract a PEO as well as the clique tree, with a loop which is similar to the MFCF, if the scoring function was the cardinality of the edges. Naturally the main difference between the MCS and the MFCF is that the MFCF does not fail but only selects the edges than make the vertex simplicial.

Blair and Peyton (1993) show that the MCS is also a generalisation of Prim's algorithm, where the edge weight is the cardinality of the separators in the clique tree.

```
$prev\_card \leftarrow 0$ ; /* Cardinality of the previous clique */
$\mathscr{L}_{n+1} \leftarrow \varnothing$ ;              /* Set of labeled vertices */
$s \leftarrow 0$ ;                           /* Clique identifier */
$\mathscr{T} \leftarrow \varnothing$;
for $i \leftarrow n$ to 1 step $-1$ do
    Choose a vertex $v \in V \setminus \mathscr{L}_{n+1}$ for which $|\mathrm{neib}(v) \cap \mathscr{L}_{n+1}|$ is maximum;
    $PEO(v) \leftarrow i$ ;                          /* $v$ becomes $v_i$ */
    $new\_card \leftarrow |adj(v_i) \cup \mathscr{L}_{n+1}|$;
    if $new\_card \leq prev\_card$ then /* begin a new clique      */
        $s \leftarrow s+1$;
        $\mathscr{C}_s \leftarrow \mathrm{neib}(v_i) \cap \mathscr{L}_{n+1}$ ;             /* $= madj(v_i)$ */
        if $new\_card \neq 0$ then /* get edge to parent        */
            $k \leftarrow \min\{j | v_j \in \mathscr{C}_s\}$;
            $p \leftarrow clique(v_k)$;
            $\mathscr{T} \leftarrow \mathscr{T} \cup \{\mathscr{C}_s, \mathscr{C}_p\}$;
        end
    end
    $clique(v_i) \leftarrow s$;
    $\mathscr{C}_s \leftarrow \mathscr{C}_s \cup v_i$;
    $\mathscr{L}_i \leftarrow \mathscr{L}_{i+1} \cup \{v_i\}$;
    $prev\_card \leftarrow new\_card$;
end
```

Algorithm 1: The Maximum Cardinality Search algorithm.

Figure 3.8 shows a possible execution of the MCS on an example graph.

3.3.3 Gain Functions

In this section we provide some examples of gain functions that can be used in several applications. Any gain function can be used with the MFCF as long as it returns a gain value and a separator. In general the identification of the separator is a classical subset selection problem, where it is required to find the subset S of the clique C that achieves the best score over the new clique $C' = S \cup v$, subject to size constraints for C'.

- If the score function is the sum of the pairwise scores between separator variables and the external variable, there is a natural order in which the separator variables must be added to the separator to maximise the gain and the gain function has only to ensure that the new clique made up of the separator variables and the external variable obeys the size constraints. For instance with a similarity matrix (see Section 3.3.3.1) the score is the sum of the pairwise scores and it is logical to add vertices to the separator in order of non decreasing similarity score.

- If the score is instead a general non-linear function of the variables, for example the determinant as in the case of a multivariate normal distribution (see Section 3.3.3.3), there is no obvious ordering of the variables that would give the optimal score. In such cases the choice is between applying an exhaustive search, which is not practical in problems of any realistic size, or to rank the separator variables according to some empirical measure of association (for instance correlation or mutual information) and add them one by one, stopping as soon as the score function decreases.

3.3.3.1 Similarity Matrix

As discussed in Massara et al. (2016) there are applications where it is required to build a network that maximises the sum of the weights of a similarity matrix subject to some constraint. Examples are the correlation networks mentioned in Section 2.2.3 (Mantegna, 1999; Di Matteo and Aste, 2002; Di Matteo et al., 2005; Tumminello et al., 2005; Aste et al., 2005a; Tumminello et al., 2007; Aste et al., 2010a; Song et al., 2012a; Pozzi et al., 2013; Musmeci et al., 2015b,a).

Let us define a symmetric matrix of weights W, where w_{ij} quantifies the "similarity" of elements i and j. If C is a subset of the row indices of W we define $Score(C) = \sum_{i \in C, j \in C} W_{ij}$.

The gain function returns the best available separator that, joined with a vertex, gives the highest possible sum of the weights. In this case the total score is the sum of the weights of the cliques minus the sum of the weights of the separators. The total score is given by

$$Score = \sum_{c \in \mathscr{C}} \sum_{i \in c, j \in c} W_{ij} - \sum_{s \in \mathscr{S}} \sum_{i \in s, j \in s} W_{ij} \tag{3.7}$$

When we perform a clique expansion and introduce a new clique $\tilde{c}$ and a new separator $\tilde{s}$ the corresponding gain in score is:

$$G(\tilde{c},\tilde{s}) = \sum_{i\in\tilde{c},j\in\tilde{c}} W_{ij} - \sum_{i\in\tilde{s},j\in\tilde{s}} W_{ij} \tag{3.8}$$

In the special case when the clique expansion results in the extension of a previous clique, such that $\tilde{C} = C \cup v$, the gain is the difference in score between the new clique and the old one (the separator is obviously zero):

$$G(\tilde{c},c) = \sum_{i\in\tilde{c},j\in\tilde{c}} W_{ij} - \sum_{i\in c,j\in c} W_{ij} = \sum_{i\in\tilde{c}} W_{iv} \tag{3.9}$$

One might also add a form of validation to this gain function and add only the edges with weights that are significantly larger than zero or exceed a given threshold.

3.3.3.2 Gain function from log-likelihood

Equation 3.6 is a likelihood for a given realisation $\mathbf{X} = \hat{\mathbf{x}}$:

$$\mathscr{L}(\mathbf{X} = \hat{\mathbf{x}}| \{c\in\mathscr{C}\},\{s\in\mathscr{S}\}) = \frac{\prod_{c\in\mathscr{C}} P(\mathbf{X}_c = \hat{\mathbf{x}}_c)}{\prod_{s\in\mathscr{S}} P(\mathbf{X}_s = \hat{\mathbf{x}}_s)} \tag{3.10}$$

and accordingly the log-likelihood is:

$$\ell(\mathbf{X} = \hat{\mathbf{x}}| \{c\in\mathscr{C}\},\{s\in\mathscr{S}\}) = \sum_{c\in\mathscr{C}} \log P(\mathbf{X}_c = \hat{\mathbf{x}}_c) - \sum_{s\in\mathscr{S}} \log P(\mathbf{X}_s = \hat{\mathbf{x}}_s) \tag{3.11}$$

When we add a new clique $\tilde{c}$ and a new separator $\tilde{s}$ the gain in log-likelihood is:

$$G(\tilde{c},\tilde{s}) = \log P(\mathbf{X}_{\tilde{c}} = \hat{\mathbf{x}}_{\tilde{c}}) - \log P(\mathbf{X}_{\tilde{s}} = \hat{\mathbf{x}}_{\tilde{s}})\ . \tag{3.12}$$

Instead, when we add a new clique $\tilde{c}$ by expanding an existing one c the gain in log-likelihood is:

$$G(\tilde{c},c) = \log P(\mathbf{X}_{\tilde{c}} = \hat{\mathbf{x}}_{\tilde{c}}) - \log P(\mathbf{X}_c = \hat{\mathbf{x}}_c)\ . \tag{3.13}$$

It is possible to add a significance test to this gain function since the model with the additional clique and separator is nested in the previous model and the difference in log-likelihood is one-half of the *deviance* (Wasserman, 2010). Under some relatively mild assumptions the deviance is asymptotically distributed as a chi-squared variable with k degrees of freedom, where k is the number of edges added to the model with the clique expansion (Lauritzen, 1996, Ch. 5.2.2). Other possible significance tests could be a cross-validation on a different set or an information criteria such as AIC or BIC (Akaike (1973); Schwarz et al. (1978)).

When a test statistic is available it is conceivable to use the p-value as a gain function. The intuitive meaning is to build a network where the links of greatest significance are added first.

3.3.3.3 Multivariate Normal Distribution

In the important specific case of a p-variate normal distribution the log-likelihood function for a given clique forest structure $\mathscr{T}$ can be written, using Equation 3.11:

$$\begin{aligned}\ell(\mathbf{X}=\hat{\mathbf{x}}|\,\{c\in\mathscr{C}\},\{s\in\mathscr{S}\}) &= p\ln(2\pi)\\ &\quad+\sum_{c\in\mathscr{C}}\left(\frac{1}{2}\ln|J_c|+(\hat{\mathbf{x}}_c-\mu_c)^t J_c(\hat{\mathbf{x}}_c-\mu_c)\right)\\ &\quad-\sum_{s\in\mathscr{S}}\left(\frac{1}{2}\ln|J_s|+(\hat{\mathbf{x}}_s-\mu_s)^t J_s(\hat{\mathbf{x}}_s-\mu_s)\right)\\ &= p\ln(2\pi)+\sum_{c\in\mathscr{C}}\left(\frac{1}{2}\ln|J_c|-\mathrm{Tr}(\hat{\Sigma}_c J_c)\right)-\sum_{s\in\mathscr{S}}\left(\frac{1}{2}\ln|J_s|-\mathrm{Tr}(\hat{\Sigma}_s J_s)\right)\end{aligned} \tag{3.14}$$

where

1. $\hat{\Sigma}_c$ (resp. $\hat{\Sigma}_s$) is the sample covariance matrix of the variables $\mathbf{X}_c$ (resp. $\mathbf{X}_s$) , and
2. J_s (resp. J_c) is the inverse covariance matrix (precision matrix).

When we perform a clique expansion and introduce a new clique $\tilde{c}$ and a new separator $\tilde{s}$ the corresponding gain in score is:

$$G(\tilde{c},\tilde{s})=\frac{1}{2}(\ln|J_{\tilde{c}}|-\ln|J_{\tilde{s}}|)=\frac{1}{2}(-\ln|\Sigma_{\tilde{c}}|+\ln|\Sigma_{\tilde{s}}|) \tag{3.15}$$

Instead, when a clique is expanded ($\tilde{c}=c\cup v$) the corresponding gain in score can be expressed as:

$$G(\tilde{c},c) = \frac{1}{2}\left(\ln|J_{\tilde{c}}| - \ln|J_c|\right) = \frac{1}{2}\left(-\ln|\Sigma_{\tilde{c}}| + \ln|\Sigma_c|\right) \qquad (3.16)$$

Note that this can be interpreted as an increase in likelihood or as a decrease in *entropy*. In this case beside the asymptotic tests of the log-likelihood ratio, there are also several small sample tests that work in the "big data" cases where $p \gg n$ (with n the number of realizations of $\mathbf{X}$).

For a given clique forest structure the likelihood is maximised by $J_c = \hat{J}_c = \hat{\Sigma}_c^{-1}$ and $J_s = \hat{J}_s = \hat{\Sigma}_s^{-1}$. In this case we have $\sum_{c\in\mathscr{C}} \mathrm{Tr}(\hat{\Sigma}_c \hat{J}_c) - \sum_{s\in\mathscr{S}} \mathrm{Tr}(\hat{\Sigma}_s \hat{J}_s) = p$ and do not change with the application of the clique expansion operator and therefore Equation 3.14 can be simplified to:

$$\ell(\mathbf{X} = \hat{\mathbf{x}} | \{c \in \mathscr{C}\}, \{s \in \mathscr{S}\}) = p\ln(2\pi) + \sum_{c\in\mathscr{C}} \frac{1}{2}\ln|\hat{J}_c| - \sum_{s\in\mathscr{S}} \frac{1}{2}\ln|\hat{J}_s| + p \qquad (3.17)$$

where the maximum likelihood estimations of the matrices $\hat{J}_c$ and $\hat{J}_s$ depend on the observations $\hat{\mathbf{x}}$ for both the structure and their values.

3.3.3.4 Multivariate Normal Distribution statistically validated

In the multivariate normal case it is possible to apply a significance test to the gain expressed in Equations 3.15 and 3.16 by applying a variant of the likelihood ratio test (Rencher, 2003, Par. 7.1). Indeed, if we have two alternative covariance matrices called for instance Σ_1 and Σ_0 it is possible to test whether they are significantly different by testing the following statistics:

$$u = \nu\left(\log|\Sigma_0| - \log|\Sigma_1| + \mathrm{Tr}\left(\Sigma_1^{-1}\Sigma_0\right)\right) . \qquad (3.18)$$

Where ν is the number of degrees of freedom of the matrix Σ_1 (and it is equal to the length of the time series n for our purposes). It is also possible to apply a small sample correction to the statistics u, see Rencher (2003, Eq. 7.2) for the details. If the two matrices are nested then u is χ^2 distributed with the degrees of freedom equal to the difference of the number of non zero parameters in the two matrices.

3.3.3.5 Random Gain Function

A very simple gain function, that can be used to generate random clique forests, is a function that returns a random number as a gain and a random subset of the input clique as a separator, subject to the constraints related to the size of the clique.

3.3.3.6 Regression

If we have a clique of variables $\mathbf{X_C} = (X_1, \dots, X_m)$ and a variable associated with an external vertex $\mathbf{X_i}$, it is possible to perform a linear regression of X_i against X_C as in Equation 3.19.

$$\mathbf{X_{v_i}} = \mathbf{X_C} \beta_\mathbf{C} + \varepsilon \tag{3.19}$$

The R^2 (or the adjusted R^2) of the regression gives an indication of how much of the variance of $\mathbf{X_i}$ is explained by $\mathbf{X_C}$ and can be taken as an indication of the strength of the association between the clique and the external vertex. In order to identify the separator we propose to examine the t-statistic of the model parameters $\beta_\mathbf{C}$[5] and to include only those vertices in C that are significant at a given confidence level, subject to the usual size constraints.

In the case of any type of regression $\mathbf{X_{v_i}} = f(X_C)\varepsilon$ it is intuitive to associate a direction from the explanatory variables towards the explained variables and this orientation assures that the resulting graph is a DAG. This is maybe not so relevant in the case of linear regression (unless one of the variables in X_C is lagged) but it might be important in other types of regression where the directionality is relevant: for instance in non-linear regression, quantile regression or logistic regression.

3.4 Other CF-Invariant Operations on Clique Forests

The clique expansion operator is by no means the only operation that can be carried out on a clique tree preserving the clique tree structure. We describe two additional operations that are useful in several cases: 1. the **direct join** of clique trees is useful to consolidate two disconnected trees in a single one, and 2. the **edge removal** can be used to "prune" a model.

[5]As calculated by most statistical packages such as the command `lm` in the R (R Core Team, 2016) language.

3.4.1 The Direct Join Operator

Definition 53 (Direct join of clique trees)**.** Let $\mathscr{F}$ be a clique forest. We say that $\mathscr{F}$ is the direct join of $\mathscr{F}_1$ and $\mathscr{F}_2$ if $\mathscr{F} = \mathscr{F}_1 \cup \mathscr{F}_2$ and $\mathscr{F}_1 \cap \mathscr{F}_2$ is complete. In other words the cliques and edges of $\mathscr{F}$ are contained in either $\mathscr{F}_1$ or $\mathscr{F}_2$ and their intersection is a clique. Figure 3.9 shows an example. Note that the intersection does not need to be a maximal clique in $\mathscr{F}$, it just needs to be complete.

Remark 54 (Relationship between direct join and graph decomposition)**.** We indicate with $G_{\mathscr{F}}(V_{\mathscr{F}}, E_{\mathscr{F}})$ the graph underlying a clique forest. The vertices are the members of the cliques and we say that $(v_i, v_i) \in E_{\mathscr{F}}$ if they belong at least to a common clique. It is easy to show that if $\mathscr{F}$ is the direct join of $\mathscr{F}_1$ and $\mathscr{F}_2$, then $(G_{\mathscr{F}_1}, G_{\mathscr{F}_2}, G_{\mathscr{F}_1 \cap \mathscr{F}_2})$ is a decomposition of $G_{\mathscr{F}}$. Lauritzen (1996, Par. 2.2) develops the theory in full generality.

Our interest is in finding a way to use this property to build a clique-tree-invariant operator, which we call the *bridge* operator.

Definition 55 (The bridge operator)**.** Given two disconnected clique forests $\mathscr{F}_a, \mathscr{F}_b$ and two cliques $C_a \in \mathscr{F}_a, C_b \in \mathscr{F}_b$ we define the *bridge* operator as the operator that joins the two forests using a clique made of two subsets $S_a \subset C_a$ and $S_b \subset C_b$, $C_s = S_a \cup S_b$.

Example 7. Figure 3.10 shows an example where $C_a = (2,3,4), C_b = (5,6,7), S_a = (4), S_b = (5,6)$ and the new clique is $C_d = (4,5,6)$.

Theorem 56 (The bridge operator preserves the CT property)**.** If $\mathscr{F}_a$ and $\mathscr{F}_b$ are as in Definition 55, the application of the bridge operator will result in a clique forest. Additionally if $C_{a1}, \ldots, C_{ah}$ is a perfect sequence of cliques for $\mathscr{F}_a$ and similarly $C_{b1}, \ldots, C_{bk}$, then $C_{a1}, \ldots, C_{ah}, C_s, C_{b1}, \ldots, C_{bk}$ is a perfect sequence of cliques for the joined clique forest.

Proof. Let us call C_s the clique introduced by the bridge operator. If we put $\mathscr{F}_1 = \mathscr{F}_a \cup C_s$ and $\mathscr{F}_2 = \mathscr{F}_b \cup C_s$, it is clear that $\mathscr{F}_1$ and $\mathscr{F}_2$ fulfil the conditions of Definition 53 and that $(G_{\mathscr{F}_1}, G_{\mathscr{F}_2}, G_{\mathscr{F}_1 \cup \mathscr{F}_2})$ is a decomposition of $G_{\mathscr{F}}$. To conclude we need to show that $G_{\mathscr{F}_1}$ and $G_{\mathscr{F}_1}$ are in turn decomposable. The cliques of $\mathscr{F}_1$ are the cliques of $\mathscr{F}_a$ with the addition of C_s and they constitute a perfect sequence of cliques because $\mathscr{F}_a$ is a clique forest; we can add C_s as the last clique to have again a perfect sequence of cliques for $G_{\mathscr{F}_1}$. The same reasoning applies to $\mathscr{F}_2$.

Finally, since no clique in $\mathscr{F}_b$ has an intersection with any clique of $\mathscr{F}_1$ other than C_s, and ordering with the cliques of $\mathscr{F}_1$ in their perfect sequence order, followed by C_s, followed by the cliques of $\mathscr{F}_2$ in their perfect sequence order fulfils the running intersection property and gives us a perfect sequence of cliques. □

3.4.1.1 Score Functions and generalisation of Kruskal algorithm

We have seen from Definition 55 that the effect of the bridge operator is to add a new clique C_s and two separators $S_a = C_s \cap G_{\mathscr{F}_a}, S_b = C_s \cap G_{\mathscr{F}_b}$. If we have a score function $S(\mathbf{X})$ the gain of the bridge operator could be the score of the new clique decreased by the score of the two separators: $Gain = S(X_{C_s}) - S(X_{S_A}) - S(X_{S_b})$. If we look at the analogy with the gain function for the clique expansion operator for the similarity matrix in Section 3.3.3.1 we see that this would be the exact generalisation, consisting in adding the weights of the new edges. If we look at the meaning of the gain function in terms of likelihood, this would represent the log-likelihood ratio of the hypothesis that the variables X_{C_S} are correlated as opposed to the hypothesis that are independent.

The bridge operator is an operator that allows building a clique forest by adding cliques to sub-forests until there is a gain to be achieved or until the forest becomes a tree. This would allow to develop a family of methods inspired by Kruskal, rather than Prim's, method. In this case the gain table would have to be indexed by pairs of cliques belonging to different trees and whenever two subtrees are joined the relative gains would have to be invalidated.

3.4.2 Pruning a Clique Forest

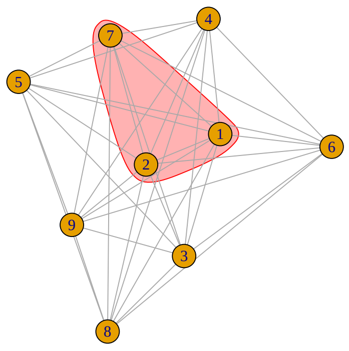
A further operation that might be useful and that is CT-invariant is the pruning of an edge in the underlying graph. The theory on which edges can be deleted retaining the chordality of the underlying graph is explained fully in Lauritzen (1996, Par. 2.1.4).

Theorem 57 (Laurizten, Lemma 2.19)**.** Given a chordal graph $G(V,E)$ the necessary and sufficient condition for an edge $e \in E$ to be deleted without introducing a chordless cycle of length greater than 4 is for the edge e to belong to one clique only. Equivalently, an edge belongs to a clique only if and only if it does not belong to any separator.

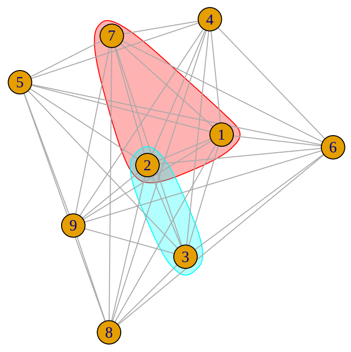
Proof. This is proven in Lauritzen (1996, Par. 2.1.4, Lemma 2.19). □

Example 8. Figure 3.11 shows the effect of pruning on a clique tree. The clique $(2,3,4,5)$ has four edges, and in particular $(3,5)$, that are contained in one clique only. The pruning operator removes $(2,3,4,5)$, introduces two new cliques $(2,3,4)$ and $(2,4,5)$. The separators stay the same, but the edges are rewired between the new cliques.

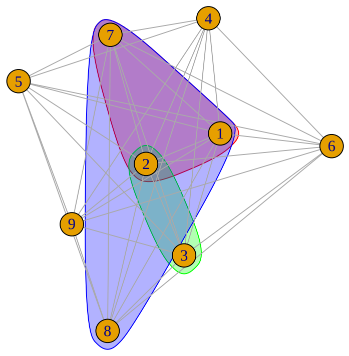
The pruning operator is also well behaved with respect to perfect sequences of cliques. It is an easy check to perform that by pruning a clique C' in correspondence with the edge (v_i, v_j) we obtain two cliques $C_i = C' \setminus v_j$ and $C_j = C' \setminus v_i$. Let us suppose we are pruning a clique forest $\mathscr{F}$ to obtain $\mathscr{F}'$. If $\mathscr{F}$ has a perfects sequence of cliques $C_1, \ldots, C', \ldots C_n$, then $C_1, \ldots, C_i, C_j, \ldots C_n$ is a perfect sequence of cliques for $\mathscr{F}'$ (Lauritzen, 1996, Lemma 2.20).



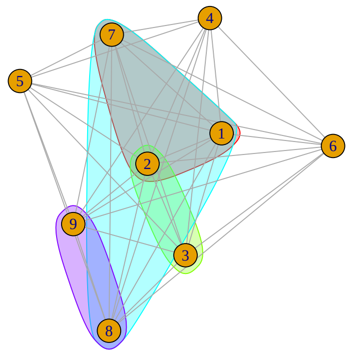
(a) Initial clique $(1,2,7)$, $PEO = \ldots, 1, 2, 7$



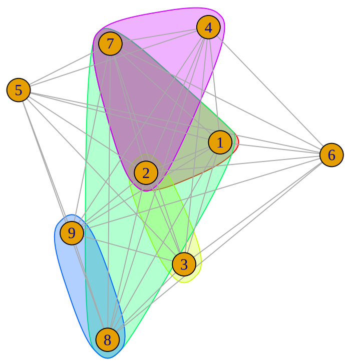
(b) New clique $(2,3)$, $PEO = \ldots 3, 1, 2, 7$



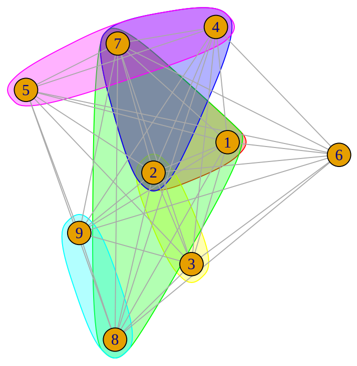
(c) New clique $(1,7,8)$, $PEO = \ldots 8, 3, 1, 2, 7$



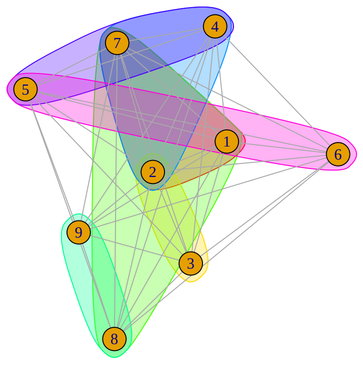
(d) New clique $(8,9)$, $PEO = \ldots 9, 8, 3, 1, 2, 7$



(e) New clique $(2,4,7)$, $PEO = \ldots 4, 9, 8, 3, 1, 2, 7$

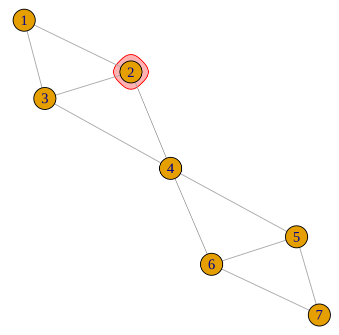


(f) New clique $(4,5,7)$, $PEO = \ldots 5, 4, 9, 8, 3, 1, 2, 7$

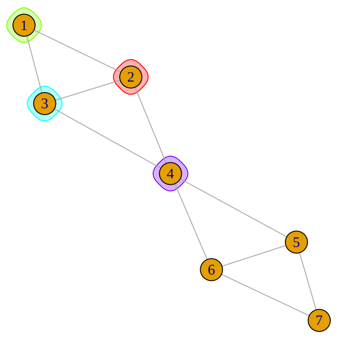


(g) New clique $(1,5,6)$, $PEO = 6, 5, 4, 9, 8, 3, 1, 2, 7$

Figure 3.7: The Maximally Filtered Clique Forest Algorithm

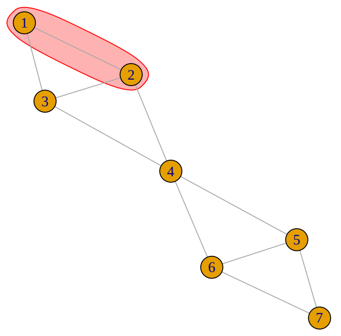


(a) Start from arbitrary node 2.

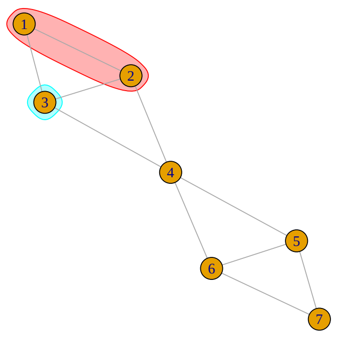


(b) Nodes 1,3,4 are the nodes with more vertices adjiacent to the set $\{2\}$

.

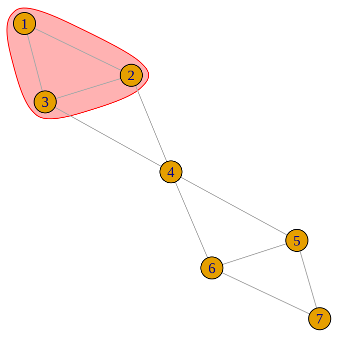


(c) We break arbitrarily the tie for 1, $PEO = \{1,2\}$.

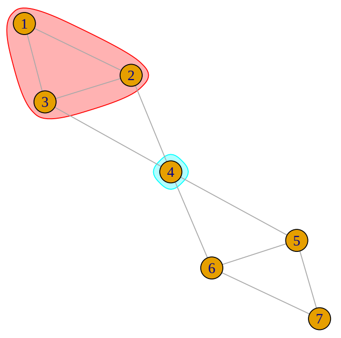


(d) 3 is the vertex whose neighbourhood has the highest cardinality with $\{1,2\}$

.

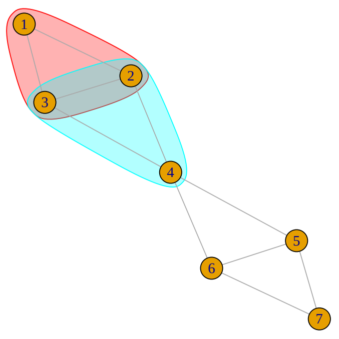


(e) $PEO = \{3,1,2\}$

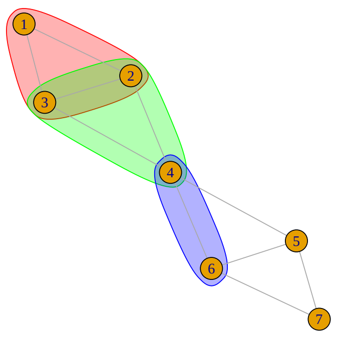


(f) The neighbourhood of 4 has the maximum cardinality intersection with the vertices already labelled.

.

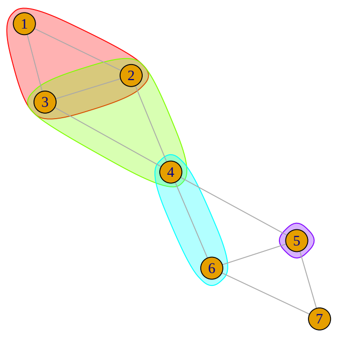


(g) 4 is added, the new candidates are 5 and 6. $PEO = \{4,3,1,2\}$.

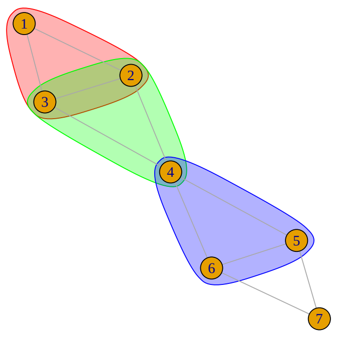


(h) The tie is broken arbitrarily in favour of 6. $PEO = \{6,4,3,1,2\}$

.

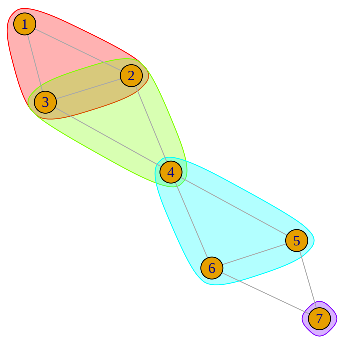


(i) 5 is the only vertex with $\mathrm{neib}(5)$ having maximum intersection with the labelled vertices.

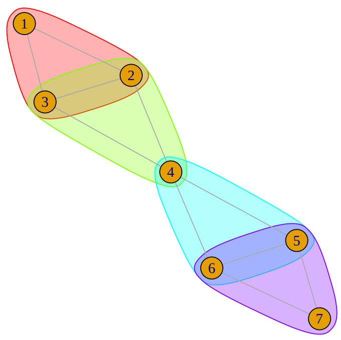


(j) $PEO = \{5,6,4,3,1,2\}$

.



(k) 7 is the remaining candidate



(l) $PEO = \{7,5,6,4,3,1,2\}$

.

Figure 3.8: The Maximum Cardinality Search Algorithm

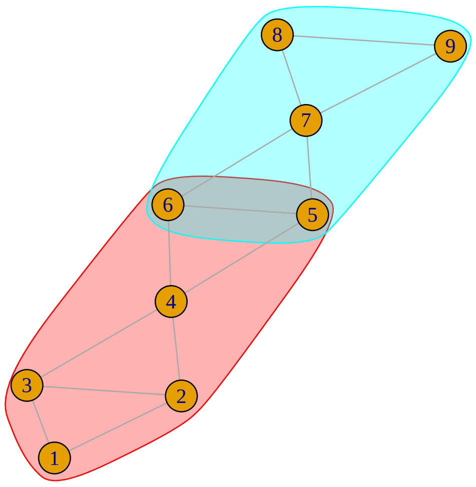
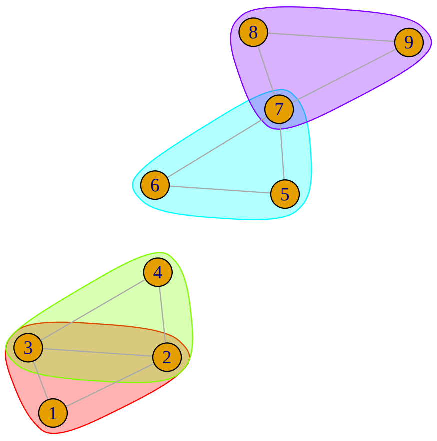
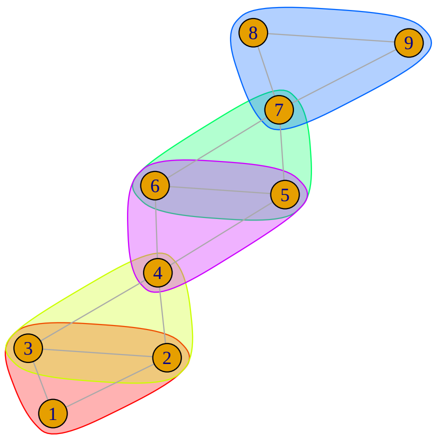


Figure 3.9: $\mathscr{F}_1 = \{(1,2,3),(2,3,4),(4,5,6)\}, \mathscr{F}_2 = \{(5,6,7),(7,8,9)\}$ and $\mathscr{F}_1 \cap \mathscr{F}_2 = \{(5,6)\}$.

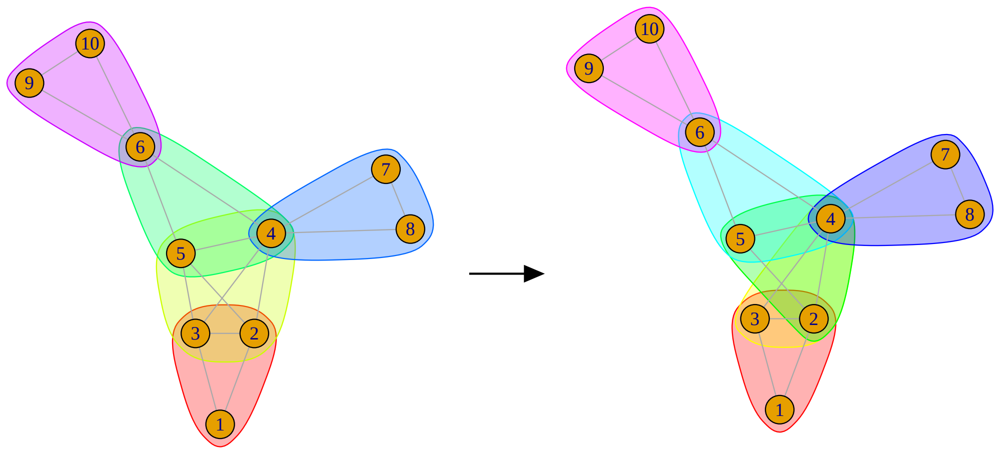


(a) Before direct join, $\mathscr{F}_1 = \{(1,2,3),(2,3,4)\}$, $\mathscr{F}_2 = \{(5,6,7),(7,8,9)\}$.



(b) After direct join. The new clique $(4,5,6)$ is used to bridge $\mathscr{F}_1$ and $\mathscr{F}_2$
.

Figure 3.10: The Direct Join Operator



(a) Before pruning, $\mathscr{F} = \{(1,2,3),(2,3,4,5),(4,5,6),(4,7,8)(6,9,10)\}$.

(b) After prining. the clique $(2,3,4,5)$ is split into $(2,3,4)$ and $(2,4,5)$

.

Figure 3.11: The edge pruning operator

Chapter 4

TMFG and other approximate solutions for the Maximum Weight Planar Subgraph Problem

In this chapter we describe a special version the MFCF that has been used to produce a solution to the Maximum Weight Planar Subgraph Problem (MWPSP); we will discuss the base algorithm (the Triangulated Maximally Filtered Graph, or TMFG henceforth) and three variants of the base algorithm specialised for the construction of planar graphs (in short "TMFG variants"). The approach is similar to the one described in chapter 3: indeed, the TMFG can be recast to an instance of the MFCF algorithm. The TMFG variants make use of an additional family of operators, which we call *planarity invariant* operators, that preserve planarity and are applied to further optimise the score achieved by the TMFG algorithm. The TMFG and the TMFG variants build the filtered networks subject to the topological constraint of planarity by adopting only planarity-invariant operators, which therefore play a similar role to the CT-invariant operators described in chapter 3 and used in the MFCF.

On a point of terminology, the basic specialised version of the MFCF, the TMFG, has been developed by us (Massara et al., 2016) before the general MFCF algorithm and is known in the literature with the original name of Triangulated Maximally Filtered Graph. Additionally, the implementation of the TMFG exploits the more regular structure of the clique forest produced and employs *ad hoc* data structures that allows for faster execution. Moreover, the three TMFG variants do not produce in general a clique forest because planarity invariant operators are not in general CT-invariant. In such cases the geometric structure of cliques and separators loses its meaning and a more natural data structure underlying the algorithms is the collection of the triangular faces of the maximal planar graph. For all of these reasons we choose to keep the original names for the base algorithm and its three variants and we specialise the exposition to the problems arising in planar graphs.

This chapter is organized as follows: in section 4.1 we discuss some background facts about Planar Maximally Filtered Graphs and describe two well known approximation algorithms used to generate such graphs; in section 4.2 we introduce the base TMFG algorithm and show how it can be considered as a particular case of the MFCF; in section 4.3 we introduce the planarity invariant operators used in section 4.4 to describe the three TMFG variants. In section 4.5 we offer some additional remarks on the characteristics of the algorithms; finally in section 4.6 we apply the TMFG and its variants to several weight distributions showing that it is computationally faster than the PMFG while achieving comparable or better results on a wide range of synthetic data.

4.1 The Maximum Weight Planar Subgraph Problem

Definition 58 (Planar and maximal planar graph)**.** A *planar graph* is a graph that can be drawn on the plane[1] without any two edges crossing each other other than in an endpoint (Bollobás, 2013; Nishizeki and Chiba, 1988). A graph is *maximal planar* if it is planar and adding any edge would destroy the property of planarity.

There are many characterisations of planar graphs. The one that is quoted most often, and also the first to appear in the literature, is the theorem from Kuratowsky which links planarity to the absence of particular subgraphs.

Theorem 59 (Kuratowski (1930))**.** A graph is planar if and only if it does not contain a subdivision of K_5 or $K_{3,3}$.

In the statement of Theorem 59, K_5 is the complete graph on five vertices and $K_{3,3}$ is the complete bipartite graph on six vertices. A *subdivision* of a graph is any graph obtained by replacing an edge (v_i, v_k) with two edges (v_i, v_j) and (v_j, v_k). It is obvious

[1]Or on a sphere, as it can be verified considering the stereographic projection.

that the two graphs, one of which is a subdivision of the other, are homeomorphic. The Kuratowsky theorem is therefore sometimes restated saying that a graph is planar if and only if it does not contain a subgraph homeomorphic to either K_5 or $K_{3,3}$.

There are many more theorems describing the necessary conditions for a graph to be planar. Wagner (1937) shows that a graph is planar if and only if it does not contain K_5 nor $K_{3,3}$ as minors[2]. Whitney (1931) establishes that a necessary and sufficient condition for a graph to be planar is to have an *abstract dual*[3]. Mac Lane (1937) shows that a planar graph must have a basis for its cycles such that no edge appears more than twice in the basis vectors. In some of the TMFG variants described in section 4.4 we explicitly build a cycle basis for the planar graphs produced. Schnyder (1989) links the planarity to the dimension of a partial order on the graph. de Verdiere (1990) associates planarity with the value of the second eigenvalue of a symmetric matrix defined from the edge set of the graph.

Definition 60 (MAXIMUM WEIGHT PLANAR SUBGRAPH PROBLEM (MWPSP))**.** Given an edge-weighted graph $G(V,E)$, with vertex set V, edge set E, and a non-negative edge weight $w(e) \geq 0$ for every $E \in E$, the MAXIMUM WEIGHT PLANAR SUB GRAPH problem requires to build a planar subgraph $G'(V,E')$, with $E' \subset E$ such that the sum of the weights $\sum_{e \in E'} w(e)$ is maximum (Liebers (2001) and Osman et al. (2003) provide a detailed description of the problem and a survey; see also (Dyer et al., 1985; Castonguay et al., 2017)).

Remark 61. When the initial graph is complete and the weights are strictly positive, as it is often the case in many applications, the Maximum Weight Planar Subgraph is also maximal[4] as it is evidently possible to add edges and increase the total weight until no more edges can be added without breaking planarity.

As Giffin (1984) has shown, the MWPSP is NP-hard (see also Dyer et al. (1985); Liebers (2001)), and therefore a vast amount of research has been focused on finding approximate solutions. We briefly review two known algorithms that have been used

[2]A graph H is a *minor* of G if it can be obtained from G by removing edges or vertices, or by contracting edges. Wagner's theorem was the beginning of *graph minor theory*, a theory that characterises minor-closed graphs in terms of excluded minors, (Lovász, 2006).

[3]A graph $G'(V',E')$ is an abstract *dual* of $G(V,E)$ if there is a bijective map ϕ between E' and E such that $T \subset E$ is a spanning tree in $G \Leftrightarrow \phi(E \setminus T)$ is a spanning tree in G'.

[4]With respect to edge inclusion

to construct planar filtered networks from a matrix of weights and that provide an approximate solution to the MWPSP:

- The PMFG (Tumminello et al., 2005), in section 4.1.1;
- The Deltahedron heuristics and subsequent improvements, including the base version of the TMFG (Foulds and Robinson, 1978; Liebers, 2001; Osman et al., 2003), in section 4.2.1.

4.1.1 Planar Maximally Filtered Graph

The PMFG algorithm (Aste et al., 2005b) searches for the maximum weighted planar subgraph by adding edges one by one (see Aste (2014)). The resulting matrix is sparse, with $3(p-2)$ edges. The algorithm starts by sorting the edges of a dense matrix of weights in non increasing order and attempts to insert every edge in the PMFG in that order. Edges that violate the planarity constraint are discarded. The most computationally intense part of the algorithm is the planarity test, which is performed every time an edge insertion is attempted. It results that the PMFG construction performs an order of p^2 ($O(p^2)$) planarity tests on any dense $p \times p$ matrix of weights W. Assuming that the complexity for a planarity test in $O(p)$ (Hopcroft and Tarjan, 1974; Boyer and Myrvold, 2001) the computational complexity of the whole algorithm results in a $O(p^3)$ (Song et al., 2012a).

Algorithm 2. PMFG: Builds a Planar Maximally Filtered Graph.

Description: *Builds a Planar Maximally Filtered Graph by adding edges in non decreasing order of weight. Adds only edges that do not break planarity.*

Input:

$\boldsymbol{W}$ [mandatory]: a p-by-p matrix of weights (e.g. matrix of squared correlation coefficients).

Output:

PMFG: filtered matrix representing a weighted planar graph.

Algorithm:

```
S1. [Initialize]. $PMFG \leftarrow I_p$, the $p-$dimensional identity matrix.
S2. [Sort edges].
    $EdgeList \leftarrow$ edges of $\boldsymbol{W}$ in non-increasing order of weight.
    $EdgesAdded \leftarrow 0$
    $EdgeIndex \leftarrow 0$
S3. [Loop over edges] **While** $(EdgesAdded \leq 3(p-2))$
    S3.1 [Get next edge] $e \leftarrow EdgeList(EdgeIndex)$.
    S3.2 [Check next edge] **If** $PMFG \cup e$ is planar **then**
         $PMFG \leftarrow PMFG \cup e$;
         $EdgesAdded \leftarrow EdgesAdded + 1$
S5. [End] ▮
```

4.2 Triangulated Maximally Filtered Graph

In this section we introduce the base version of the TMFG, and in the next section we introduce three variants.

The TMFG algorithm is not greedy with respect to edge insertion in the same way as the PMFG in the sense that the PMFG chooses the best possible move from a *subset* of all the feasible edge insertions that preserve planarity, while the TMFG optimises the result of the T_2 operator, which means that the optimisation is done over sets of triples of edge insertions and could miss an optimal single insertion. Nonetheless, we shall see that TMFG performs as well as – or better than – the PMFG for a large class of weight matrices, including squared correlation coefficient matrices from empirical time series which are relevant for modeling (Barfuss et al., 2016).

4.2.1 TMFG and Deltahedron heuristic

The deltahedron heuristic (Foulds and Robinson, 1978; Liebers, 2001) searches for approximate solutions of the MWPSP problem starting from a tetrahedron, K_4, which is planar and chordal. At each successive step a vertex is added into a triangular face and three edges are added connecting the newly inserted vertex to the vertices of the triangular face. This vertex insertion in a triangular face is called T_2 move (see Fig.4.1 and Aste and Sherrington (1999); Aste et al. (2012b,a); Dubertret et al. (1998); Andrade Jr. et al. (2005)).

It is easily seen that the T_2 operator is a special case of the clique expansion operator, with the additional constraint that the minimum and maximum clique size is 4 and that separators can only be between two cliques. As such the output of the deltahedron

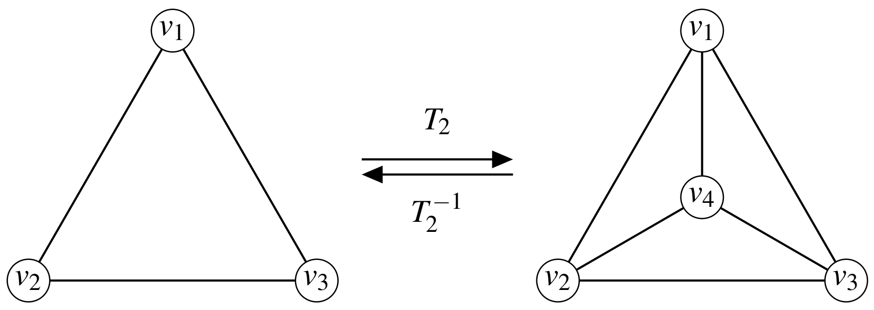


Figure 4.1: T_2 move: addition of one vertex within a triangular face (Aste and Sherrington, 1999; Aste et al., 2012b,a; Dubertret et al., 1998; Andrade Jr. et al., 2005). Its inverse, T_2^{-1}, removes a vertex from inside a three-clique (in this case the clique $\{v_1, v_2, v_3\}$).

heuristic is a clique forest, since it begins with a complete graph and expands it with a CF-invariant operator. Additionally since the dimension of the cliques is the same, the output is also a *pure (or homogeneous)* simplicial complex[5].

The T_2 operator is also a planarity invariant operator, as it acts without breaking planarity. Since the initial clique is planar the final filtered network is also planar. In the deltahedron heuristic the triangular face is chosen in order to maximise the sum of the newly inserted edges, while the vertices to be inserted are extracted from a pre-sorted list. The newly inserted vertex is simplicial in the graph at the point of insertion, and therefore the vertices are introduced in reverse PEO.

An example of the application of the T_2 operator is shown in Fig.4.1, where vertex v_4 is inserted into the triangular face $\{v_1, v_2, v_3\}$ splitting it into three triangular faces $\{v_1, v_2, v_4\}$, $\{v_1, v_4, v_3\}$, and $\{v_4, v_2, v_3\}$. In the following we will call *face* a three-clique that does not contain any vertex in the given embedding[6], reserving the word *triangle* for a generic 3-clique. We see that, after the T_2 move, $\{v_1, v_2, v_3\}$ is no longer a face but rather a triangle. A complementary way to see this is by using the concept of a cycle basis. In a graph it is possible to define the addition of cycles as the (set-theoretic) symmetric difference of the edges of the two cycles. With this definition we can say that $\{v_1, v_2, v_3\}$ is a linear combination (the sum, in fact) of $\{v_1, v_2, v_4\}$, $\{v_1, v_4, v_3\}$, and $\{v_4, v_2, v_3\}$. Thus the effect of the T_2 move is to remove one cycle from the basis and augment it with three more cycles. It can be shown that in a planar graph

[5]Pure simplicial complexes of dimension k are those where every face of dimension less than k is contained in a face of dimension k

[6]An embedding is an actual drawing on the plane on a planar graph. It can be described by enumerating, for every vertex, the list of all the adjacent vertices in the order in which they appear around the vertex.

(Bollobás, 2013) the faces constitute a cycle basis. In our algorithm when a face is expanded it becomes a separator and disappears from the basis.

In the literature (Liebers, 2001) the vertex list is sorted according to two functions[7] of the edge weights incident to the vertex, yielding two possible variants of the deltahedron heuristic: 1. the sum of the incident edge-weights; 2. the maximum incident edge-weight. Different weightings lead to different ordering for the vertices and different results.

The deltahedron heuristic algorithm is not "greedy" (unlike the PMFG that chooses the heaviest feasible edge at every step) since the choice of the ordering of the vertices is done once at the beginning and there is no subsequent attempt at optimising the order of the vertices taking into account the evolution of the local configuration.

However, the algorithm is considerably faster than the PMFG, since the T_2 move is planarity-invariant and therefore there is no need to test for planarity at each stage.

In Green and Al-Hakim (1985) and Osman et al. (2003) the deltahedron heuristic is improved by maintaining a record of the most favourable vertex insertion moves (Green and Al-Hakim heuristic – the GH-heuristic henceforth), essentially keeping a cache of the best and next-to-best options for inserting any of the remaining vertices. The cache is updated as the algorithm progresses. Optionally Osman et al. (2003) allow for a parameter that governs the "greediness" of the algorithm.

The TMFG is an improved version of the GH-heuristic that records the gains of the possible applications of the T_2 operator to different triangular faces (or, with the terminology of the clique expansion operator, the separators) and to the outstanding vertices. Differently from the GH-heuristic the table is updated after every operation. The main difference between the deltahedron heuristic and the base TMFG algorithm is in the fact that the TMFG allows much greater flexibility in the choice of the scoring function: in the deltahedron heuristic it is the sum of the weights for the newly inserted vertices. In the TMFG the scoring function could be any scoring function as described in section 3.3.3. In section 5.1 we provide an information-theoretical perspective on a specific gain function.

Remark 62 (The Deltahedron heuristic as a special case of the MFCF)**.** We have ob-

[7]Of course there are many more conceivable functions of the edge weights incident to a vertex, but the two mentioned in the main body appear to have been studied in the literature.

served that the T_2 operator inserts a simplicial vertex. We also observe that every new clique is a tetrahedron and this can be enforced in the MFCF by requiring that the minimum clique size and the maximum clique size are both fixed at four[8]. The list of cliques contains the initial K_4 and all the subsequent cliques introduced by the clique expansion operator. The list of separators is made up of the triangular faces where all the vertices are inserted. Every separator can be used only once to keep the planarity: this means that the separator is saturated after one use.

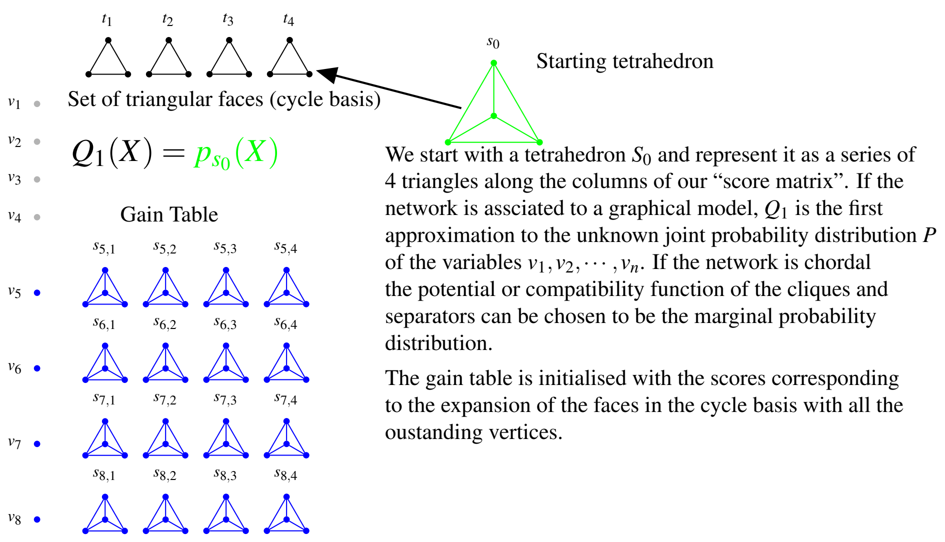
4.2.2 TMFG construction

The TMFG algorithm starts from a clique of order 4 (K_4) and adds vertices by using the local move T_2. The novelty of this algorithm with respect to the Deltahedron heuristics is that, at each step, the algorithm optimizes a *score function* (e.g. the sum of the weights of the edges). Similarly to the GH-heuristics, the method does not rely on any particular ordering of the vertices but, at every step, it calculates the score that would be obtained by adding any of the remaining vertices inside any feasible face. T_2 is applied to the vertex and face pair that leads to the maximum increase in score. A naive implementation would require to evaluate the gain function for every pair consisting of a feasible vertex and a feasible face, thus resulting in an $O(p^2)$ calculations at every step and therefore $O(p^3)$ overall computational complexity. However, it is possible to maintain and update incrementally a cache[9] with the information about the best possible pairing, and update only the records affected by a move. This cache contains as many elements as there are feasible faces ($O(p)$). Since the calculation of the maximum of a vector of $O(p)$ elements requires $O(p)$ calculations, the overall number of calculations for the score functions is $O(p^2)$. This results in much faster computational times with respect to the $PMFG$. Differently from Osman et al. (2003) we use a slightly different data structure to keep track of the vertices to insert into the feasible faces. We also keep track of the triangles that are no longer faces because these are the separators of the clique forest and are relevant for probabilistic modelling (see section 5.1). The stages

[8]Size 4 means the number of vertices allowed in a clique, and not the geometric dimension of the clique.

[9]The cache is similar to the gain table in the MFCF. The difference is that the cache contains all the possible separators for every clique and vertex, and not only the best possible separator. This is made possible by the simpler geometric structure of a clique forest with fixed treewidth: every clique has three possible separators for every vertex, and therefore it is easy to maintain the information in a table of fixed size, but it is not practical when the cliques can have different or lage sizes.

of the calculation are described graphically in Figures 4.2-4.4.


t_1 t_2 t_3 t_4
s_0
Starting tetrahedron
Set of triangular faces (cycle basis)
v_1 v_2 v_3 v_4 v_5 v_6 v_7 v_8
$Q_1(X) = p_{s_0}(X)$
Gain Table
$s_{5,1}$ $s_{5,2}$ $s_{5,3}$ $s_{5,4}$
$s_{6,1}$ $s_{6,2}$ $s_{6,3}$ $s_{6,4}$
$s_{7,1}$ $s_{7,2}$ $s_{7,3}$ $s_{7,4}$
$s_{8,1}$ $s_{8,2}$ $s_{8,3}$ $s_{8,4}$

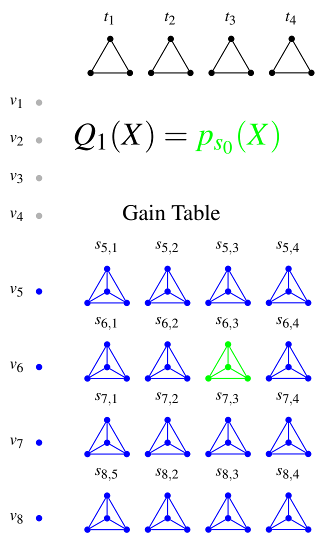
We start with a tetrahedron S_0 and represent it as a series of 4 triangles along the columns of our "score matrix". If the network is assciated to a graphical model, Q_1 is the first approximation to the unknown joint probability distribution P of the variables $v_1, v_2, \cdots, v_n$. If the network is chordal the potential or compatibility function of the cliques and separators can be chosen to be the marginal probability distribution.

The gain table is initialised with the scores corresponding to the expansion of the faces in the cycle basis with all the oustanding vertices.

Figure 4.2: Initialisation stage of the TMFG, see detailed description in Remark 63

Remark 63 (Initialisation stage of the TMFG)**.** Figure 4.2 shows the initialisation of the TMFG algorithm. The first tetrahedron is built using an heuristic where the four vertices with the higher overall connectivity are joined together. The four triangles that constitute the tetrahedron are put in the initial set of faces. Note that the set of faces is a basis for the cycle space of the planar network built so far. The gain table is initialised by calculating the scores corresponding to every pairing between the feasible faces t_1, t_2, t_3, t_4 and the vertices not yet added to the network (i.e. $v_4, \ldots, v_8$ in this simplified example). In the case that we are looking for an approximate solution to the MWPSP, the gain is the sum of the weights of the edges introduced; in case we are building a graphical model from empirical data, the gain function can be a measure of association between the variables such as mutual information. In such case the first approximation $Q_1(\mathbf{X})$ to the "true" probability distribution is the product of the marginal distribution of $\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3, \mathbf{X}_4$, and assumes that the remaining variables are independent from the initial clique.

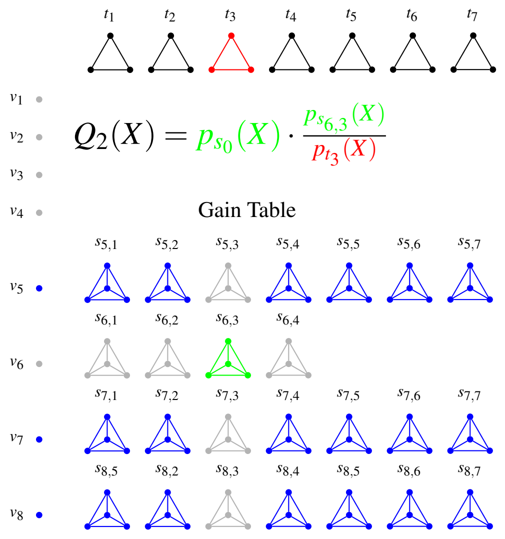
Remark 64 (Selection of the next vertex)**.** Figure 4.3 shows the first part of the main iterative loop in the TMFG, which is executed once for every outstanding vertex. By looking up the highest value in the gain table we identify the best candidate vertex to be added next to the network by means of a T_2 operator. In this example the vertex is v_6


t_1 t_2 t_3 t_4
v_1 v_2 v_3 v_4 v_5 v_6 v_7 v_8
$Q_1(X) = p_{s_0}(X)$
Gain Table
$s_{5,1}$ $s_{5,2}$ $s_{5,3}$ $s_{5,4}$
$s_{6,1}$ $s_{6,2}$ $s_{6,3}$ $s_{6,4}$
$s_{7,1}$ $s_{7,2}$ $s_{7,3}$ $s_{7,4}$
$s_{8,5}$ $s_{8,2}$ $s_{8,3}$ $s_{8,4}$

In the case of a similarity matrix W the gain is $\sum_{i \in t_3} W(6,i)$, the sum of the weights of the edges introduced.
Other choices of local score functions are possible.
For instance in Massara et al. (2016) we use the increase in mutual information obtained by inserting e.g. v_6 into t_3:
$S(v_6, t_3) = \sum_{\mathbf{X}} p_{s_{6,3}}(\mathbf{X}) \log \left(p_{s_{6,3}}(\mathbf{X})\right)$
$- \sum_{\mathbf{X}} p_{t_3}(\mathbf{X}) \log \left(p_{t_3}(\mathbf{X})\right)$
the algorithm tries to maximise this quantity at each step.
Maximising this quantity is equivalent to minimising the Kullback-Leibler divergence between P and Q .

Figure 4.3: Selection stage of the TMFG, see detailed description in remarks 64

to be inserted in t_3. This description is simplified for clarity: in reality the gain table is not looked up every time, as it would be inefficient from a computational point of view. The same information is, in the TMFG implementation, kept in two vectors with the same cardinality as the triangular faces set: the vector *MaxGain* contains the best gains for every triangle in the base, and the vector *BestVertex* contains the vertex for which the maximum value for that triangle is achieved. An important part of the TMFG is in the bookkeeping of this data structure as the network is updated.


t_1 t_2 t_3 t_4 t_5 t_6 t_7
v_1 v_2 v_3 v_4 v_5 v_6 v_7 v_8
$Q_2(X) = p_{s_0}(X) \cdot \frac{p_{s_{6,3}}(X)}{p_{t_3}(X)}$
Gain Table
$s_{5,1}$ $s_{5,2}$ $s_{5,3}$ $s_{5,4}$ $s_{5,5}$ $s_{5,6}$ $s_{5,7}$
$s_{6,1}$ $s_{6,2}$ $s_{6,3}$ $s_{6,4}$
$s_{7,1}$ $s_{7,2}$ $s_{7,3}$ $s_{7,4}$ $s_{7,5}$ $s_{7,6}$ $s_{7,7}$
$s_{8,5}$ $s_{8,2}$ $s_{8,3}$ $s_{8,4}$ $s_{8,5}$ $s_{8,6}$ $s_{8,7}$

The entries that are greyed out to the left are the ones that are no longer admissible. t_3 becomes a separator and is removed from the list of triangular faces. All the entries in the gain table that involve either t_3 or v_6 must be put to zero

Figure 4.4: Update stage of the TMFG, see detailed description in Remark 65

Remark 65 (Post-selection updates: cliques and separators, probability distributions, gain table)**.** Figure 4.4 shows the second part of the main iterative loop in the TMFG, and describes the updates required after the selection of the next vertex. The first step is the assembly of a new clique from the chosen triangle and vertex. The clique and triangle (separator) are added to the list of cliques and separators, which constitutes an output of the algorithm. Next we need to update the gain table: the entries that involve the vertex just inserted are no longer feasible and have to be removed by setting them to zero, the set of triangular faces must be updated by removing the used triangle t_3 and by adding the new triangles introduced by the T_2 operator (t_5, t_6, t_7 in our example). Finally we observe that in case we are building a graphical model we have a new approximation to the probability P; Q_2 is obtained from Q_1 by multiplying it by the marginal density of the new clique and dividing by the marginal density of the new separator.

Remark 66 (Performance improvements to the gain table cache)**.** As explained in remark 64 the TMFG does not use the gain table directly, as it would require an $O(p^2)$ lookup operation for every vertex. The same result can be achieved more efficiently by maintaining two vectors that contain the information needed by the TMFG. After every application of T_2 the cache is updated: some scores that were previously achievable are no longer feasible, while others become feasible and the corresponding score is calculated.

More formally, we define a *score function* $S(v_h, \{v_a, v_b, v_c\})$ that quantifies the gain achievable by adding vertex v_h inside the triangle $\{v_a, v_b, v_c\}$.

For instance, for a given, dense, matrix of weights W, the gain function can be the sum of the weights of the edges that will be added by inserting v_h in face $\{v_a, v_b, v_c\}$: $S(v_h, \{v_a, v_b, v_c\}) = W(v_h, v_a) + W(v_h, v_b) + W(v_h, v_c)$[10]

The cache is a structure made up of two vectors (*MaxGain* and *BestVertex*) indexed by the faces in the planar graph present up to that point. Let us consider a given stage of the construction with m triangular faces t_i, $i \in \{1, 2, \cdots, m\}$ and k remaining uninserted vertices $v \in \{v_1 \cdots v_k\}$. The *MaxGain* vector contains the value of the maximum gain over all remaining vertices for all triangular faces:

$$\textbf{MaxGain} = \left(\max_{v \in \{v_1 \cdots v_k\}} S(v, t_1), \max_{v \in \{v_1 \cdots v_k\}} S(v, t_2), \ldots \max_{v \in \{v_1 \cdots v_k\}} S(v, t_m) \right) . \quad (4.1)$$

[10]In the next chapter we discuss an information theoretic interpretation of the score function.

The *BestVertex* vector contains inside the list of vertices that attains the maximum gain for the specific triangular face:

$$\mathbf{BestVertex} = \left(\underset{v \in \{v_1 \cdots v_k\}}{\arg\max}\ S(v,t_1),\ \underset{v \in \{v_1 \cdots v_k\}}{\arg\max}\ S(v,t_2), \ldots,\ \underset{v \in \{v_1 \cdots v_k\}}{\arg\max}\ S(v,t_m) \right). \tag{4.2}$$

When a vertex (say vertex v_h) is added to a certain triangular face (say face t_j) the two cache vectors must be updated by removing vertex v_h from the list of remaining vertices, removing face t_j and adding three new faces.

The TMFG pseudocode is shown in Algorithm 3. We note that the algorithm produces a highly regular structure where the size and number of cliques and separators is known exactly in advance; this is very convenient from a computational point of view because it allows to allocate directly the memory required and to access exactly by position every clique and separator. Also, since the geometric structure is simplified, the gain function does not have to calculate the optimal size of the separator as in chapter 3. For these reasons the TMFG in its current implementation is more efficient than the MFCF.

Algorithm 3. TMFG: Builds a planar Triangulated Maximally Filtered Graph.

Description: *Builds a planar Triangulated Maximally Filtered Graph by successively adding vertices with a T_2 move.*

Input:

> $\boldsymbol{W}$ [mandatory]: A dense $p \times p$ square matrix $\boldsymbol{W}$ with positive weights (e.g. a matrix of squared correlation coefficients). Alternatively, a data matrix of n observations and p variables and a gain function that operates on a triangle and a separator.

Output:

> A sparse matrix, $\boldsymbol{P}$, a filtered version of $\boldsymbol{W}$ fulfilling the planarity constraint. Or a matrix built from the cliques and separators produced by the algorithm.
> A list of cliques, $\mathscr{C}$.
> A list of separators, $\mathscr{S}$.
> A list of triangular faces $\mathscr{T}$.

Algorithm:

```
S1. [Initialize].
    $\mathscr{C} \leftarrow \varnothing$; $\mathscr{S} \leftarrow \varnothing$; $\mathscr{T} \leftarrow \varnothing$;
    $\mathscr{C}_1 \leftarrow \{v_1, v_2, v_3, v_4\}$. Assign to $\mathscr{C}_1$ the tetrahedron with highest score.
    $\mathscr{T} \leftarrow \{\{v_1,v_2,v_3\},\{v_1,v_2,v_4\},\{v_1,v_3,v_4\},\{v_2,v_3,v_4\}\}$. Assign the 4 triangular faces of
    $\mathscr{C}_1$ to $\mathscr{T}$.
    $\mathscr{C} \leftarrow \mathscr{C}_1$. Assign the first clique to the list of cliques.
    $\mathscr{V} \leftarrow \{v_5, \cdots, v_p\}$. Put the vertices not in $\mathscr{C}_1$ in the array of outstanding vertices.
    $\mathbf{P} \leftarrow \mathbf{W}(\mathscr{C}_1, \mathscr{C}_1)$. Assign the weights of the full matrix, restricted to the first clique, to the
    filtered matrix $\boldsymbol{P}$. If the input is a data-matrix, then $\mathbf{W}(\mathscr{C}_1, \mathscr{C}_1)$ must be calculated from the
    variables $\mathbf{X}_{\mathscr{C}_1}$
S2. [Build and update the gain table]. Calculate MaxGain and BestVertex for $\mathscr{T}$ and $\mathscr{V}$ as in
    Equation 66.
S3. While $\mathscr{V} \neq \varnothing$ do:
    S3.1 Find $t_i$ such that MaxGain is maximum.
    S3.2 Find the corresponding vertex $v_i$ in BestVertex.
    S3.3 Apply $T_2$: insert $v_i$ into $T_i$. This creates three new triangular faces: $t_a, t_b, t_c$.
    S3.4 [Update cliques, faces, outstanding vertices, separators and $\boldsymbol{P}$]
         $\mathscr{V} \leftarrow \mathscr{V} \setminus v_i$. Remove the vertex just added from outstanding vertices.
         $\mathscr{T} \leftarrow (\mathscr{T} \setminus \{t_i\}) \cup \{t_a, t_b, t_c\}$. Remove the triangular face from admissible triangular
         faces and add the three new faces introduced by the $T_2$ operator.
         $\mathscr{S} \leftarrow \mathscr{S} \cup t_i$. Add separator.
         $\mathscr{C}_i \leftarrow \mathscr{C}_i \cup \boldsymbol{Clique}(t_i, v_i)$. Add the new clique obtained from applying $T_2$ to $T_i$ and $v_i$.
         $\mathscr{P} \leftarrow \mathscr{P} + \mathscr{W}(\mathscr{C}_i, \mathscr{C}_i) - \mathscr{W}(\mathscr{S}_i, \mathscr{S}_i)$
    S3.5 [Update gain table]
         Remove from MaxGain and BestVertex where $t = t_i$.
         Update MaxGain and BestVertex where $v = v_i$.
         Calculate three new entries in MaxGain and BestVertex for $t_a, t_b, t_c$.
```

Remark 67 (Use of a matrix of data observations)**.** When the input is not a matrix of weights, the score function must be calculated from the observations of the variables. In these cases the meaning of the filtered matrix produced by the algorithm depends on the type of the score function. For instance, if the data are multivariate normal, then the natural filtered matrix to model is the inverse of the covariance matrix, as we will see in chapter 6. In other cases there might not be a natural specific meaning to the filtered matrix produced by the TMFG (for instance when the data is binomial or multinomial), but what retains meaning and should be used in the applications is the structure of cliques and separators.

4.3 Planarity invariant operators: $\mathbf{T_1}$, $\mathbf{T_2}$, $\mathbf{A}$, & $\mathbf{S}$

In addition to the PMFG and the deltahedron heuristic algorithms, which have inspired our approach, there have been other attempts along different lines. Jünger and Mutzel (1993) propose an approximate solution based on branch and cut. Poranen (2004) suggests a solution based on simulated annealing.

We now set out our approach to graph planarization in a way that is similar to the one to the one used for CF-invariant operators. We define *planarity invariant* operators and investigate some that will be used in section 4.4.

Definition 68 (Planarity invariant operators)**.** Let $G(V,E)$ a planar graph. A *planarity invariant* operator is a transformation of a planar graph (obtained from the combination of elementary operations such as addition of new vertices, swapping of vertices, addition of new edges, removal of existing edges etc.) such that the resulting graph is still planar.

The use of planarity invariant operators allows to explore only the graph construction strategies that do not violate planarity and therefore it provides scope for large performance improvements, as we do not need to test for the planarity of the graph at each stage.

We have seen how the basic deltahedron heuristic algorithm is in essence an instance of the MFCF algorithm where the size of the cliques is fixed to three (that is, it includes four vertices) and the separators are saturated after one expansion. In order to improve the performance of the algorithm we need to widen the set of topological operators allowed and we focus on planarity preserving operators.

In order to do that we relax the requirement to maintain chordality as an invariant and we move to analyse some further topological moves that maintain the planarity, but not necessarily the chordality, of the underlying graph.

These topological moves are sketched in Figs. 4.1, 4.6, 4.8 and 4.9. The T_1 operator acts on two triangles which share an edge (we will call such a construct a "plaquette") and replaces the shared edge with a new edge joining the (previously) opposite vertices (see Fig. 4.6); the T_2 move adds a vertex inside a triangle and connects this vertex to the triangle's vertices with three new edges (see Fig. 4.1); the *A* move operates on two triangles which share an edge by deleting the shared edge, adding a new vertex inside the resulting rhombus and joining the added vertex to the rhombus' vertices (see Fig. 4.8); finally the *S* operator swaps two vertices of the graph keeping fixed the neighbours (see Fig. 4.9). To complete the specification of the topological moves we should also add a score function and a description of the gain in score achieved in by every move. For the purposes of the MWPSP we will always use as a scoring function the sum of the edge weights introduced by the operators.

In an extension of the deltahedron heuristic method, suggested by Leung (Leung, 1992), vertex insertion can happen either one vertex at a time (the T_2 move, as in Fig.4.1) or three vertices at a time as in Fig.4.5. This corresponds to the insertion of an octahedron within a triangular face (Song et al., 2012b); clearly this is different from the T_2 move that instead corresponds to the insertion of a tetrahedron. However, such a move can be obtained by combining T_2 with another local move, called T_1 (Aste and Sherrington, 1999; Aste et al., 2012b,a; Dubertret et al., 1998), consisting in switching neighbours among two adjacent triangles, as shown in Fig.4.6. In general, any local topological change of a surface triangulation that preserves embedding and results in a triangulation can be realized through the combination of the two elementary moves T_1 and T_2 (Alexander, 1930). However it should be pointed out that the application of T_1 could cause the graph to become no longer chordal. For instance, the Leung extension (Fig.4.5) can be produced via two T_2 and one T_1, as demonstrated in Fig.4.7.

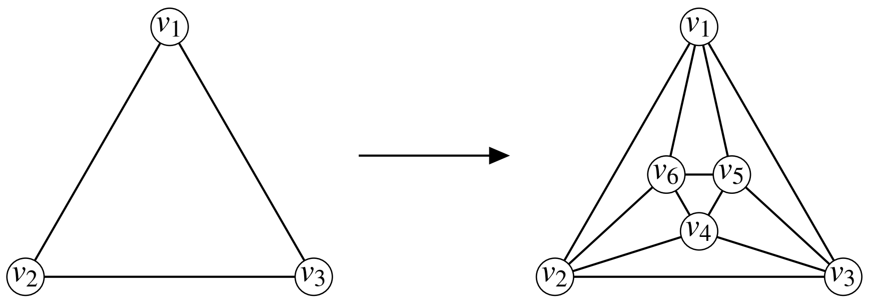


Figure 4.5: Addition of three vertices in Leung's extension of the deltahedron heuristic.

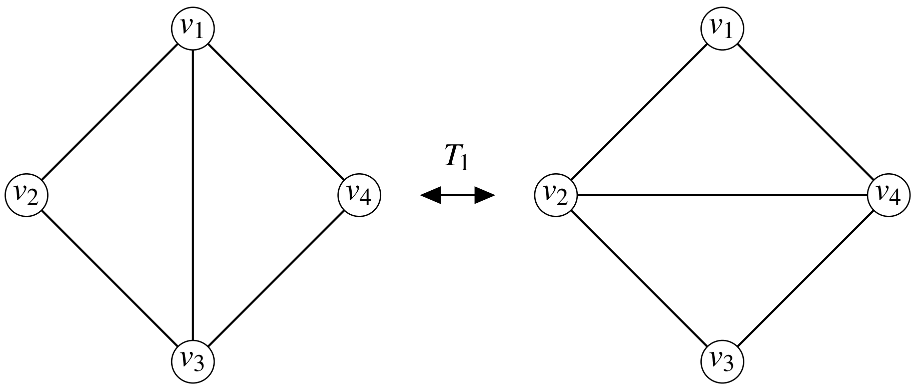


Figure 4.6: T_1 move: rewiring of a shared edge between neighbouring triangular faces.

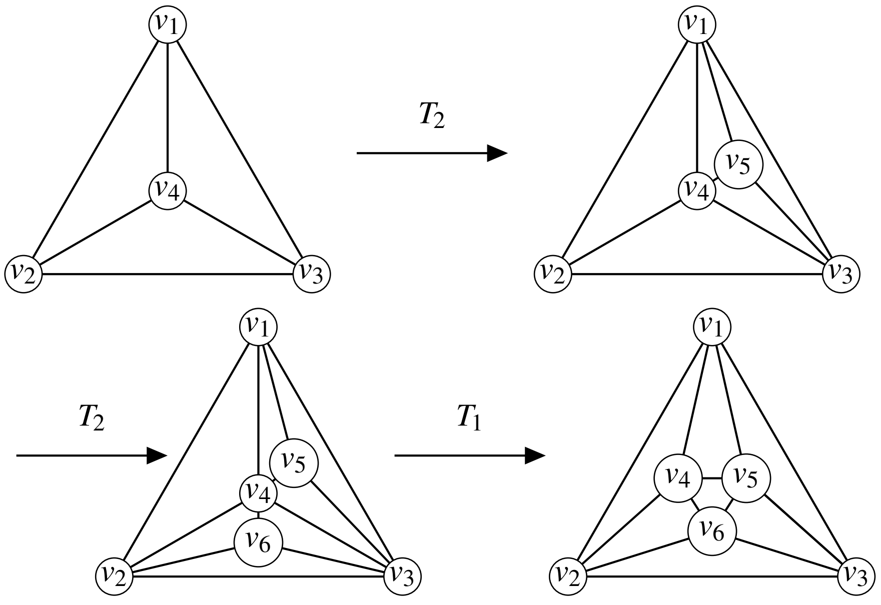


Figure 4.7: Demonstration that the Leung's extension in Fig.4.5 can be generated by using two T_2 and one T_1 moves.

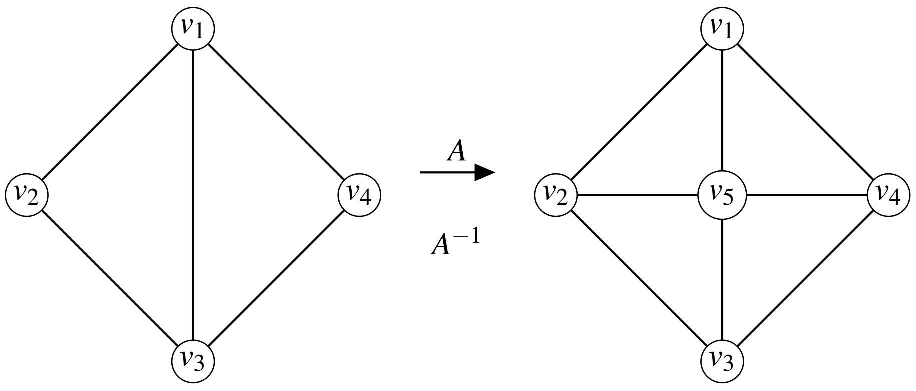


Figure 4.8: A move: insertion of a vertex inside a plaquette made of two neighbouring triangular faces.

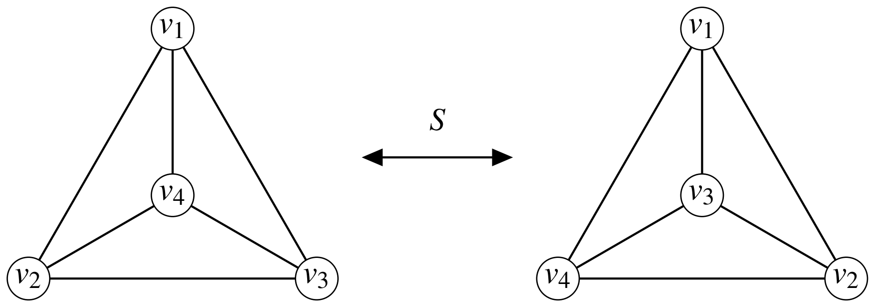


Figure 4.9: S move: relabelling of the vertices of a 4-simplex. Note that the topology of the graph is unchanged.

Another move that we will use to build planar graphs is the A move as described in Fig.4.8. Also in this case the move can be produced combining T_1 and T_2 and leads to non-chordal graphs.

Finally, we will use the 'swap' operator, S, that re-labels sub sets of vertices of a graph as shown in Fig.4.9, where it is acting on the vertices of a 4-simplex. This operation is trivial when the weights are identical, but will in general affect aggregate functions of the weights in a non-trivial way. The peculiarity of this operator is that it does not, in general operate locally and it keeps the graph topology unchanged preserving therefore planarity and chordality.

The TMFG algorithm can be extended to include T_1 and A moves as well (see Section 4.4). In this case, the moves are local, internal to the plaquette made by two joint triangles (i.e. $\{v_1, v_2, v_3\}$ and $\{v_2, v_3, v_4\}$ in Fig.4.6). The gain function for a T_1 move is associated to the removal of an edge (i.e. (v_1, v_3) in Fig.4.6) and the simultaneous addition of another edge (i.e. (v_2, v_4) in Fig.4.6). Similarly the gain for a move of type A (as shown in Fig.4.8) results from the removal of one edge and the insertion of a new vertex and four new edges. The use of T_1 and A moves generally improve gain; however, we have verified that the algorithm with T_2 only produces very similar results. Furthermore, planar filtered graph with T_1 or A moves are no longer clique trees but rather bubble-trees (Song et al., 2012a) which are in general no longer chordal. For instance see Fig.4.7, where the application of T_1 creates a non-chordal graph: the cycle $v_1 - v_2 - v_6 - v_5 - v_1$ has length grater than 3 without internal chords. This has implications for dependency modeling, as we shall discuss in Section 5.1. In many cases the application of the swap operator S results in higher overall gains. This operator has the advantage of leaving the overall topology unchanged but its use should be regulated by few local or heuristic criteria to avoid an increase in the complexity of the algorithm due to the increasing number of possible combinations. The S operator is the only operator that is not, in principle, local and therefore it could be applied to the whole network with clear implications for the computational complexity of the algorithm. To limit this issue we have chosen to use the S operator only locally (between adjacent vertices). Specifically, after every T_2 move we only execute the swaps that permute the vertices in the new clique. This requires some further changes to the cache vectors, but – being applied locally – it does not increase the overall computational complexity that remains $O(p^2)$.

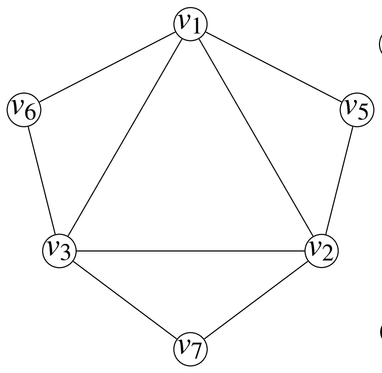
4.4 Variants of the TMFG algorithm

We have studied a number of variants of the basic TMFG algorithm. They all have in common the fact that one or more planarity preserving operators are applied after every execution of a T_2 operator. Since these operators are not in general CT-invariants the chordality of the filtered graph is not assured. Another consequence of this fact is that the the representation as a clique tree with cliques and separators is no longer valid. The TMFG variants exposed in this section must maintain the bookkeeping of the triangular faces for which a T_2 operator is feasible. The triangular faces constitute a basis for the cycle space of the graph (Diestel, 2010, Par. 4.5).

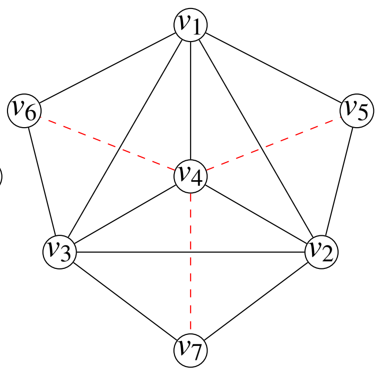
4.4.1 TMFG-T1

The TMFG-T1 (described in Algorithm 4) is the first proposed variant of the basic TMFG algorithm. The characteristic of this variant is a combination of a T_2 move followed by the evaluation of three T_1 moves, as shown in Figure 4.10. Specifically Figure 4.10a shows a triangular face (v_1, v_2, v_3) before the T_2 operator; in Figure 4.10b the T_2 is applied adding v_4 to the triangular face; the result id to create three plaquettes $((v_1, v_4, v_2, v_5), (v_2, v_4, v_3, v_7), (v_3, v_4, v_1, v_6))$ to which the T_1 operator could be applied. The TMFG-T1 calculates which of these edge flips would improve the total score and if one such edge flip is found, the corresponding T_1 operator is applied to the graph. The result for an hypothetical edges flip is shown in Figure 4.10c. After the T1 the cycle basis needs to be updated: with reference to figure 4.10 the two faces (v_1, v_2, v_4)

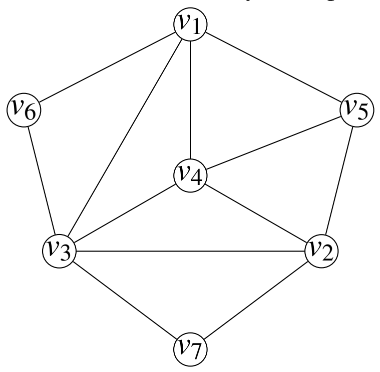
and (v_1, v_2, v_5) must be replaced by the two new faces (v_1, v_4, v_5) and (v_2, v_4, v_5). Additionally the algorithm keeps track of the contact structure between adjacent triangles to assess the possible edge flips. After every move the contact structure needs to be updated.



(a) Graph before the T_2 operator.



(b) After the T_2 operator. Three plaquettes are introduced: the dashed red lines show the edges that could be introduced by a T_1 operator.



(c) After the execution of the T_1 operator on one of the plaquettes.

Figure 4.10: Combination of T_2 and T_1 moves in the TMFG-T1 variant.

Remark 69. [Application of the T_1 move after a T_2 in the TMFG] Figure 4.10 shows how the T_1 operator is used in the TMFG-T1 variant to improve the total score after the execution of a T_2 move. Figure 4.10a shows a portion of a planar network before the T_2 is executed. In figure 4.10b we see that the execution of the T_2 creates the possibility for three T_1 swaps on three plaquettes: in (v_1, v_4, v_3, v_6) we could connect v_4 and v_6 and disconnect v_1 and v_3, in (v_1, v_4, v_2, v_5) we could connect v_1 and v_2 and disconnect v_4 and v_5, and in (v_3, v_4, v_2, v_7) we could connect v_4 and v_7 and disconnect v_2 and v_3. In each of the three cases the gain from the swaps results from the weight of the edge that would be connected minus the weight of the edge that would be disconnected by the move. If there are more than one positive gain to be achieved, the move with the maximum gain is carried out. In figure 4.10c we show the effect of applying the T_1 operator to the plaquette (v_1, v_4, v_2, v_5). As already observed, the result is not in general a clique tree, therefore the structure of cliques and separators is not useful. From an implementation point of view it is instead very useful to maintain in a data structure the composition of all the plaquettes. We have achieved that maintaining a parallel adjacency matrix for the triangles. We define that two traingles are adjacent only if they share an edge. In this way the plaquettes are the non zero elements of the adjacency matrix.

Algorithm 4. TMFG-T1: Builds a planar Maximally Filtered Graph. The graph is built out of a combination of T_2 and T_1 moves.

Description: *Builds a planar Maximally Filtered Graph by successively adding vertices with a T_2 move and a local optimisation based on T_1.*

Input:

$\boldsymbol{W}$ [mandatory]: A dense $p \times p$ square matrix $\boldsymbol{W}$ with positive weights (e.g. a matrix of squared correlation coefficients).

Output:

A sparse matrix, $\boldsymbol{P}$, a filtered version of $\boldsymbol{W}$ fulfilling the planarity constraint.

A list of cliques, $\mathscr{C}$.

A list of triangular faces $\mathscr{T}$.

A structure describing the adjacency relationship of the triangular faces, $\boldsymbol{A}$.

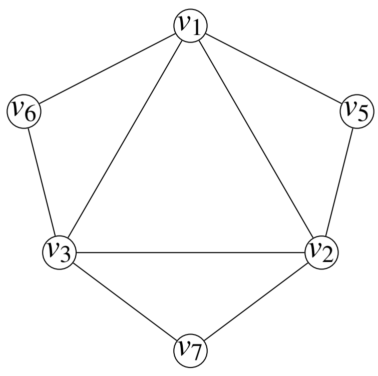
Algorithm:

```
S1. [Initialize].
    $\mathscr{C} \leftarrow \varnothing$. $\mathscr{T} \leftarrow \varnothing$.
    $\mathscr{A} \leftarrow 0$. Zero matrix.
    $\mathscr{C}_1 \leftarrow \{v_1, v_2, v_3, v_4\}$. Assign to $\mathscr{C}_1$ the tetrahedron with highest score.
    $\mathscr{T} \leftarrow \{\{v_1,v_2,v_3\},\{v_1,v_2,v_4\},\{v_1,v_3,v_4\},\{v_2,v_3,v_4\}\}$. Assign the 4 triangular faces of
    $\mathscr{C}_1$ to $\mathscr{T}$.
    $\mathscr{C} \leftarrow \mathscr{C}_1$. Assign the first clique to the list of cliques.
    $\mathscr{V} \leftarrow \{v_5, \cdots, v_p\}$. Put the vertices not in $\mathscr{C}_1$ in the array of outstanding vertices.
    $\mathbf{P} \leftarrow \mathbf{W}(\mathscr{C}_1, \mathscr{C}_1)$. Assign the weights of the full matrix, restricted to the first clique, to the
    filtered matrix $\boldsymbol{P}$.
```

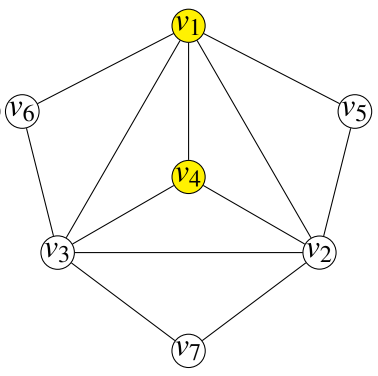
```
S2. [Build and update the gain table]. Calculate $\boldsymbol{MaxGain}$ and $\boldsymbol{BestVertex}$ for $\mathscr{T}$ and $\mathscr{V}$ as in Equation 66.
S3. While $\mathscr{V} \neq \varnothing$ do:
    S3.1 Find $t_i$ such that $\boldsymbol{MaxGain}$ is maximum.
    S3.2 Find the corresponding vertex $v_i$ in $\boldsymbol{BestVertex}$.
    S3.3 Apply $T_2$: insert $v_i$ into $T_i$. This creates three new triangular faces: $t_a, t_b, t_c$.
    S3.4 [Update cliques, faces, outstanding vertices, triangles adjacency matrix and $\boldsymbol{P}$]
         $\mathscr{V} \leftarrow \mathscr{V} \setminus v_i$. Remove the vertex just added from outstanding vertices.
         $\mathscr{T} \leftarrow (\mathscr{T} \setminus \{t_i\}) \cup \{t_a, t_b, t_c\}$. Remove the triangular face from admissible triangular faces and add the three new faces introduced by the $T_2$ operator.
         Update $\boldsymbol{A}$ by setting $t_a, t_b, t_c$ adjacent.
         TrianglesToUpdate $\leftarrow \{t_a, t_b, t_c\}$
         $\mathscr{C}_i \leftarrow \mathscr{C}_i \cup \boldsymbol{Clique}(t_i, v_i)$. Add the new clique obtained from applying $T_2$ to $T_i$ and $v_i$.
         $\mathscr{P} \leftarrow \mathscr{P} + \mathscr{W}(\mathscr{C}_i, \mathscr{C}_i) - \mathscr{W}(\mathscr{S}_i, \mathscr{S}_i)$
    S3.5 [Evaluate $T_1$]
         Get from $\boldsymbol{A}$ the neighbouring triangles of $t_i$, because those are the ones that might be altered by a $T_1$ move.
         For all triangles $t_a, t_b, t_c$ and their respective neighbouring triangles $t_{na}, t_{nb}, t_{nc}$ assess the gain from a $T_1$ move and execute it if there is a gain.
         If $T_1$ is executed for any of $t_{na}, t_{nb}, t_{nc}$ add the respective triangular face to the list $\boldsymbol{TrianglesToUpdate}$.
    S3.6 [Update gain table]
         Remove from $\boldsymbol{MaxGain}$ and $\boldsymbol{BestVertex}$ where $t = t_i$.
         Update $\boldsymbol{MaxGain}$ and $\boldsymbol{BestVertex}$ where $v = v_i$.
         Calculate the new entries in $\boldsymbol{MaxGain}$ and $\boldsymbol{BestVertex}$ for $\boldsymbol{TrianglesToUpdate}$.
```

4.4.2 TMFG-S

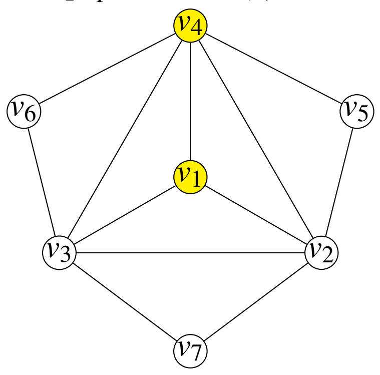
The TMFG-S algorithm (described in Algorithm 5) is a variant of the TMFG that evaluates potential gains due to possible swaps of vertices after every application of the T_4 operator, as shown in Figure 4.11. As both the T_2 and the S operators are CF-invariant, the end result is still chordal. However, care must be taken because after every swap the separator changes: for instance in Figure 4.11b the newly introduced separator is (v_1, v_2, v_3), but, after the swap, the separator is (v_2, v_3, v_4). A similar update must be performed for all the cliques and triangles containing the vertex that has been swapped: taking again Figure 4.11 as a reference all the triangles, such as (v_1, v_6, v_3) and (v_1, v_2, v_5) in Figure 4.11b that contain v_1 before the swap need to be updated to (v_4, v_6, v_3) and (v_4, v_2, v_5) as in Figure 4.11c.



(a) Graph before the T_2 operator.



(b) After the T_2 operator.



(c) After the swap of v_1 and v_4.

Figure 4.11: T_2 and subsequent swap of vertices v_1 and v_4, see remark 70 for a full description.

Remark 70 (Application of the S move after a T_2 in the TMFG)**.** Figure 4.11 shows how the S operator is used in the TMFG-S variant to improve the total score after the execution of a T_2 move. Since the S operator is non-local, we have restricted the scope to swaps between adjacent vertices after the execution of a T_2 operation. As the figure shows, the new vertex introduced by the T_2 is v_4. In the TMFG-S, the only swaps that are performed are those involving v_4 and its neighbours. The calculation of the gain involves the sum of the weights introduced by the swap minus the weights removed by the swap.

Algorithm 5. TMFGS: Builds a planar Triangulated Maximally Filtered Graph using a T_2 move and a series of vertex swaps.

Description: *Builds a planar Triangulated Maximally Filtered Graph by successively adding vertices with a T_2 move and executing swaps on the newly inserted clique..*

Input:

$\boldsymbol{W}$ [mandatory]: A dense $p \times p$ square matrix $\boldsymbol{W}$ with positive weights (e.g. a matrix of squared correlation coefficients).

Output:

A sparse matrix, $\boldsymbol{P}$, a filtered version of $\boldsymbol{W}$ fulfilling the planarity constraint.

A list of cliques, $\mathscr{C}$.

A list of separators, $\mathscr{S}$.

A list of triangular faces $\mathscr{T}$.

Algorithm:

```
S1. [Initialize].
    $\mathscr{C} \leftarrow \varnothing$; $\mathscr{S} \leftarrow \varnothing$; $\mathscr{T} \leftarrow \varnothing$;
    $\mathscr{C}_1 \leftarrow \{v_1, v_2, v_3, v_4\}$. Assign to $\mathscr{C}_1$ the tetrahedron with highest score.
    $\mathscr{T} \leftarrow \{\{v_1,v_2,v_3\},\{v_1,v_2,v_4\},\{v_1,v_3,v_4\},\{v_2,v_3,v_4\}\}$. Assign the 4 triangular faces of
    $\mathscr{C}_1$ to $\mathscr{T}$.
    $\mathscr{C} \leftarrow \mathscr{C}_1$. Assign the first clique to the list of cliques.
    $\mathscr{V} \leftarrow \{v_5, \cdots, v_p\}$. Put the vertices not in $\mathscr{C}_1$ in the array of outstanding vertices.
    $\mathbf{P} \leftarrow \mathbf{W}(\mathscr{C}_1, \mathscr{C}_1)$. Assign the weights of the full matrix, restricted to the first clique, to the
    filtered matrix P.
S2. [Build and update the gain table]. Calculate MaxGain and BestVertex for $\mathscr{T}$ and $\mathscr{V}$ as in
    Equation 66.
S3. While $\mathscr{V} \neq \varnothing$ do:
    S3.1 Find $t_i$ such that MaxGain is maximum.
    S3.2 Find the corresponding vertex $v_i$ in BestVertex.
    S3.3 Apply $T_2$: insert $v_i$ into $T_i$. This creates three new triangular faces: $t_a, t_b, t_c$.
    S3.4 [Update cliques, faces, outstanding vertices, separators and P]
         $\mathscr{V} \leftarrow \mathscr{V} \setminus v_i$. Remove the vertex just added from outstanding vertices.
         $\mathscr{T} \leftarrow (\mathscr{T} \setminus \{t_i\}) \cup \{t_a, t_b, t_c\}$. Remove the triangular face from admissible triangular
         faces and add the three new faces introduced by the $T_2$ operator.
         $\mathscr{C}_i \leftarrow \mathscr{C}_i \cup \boldsymbol{Clique}(t_i, v_i)$. Add the new clique obtained from applying $T_2$ to $T_i$ and $v_i$.
         $\mathscr{P} \leftarrow \mathscr{P} + \mathscr{W}(\mathscr{C}_i, \mathscr{C}_i) - \mathscr{W}(\mathscr{S}_i, \mathscr{S}_i)$
    S3.5 [Evaluate gains for all possible swaps of the clique]
         For all the possible permutations of the vertices of $\mathscr{C}_i$ assess the possible gain. This
         might change the separator.
    S3.6 [Add the separator] $\mathscr{S} \leftarrow \mathscr{S} \cup t_i$.
    S3.7 [Identify triangular faces affected by the swap] Identify all triangles containing one
         of the swapped vertices and update the triangular face.
    S3.8 [Update gain table]
         Remove from MaxGain and BestVertex where $t = t_i$.
```

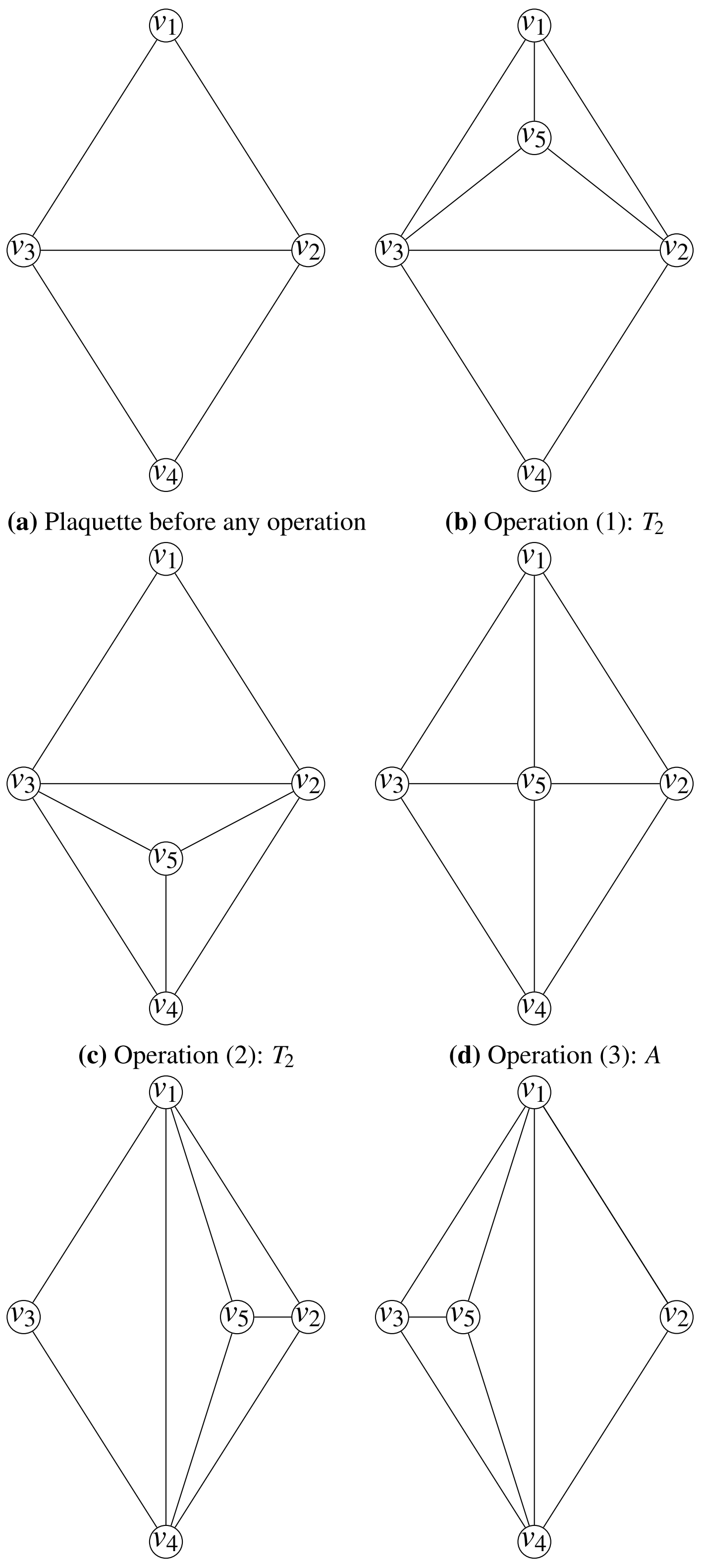
```
        Update MaxGain and BestVertex where $v = v_i$.
        Calculate three new entries in MaxGain and BestVertex for $t_a, t_b, t_c$.
        Update the gain table for all triangular faces that have been changed by the swap.
```

4.4.3 TMFG-A

The TMFG-A variant of the TMFG is a specialised version of the algorithm that works on plaquettes, as shown in 4.12 and algorithm 6. In this variant the gain table is a matrix with the rows and the columns indexed by the triangular faces of the graph. The non-zero entries of the matrix identify pair of adjacent triangles[11]. The gain is calculated by evaluating the change in score for each of the five operations described pictorially in Figure 4.12 in correspondence with every outstanding vertex. Differently from the previous variants here we have an additional explicit degree of freedom, the operation to be applied to the plaquette. The structure is initialised by building a 4-clique as in the other cases, but any subsequent move could potentially break the planarity of the graph and the end result will not be, in general, chordal. Every operation can potentially add new triangular faces or delete some existing ones, and change the matrix that records the adjacency of triangles. As a consequence of additions, removals or changes in structure, the gains related to the faces affected must be calculated or updated. Since the end result is not in general a clique forest, the TMFG-A does not return a list of cliques and separators, but simply the list of triangular faces. After the execution of one of the five operations, the TMFG-A can perform a further step where all the plaquettes are evaluated and a T_1 is executed if it improves the total score. This further optimisation step can be repeated a configurable number of times. This algorithm is rather more complex than the previous ones, and therefore we will sketch the high level steps, please refer to the full code listing in Appendix 9.1.4.

Remark 71. Figure 4.12a shows a plaquette in a planar graph. The TMFG-A works at the plaquette level because all the operations must be comparable and the operator A can only be defined on a plaquette. By looking at Figure 4.12 we can see the net effect of any operation is to add three edges: operations (1) and (2) add three new edges, operations (3)-(5) remove one edge and adds four new edges. In this way the gains can

[11]As we have done in section 4.4.1, we consider two triangles adjacent if they share an edge.



(a) Plaquette before any operation

(b) Operation (1): T_2

(c) Operation (2): T_2

(d) Operation (3): A

(e) Operation (4): $T_2 \circ T_1$

(f) Operation (5): $T_2 \circ T_1$

Figure 4.12: TMFG-A: a plaquette and the result of the five operations used by the algorithm, see remark 71 for a full description.

be compared. Operations (1) and (2) are no more than the T_2 applied to either triangle of the plaquette. Operation (3) is the A operator. Operations (4) and (5) are the T_2 operator preceded by a T_1. As in the case of the TMFG-T1, an important part of the algorithm is the maintenance of the triangle basis.

Algorithm 6. TMFG-A: Builds a planar Maximally Filtered Graph. The graph is built out of a combination of T_2, T_1 and A moves that operate on plaquettes.

Description: *Builds a planar Maximally Filtered Graph by successively adding vertices with a T_2 move and a local optimisation based on T_1.*

Input:

$\boldsymbol{W}$ [mandatory]: A dense $p \times p$ square matrix $\boldsymbol{W}$ with positive weights (e.g. a matrix of squared correlation coefficients).

NumFlips [mandatory]: the number of cycles of T_1 edge flips that must be considered after every main operation.

Output:

A sparse matrix, $\boldsymbol{P}$, a filtered version of $\boldsymbol{W}$ fulfilling the planarity constraint.

A list of triangular faces $\mathscr{T}$.

A structure describing the adjacency relationship of the triangular faces, $\mathscr{A}_t$.

Algorithm:

```
S1. [Initialize].
    $\mathscr{T} \leftarrow \varnothing$.
    $\mathscr{A}_t \leftarrow 0$. Zero matrix.
    $\mathscr{C}_1 \leftarrow \{v_1, v_2, v_3, v_4\}$. Assign to $\mathscr{C}_1$ the tetrahedron with highest score.
    $\mathscr{T} \leftarrow \{\{v_1,v_2,v_3\},\{v_1,v_2,v_4\},\{v_1,v_3,v_4\},\{v_2,v_3,v_4\}\}$. Assign the 4 triangular faces of
    $\mathscr{C}_1$ to $\mathscr{T}$.
    $\mathscr{V} \leftarrow \{v_5, \cdots, v_p\}$. Put the vertices not in $\mathscr{C}_1$ in the array of outstanding vertices.
    $\mathbf{P} \leftarrow \mathbf{W}(\mathscr{C}_1, \mathscr{C}_1)$. Assign the weights of the full matrix, restricted to the first clique, to the
    filtered matrix $\boldsymbol{P}$.
    The triangles $t_1 = \{v_2,v_3,v_4\}, t_2 = \{v_1,v_3,v_4\}, t_3 = \{v_1,v_2,v_4\}, t_4 = \{v_1,v_2,v_3\}$ are all ad-
    jacent to each other, therefore they are marked as adjacent in $\mathscr{A}_t$. $\mathscr{A}_t(t_1,t_2) \leftarrow 1, \mathscr{A}_t(t_2,t_3) \leftarrow$
    $1, \mathscr{A}_t(t_1,t_3) \leftarrow 1, \mathscr{A}_t(t_1,t_4) \leftarrow 1$.
S2. [Build and update the gain table]. Calculate MaxGain, BestVertex, and BestOp for $\mathscr{T}$ and
    $\mathscr{V}$. In this algorithm these three entries in the gain table are indexed by pairs of adjacent
    traingles, or plaquettes. The gain table in this case contains also the operation, described in
    Figure 4.12.
```

```
S3. While $\mathscr{V} \neq \varnothing$ do:
    S3.1 Find $t_i, t_j$ such that MaxGain is maximum.
    S3.2 Find the corresponding vertex $v_k$ in BestVertex.
    S3.3 Find the corresponding operation $o_k$ in BestOp.
    S3.4 Apply $o_k$ to $v_k$ and $t_i, t_j$. This creates three (for operations 1,2,4,5) or four (operation 3) new triangular faces ($NewFaces$) and removes one (operations 1,2,4,5) or two (operation 3) of the existing triangular faces ($OldFaces$).
    S3.5 [Update faces, outstanding vertices, triangles adjacency matrix and P]
         $\mathscr{V} \leftarrow \mathscr{V} \setminus v_i$. Remove the vertex just added from outstanding vertices.
         Remove the triangular face(s) from admissible triangular faces and add the new triangular faces introduced by the $o_k$ operator.
         Update $\mathscr{A}_t$ by marking as adjacent the appropriate faces on $NewFaces$.
         TrianglesToUpdate $\leftarrow NewFaces$
         Update $\mathscr{P}$ by adding the newly introduced edges and removing the edges from the triangles that have been removed.
    S3.6 [Update gain table]
         Remove from MaxGain, BestVertex and BestOp where the plaquette is $t_i, t_j$.
         Update MaxGain, BestVertex and BestOp where $v = v_k$.
         Perform NumSwaps iterations of $T_1$ operations on all the triangles, updating TrianglesToUpdate. Calculate the new entries in MaxGain, BestVertex and BestOp for TrianglesToUpdate.
```

4.5 Additional observations

In this section we collect some observation related to practical implementation issues or to potential improvements to the algorithm. Some of these topics will be also mentioned in chapter 8 since they concern future research.

4.5.1 Dynamical adaptability

Due to the local nature of the operators, T_1, T_2, T_2^{-1}, A, A^{-1} and (local) S, used to construct the TMFG one can continuously modify the network allowing ‘online’ adaptability while new data are generated. This is of practical importance because in real, big data, applications information is changing dynamically with new data continuously fed causing changes in the matrix of weights that require modifications of the filtered graph. Further, creation of new nodes is required when new elements/variables become relevant in the system. Conversely, elements/variables can eventually become irrelevant and the corresponding vertices should be eliminated from the graph. The implementation of these moves requires keeping a cache matrix of gains continuously updated and dynamically checking for moves that improve total gains. An interesting topic for further research would be the development of effective criteria that indicate whether the network structure is no longer adequate to the data under analysis and should therefore be updated.

4.5.2 Parallelization and big data

The local nature of TMFG construction and dynamical adaptation through T_1, T_2, T_2^{-1}, A, A^{-1} and (local) S, moves make it suitable for parallelization. Coelho et al. (2016) proposes several possibilities for parallelization based on the TMFG algorithm. Let us however discuss briefly a possible parallel implementation of the TMFG. One of the main features of planar triangulations is that three-cliques uniquely divide the network into two 'inside' and 'outside' subgraphs within a nested hierarchical structure (Song et al., 2011). This means that, given a seed structure of three-cliques, each clique can develop its inside subgraph independently. A processor can be assigned to each seed clique and calculations can be performed locally. Given that each separating clique divides roughly the graph into two parts one can compute the TMFG in $O(p)$ using $O(\log p)$ processors. Another issue related to big data is the size of the score vectors. It is clear from the construction that the size of the cache grows linearly with the dimension of the problem and that triangles in the basis can be assigned to different processors, allowing parallel updates of the cache.

4.5.3 Memory usage

In the case of pair-wise dependence (such as correlation) both the deltahedron heuristic and the PMFG require to compute in advance the entire correlation matrix, while the TMFG does not use the full information from the correlation matrix and could calculate only the correlations necessary for the incremental update of the gain vectors. This is an advantage already for correlation measures, in the (numerous) cases where the number of observations (q) is less than the number of variables (p): in fact it could require much less memory (approximately $p \times q$) to store the time series of the observations and calculate the correlations on-demand, rather than calculating and storing a large correlation matrix (approximately $\frac{p\times(p-1)}{2}$). This fact is even more relevant for multi-point dependencies (e.g. partial correlation, mutual information, ...): in these cases the

TMFG would still require only to store the time series in memory and would require the calculation of the relevant gain functions only, while other methods would require the storage of large amounts of data (e.g. order of p^3 for a three-points dependency measure).

4.6 Comparison between TMFG and PMFG

A vast literature has demonstrated that PMFG can be used to retrieve meaningful information about the structure of interdependency in complex datasets (Tumminello et al., 2005; Song et al., 2012a; Musmeci et al., 2015a,b; Pozzi et al., 2013), it is therefore natural to compare the performances of the TMFG with the ones of the PMFG.

Let us first look at the scaling of execution times for TMFG and PMFG algorithms as function of the size p of the weight matrix W; results are reported in Fig.4.13 (seconds on a 2.6 GHz Intel Core i7®). We observe that TMFG execution times scale with the matrix dimension size p approximately as $O(p^2)$ while PMFG scales approximately as $O(p^3)$. The 2-parameters best polynomial fits give respectively: $T_{TMFG} \sim 2 \cdot 10^{-7} \cdot p^2 + 6 \cdot 10^{-4} p$ and $T_{PMFG} \sim 2 \cdot 10^{-6} \cdot p^3 + 3 \cdot 10^{-5} p^2$. Overall we can see that execution times are several orders of magnitude faster for TMFG than PMFG.

4.6.1 Comparison between the performances of the various methods

We have then compared the total retained edge weight for the following four variants of the TMFG construction:

1. TMFG: the base version of the algorithm described in Section 4.2
2. TMFG-T1: the variant described in 4.4.1.
3. TMFG-S: the variant described in 4.4.2.
4. TMFG-A: the variant described in 4.4.3.

In choosing the matrices to use for the test we have tried to cover a number of distributions: with limited or unlimited range, with or without fat tails, with very high right or left skew. Besides, 5 of the matrices have been sampled element-wise, and therefore there is no correlation between the edge weights. Three of the matrices have

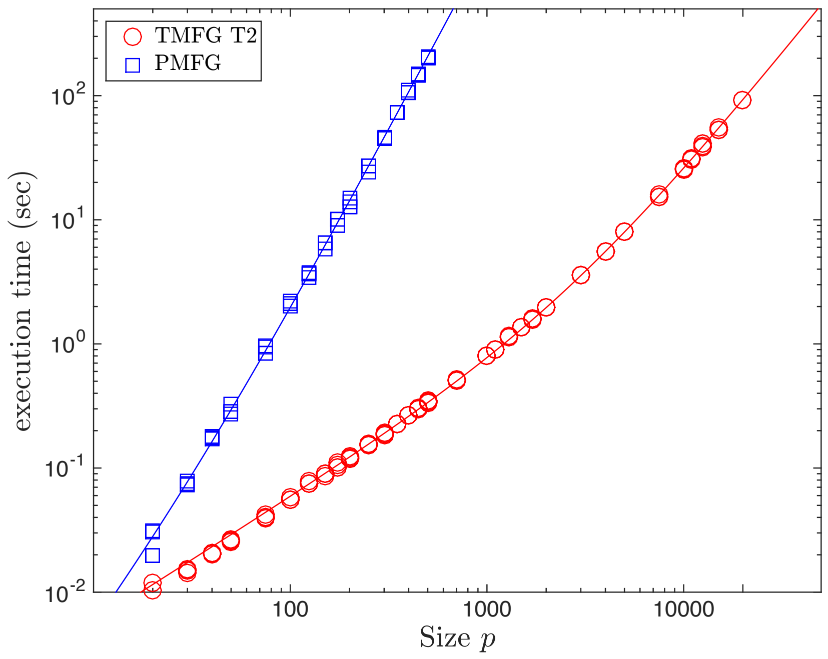


Figure 4.13: Demonstration that TMFG is faster and scalable with respect to the PMFG. Comparison between execution times for TMFG and PMFG for different values of p ranging between 50 and 10000. Lines are the 2-parameters best polynomial fits (see text).

been simulated using a factor model with an increasing number of factors; these matrices should show how the algorithms behave when there is an underlying structure in the matrix of weights. Finally we have included a matrix built from real data using the correlation coefficients of the log-returns of a set of 342 US stocks (the dataset is described in (Massara et al., 2016)). We will see in the discussion of the results that this choice allows to highlight differences between the two algorithms.

In total we have tested 9 types of random weight matrices W with different weight distributions:

1. Beta distribution with shape parameters $\alpha = 0.5$ and $\beta = 3$. This distribution is heavily skewed and is characterised by very low density on the right side of the interval $[0, 1]$ and therefore very few elements should be close to one. In this distribution the elements of the matrix are independent from one another.

2. Beta distribution with shape parameters $\alpha = 3$ and $\beta = 0.5$. This distribution is skewed in the opposite direction and has a high density near the right extreme

of the interval $[0,1]$, with many elements close to 1. The matrix elements are independent.

3. Pareto distribution with power law exponent equal to 1. This distribution has a fat tail, and this means that there are isolated cases with very high weights. Also in this case the elements are independent.

4. Pareto distribution with power law exponent equal to 2. This distribution has still a fat tail, but thinner than the previous one. The elements are independent.

5. Random matrix of correlations of 400 time series generated by simulating 20 normally distributed common factors. Differently from the previous distributions, where the edges are assigned weights independently from one another, we have chosen this distribution because it is commonly used to model cases where variables react similarly to common factors.

6. Random matrix of correlations of 400 time series generated by simulating 50 common factors. This matrix shows less structure than the one generated using 20 factors.

7. Random matrix of correlations of 400 time series generated by simulating 100 common factors. This matrix shows less structure than the two above and should be more similar to a matrix with element-wise random elements, due to the very high number of model parameters (400×100 factor loadings).

8. Uniform distribution over $[0,1]$. This distribution has independent elements but a relatively high number of weights close to 1.

9. Square of a real correlation matrix coefficients computed from daily log-returns of 342 US stocks, across a period of 15 years (form Jan 1997 to Jul 2012) (Musmeci et al., 2015a).

All matrices are symmetric and have size $p = 400$ except the real correlation data that have sizes $p = 342$. For all the weight matrices (excepting for the real correlation matrix) we have compared results for 100 samples. For the real correlation matrices we generated matrices by random sampling the starting point of 100 time windows of length 1000 data points over a period of 4500 points in total. Table 4.1 reports the average relative performance, defined as the ratio between the sum of the edge weight in the four variants of TMFG with respect to the sum of the edge weight in the PMFG. It shows that the PMFG is usually more effective when the density of high weights is low, while the TMFG is more effective when the density of high weights is high or the range of weights is limited. This result is to be expected since the PMFG is less constrained than the TMFG in picking up isolated high-weight edges one at a time, while the TMFG is more efficient in selecting subsets of edges with a high total sum. For the random matrices of correlation we see that the TMFG performs better than the PMFG in filtering the more structured matrix generated using 20 factors. In the real case we see that the TMFG is marginally better than the PMFG. We conclude that TMFG is in general performing comparably well, and sometimes better than the PMFG. We observe that the TMFG tends to improve relative performance as the size of the matrix increases (see Table 4.2). We observe that in this case, even with an unfavourable distribution the TMFG variants will eventually outperform the PMFG. We explain this by observing that the number of elements grows approximately as $0.5 \times p^2$, while the number of edges in the filtered graphs grows as $3 \times p$; this means that there are relatively more edges with high weights in the right side of the distribution allowing the TMFG to optimise better the sum by cliques. All the TMFG variants exhibit a performance improvement with the best result obtained by the TMFG-A variant.

Weight matrix coefficients distribution	TMFG/ PMFG	TMFG-T1/ PMFG	TMFG-S/ PMFG	TMFG-A/ PMFG	TMFG (Time)/ PMFG (Time)
Beta(0.5,3)	95.42%	96.24%	95.72%	99.89%	0.16%
Beta(3, 0.5)	104.70%	104.73%	104.77%	104.80%	0.14%
Pareto(1)	99.97%	99.97%	99.97%	99.97%	0.17%
Pareto(2)	97.94%	98.00%	98.02%	98.32%	0.17%
Random Matrix (20 factors)	102.23%	102.63%	102.57%	103.77%	0.22%
Random Matrix (50 factors)	100.30%	100.82%	100.64%	102.54%	0.21%
Random Matrix (100 factors)	98.46%	99.14%	98.86%	101.42 %	0.21%
Uniform	116.27%	116.29%	116.34%	116.89%	0.15%
Real correlation matrix	100.11%	100.17%	100.24%	100.42%	0.15%

Table 4.1: Average relative performances (ratio between sum of edge weights) of the TMFG algorithm with respect to the PMFG. Four TMFG variants and nine different weight distributions. Note that TMFG and TMFG-S are chordal graphs.

Weight matrix size p	TMFG /PMFG	TMFG-T1 /PMFG	TMFG-S /PMFG	TMFG-A /PMFG
50	88.68%	88.82%	89.35%	95.93%
100	90.49%	93.13%	92.31%	98.14%
150	92.14%	93.77%	90.44%	95.26%
300	93.73%	95.6%	94.63%	100.06%
500	96.36%	96.79%	96.6%	100.98%
700	98.83%	100.49%	98.92%	103.58%
850	98.93%	99.95%	99.99%	103.83%
1000	100.33%	100.56%	100.71%	105.39%
1200	101.34%	102.16%	101.16%	105.24%

Table 4.2: Example of relative increase in performance of the TMFG algorithm with respect to PMFG when dimensionality p increases. The underlying distribution is a Beta(0.5, 3).

Chapter 5

Probabilistic modelling with TMFG / MFCF and Financial Applications

In chapter 3 we have discussed in general terms the MFCF methodology and in chapter 4 we have discussed the TMFG and its variants with a view towards the solution of the MWPSP.

In this chapter we show how probabilistic models can be built upon these network structures and we will discuss some financial risk management applications.

We first describe in in Section 5.1 the use of Gaussian Markov Random Fields (GMRF) in relation to the TMFG. The algorithms described in this chapter have been initially discussed in Barfuss et al. (2016) using 4-dimensional cliques, but the the theory holds for any clique forests and therefore these results can be applied to any network generated by the MFCF. The methodology described in Barfuss et al. (2016), which describes the use of the the TMFG to build a GMRF where all the cliques have the same size of 4, goes under the name of LoGo (Local / Global optimisation). We provide two applications to stress testing and risk allocation in Section 5.2.

We the describe in Section 5.3 the implementation of the MFCF with a gain function based on linear regression. We show how the local nature of the MFCF can be used to reliably build a network of financial and macroeconomic variables when the time series have mixed frequencies.

5.1 Modeling with TMFG: information theoretic perspective

In complex systems, such as financial markets, a large number of interdependent variables are typically involved. The TMFG, as a particular case of the MFCF, is a way of filtering the dependence structure between the variables reducing it to a network consisting of the most relevant interactions.

This section is an extension of Section 3.3.3, where several gain functions have been discussed in the general case of a clique tree. In this case the score function discussed is the mutual information, or the negative of the Kullback-Leibler divergence of the joint probability distribution and the product of the marginals. We present the results in the original setting of a four-dimensional clique tree[1] since this can be directly compared with the simpler setting of the Chow-Liu tree (Chow and Liu, 1968) and the more general setting of a general clique tree underlying a multivariate normal distribution as previously described in Section 3.3.3.3.

Modeling the system statistically consists in identifying the joint probability distribution that best describes the observed collective behavior of the variables.

Specifically, given a set of observations $\{x_1(1),...,x_1(q)\}$, $\{x_2(1),...,x_2(q)\}$, ... $\{x_p(1),...,x_p(q)\}$ of p random variables $\mathbf{X} = \{X_1, X_2, ..., X_p\}$, one aims to estimate a joint probability distribution function $Q(\mathbf{X})$ that is the best representation of the 'true' multivariate probability distribution function $P(\mathbf{X})$ from which the set of observations are drawn. Clearly, $P(\mathbf{X})$ is unknown and the only information available are the observations $\{x_1(1),...,x_1(q)\}$, $\{x_2(1),...,x_2(q)\}$, ... $\{x_p(1),...,x_p(q)\}$ from which $Q(\mathbf{X})$ must be estimated.

Information filtering graphs can be used to compute $Q(\mathbf{X})$. The main advantage is that these graphs are locally low dimensional (e.g. the largest clique is K_4 when planarity is enforced) which makes sampling tractable also with limited amount of data (Barfuss et al., 2016). We have explained in Chapter 3 how the repeated operation of the clique expansion operator creates a clique forest[2] and thus $Q(\mathbf{X})$ admits a representation based on clique and separators potentials as per Equation 5.1:

[1] In this context "four-dimensional" refers to the maximum number of vertices allowed in the clique, not to the dimension of the underlying simplicial complex, which is three.

[2] And the T_2 operator as a particular case of the clique expansion operator is no exception.

$$Q(\mathbf{X}) = \frac{\prod_{c \in \mathscr{C}} P_c(\mathbf{X}_c)}{\prod_{s \in \mathscr{S}} P_s(\mathbf{X}_s)} \ . \tag{5.1}$$

Where: $\mathscr{C}$ and $\mathscr{S}$ are respectively the set of cliques and separators of the graph and $P_c(\mathbf{X}_c)$ and $P_s(\mathbf{X}_s)$ are the marginal probabilities of the sub sets of variables X_c and X_s associated respectively with the 4-clique c and the triangular separator s.

Equation (5.1) reduces the p-dimensional problem of estimating the joint probability distribution function $Q(\mathbf{X})$ to the estimation of a set of 3- and 4-dimensional local marginal probabilities $P_s(\mathbf{X}_s)$ and $P_c(\mathbf{X}_c)$ (Barfuss et al., 2016)[3]. Such a reduction of a global high-dimensional problem to a set of local low-dimensional problems helps greatly in the estimation of the joint probability. The open question is now to measure how well the, unknown, true joint distribution $P(\mathbf{X})$ is represented by the model estimation $Q(\mathbf{X})$ factorized over the TMFG. To this end we can measure the dissimilarity between the two probability density functions which is given by the Kullback-Leibler divergence (Kullback and Leibler, 1951):

$$D_{KL}(P \parallel Q) = \sum_{\mathbf{X}} P(\mathbf{X}) \log\left(\frac{P(\mathbf{X})}{Q(\mathbf{X})}\right) \ ; \tag{5.2}$$

By substituting Eq.5.1 into Eq.5.2 we write the Kullback-Leibler divergence as follows:

$$\begin{aligned} D_{KL}(P \parallel Q) = & \sum_{\mathbf{X}} P(\mathbf{X}) \log(P(\mathbf{X})) \\ & - \sum_{c \in \mathscr{C}} \sum_{\mathbf{X}_c} P_c(\mathbf{X}_c) \log(P_c(\mathbf{X}_c)) \\ & + \sum_{s \in \mathscr{S}} \sum_{\mathbf{X}_s} P_s(\mathbf{X}_s) \log(P_s(\mathbf{X}_s)) \ . \end{aligned} \tag{5.3}$$

From a information theoretic perspective the first term in Eq.5.3 (with a minus sign),

$$H = -\sum_{\mathbf{X}} P(\mathbf{X}) \log(P(\mathbf{X})) \ , \tag{5.4}$$

[3]However, note that equation 5.1 is completely general and can accommodate any structure of cliques and separators as long as they represent a clique forest. The problem of finding the best trade off between the goodness of fit and the size of the cliques is rather subtle, and it is discussed in greater detail in chapter 6 in the context of the problem of covariance selection.

quantifies the total amount of uncertainty in the system, measuring the number of bytes (if base-2 logarithms are used) necessary to define a state. The other two remaining terms in Eq.5.3:

$$H_m = -\sum_{c\in\mathscr{C}}\sum_{\mathbf{X}_c} P_c(\mathbf{X}_c)\log\left(P_c(\mathbf{X}_c)\right) + \sum_{s\in\mathscr{S}}\sum_{\mathbf{X}_s} P_s(\mathbf{X}_s)\log\left(P_s(\mathbf{X}_s)\right) \tag{5.5}$$

also quantify an uncertainty, but in this case, associated with the model of the system. In other words, by adopting the TMFG structure of interactions, H_m measures the number of bytes necessary to define the state of the system when we consider only the dependencies among variables associated with edges in the TMFG.

An algorithm to construct the TMFG with the aim of minimizing $D_{KL}(P \parallel Q)$ can be implemented by choosing at every stage the move that minimally increases H_m consistently with all other constraints.

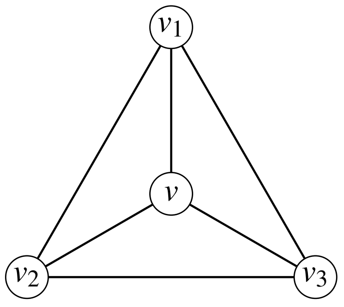


Figure 5.1: Illustration of Equation 5.3, here $t = v_1, v_2, v_3$, the new vertex is v and $u = v_1, v_2, v_3, v$.

In particular, considering the TMFG construction via T_2 moves (as in Figure 5.1, the contribution to $D_{KL}(P \parallel Q)$ from the insertion of a vertex v added inside an existing triangular face t generating a 4-clique $u = t \cup v$ is:

$$S(v,t) = \sum_{\mathbf{X}_u} P_u(\mathbf{X}_u)\log\left(P_u(\mathbf{X}_u)\right) - \sum_{\mathbf{X}_t} P_t(\mathbf{X}_t)\log\left(P_t(\mathbf{X}_t)\right) \ . \tag{5.6}$$

From an information theoretic perspective $-S$ is the amount of uncertainty introduced in the model by including a variable v, the TMFG structure should be constructed in a way to minimize such uncertainty. (Barfuss et al., 2016) discusses in detail the case for normal multivariate distributions.

Note that, in practice, to compute Eq.5.3 one must substitute the – unknown – marginal distributions of the real probability $P_c(\mathbf{X}_c)$ and $P_e(\mathbf{X}_s)$ with the corresponding empirical estimators $\hat{P}_c(\mathbf{X}_c)$ and $\hat{P}_c(\mathbf{X}_s)$ and the equality in Eq.5.3 would therefore become an approximate estimate. This is consistent with our approach as far as the empirical estimators are the MLE estimate of the real marginal distributions of cliques and separators.

It is a known fact that GMRFs are completely defined by the mean and by the inverse covariance matrix (Rue and Held, 2005). Following the theory of decomposable graphical models it is possible to show that the inverse covariance matrix of a clique forest can be estimated as[4]:

$$J = \sum_{c \in \mathscr{C}} \left[(\Sigma_c)^{-1} \right]^V - \sum_{s \in \mathscr{S}} \left[(\Sigma_s)^{-1} \right]^V \tag{5.7}$$

The same equation is discussed in section 6.2.1 in the context of general clique forests. We have, therefore, an easy way to estimate a sparse precision matrix given the structure of the clique forest. The estimation of the matrix allows to develop a number of applications in financial risk management. We note how the estimation of the inverse (Equation 5.8) can be performed by assembling the local clique estimates of the inverse matrix.

$$J = \sum_{c \in \mathscr{C}} [\tilde{J}_c]^V - \sum_{s \in \mathscr{S}} [\tilde{J}_s]^V \tag{5.8}$$

5.2 Financial Applications of the TMFG

5.2.1 Financial applications: Stress Testing

A typical stress test for financial institutions, required by regulatory bodies, consists in forecasting the effect of severe financial and economic shocks on the balance sheet of a financial institution. In this context let us reformulate the previous results by considering X_1 the set of economic and financial variables that can be shocked and X_2 the set of the securities held in an institution's portfolio. Assuming that all the changes in the economic and financial variables and in the assets of the portfolio can be modelled as a GMRF, the distribution of the returns of the portfolio (X_2) conditional

[4] See Lauritzen (1996, Prop. 5.9), Barfuss et al. (2016) or section 6.2.1.

on the realization of the economic and financial shocks (X_1) can be written as[5]:

$$\mathbf{J}_{2,2}\mu_{2|1} = -\mathbf{J}_{2,1}X_1 \quad , \tag{5.9}$$

Where the conditional expectation values $\mu_{2|1}$ can be calculated from the conditional joint distribution function, which, from the Bayes theorem, is $f(X_2|X_1) = f(X_2,X_1)/f(X_1)$.

An approach along similar lines was proposed in Rebonato (2010a); Rebonato and Denev (2014); Denev (2015). We note that with the LoGo approach we have a sparse relationship between the financial variables and the securities. This makes calibration more robust and it can be insightful to identify mechanisms of impact and vulnerabilities.

5.2.2 Risk Allocation

A second application is the calculation of conditional statistics in the presence of linear constraints (see Rue and Held (2005)). In this case we indicate with $\mathbf{X}$ a set of p random variables associated with the returns in a portfolio of p assets and with $\mathbf{J}$ the associated sparse inverse covariance matrix. Let $\mathbf{w} \in \mathbb{R}^{p\times 1}$ be the vector of holdings of the portfolio, then $\mathbf{w}^\top \cdot \mathbf{X}$ is the return of the portfolio. An important question in portfolio management is to allocate profits and losses to different assets conditional on a given level of profit or loss, which is equivalent to knowing the distribution of returns conditional on a given level of loss $\mathbf{X}|\mathbf{w}^\top \cdot \mathbf{X} = \mathbf{L}$. More generally we want to estimate $\mathbf{X}|\mathbf{A}\cdot\mathbf{X} = \mathbf{z}$ where $\mathbf{A} \in \mathbb{R}^{k\times p}$ is generally a low rank k ($k = 1$ in our example) matrix that specifies k hard linear constraints. Using the Lagrange Multipliers method (see Strang (1986) for an introduction) the conditional mean is calculated as (Rue and Held, 2005):

$$\mathbb{E}\left(\mathbf{X}|\mathbf{A}\cdot\mathbf{X} = \mathbf{z}\right) = \mathbf{A}\mathbf{J}^{-1}\left(\mathbf{A}\mathbf{J}^{-1}\mathbf{A}^\top\right)^{-1}\mathbf{z} \tag{5.10}$$

and the conditional covariance is:

$$\mathbf{Cov}\left(\mathbf{X}|\mathbf{A}\cdot\mathbf{X} = \mathbf{z}\right) = \mathbf{J}^{-1} - \mathbf{J}^{-1}\mathbf{A}^\top\left(\mathbf{A}\mathbf{J}^{-1}\mathbf{A}^\top\right)^{-1}\mathbf{A}\mathbf{J}^{-1} \quad . \tag{5.11}$$

[5]This is a standard result on multivariate normal distributions, for a proof see e.g. Lauritzen (1996, Prop. C.5)

In case **J** is estimated using decomposable Information Filtering Networks (MST or TMFG) then it can be written as a sum of smaller matrices involving cliques and separators:

$$\mathbf{J} = \sum_{\mathscr{C} \in Cliques} \mathbf{J}_{\mathscr{C}} - \sum_{\mathscr{S} \in Separators} \mathbf{J}_{\mathscr{S}} \tag{5.12}$$

This decomposition allows for a sparse and potentially parallel evaluation of the matrix products in Eqs. 5.10 and 5.11.

This framework can therefore be used to build the Profit/Loss (P/L) distribution of a portfolio, conditionally on a number of explanatory variables, and to allocate the P/L to the different assets conditional on the realization of a given level of profit and loss. The solution is analytical and therefore extremely quick. Besides, given the decomposability of the portfolio, Eq.5.12 allows to calculate important statistics in parallel, by applying the calculations locally to the cliques and to the separators. For instance, it is a simple exercise to show that the unconditional expected P/L and the unconditional volatility can be calculated in parallel by adding the contributions of the cliques and subtracting the contributions of the separators. In summary LoGo provides the possibility to build a basic risk management framework that allows risk aggregation, allocation, stress testing and scenario analysis in a multivariate Gaussian framework in a quick and potentially parallel fashion.

5.3 Financial Applications of the MFCF

One of the problems that affect LoGo, and indeed one of the main motivations for the development of the MFCF, is that the dimension of the cliques is fixed and this means that, while the performance is overall satisfactory, there are edges included in the network that are not necessarily significant and, conversely, some significant edges cannot be included in the network due to the constraint on the size of cliques. This issue is more acute when the analysis must be performed on time series that have mixed frequencies, as this affects the statistical significance of the regression or correlation coefficients. On the other hand is a common feature of all regression packages to provide the number of significant coefficients in a given regression. In the next section we show how it is possible to combine time series with mixed frequencies, or different time spans, in a coherent framework using the MFCF and a particular choice of gain function. This section is, in our view, of great practical importance because it fits exactly with the requirements for stress testing in virtually all financial institutions.

5.3.1 Learning with Mixed Frequency Time Series

Quantitative Financial Risk Management requires to analyse the dependency between time series of financial assets, such as the returns of stocks and commodities prices, interest rates, credit spreads and macroeconomic time series such as Gross Domestic Product (GDP), unemployment, inflation. The applications are numerous, for instance credit modelling, stress testing, forecast, correlation modelling and many more. One issue of these types of series is that they are characterised by different frequencies, varying from daily to quarterly[6].

The practitioner that need to build a correlation matrix of mixed time series usually resorts to two strategies: 1. Build a correlation matrix using the data points for which all the observations are available, or 2. assemble a correlation matrix from all the pairwise correlations calculated using the data points available for the pairs of time series. In the first case there will be loss of accuracy and the correlation matrix might not be positive definite[7]. In the second case the correlation matrix will include correlation coefficients with different level of noise and significance level, and this will make dubious any attempt of filtering the network using the correlation matrix; besides, there is no guarantee that the matrix filtered in this way is positive definite.

Moreover, in many cases, and especially when regressing macroeconomic variables, there is a need to apply a lag to one or more of the regressors in order to build or test a model where some macroeconomic indicators are leading or lagging. To build a correlation matrix taking into account lagging as well would be prohibitive because it would mean to add a new variable for every lag applied to a time series, so that every variable would in effect become a block of lagged variables. The researcher or practitioner would then need to perform their analysis bearing in mind that some correlations are in reality autocorrelations between lagged versions of the same variable and any filtering procedure should have to similarly work across the blocks.

In section 3.3.3.6 we have described a gain function that expresses the effective-

[6]We do not enter into the field of high frequency data for lack of personal knowledge

[7]In case the common observations are so few that they are less tahn the number of variables.

ness of the regression of a variable against a set of other variables in a clique. The gain function utilises concepts that are fairly common in regression and it is easy to generalise, but for the sake of simplicity we will limit ourselves to the linear case.

In multiple linear regression the variation of a dependent variable is expressed in terms of the variation of a number of explanatory, or independent, variables. The total variance in the dependent variable explained by the independent variables is the R^2 of the regression. The higher the R^2, the better the predictive value of the linear model. Obviously, there is a risk of overfitting the data if too many independent variables are used. The classical solution to this problem is to keep only the independent variables whose coefficient in the linear combination modelling the dependent variable is significantly different from zero. This classical setting gives us an almost direct method to build a gain function: the R^2 of the regression is the gain, and the independent variables that have significant coefficients are the separator returned by the gain function, so that the new clique is made of the dependent variable and the significant independent variables.

The regression should be performed on transformed variables, for instance differentiated or log-differentiated to make them stationary. The gain function should extract therefore the complete observations in the clique and the new candidate vertex and perform the differentiation or the calculation of returns ahead of the linear regression. It is worth noting that in this way one raw time series could be used as a series of daily returns in one clique, and as a series of monthly returns in a different clique, depending on set of dependent and independent variables present in the clique. Finally the gain function can perform the lagging of the independent variables. Due to the local nature of the MFCF, each regression can have different frequencies and lags. Finally, because the cliques are of limited size compared to the number of data points used, the regression coefficients would be well determined and significant.

Should the researcher be interested purely in the shape of the network, there would be no explicit need to store the lags, frequency and direction[8] of the regressions; but if the researcher wanted to simulate the evolution of the system it would be necessary to take into account frequency, lags, and direction of the regression. Because the MFCF

[8]By "direction" we allude to the fact that the flow of information goes from the independent to the dependent variable.

produces a DAG, the evolution should always be well-behaved.

Chapter 6

Application to the Covariance Selection Problem

Covariance selection is the problem of estimating a covariance matrix with a sparse structure. In multivariate normal populations this means that the inverse of the covariance matrix should be sparse. The benefits of a sparse covariance matrix are the same as those found in any sparse model: less sensitivity to noise, simpler explanation and analysis of the dependence relationships, computational benefits. While the applications are various (e.g. bioinformatics, speech processing, computer vision) the main motive of interest for us is in finance and portfolio management, where a robust estimation of correlation can prevent a portfolio manager from committing costly mistakes in asset allocation (Engle and Colacito, 2012).

In this chapter we briefly introduce the problem of covariance selection in Section 6.1 and, after a brief introduction to the Graphical Lasso (GLASSO) approach (Section 6.1.2), we frame the problem in the MFCF framework and describe in detail the construction of the estimators, including a shrinkage target based on the chordal structure of the clique tree (Section 6.2). In Section 6.3 we describe the testing methodology and in Section 6.4 we compare the performances of the various algorithms.

6.1 Covariance Selection

The study of the problem of *covariance selection* has been initiated in his seminal paper by Dempster (1972). In the paper the author emphasises the principle of parsimony, where the possibility to set to zero the number of parameters in a statistical model is see as beneficial in terms of simplicity and reduction of estimation error due to noise.

The author tackles the problem of estimating of the covariance structure of a set of data generated by a multivariate normal process by suggesting to set to zero the elements of the *inverse* of the covariance matrix (which he calls *concentration matrix*) instead of the elements of the covariance matrix itself. In the paper the theoretical motivation for setting the elements of the concentration matrix, rather than the covariance matrix, to zero comes from the analysis of the multivariate normal distribution as a member of the exponential family, where the elements of the concentration matrix feature as the natural parameters of the distribution. In this setting, therefore, setting the elements of the concentration matrix to zero is the same as reducing the number of parameters of the multivariate normal model. Within the framework of Gaussian graphical models, we have seen that setting elements of the concentration matrix to zero (as shown in Example 1) is equivalent to state the conditional independence of the pair of variables associated with the matrix element. The set I of indices where the concentration coefficients should be zero can be decided *a priori*, considering the nature of the problem, or estimated from the sample. Taking I as a given, and calling N the complementary list of indices where the concentration coefficients are not zero[1], Dempster (1972) defines a solution to the covariance selection problem as the matrix that assumes the same values in the direct matriix as the sample correlation matrix (that is, on N), and such that the the elements of the concentration matrix are zero in I; in other words the elements of the proposed solution in N are the same as the sample, while the values in I are adjusted in a way to make the corresponding elements of the inverse equal to zero. It is then proved that the proposed solution has three desirable properties:

1. The solution to this problem exists and it is unique.
2. Among all normal models that agree with the sample covariance matrix in N and that can assume any value in I, the proposed solution is the one with the maximum entropy.
3. Among all the normal models that have the elements of the concentration matrix equal to zero in I, the proposed solution is the one with the maximum likelihood.

[1]The paper by Dempster and a large part of the literature calls J what we have called N, that is the list of indices where the concentration matrix is not zero. We have chosen to deviate from convention in order not to confuse the reader, as we use the symbol J to indicate the concentration matrix.

If the set N is not chosen *a priori*, two options are to start with a minimal model assuming complete independence (where the inverse of the covariance matrix has only diagonal elements) and add off-diagonal elements as long as they are significant (forward selection, used by the MFCF), or start with a saturated model and force the off-diagonal elements to zero (backward selection, used by the Graphical Lasso).

We observe, however, that if the underlying pattern is non-chordal, most uses in actual applications (such as QR-factorisation, Gaussian elimination, to name a few) would result in significant *fill-in*. If the pattern is instead chordal and operations are carried out respecting the PEO, no *fill-in* is introduced.

6.1.1 Penalised Likelihood Maximisation

The log likelihood of a multivariate normal model is:

$$\ell(\mathbf{X}=\hat{\mathbf{x}}) = p\ln(2\pi) + \ln|J| - \mathrm{Tr}(\Sigma \cdot J) \simeq \ln|J_c| - \mathrm{Tr}(\Sigma \cdot J) \tag{6.1}$$

A way to find a trade-off between high likelihood and limited number of parameters would be to optimise a penalised likelihood that could be (removing the constant term in the likelihood):

$$\ell(\mathbf{X}=\hat{\mathbf{x}})_\rho = \ln|J| - \mathrm{Tr}(\Sigma \cdot J) - \rho\,\mathbf{Card}(J) \tag{6.2}$$

where $\mathbf{Card}(J)$ is the number of non-zero elements in the concentration matrix J. Using $\rho = \frac{2}{p+1}$ yields the Akaike Information Criterion (AIC) (Akaike, 1973), while $\rho = \frac{\ln(p+1)}{p+1}$ would give the Bayesian Information Criterion (BIC) (Schwarz et al., 1978).

6.1.2 Graphical Lasso

The optimisation of the quantity in Equation 6.2 is difficult because the function is not convex and not amenable to standard optimization procedures. The MFCF, however, tries to find a solution by limiting the maximum dimension of the cliques and tries to solve the problem in equation 6.2 directly. A different approach is to apply a convex relation to the problem and optimise instead the quantity:

$$\tilde{\ell}(\mathbf{X}=\hat{\mathbf{x}})_\rho = \ln|J| - \mathrm{Tr}(\Sigma \cdot J) - \lambda \parallel J \parallel_1 \tag{6.3}$$

where $\parallel J \parallel_1$ is the L_1 norm of the vector of parameters of the model, that is the sum of the absolute values of the parameters. It is a well known fact that the use of the L_1 norm drives some coefficients to zero (see for instance Hastie et al. (2015)), while other norms (such as L_2) just shrink the coefficients without necessarily driving them towards zero. This line of research has been initiated by the seminal work (Tikhonov, 1943) on regularisation.

The Graphical Lasso method (Friedman et al., 2008; Hastie et al., 2015) is a numerical procedure to find the matrix $\hat{\Theta}$ that solves optimization problem:

$$\hat{\Theta} \in \underset{\Theta \succ 0}{\arg\max} \{\ln |\Theta| - \mathrm{Tr}(\Sigma\Theta) - \lambda_0 \parallel J \parallel_1\} \tag{6.4}$$

where $\Theta \succ 0$ means that we restrict Θ to the set of positive definite matrices.

Remark 72. The Graphical Lasso finds a solution to the problem 6.4 for a given value λ_0 of the penalisation parameter λ. The choice of the appropriate value for λ is difficult and it is usually set by cross-validation. The problem is a classic convex relaxation such as those often found in other Lasso regressions, see Hastie et al. (2015, Par. 9.3.2) for a description of the algorithm.

6.2 The MFCF approach to to covariance selection

In our proposed approach we build a clique forest using the MFCF algorithm and we use two distinct gain functions. Since the Graphical Lasso optimises a penalised version of Gaussian log-likelihood, we have used two score functions based on multivariate Gaussian log-likelihood, so that the results are effectively comparable:

- Multivariate Gaussian log-likelihood, described in Section 3.3.3.3. In this case the limitation of the number of coefficients is achieved by fixing the size of the cliques to a given constant value. The clique size is a hyper-parameters we have to fit. As with the GLASSO, this parameter can be estimated by cross-validation.

- Gaussian log-likelihood statistically validated, described in Section 3.3.3.4,where we allow cliques of any size up to a maximum value. In this case the number of parameters of the model is bound by two factors: the maximum allowed clique size, and the significance level of the statistical test. As the clique size it is now only an upper bound, it is convenient to choose a relatively high clique size and let the statistical test drive the selection of significant parameters.

It is important to note that the GLASSO optimisation has two effects, that cannot be put in isolation: on the one hand it learns the structure of the concentration matrix by putting some model parameters to zero, on the other hand it also shrinks towards zero the remaining parameters, because the penalty is on the L_1 norm. The MFCF, instead, allows to separate the structure learning phase, where some parameters are set to zero, to the parameters tuning phase, where the remaining parameters are tuned. Therefore, with the MFCF, we have split the calibration of the model in a structure learning phase and a parameter tuning phase. The tuning of the parameters has been achieved by applying a shrinking procedure to the maximum likelihood estimate obtained from the training data, using two different targets as described in Section 6.2.2 below.

6.2.1 Construction of the precision matrix in the multivariate Gaussian case

In the Gaussian case, once the clique forest structure is known the maximum likelihood estimate of the precision matrix is given in explicit form (see Lauritzen (1996, Prop. 5.9) or Barfuss et al. (2016)):

$$J = \sum_{c \in \mathscr{C}} \left[(\Sigma_c)^{-1} \right]^V - \sum_{s \in \mathscr{S}} \left[(\Sigma_s)^{-1} \right]^V \tag{6.5}$$

where the notation $[M_c]^V$ in Equation 6.5 means a matrix of dimension $p = |V|$ where all the elements are zero, excepting for the ones with the indices in the clique c; that is $[M_c]^V_{ij} = M_{ij}$ if $i \in c$ and $j \in c$, $[M_c]^V_{ij} = 0$ otherwise.

6.2.2 Shrinkage procedures

In all the experiments we have applied some shrinkage[2] to the maximum likelihood estimate obtained from the training data set. The shrinkage parameter has been calibrated by performing a grid search and optimising the likelihood over the validation data set. We have used two shrinkage targets: the commonly used identity matrix (Ledoit and Wolf, 2003, Sec. 3.3), and a new shrinkage target we call "the clique tree target". The

[2]We refer the reader to Ledoit and Wolf (2003) for a useful introduction to shrinkage in the context of financial portolfio management.

clique tree target is a generalisation of the constant correlation target (Ledoit and Wolf, 2004). In the constant correlation target the target matrix is a matrix where the off-diagonal elements are equal to the mean of the correlation coefficients, and the shrinkage estimator is a convex combination of the original, un-shrunk, matrix and of the constant correlation target. The "clique tree target" is created gluing together smaller correlation target matrices that represent the cliques of a clique tree. Every correlation coefficient is naturally associated with an edge in the network and its target value is the average of the correlation coefficients of the cliques the edge belongs to.

The matrix is built in three steps, starting from the calculation of the average correlation between elements in every clique c in the clique tree.

6.2.2.1 Step 1

$$\hat{\rho}_c = \frac{\sum_{i\in c, j\in c, j>i} (\Sigma_c)_{ij}}{\sum_{i\in c, j\in c, j>i} 1} \tag{6.6}$$

6.2.2.2 Step 2

Next we build a clique level correlation matrix for every clique c using the following rules:

- If $i \in c, j \in c, i = j$ then $(\hat{\hat{\Sigma}}_c)_{ij} = 1$
- If $i \in c, j \in c, i \neq j$ then $(\hat{\hat{\Sigma}}_c)_{ij} = \frac{\sum_{c'\in\mathscr{C}, i\in c', j\in c'} \hat{\rho}_{c'}}{\sum_{c'\in\mathscr{C}, i\in c', j\in c'} 1}$, that is we calculate the target correlation as the mean of the average correlations of all the cliques the element belongs to.

6.2.2.3 Step 3

And finally we build the estimate for the inverse applying the "Lauritzen formula" (6.5)

$$\hat{J} = \sum_{c\in\mathscr{C}} \left[\left((1-\theta)\hat{\Sigma}_c + \theta\hat{\hat{\Sigma}}_c \right)^{-1} \right]^V - \sum_{s\in\mathscr{S}} \left[\left((1-\theta)\hat{\Sigma}_s + \theta\hat{\hat{\Sigma}}_s \right)^{-1} \right]^V \tag{6.7}$$

Remark 73. The constant correlation estimates $\hat{\hat{\Sigma}}$ are positive definite because every $\hat{\hat{\Sigma}}_c$ $(\hat{\hat{\Sigma}}_s)$ is the normalized sum of positive definite matrices.

6.3 Testing Methodology

We now report the result of the computational experiments where we have compared the performances of the MFCF, the GLASSO, and a shrinkage estimator on a range of data sets. The general approach is to use a training dataset to estimate the structure of the models, to tune the model hyperparameters [3] on a validation data set and finally to assess the performance of the models on a test dataset.

The process used to generate the synthetic data used in the tests is described in Section 6.3.1. The treatment of real data is described in Section 6.3.2. The full specification of the algorithms used in the test is reported in Section 6.3.3. Finally, we describe the performance measures in Section 6.3.4.

6.3.1 Generation of Synthetic Data

We test the performance of the algorithm on three types of synthetic data and on a real dataset of stocks returns. The synthetic data are multivariate Gaussian generated using respectively: (1) a sparse chordal inverse matrix with known sparsity pattern; (2) a factor model; (3) a random positive definite matrix generated from random eigenvalues and a random rotation. The real example is generated from a long-return series of stock prices. All the datasets used in the experiments have been produced for 100 variables ($p = 100$) and varying time series lengths ($n \in \{25, 50, 75, 100, 200, 300, 400, 500, 750, 1000, 1500\}$). The details about the data generation process are described in the sub-Sections 6.3.1.1, 6.3.1.2, 6.3.1.3 and 6.3.2.

For every type of data we generate the following datasets:

1. The *train data set* which is used to learn the model parameters, such as the MFCF network and the elements of the precision matrix. For every type of data we generate 5 distinct training data sets to test reproducibility.

2. The *validation data set* is used to select the model hyper-parameters: these are the L_1 penalty for the graphical lasso, the shrinkage parameter for the shrinkage method, and the maximum clique size and shrinkage parameter for the MFCF. For all methods we perform a grid search over the hyper-parameters and select the model that achieves the best likelihood on the validation dataset. In analogy with the train data we generate 5 distinct validation data sets.

3. The *test data set* is used to assess the performance of the models. We use 10 distinct test datasets for every training/validation data set and therefore for every

[3]that is, the λ parameter of the GLASSO and the shrinkage parameters of the other models

data type we have 50 test datasets.

6.3.1.1 Synthetic data: sparse decomposable precision matrix

This data has been produced with a multivariate model from a sparse inverse covariance matrix (the benchmark precision matrix) where the non-zero structure pattern is a clique forest. The clique forest was generated by applying repeatedly the clique expansion operator with a random choice of the vertices, cliques and separators that were available at any steps (this is an application of the gain function described in Section 3.3.3.5).

For every clique $c \in \mathscr{C}$ we have defined a factor F_c distributed as $\mathscr{N}(1,1)$ and for every $X_i, i \in c$ we have defined $X_i = F_c + \varepsilon_{i,c}$, where $\varepsilon_{i,c} \sim \mathscr{N}(0,0.1)$ is a small noise factor to avoid perfect correlation between the variables in the clique c. Finally for a variable X_i that belongs to more than one clique we define $X_i = \sum_{i \in c} F_c + \varepsilon_{i,c}$. The final inverse correlation matrix is assembled using Equation 6.5.

The exact inverse of the precision matrix has been used to generate the training, validation and test data sets, using the function `mvrnorm` of the package MASS (Venables and Ripley, 2002) developed for the R language (R Core Team, 2016).

6.3.1.2 Synthetic data: Full Positive Definite Matrix from package "clusterGeneration"

This data has been generated using the R package "clusterGeneration" (Qiu and Joe. (2015)). The methodology is to produce a vector of random eigenvalues ($p = 100$ values in the range $[0.01, 100]$ in this experiment) and to rotate the diagonal matrix of eigenvalues with a random orthogonal matrix to produce a dense positive definite matrix that is used as the benchmark reference covariance. As in the previous example the generation of the data sets has been carried out using the package 'MASS' as described in 6.3.1.1.

6.3.1.3 Random Factor Model with noise

This data set has been generated by building a factor model with 5 factors. For a review of factor models, with particular regards to large factor models see Bai et al. (2008); here we follow their conventions and model the variables $\mathbf{X}$ as $\mathbf{X} = \Lambda\mathbf{F} + \varepsilon$ where: $\mathbf{F}$ is an $f \times n$ matrix, with $f < p$ the number of factors, Λ is the $p \times f$ matrix of *factor loadings* and ε is the $p \times n$ idiosyncratic term.

Accordingly the correlation matrix breaks down in two parts: $\Sigma = \Lambda\Lambda' + \Omega$, where $\Lambda\Lambda'$ is the systematic component and Ω is the idiosyncratic component .

The training, validation and test matrices have been generated using $f = 5$. The factor loadings have been randomly generated from independent normal distribution and the factors have been generated as independent normal variates. As the factor loadings are in general different from zero, this model is dense.

6.3.2 Usage and Treatment of Real Data

This data set contains a set of stock returns for 342 companies over 4025 trading days, as described in Barfuss et al. (2016). For every training, validation and test execution we have sampled randomly without replacement $p = 100$ time series. The training, validation and testing datasets have been sampled, with replacement, from the total time series of 4025 trading days.

The estimate of the 'real' reference covariance matrix has been produced using the full dataset, and this has been used as a benchmark for the estimates produced by the models.

6.3.3 Algorithms used in the Testing

We have generated sparse inverse covariance estimates with different implementations of the MFCF algorithm and compared their performances with the GLASSO and shrinkage estimators, the real benchmark and the null hypothesis. The description of the methodology to generate the estimates for all algorithms is below.

1. GLASSO_XVAL: the Graphical Lasso (Friedman et al., 2008); we use the implementation provided by the R package `huge` (Zhao et al., 2015). The penalty parameter is estimated through cross-validation using an adaptive grid search in the interval $[0.01, 1]$. The precision matrix is estimated, for a given penalty parameter, on the training data set; the penalty parameter selected is the one that produces the estimate with the highest log-likelihood on the validation data set[4]. Performances are assessed on the test data sets.

2. SHRINKAGE: a shrinkage estimator with target the identity matrix. We produce shrunk correlation matrices estimators from the training dataset using a

[4]The minimum penalty of 0.01 has been used because for smaller values we have encountered convergence problems with some of the test cases.

grid search for the shrinkage parameter associated with the highest likelihood on the validation data set. Performances are assessed on the test data sets. Recall that this method does not produce sparse precision matrices, and it is therefore used purely as a benchmark to assess the performance of the remaining models in terms of likelihood.

3. MFCF_FIX: the MFCF algorithm with fixed clique size, the shrinkage target is the clique tree target described in 6.2.2, and the gain function described in 3.3.3.3. We proceed in two steps: initially the correlation matrix built from the training set is shrunk by a small parameter $\varepsilon = 0.05$ using the identity matrix as a target[5]. Then we produce a set of models with clique sizes between 2 and 20.[6] The precision matrix estimates are produced using the training datasets and the shrinkage procedure described in Section 6.2.2. The shrinkage parameter is the one that achieves the best likelihood on the validation data set, estimated with a grid search as we do for the graphical lasso and the shrinkage estimators.

4. MFCF_FIX_ID: same as MFCF_FIX, excepting for the shrinkage target where we use the identity matrix.

5. MFCF_VAR: same as MFCF_FIX (in particular using the clique tree target as a shrinkage target) but with variable clique sizes between 2 and 20 and the gain function described in 3.3.3.4. The p-value used for the likelihood ratio test (used in 3.3.3.4) was 0.05.

6. MFCF_VAR_ID: same as MFCF_VAR excepting the shrinkage target, where we use the identity matrix.

7. REAL_OR_ML: the benchmark ‘real’ precision matrix. For synthetic data, when the structure of the correlation matrix is known exactly, we use the exact inverse; in the case of real data, for which we do not know the real correlation matrix, we use the inverse of the sample correlation matrix computed on the entire time series.

[5]This step is performed to stabilise numerically the algorithm; otherwise the matrices for some small cliques might be near singular numerically and lead to problems in the calculation of the gains. The parameter 0.05 has not been tuned but just used as a reasonably small number.

[6]Larger values would lead to essentially dense models.

8. NULL hypothesis: the identity matrix as the inverse precision matrix.

6.3.4 Performances indicators

For every test set we collect the following performance indicators: [7]

1. Log likelihood, which is $-\frac{1}{2}p\left(-\log|J| + \text{Tr}\left(\hat{\Sigma} \cdot J\right)\right)$ (consistently with the definition of the objective function used in the R package glasso we omit the constant). Please note that J is the precision matrix estimated using the training dataset, while $\hat{\Sigma}$ is the sample correlation estimated on the test dataset.

2. $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$, which is the fraction of entries in the precision matrix J that are correctly predicted as zero or non-zero.

3. $Sensitivity = \frac{TP}{TP+FN}$, which is the fraction of non-zero entries in the precision matrix J that are correctly predicted by the models.

4. $Specificicty = \frac{TN}{TN+FP}$, which is the fraction of zero entries in the precision matrix J that are correctly predicted by the model.

5. The *correlation* of the estimated precision matrix with the true precision matrix (which is known in the case of synthetic data) or with the maximum likelihood estimate of the precision matrix computed on the longest possible data set (in the case of real data). The correlation is calculated as if the two matrices were vectors in $\mathbb{R}^{p^2}$.

6. *Eigenvalue distance* is the $\mathbb{R}^2$ norm of the vector of differences of the eigenvalues of the real or maximum-likelihood estimate precision matrix and the estimated precision matrix $\left(\sum_{i=1}^{p}(\hat{\lambda}_i - \lambda_i)^2\right)^{\frac{1}{2}}$.

7. *Eigenvalue inverse distance* is the $\mathbb{R}^2$ norm of the the vector of differences of the reciprocal of the eigenvalues of the real or maximum-likelihood estimate precision matrix and the estimated precision matrix $\left(\sum_{i=1}^{p}(\hat{\lambda}_i^{-1} - \lambda_i^{-1})^2\right)^{\frac{1}{2}}$.

[7] We define: TP (True Positives) as the count of elements in the precision matrix that are correctly predicted as different from zero, TN (True Negatives) as the count of elements in the precision matrix that are correctly predicted as zero, and FP (False Positives) and FN (False Negatives) in analogous fashion. This is possible only when the 'true' precision matrix is known (synthetic data) and these measures are meaningful only when it is sparse.

6.4 Results

6.4.1 Synthetic data: sparse decomposable precision matrix

Figure 6.1 provides a box plot representing the mean the confidence interval and the extreme values of the log-likelihood achieved by the algorithms over the test data sets, broken down by the length of the series[8]. We observe that, in all cases, the MFCF algorithms outperform both the graphical lasso and the shrinkage estimator. The graphical lasso improves performances as the length increases but does not exceeds MFCFs. The dispersion around the mean is similar for all methods and it has been computed by repeating the experiments on 50 independent datasets (10 testing sets for each 5 training and validating sets, as discussed in 6.3.1).

Table 6.1 reports the average value of the graphical lasso penalty parameter and of the shrinkage parameter as selected by the grid search. As expected, the parameters become smaller as the series length grows, with MFCFs requiring less shrinkage/penalisation than the other methodologies[9], especially with short time series. We believe that this is a desirable feature of the MFCF algorithm: the topological constraint allows to estimate with good accuracy the cliques with high likelihood, and excludes edges with low likelihood with the end effect of requiring less shrinking.

Table 6.2 reports the number of non zero elements in the precision matrix for every length of the time series. One can observe that the MFCF algorithms are much more parsimonious than the graphical lasso (the shrinkage method produces always a full precision matrix).

Figure 6.2 shows a summary of the performance measures. We observe that that the MFCF family is overall more accurate than the graphical lasso especially for what concerns accuracy and specificity. While the graphical lasso is more sensitive picking up more true positives. However, it is also less selective and produces denser precision matrices with a much higher number of false negatives. We observe that the perfor-

[8]The boxplots in this paper have been produced with the R (R Core Team, 2016) package GGPLOT2 (Wickham, 2009). According to the package documentation the first lower and upper hinges correspond to the first and third quantile, the upper whisker covers the values form the third quartile hinge to 1.5 times the inter-quartile range away from the hinge, and similarly the lower whisker covers the values between the first quartile hinge and 1.5 times the interquartile range below the hinge. The remaining points are considered outliers and plotted individually.

[9]The comparison of penalty and shrinkage parameter is purely indicative, as the two parameters are not directly comparable.

Series length	GLASSO XVAL	MFCF FIX	MFCF FIX ID	MFCF VAR	MFCF VAR ID	SHRINKAGE
25	0.150	0.005	0.005	0.005	0.005	0.726
50	0.120	0.002	0.002	0.002	0.001	0.522
75	0.102	0.002	0.001	0.002	0.001	0.374
100	0.056	0.001	0.001	0.001	0.001	0.257
200	0.014	0.001	0.001	0.001	0.001	0.074
300	0.011	0.001	0.000	0.001	0.000	0.022
400	0.010	0.000	0.000	0.000	0.000	0.020
500	0.010	0.000	0.000	0.000	0.000	0.004
750	0.010	0.000	0.000	0.000	0.000	0.000
1000	0.010	0.000	0.000	0.000	0.000	0.000
1500	0.010	0.000	0.000	0.000	0.000	0.000

Table 6.1: Mean penalty (GLASSO_XVAL) or shrinkage parameters by length of time series. The statistics is based on 5 different calibrations (one per each training / validation set) of the shrinkage parameters per each length of the time series.

Series length	GLASSO XVAL	MFCF FIX	MFCF FIX ID	MFCF VAR	MFCF VAR ID	SHRINKAGE
25	1194	99	99	98	98	4950
50	1118	99	99	135	135	4950
75	1161	138	138	136	136	4950
100	1730	158	158	191	191	4950
200	2586	216	216	195	176	4950
300	2632	216	216	231	231	4950
400	2549	216	216	231	231	4950
500	2451	216	236	213	231	4950
750	2254	255	255	246	246	4950
1000	2108	255	255	244	244	4950
1500	1867	294	294	281	281	4950

Table 6.2: Mean number of non-zero coefficient in the precision matrix by length of time series. The statistics is based on 5 different calibrations (one per each training / validation set) per each length of the time series.

mance of the graphical lasso improves in all measures for time series of length greater than 200, when the penalty parameter is essentially fixed at 0.01. The MFCF exhibit better log-likelihood, as already observed, and also larger correlations with the true precision matrix.

Figure 6.3 shows the distance between the spectra of the precision matrix produced by the models and the true precision matrix. The measure is normalised so that the identity matrix has distance one. We observe that the MFCF algorithms always perform better and the performance improves for all methods as the time series length increases, with the exception of the graphical lasso in the region where the penalty parameter is floored at 0.01.

Figure 6.4 shows the distance between the inverse spectra of the precision matrix

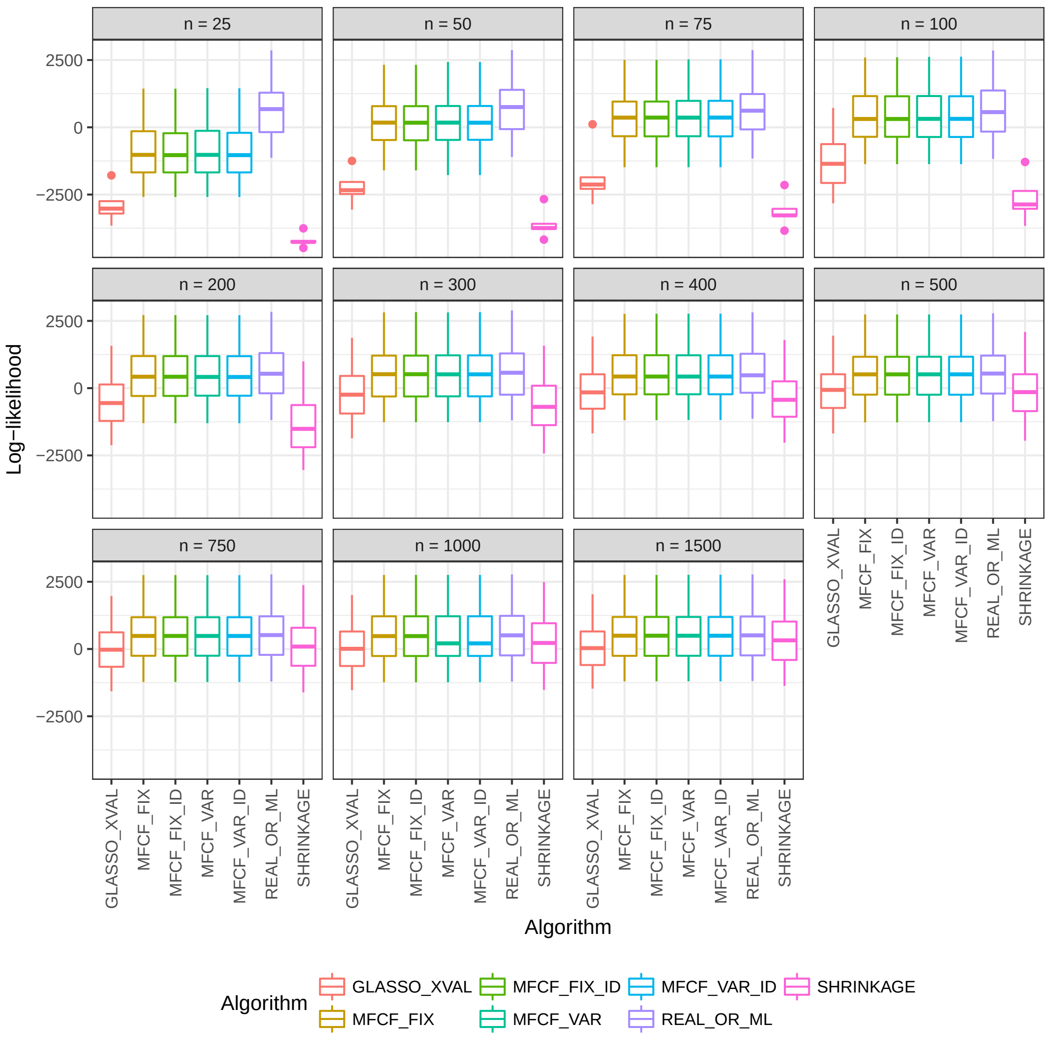


Figure 6.1: Box plot for the log-likelihood of the algorithms on synthetic data (sparse decomposable precision matrix) for different lengths of the series. The statistics is based on a total of 50 test sets (10 test sets for each of 5 different training / validation sets).

(λ_i^{-1}) produced by the models and the ones for the true precision matrix. We observe that the MFCF algorithms perform slightly better than the graphical lasso and shrinkage but performance is very similar. Interestingly, in this case, the distance decreases with the time series length for all algorithms and there is no apparent effect due to the flooring of the graphical lasso penalty parameter.

Figure 6.5 shows the number of cliques of different size produced by the MFCF_VAR algorithm as a function of the maximum allowed clique size and of the time series length. We note that as the time series length increases the test becomes less stringent with a higher number of large cliques in the model; conversely, when the time series is shorter $(n < p)$, the models produced are more parsimonious. The number

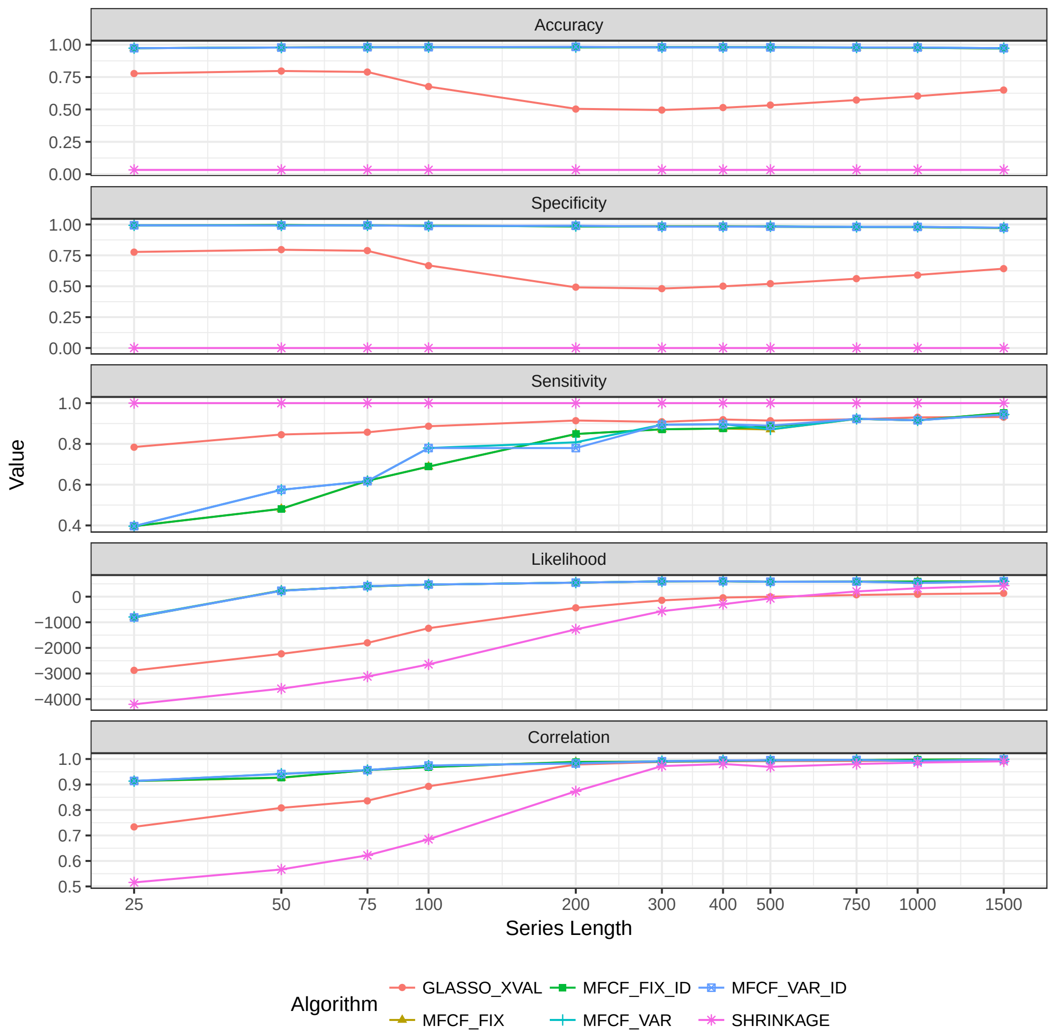


Figure 6.2: Performance measures of the algorithms on synthetic data (sparse decomposable precision matrix) for different lengths of the series. The statistics is based on a total of 50 test sets (10 test sets for each of 5 different training / validation sets).

of cliques of size smaller than the maximum is linked to the degree of sparsity of the model. We will see in Section 6.4.3 that in the case of systems that are inherently dense the vast majority of the cliques will have the maximum allowed clique size.

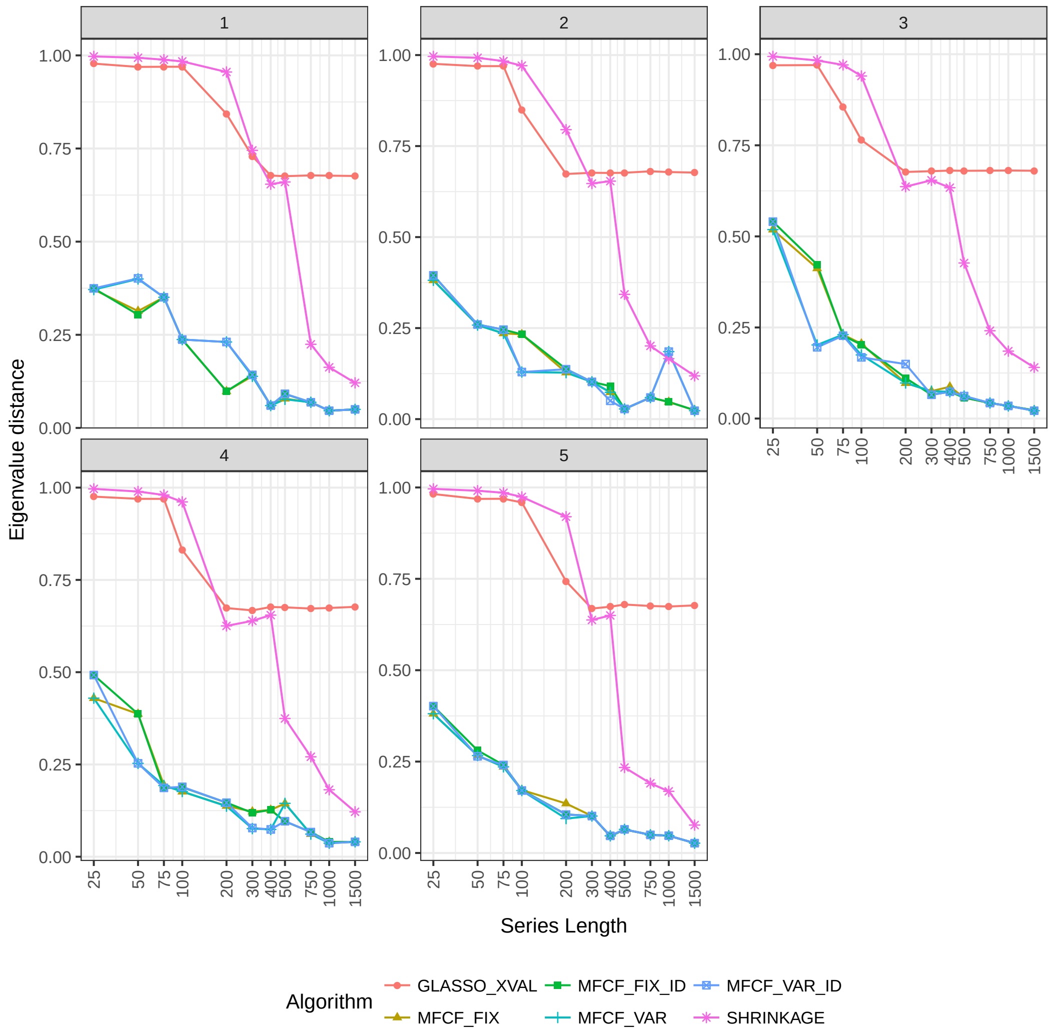


Figure 6.3: Eigenvalue distance for synthetic data (sparse decomposable precision matrix). The five panels show the values at different time series lengths for 5 training / validation datasets.

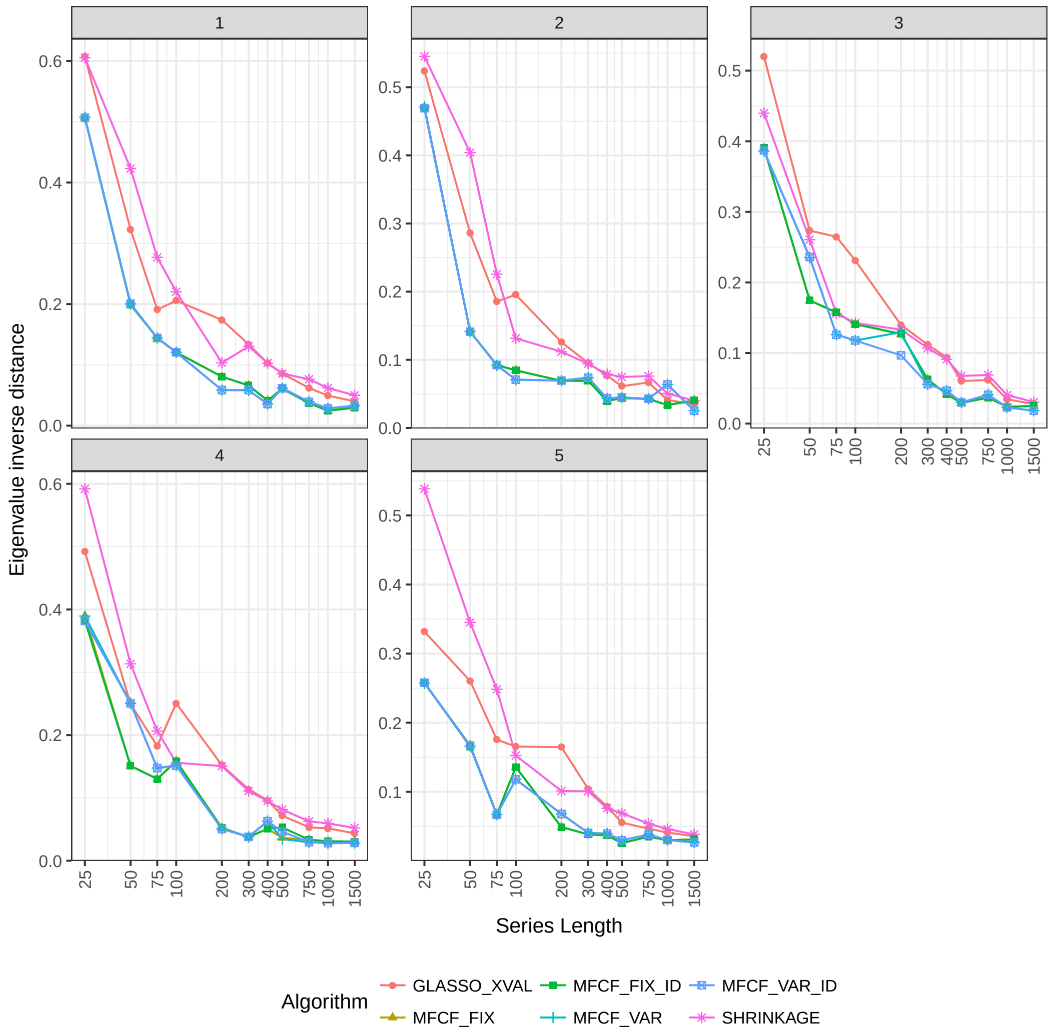


Figure 6.4: Inverse eigenvalue distance for synthetic data (sparse decomposable precision matrix). The five panels show the values at different time series lengths for 5 training / validation datasets.

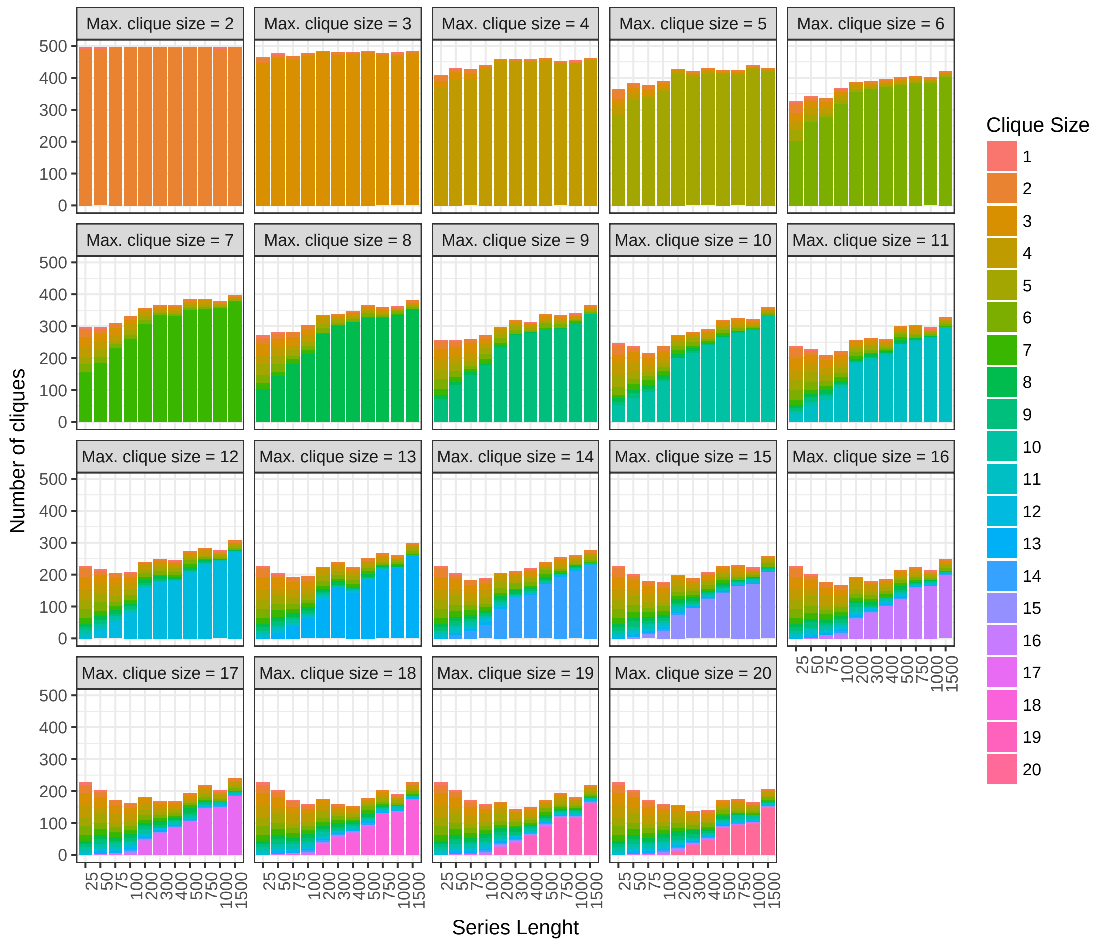


Figure 6.5: Composition of cliques for synthetic data (sparse decomposable precision matrix). The statistics is based on a total of 5 different training / validation sets.

6.4.2 Synthetic data: Full Positive Definite Matrix from package “clusterGeneration”.

In this sub-section and in the next two we repeat on different datasets all the analyses described in the previous subsection 6.4.1. Figure 6.6 displays the log-likelihood of the models. We observe that MFCF algorithms perform overall better than either GLASSO or SHRINKAGE, but is worth noting the overall low level of the log-likelihood for all models. In particular the GLASSO performs worse than the null hypothesis (which has log-likelihood of 5000) for short time series. From Table 6.3 we observe that the penalty or shrinkage parameters decrease but they retain higher overall values than in the other examples.

Series length	GLASSO XVAL	MFCF FIX	MFCF FIX ID	MFCF VAR	MFCF VAR ID	SHRINKAGE
25	0.33	0.97	0.95	0.99	0.99	0.99
50	0.24	0.97	0.89	0.97	0.91	0.99
75	0.22	0.94	0.88	0.95	0.87	0.95
100	0.17	0.88	0.83	0.90	0.83	0.93
200	0.12	0.85	0.75	0.86	0.77	0.89
300	0.12	0.73	0.64	0.72	0.62	0.86
400	0.12	0.63	0.55	0.63	0.54	0.84
500	0.12	0.56	0.48	0.54	0.46	0.80
750	0.10	0.40	0.36	0.39	0.38	0.76
1000	0.08	0.34	0.31	0.32	0.28	0.71
1500	0.05	0.20	0.18	0.20	0.18	0.62

Table 6.3: Mean penalty/shrinkage parameter by length of time series. The statistics is based on a total of 5 different training / validation sets per length of the time series.

Table 6.4 shows the number of non zero elements in the precision matrix for every length of the time series. We note that the statistically validated methods MFCF_VAR and MFCF_VAR_ID produce consistently sparser models, without significant deterioration on the performance in terms of log-likelihood or correlation.

The measures of performance are reported in Figures 6.7-6.10. IWe note that the MFCF_FIX and MFCF_FIX_ID are more accurate for short time series as they pick up many more matrix elements than the validated methods, but this does not translate in improvements for the other measures of performance. The MFCF methods seem to perform better than GLASSO and SHRINKAGE also when it comes to distance of the eigenvalues, especially with short time series. Interestingly, the composition of the clique structure produced by the MFCF_VAR shown in Figure 6.10suggests that even for medium and long time series the algorithm produces a mostly small or large cliques

Series length	GLASSO XVAL	MFCF FIX	MFCF FIX ID	MFCF VAR	MFCF VAR ID	SHRINKAGE
25	484	1062	1164	234	36	4950
50	459	465	679	250	276	4950
75	343	592	1129	255	353	4950
100	472	555	757	247	308	4950
200	472	351	687	238	301	4950
300	260	352	466	272	382	4950
400	155	331	369	272	343	4950
500	108	294	351	297	364	4950
750	165	370	408	330	428	4950
1000	171	313	427	292	336	4950
1500	922	313	313	282	349	4950

Table 6.4: Mean number of non-zero coefficient in the precision matrix by length of time series. The statistics is based on a total of 5 different training / validation sets per length of the time series.

with only a small fraction of cliques with intermediate sizes.

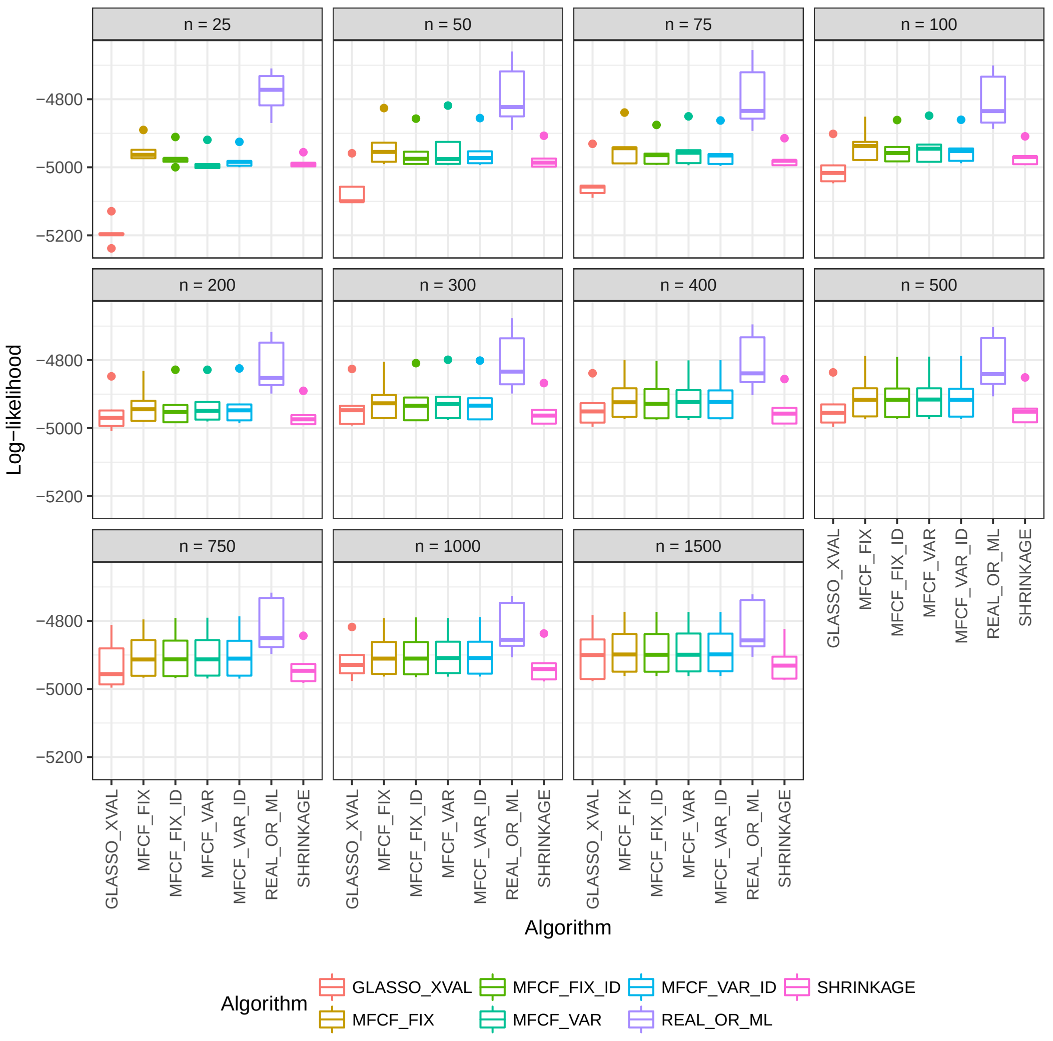


Figure 6.6: Box plot for the likelihood of the algorithms on synthetic data (random positive definite matrix generated by ClusterGen) for different lengths of the series. The statistics is based on a total of 50 test sets (10 test sets for each of 5 different training / validation sets) per length of the time series.

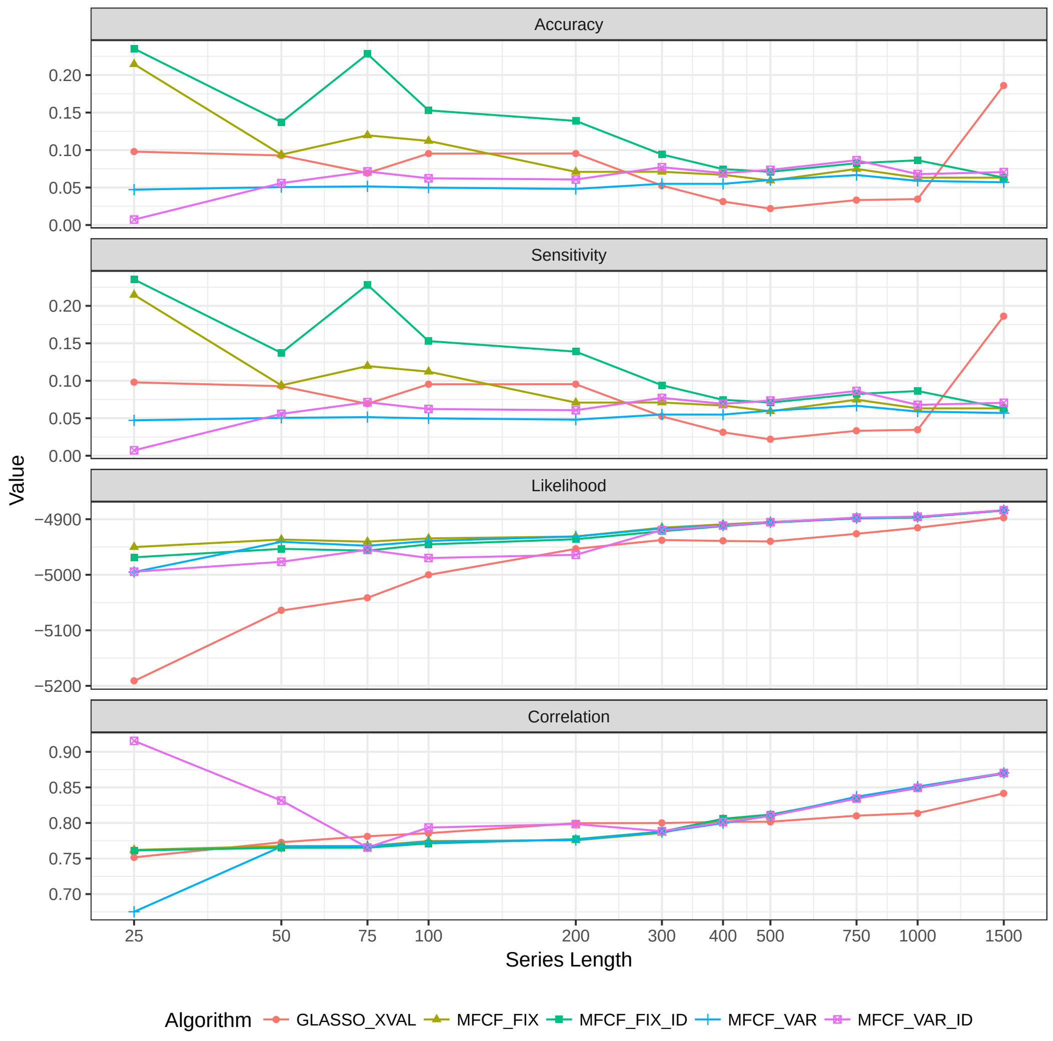


Figure 6.7: Performance of the algorithms on synthetic data (random positive definite matrix generated by ClusterGen) for different lengths of the series. The statistics is based on a total of 50 test sets (10 test sets for each of 5 different training / validation sets).

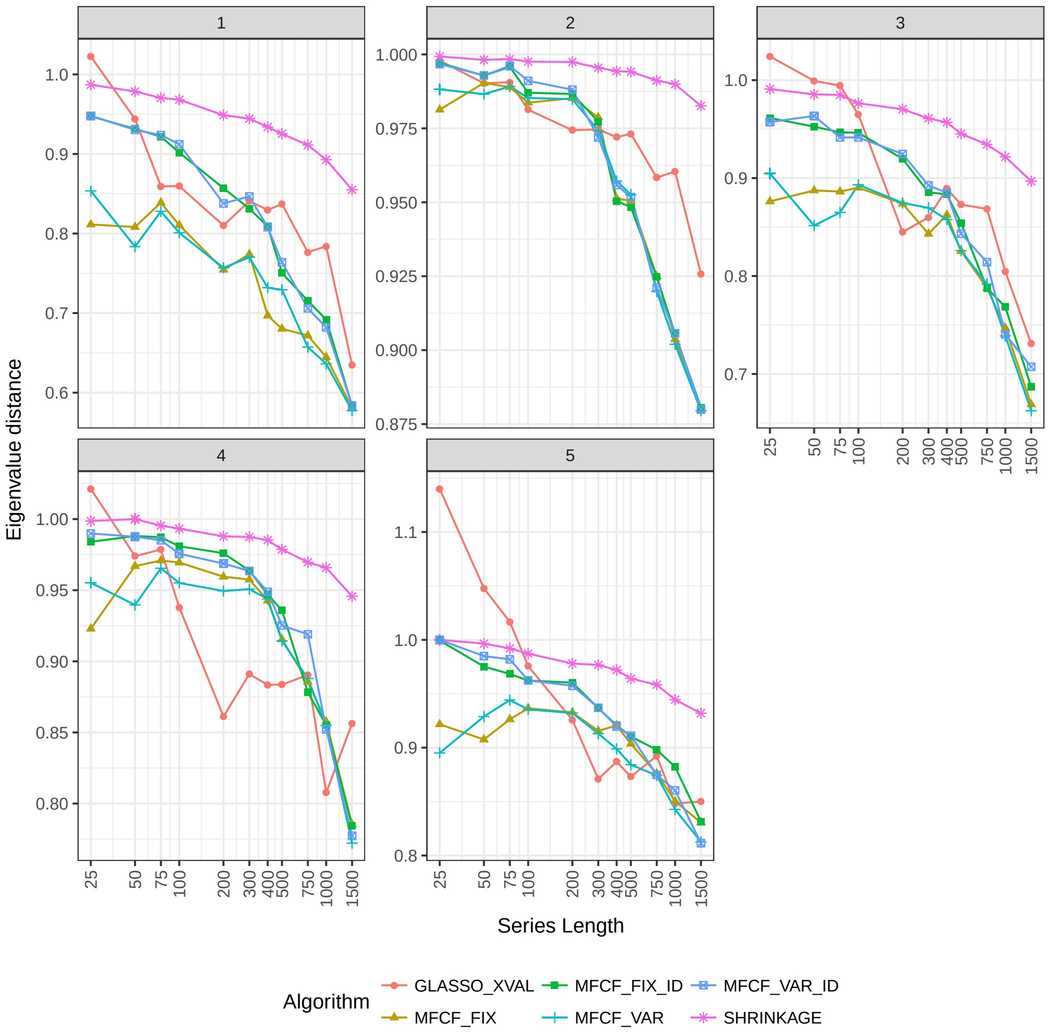


Figure 6.8: Eigenvalue distance for synthetic data (random positive definite matrix generated by ClusterGen). The five panels show the values at different time series lengths for 5 training / validation datasets.

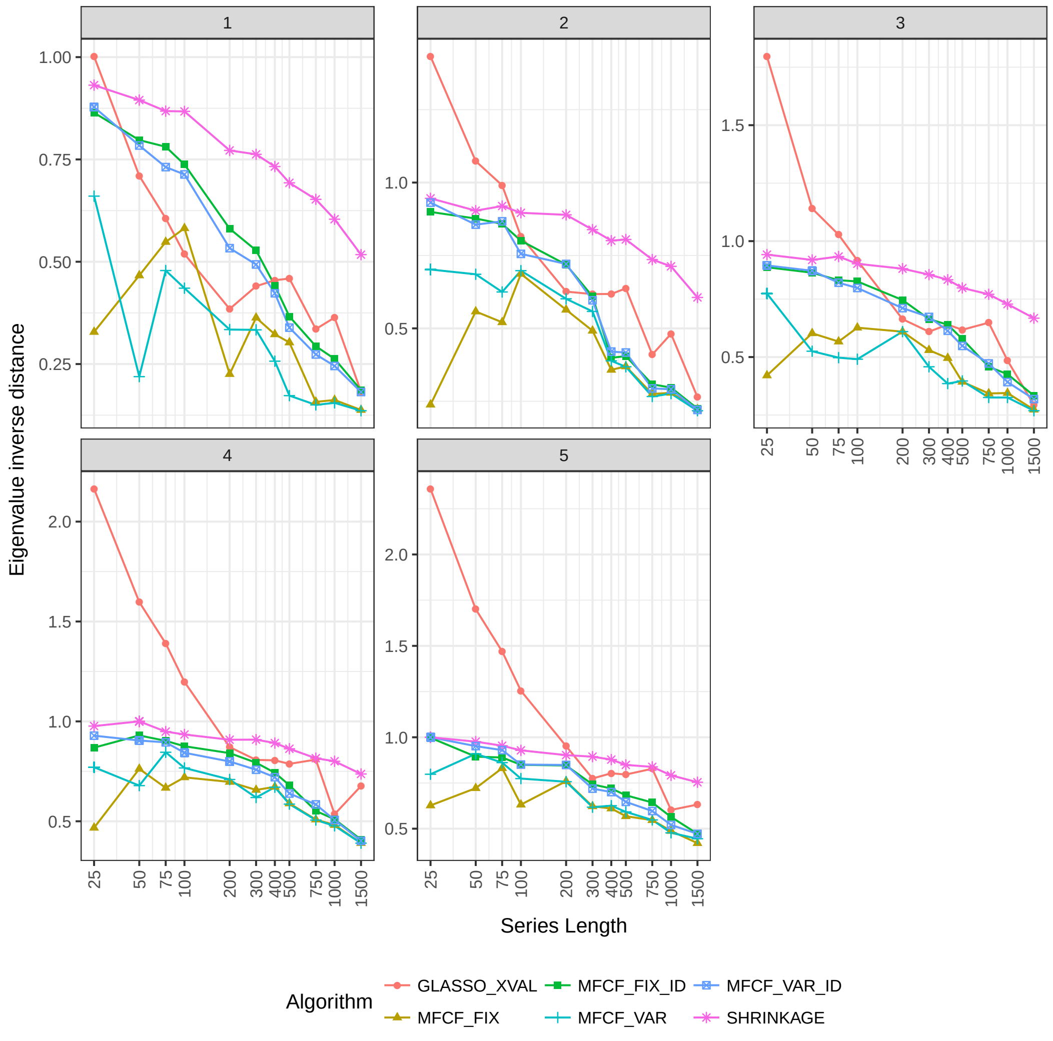


Figure 6.9: Inverse eigenvalue distance for synthetic data (random positive definite matrix generated by ClusterGen). The five panels show the values at different time series lengths for 5 training / validation datasets.

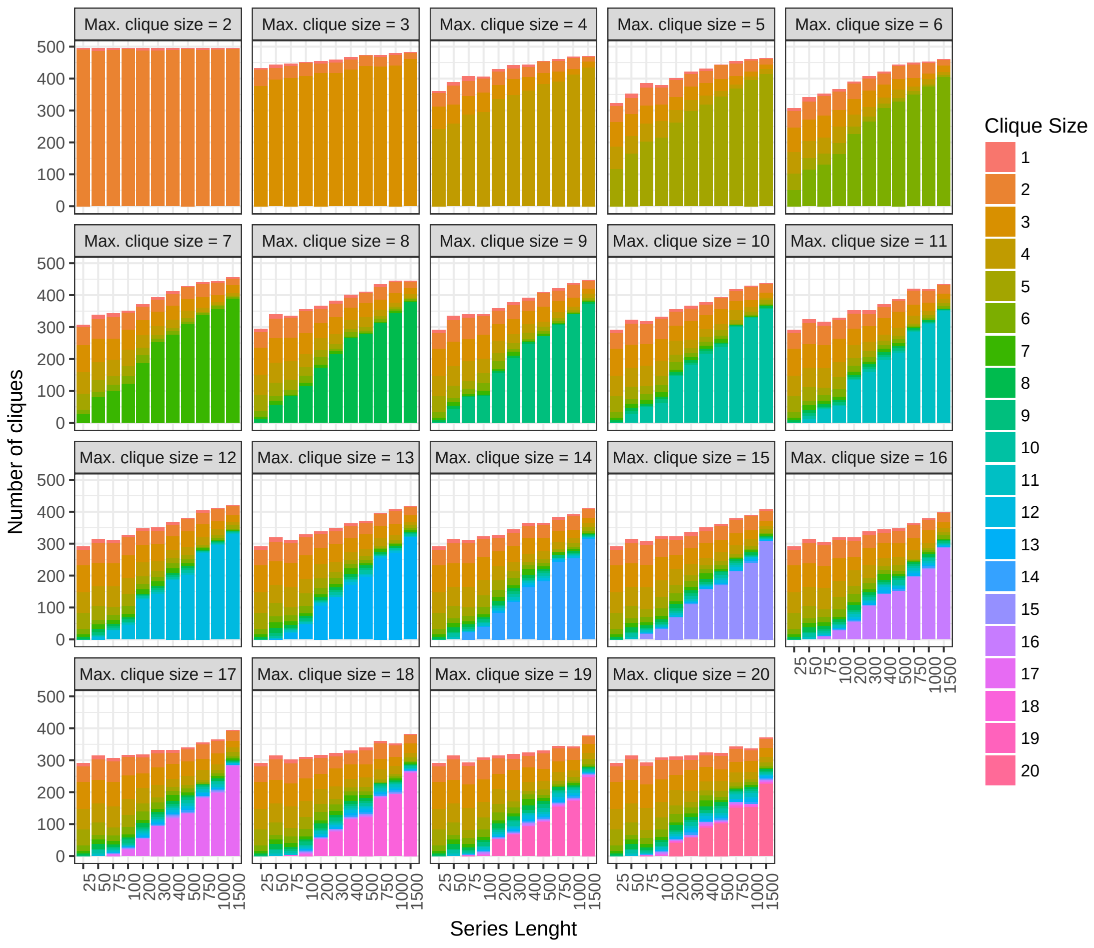


Figure 6.10: Composition of cliques for synthetic data (random positive definite matrix generated by ClusterGen). The statistics is based on a total of 5 different training / validation sets per length of the time series.

6.4.3 Random Factor Model with noise

Performance measures are reported in Figures 6.11-6.15. As discussed in Section 6.3.1, the loadings to the 5 factors are different from zero for every time series, and therefore the model is not suitable for local algorithms such as the MFCF; this would probably explain why in this instance the GLASSO performs better in terms of almost all measures. Table 6.5 shows that the behaviour of the shrinkage or penalty parameters are decreasing with series length as expected. Table 6.6 highlights how all the models tend to use as many parameters as possible, consistently with the constraints imposed on penalty and clique size. Figure 6.15 supports the idea that the underlying model is non local, since the validated methods tend to use exclusively the largest cliques allowed by the constraints. This suggests that the analysis of the clique sizes might provide insight into the sparsity of the data set, when the data generation process in not known.

Series length	GLASSO XVAL	MFCF FIX	MFCF FIX ID	MFCF VAR	MFCF VAR ID	SHRINKAGE
25	0.12	0.27	0.26	0.25	0.25	0.30
50	0.12	0.19	0.18	0.18	0.18	0.23
75	0.09	0.15	0.14	0.15	0.15	0.20
100	0.06	0.13	0.13	0.12	0.12	0.18
200	0.02	0.08	0.08	0.08	0.08	0.14
300	0.01	0.06	0.06	0.06	0.06	0.11
400	0.01	0.05	0.05	0.05	0.05	0.09
500	0.01	0.04	0.04	0.04	0.04	0.08
750	0.01	0.03	0.03	0.03	0.03	0.06
1000	0.01	0.02	0.02	0.02	0.02	0.04
1500	0.01	0.01	0.01	0.01	0.01	0.04

Table 6.5: Mean penalty/shrinkage parameter by length of time series. The statistics is based on a total of 5 different training / validation sets per length of the time series.

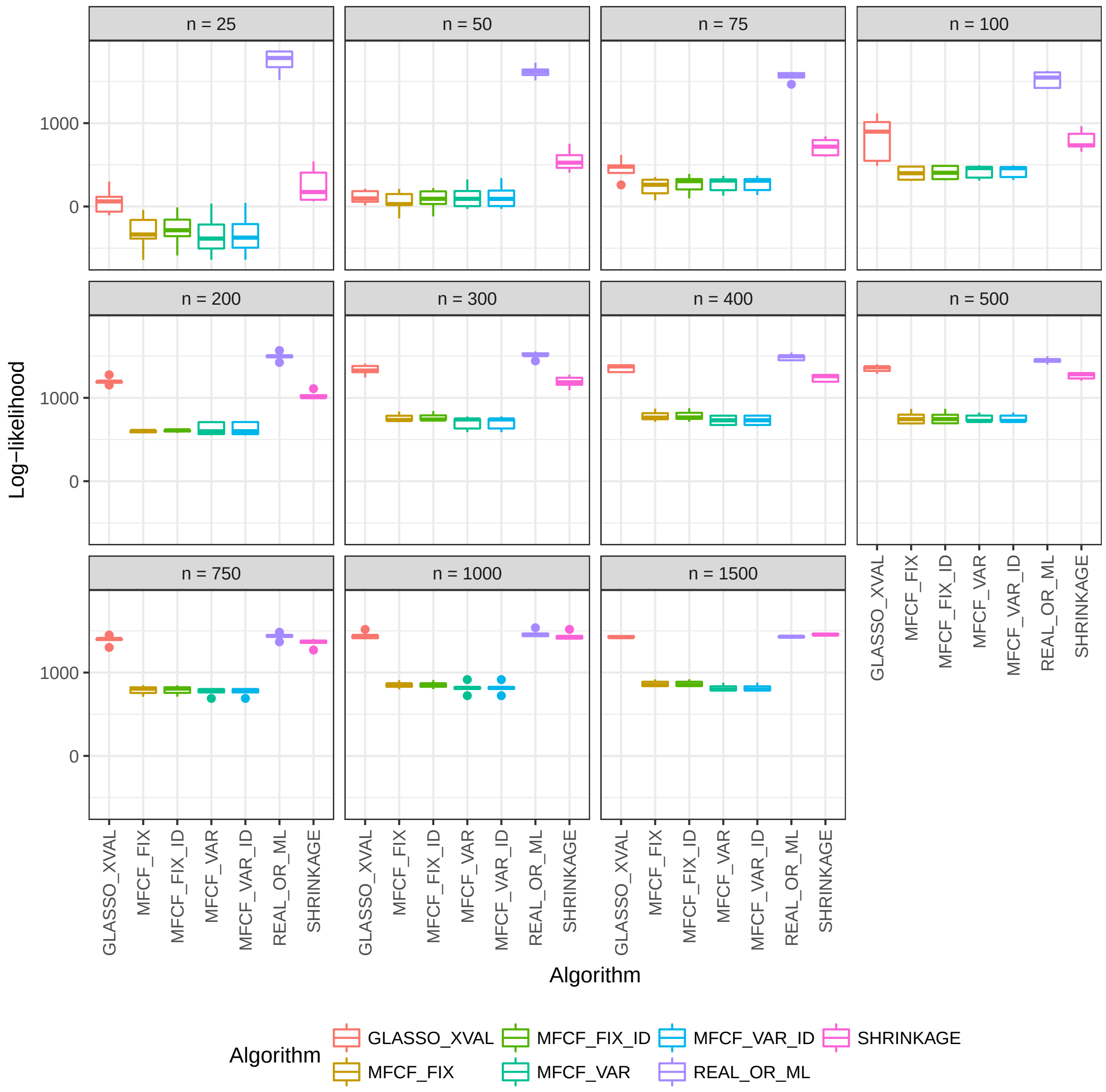


Figure 6.11: Box plot for the likelihood of the algorithms on synthetic data (factor model) for different lengths of the series. The statistics is based on a total of 50 test sets (10 test sets for each of 5 different training / validation sets) per length of the time series.

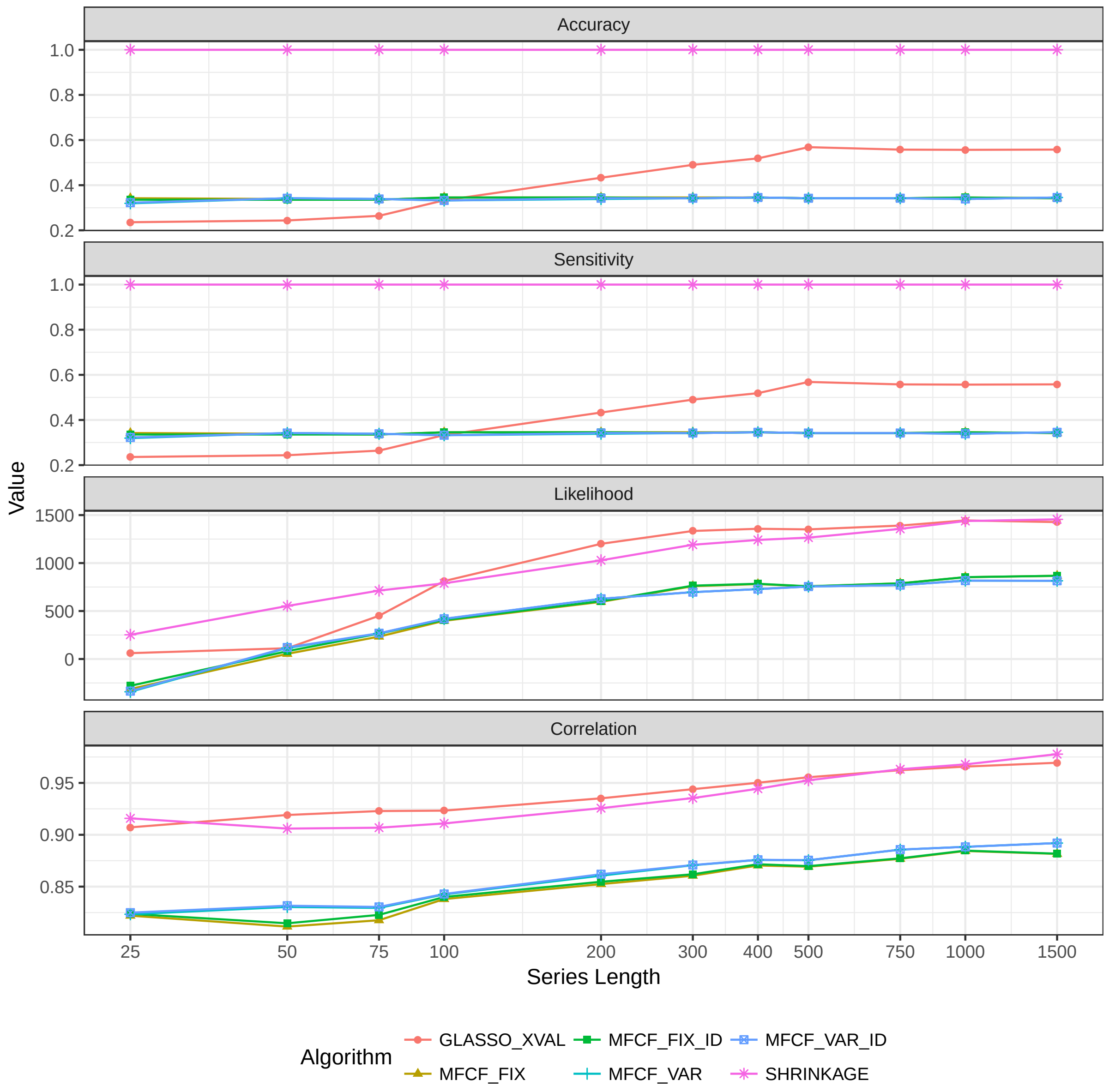


Figure 6.12: Performance of the algorithms on synthetic data (factor model) for different lengths of the series. The statistics is based on a total of 50 test sets (10 test sets for each of 5 different training / validation sets) per length of the time series.

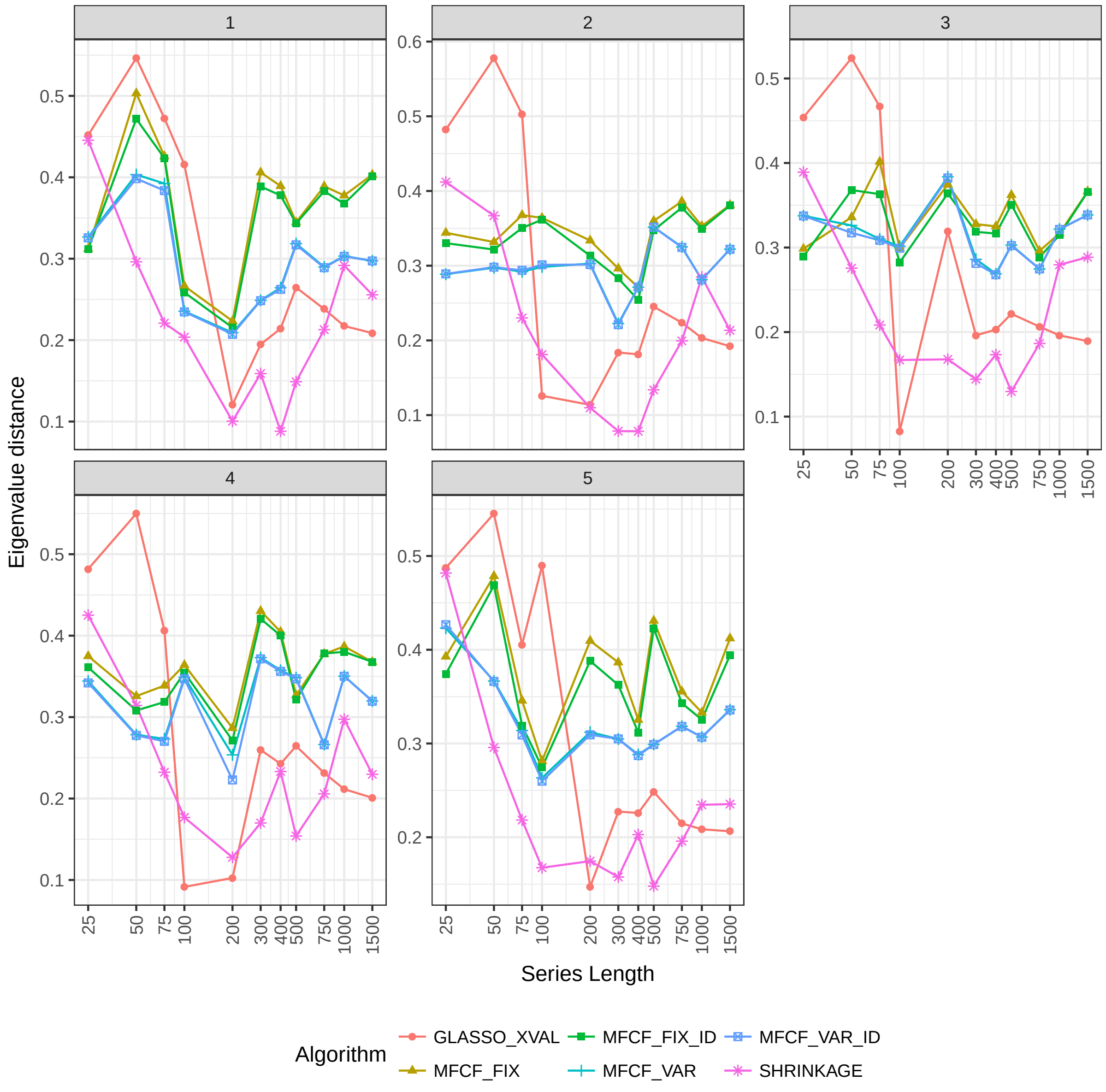


Figure 6.13: Eigenvalue distance for synthetic data (factor model). The five panels show the values at different time series lengths for 5 training / validation datasets.

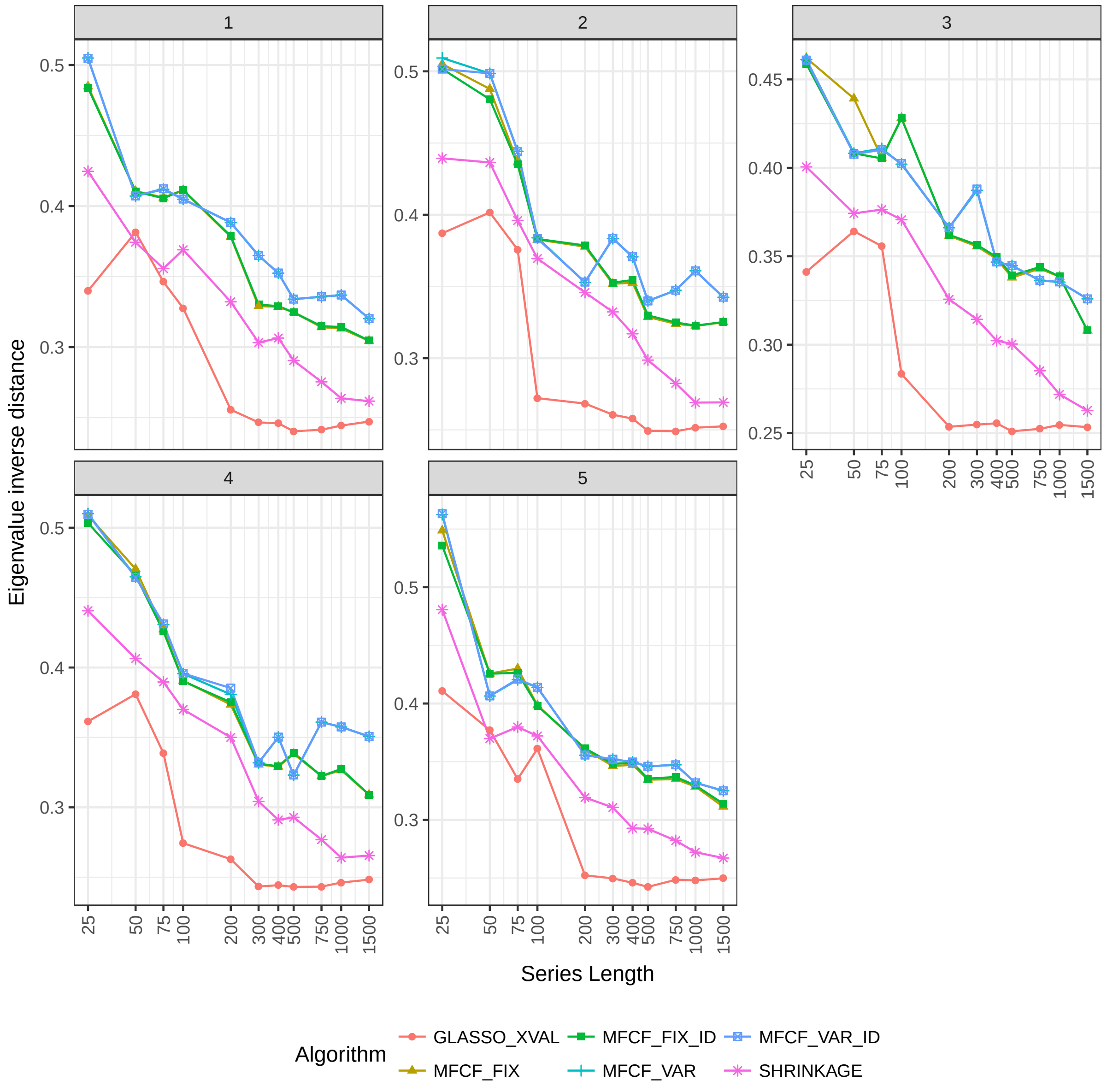


Figure 6.14: Inverse eigenvalue distance for synthetic data (factor model). The five panels show the values at different time series lengths for 5 training / validation datasets.

Series length	GLASSO XVAL	MFCF FIX	MFCF FIX ID	MFCF VAR	MFCF VAR ID	SHRINKAGE
25	1168	1694	1661	1582	1597	4950
50	1206	1677	1661	1694	1694	4950
75	1308	1661	1661	1678	1678	4950
100	1648	1710	1710	1645	1645	4950
200	2143	1710	1710	1678	1694	4950
300	2426	1710	1694	1694	1694	4950
400	2568	1710	1710	1710	1710	4950
500	2814	1694	1694	1694	1694	4950
750	2760	1694	1694	1694	1694	4950
1000	2755	1710	1710	1678	1678	4950
1500	2761	1694	1694	1710	1710	4950

Table 6.6: Mean number of non-zero coefficient in the precision matrix by length of time series. The statistics is based on a total of 5 different training / validation sets per length of the time series.

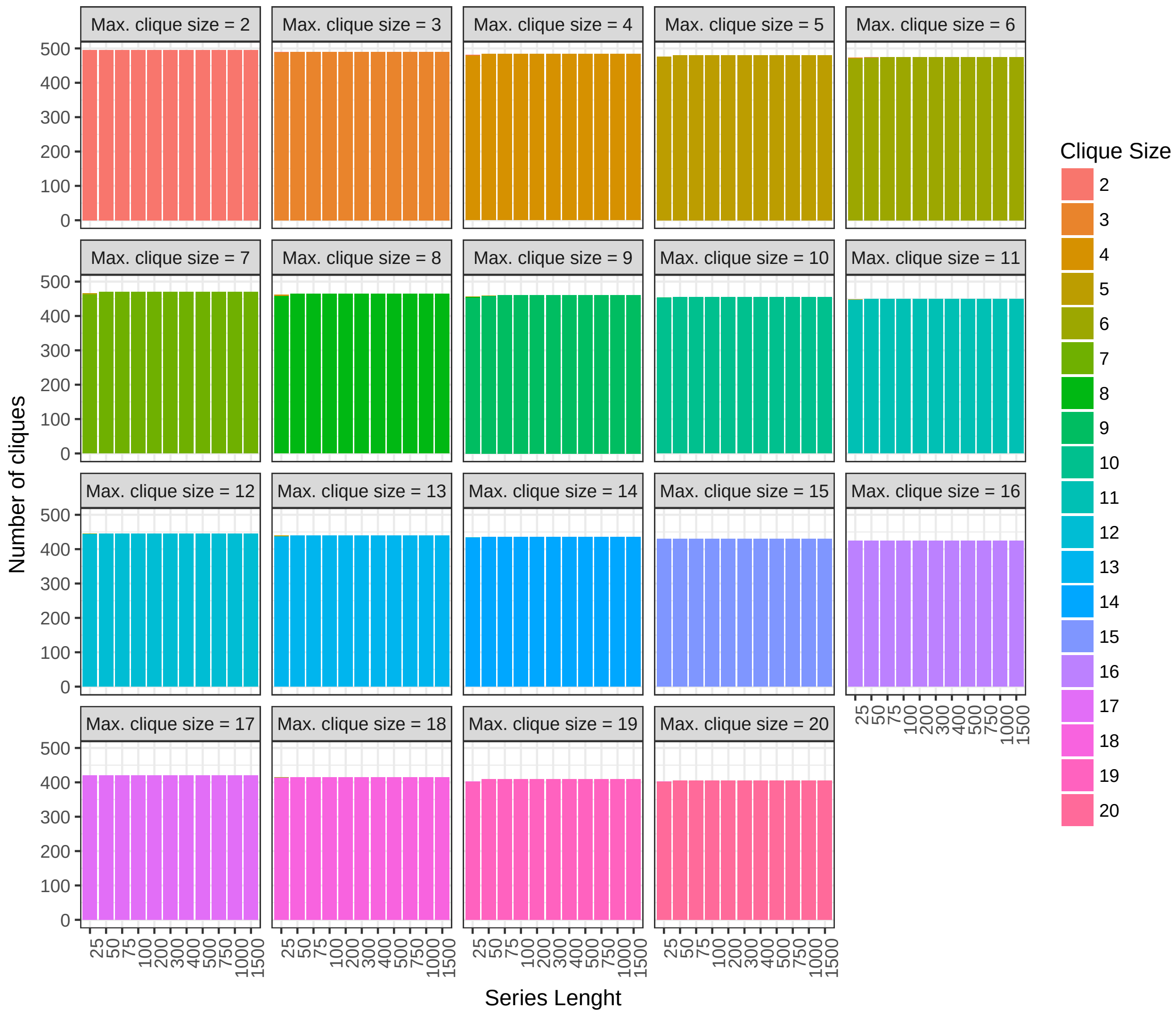


Figure 6.15: Composition of cliques for synthetic data (factor model). The statistics is based on a total of 5 different training / validation sets per length of the time series.

6.4.4 Real Data

Performance measures are reported in Figures 6.16-6.20. We see from the inspection these Figures 6.16 and 6.17 that with real data the log-likelihood is comparable across all models, with slight better values for MFCF_FIX and MFCF_VAR for shorter time series. It is worth noting that, in the family of the MFCF algorithms, the two that use the clique tree shrinkage target described in Equation 6.7 (MFCF_FIX and MFCF_VAR) perform significantly better, for short time series, than the models with the same structure but the simpler identity matrix (MFCF_FIX_ID and MFCF_VAR_ID) as a shrinkage target. Table 6.7 shows that the penalty and shrinkage parameters decrease, as expected, with the length of the time series. Table 6.8 shows that from time series lengths above 500 the GLASSO produces matrices with a significantly higher number of parameters different from zero, including almost 50% of the total number of elements. The growth in performance for MFCF family of algorithms is in this case constrained by the maximum clique size. As Figure 6.18 shows the MFCF algorithms performs better in the approximation of the eigenvalues of the precision matrix, and slightly worse (Figure 6.19 in the representation of the eigenvalues of the inverse precision matrix. In this experiment, since we don't know the "true" correlation matrix we have used the maximum likelihood estimate of the correlation matrix over the full time series. Overall the results for this dataset demonstrate that MFCF_FIX and MFCF_VAR are the best performer. The analysis in Figure 6.20 showing the composition of the cliques of different sizes shows that the synthetic model closest to the real data is the factor model (see 6.4.3). This seems to justify the modelling of portfolios of stocks reacting to a set of external factors.

Figure 6.20 shows that, excepting for the shortest time series, the MFCF_VAR algorithms almost always use the largest allowed clique size.

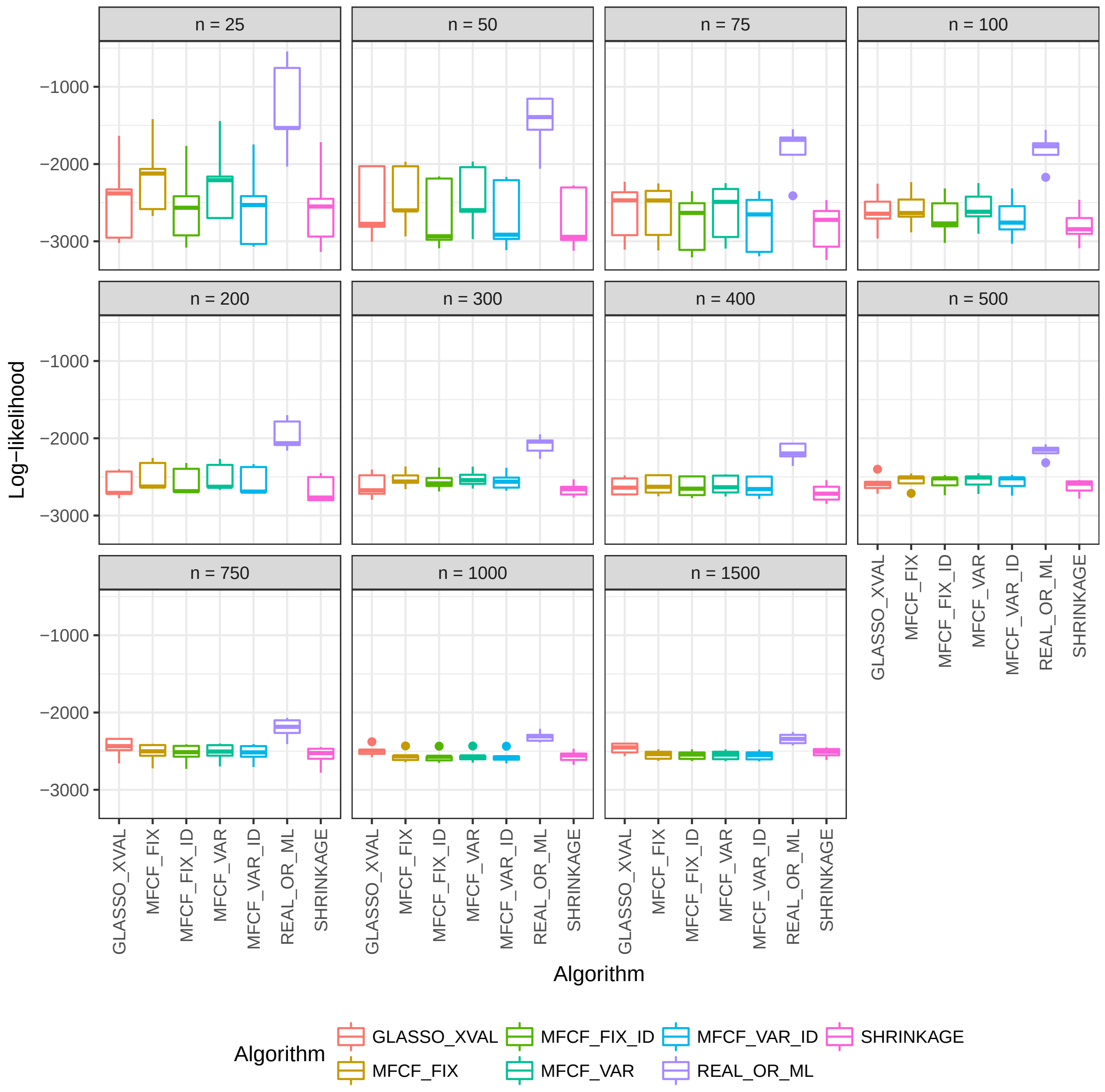


Figure 6.16: Log-likelihood of the algorithms on real data (stock returns) for different length of the series. The statistics is based on a total of 50 test sets (10 test sets for each of 5 different training / validation sets) per length of the time series.

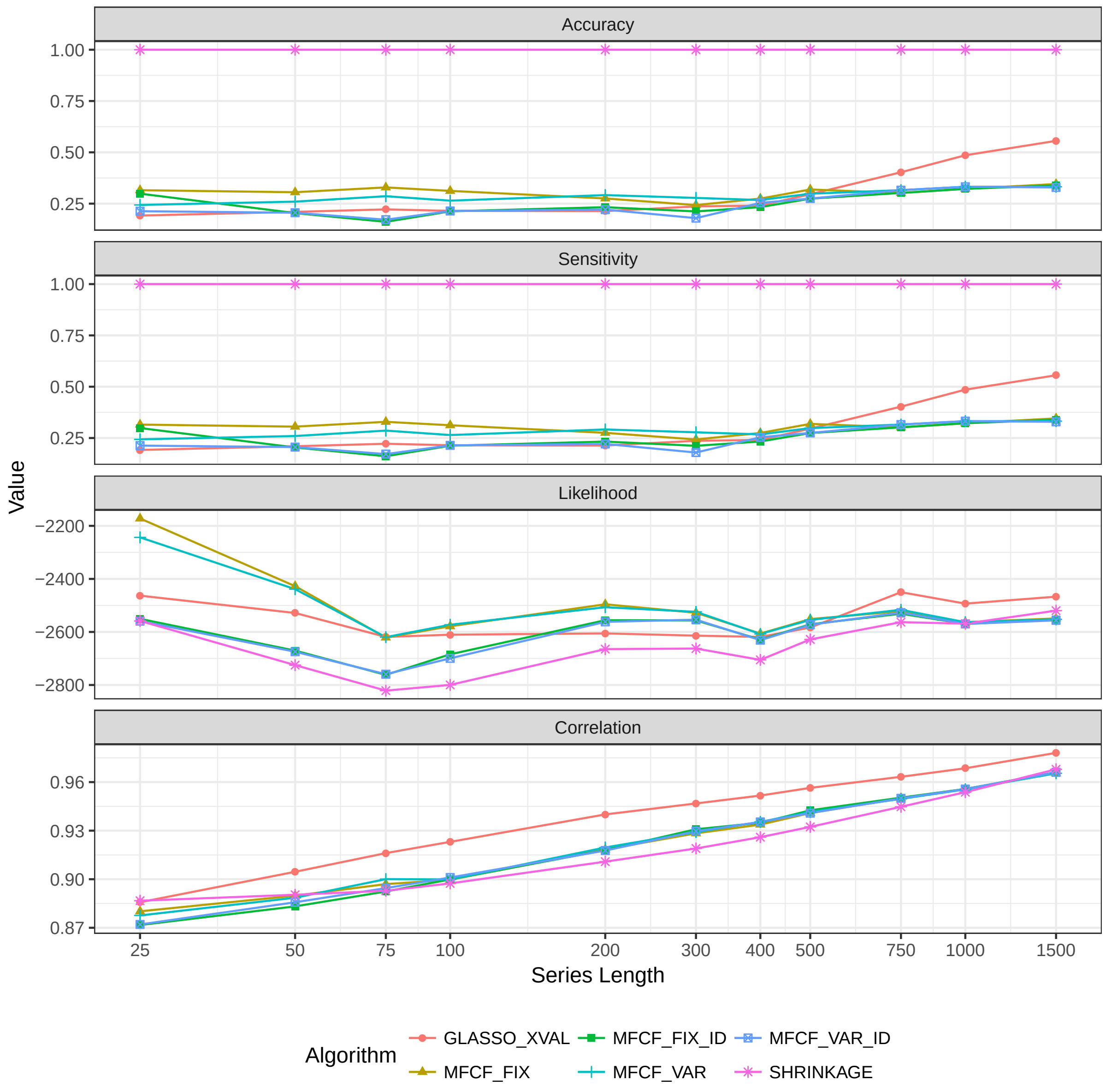


Figure 6.17: Performance of the algorithms on real data (stock returns) for different lengths of the series. The statistics is based on a total of 50 test sets (10 test sets for each of 5 different training / validation sets) per length of the time series.

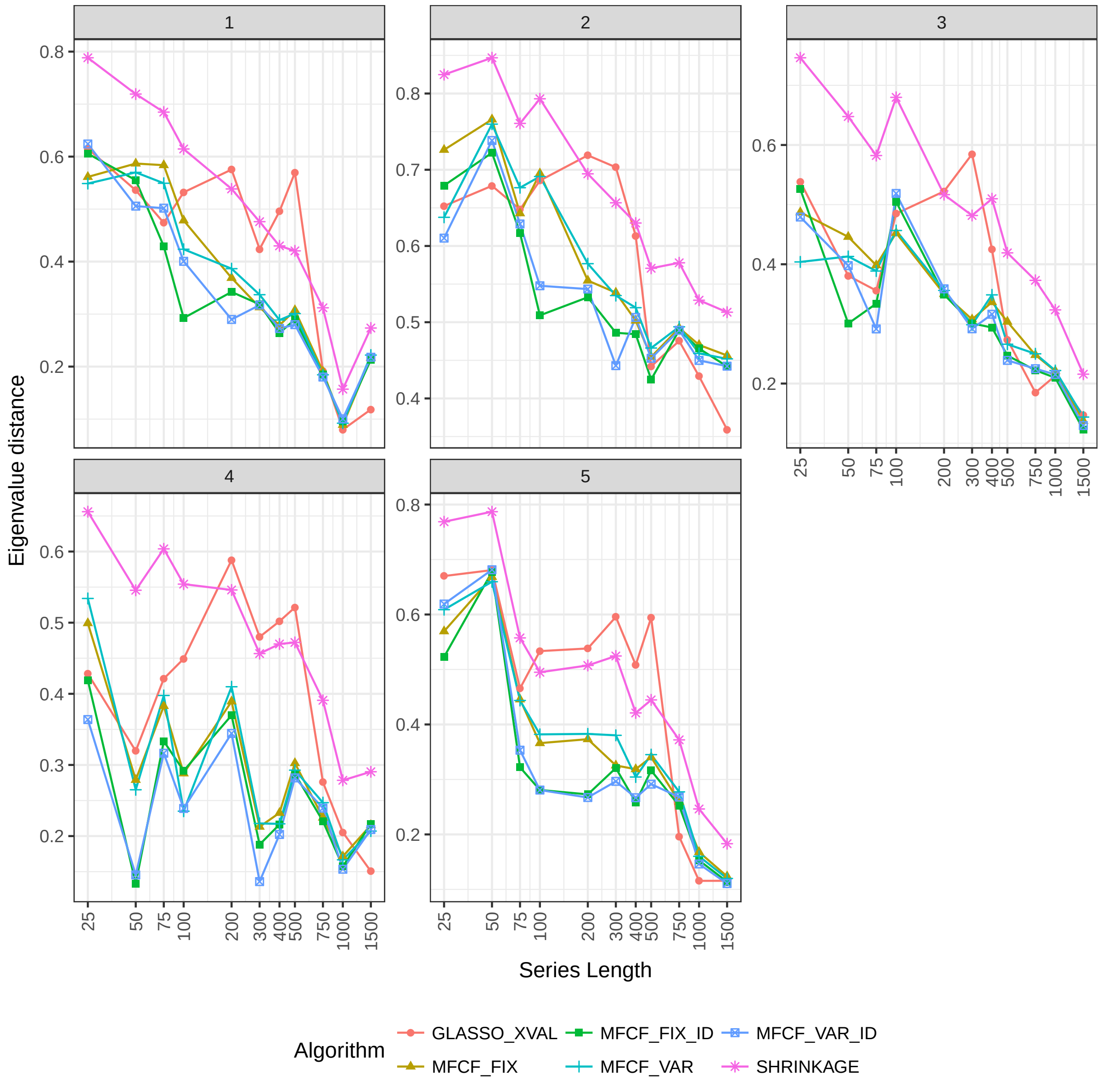


Figure 6.18: Eigenvalue distance for real data (stock returns) for 5 training sets. The five panels show the values at different time series lengths for 5 training / validation datasets.

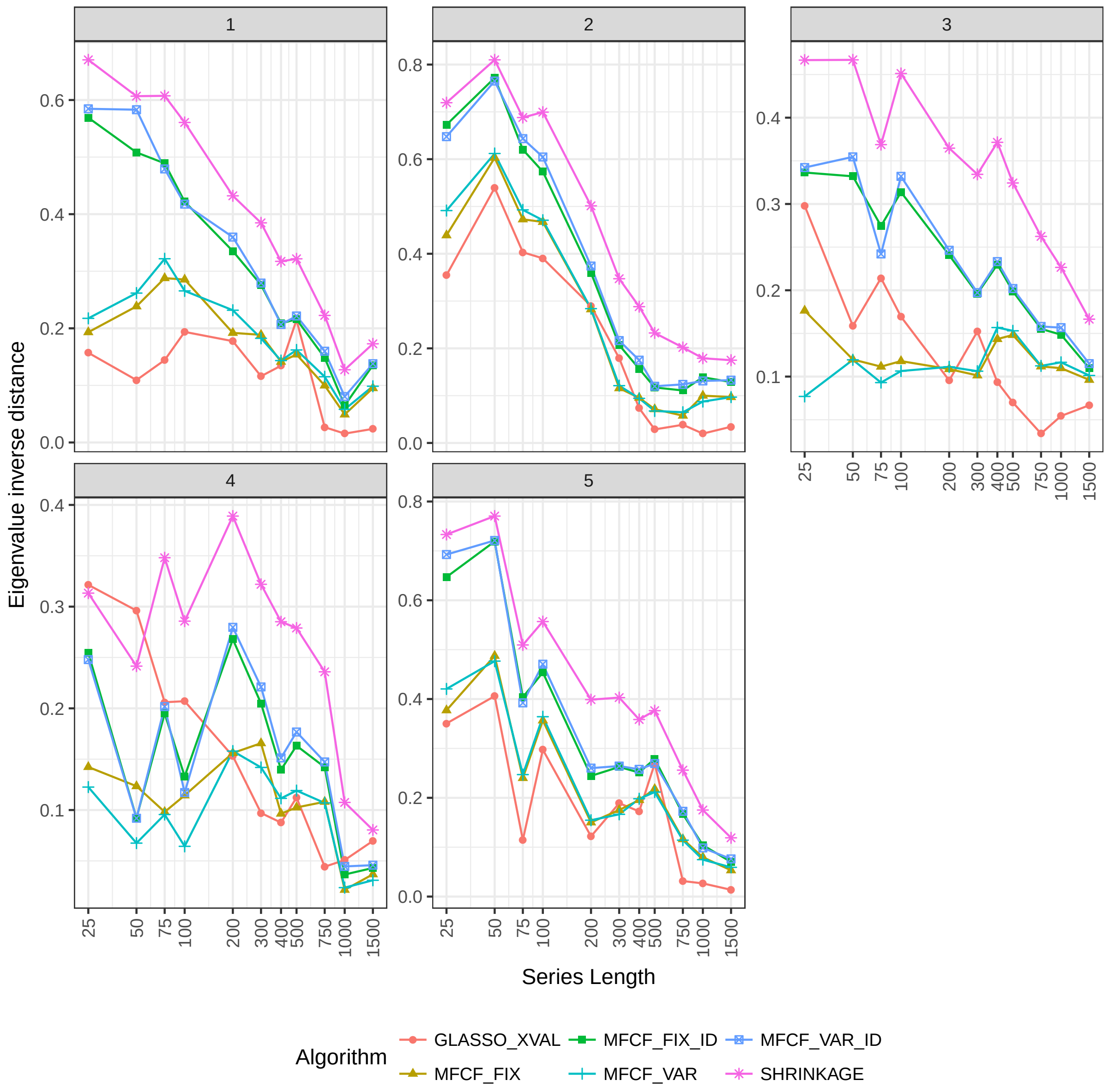


Figure 6.19: Inverse eigenvalue distance for real data (stock returns) for 5 training sets. The five panels show the values at different time series lengths for 5 training / validation datasets.

Series length	GLASSO XVAL	MFCF FIX	MFCF FIX ID	MFCF VAR	MFCF VAR ID	SHRINKAGE
25	0.21	0.81	0.57	0.80	0.54	0.67
50	0.15	0.73	0.47	0.70	0.47	0.63
75	0.12	0.62	0.35	0.61	0.36	0.55
100	0.12	0.56	0.31	0.54	0.33	0.54
200	0.12	0.43	0.26	0.44	0.26	0.44
300	0.10	0.32	0.20	0.34	0.18	0.39
400	0.08	0.28	0.17	0.28	0.17	0.34
500	0.07	0.25	0.15	0.24	0.15	0.30
750	0.03	0.18	0.12	0.19	0.13	0.24
1000	0.02	0.13	0.08	0.13	0.08	0.17
1500	0.01	0.11	0.07	0.11	0.07	0.14

Table 6.7: Mean penalty/shrinkage parameter by length of time series. The statistics is based on a total of 5 different training / validation sets per length of the time series.

Series length	GLASSO XVAL	MFCF FIX	MFCF FIX ID	MFCF VAR	MFCF VAR ID	SHRINKAGE
25	950	1561	1478	1206	1055	4950
50	1039	1512	1010	1286	1018	4950
75	1099	1628	798	1416	850	4950
100	1063	1545	1053	1310	1060	4950
200	1057	1363	1152	1443	1094	4950
300	1170	1202	1049	1375	888	4950
400	1191	1359	1154	1325	1255	4950
500	1464	1578	1358	1477	1360	4950
750	1994	1495	1495	1562	1562	4950
1000	2404	1595	1595	1645	1645	4950
1500	2752	1710	1677	1645	1629	4950

Table 6.8: Mean number of non-zero coefficient in the precision matrix by length of time series. The statistics is based on a total of 5 different training / validation sets per length of the time series.

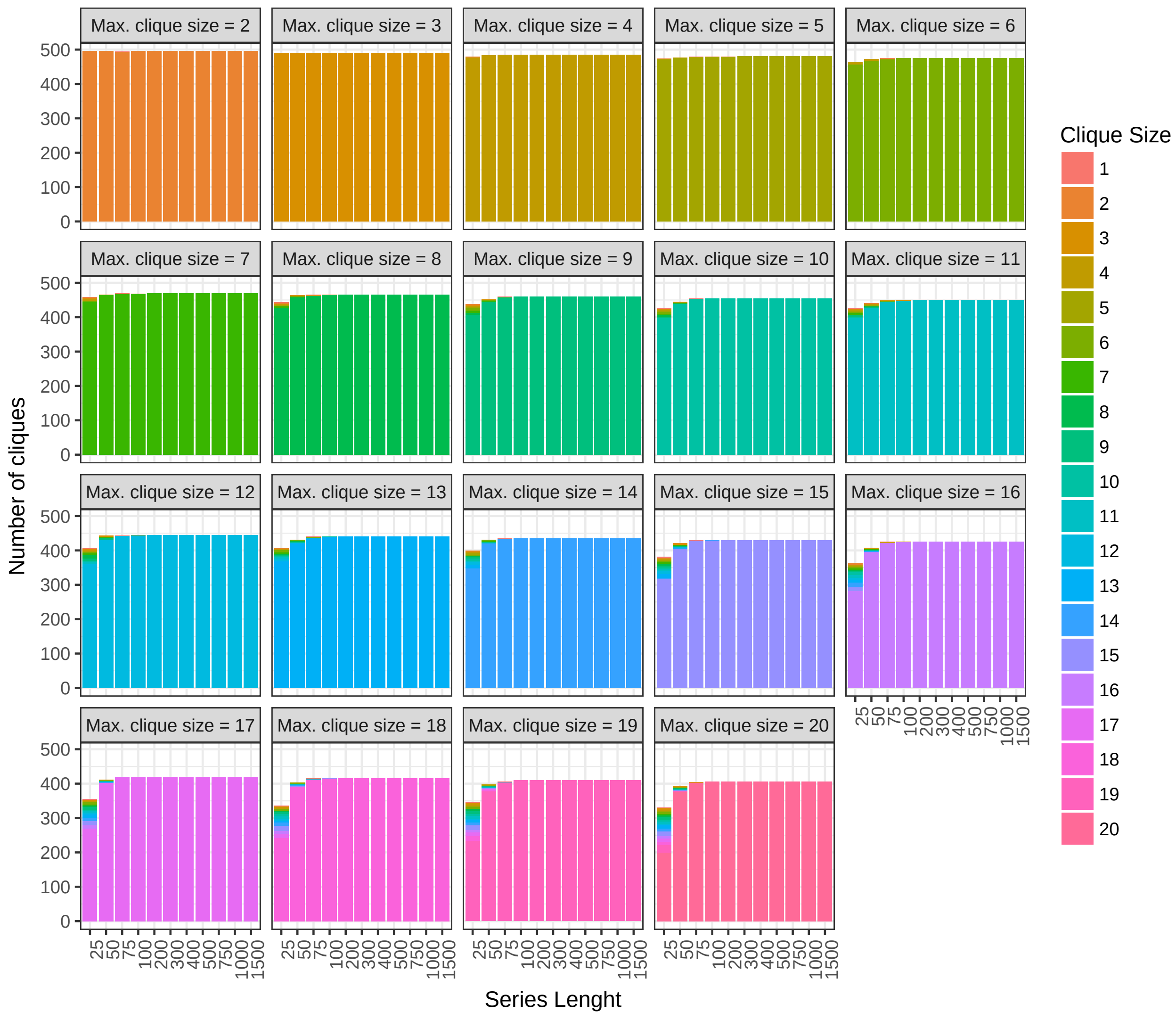


Figure 6.20: Composition of cliques for real data (stock returns). The statistics is based on a total of 5 different training / validation sets per length of the time series.

Chapter 7

Topological Data Analysis with the MFCF

In this chapter we describe some applications of the MFCF in relation to the fast growing field of Topological Data Analysis (TDA). In particular we show that it is possible and useful to analyse the clique trees produced by the MFCF as a simplicial complex and that the calculation of Betti numbers allows in some circumstances to draw interesting conclusions on the distribution that generated the data. Our examples show that the analysis of the structure of the cliques and separators provides information equivalent to the Betti numbers. Finally, we illustrate a new gain function that allows to formulate hypotheses about the dimensionality of the data and the number of independent factors in a factor model.

7.1 Applications to Topology of Data

In chapter 6 we have already shown how the MFCF allows to perform data analysis at different levels of resolution, or *multiscale analysis*, by changing the maximum allowed clique size or the level of confidence of the tests such as in the gain function for the multivariate likelihood statistically validated (described in Section 3.3.3.4). From the analysis of the clique sizes we have been able to highlight differences between sparse models and factor models, which are in general non sparse. The count of the clique sizes achieved with the gain functions used in Chapter 6 is a rather crude description of the geometry of the data; therefore, a legitimate question would be whether it is possible to use more sophisticated tools to understand better the nature of the data. One common tool used in TDA is the persistent homology, where for every data set one builds a *filtration* of simplicial complexes. In Sections 7.1.1 and 7.1.2 we propose a way to build (quasi-)filtrations that can be analysed using homological tools. In Section 7.1.3.3 we propose a new gain function to study the dimensionality of a dataset generated by a factor model. Topological Data Analysis is a well-developed, mathematically sophisticated field and, in order to keep the size of this chapter to an acceptable level, the preliminary definitions and theorems will be necessarily succinct, and we invite the reader to consult the literature cited to achieve a better understanding of the techniques and the context of this rapidly expanding field.

7.1.1 Abstract Simplicial Complexes

This basic introduction is based on Dumas et al. (2003), more detailed information can be found in standard algebraic topology textbooks such as Munkres (1984) or, for a computational perspective, Zomorodian (2009). Good introductions to the topological analysis of data are Ghrist (2008) and Carlsson (2009).

Definition 74 (Finite simplicial complex)**.** A finite simplicial complex Δ over the ground set Ω[1] is a set of finite subsets of Ω with the property of being closed with respect to set inclusion: that is if $B \in \Omega$ and $A \subseteq B$, then $A \in \Omega$. An element of Ω is called a face. A face which is maximal with respect to inclusion is called a facet. The dimension of a face A is the cardinality of the face (as a set) minus one. The dimension of a simplicial complex is the dimension of its largest facet. It is usual to denote with the symbol f_i the number of i-dimensional faces of Δ and to call the vector $(f_0, f_1, \dots, f_d)$ the f-vector of Δ[2].

An abstract simplicial complex is the counterpart of a geometric simplicial complex, where the sets are real, geometric, simplices that meet each other along lower-dimensional faces. For instance a simple graph is a 2-dimensional simplicial complex where the facets can be edges (that is 1-dimensional simplices) or isolated vertices (that is 0-dimensional simplices). Edges can only meet at one common vertex. Similarly a triangulation of a geometric object is a 2-dimensional simplicial complex made of triangular faces, segments and isolated points; simplexes can only meet in isolated points or along a common segment. The idea is to represent a complex space as the union of

[1]In the case of clique trees the ground set is the set of vertices V.

[2]The points of Ω are conventionally taken to have dimension 0.

simpler building blocks of varying dimension.

Example 9 (The clique simplex of a graph)**.** The set of all complete subsets of a graph is a simplicial complex. It is immediate to verify that the induced subset A of a complete set is also complete, and therefore it enjoys the property of Definition 74. Any clique is a face and the maximal cliques are the facets. A clique tree, being a particular case of a graph, is naturally endowed with a simplicial complex structure.

It is possible to consider formal linear combinations of simplices, or *chains*. For instance if we have a triangle $1,2,3$ we can represent the boundary of the triangle as the formal sum of its segments $1\cdot\{1,2\}+1\cdot\{2,3\}+1\cdot\{3,1\}$. Note that a full generic treatment of the simplicial homology should take into account also the *orientation* of the simplices, such that the orientation of the segments is important: $1\cdot\{1,2\}+1\cdot\{2,3\}+1\cdot\{3,1\} = 1\cdot\{1,2\}+1\cdot\{2,3\}-1\cdot\{1,3\}$. A path can be represented as a formal linear combination of the edges, possibly taking into account their orientation if the graph is directed. This motivates the following definition.

Definition 75 (Chain group of a simplicial complex)**.** Given a commutative ring with unity R (for instance the integers $\mathbb{Z}$ or the Galois field $\mathbb{F}_2$) and the set Δ_i of all $i-$dimensional faces of a simplex Δ, we can build the set of formal sums $\sum_{e\in\Delta_i} r_e e$ and observe that it can be given the structure of a commutative group which we will call the $i-$th chain group and will denote with $C_i(\Delta, R)$.

Example 10 (Simplicial complex example)**.** In Figure 7.1 we see a simplicial complex with the two 2-dimensional simplices $\{1,2,3\}$ and $\{2,4,3\}$. It could be seen as a formal linear combination $1\cdot\{1,2,3\}+1\cdot\{2,4,3\}$. We have given an orientation to the simplices by ordering the vertices in a counter-clockwise fashion.

Definition 76 (The $i-th$ differential ∂_i (boundary operator))**.** Given a face $E = \{e_0,\dots,e_i\}$ we define the $i-$th differential $\partial_i : C_i(\Delta,R) \to C_{i-1}(\Delta,R)$ as the ring homomorphism defined on the element E as $\partial_i E = \sum_{j=0}^{i}(-1)^j(E\setminus e_j)$. This homomorphism can be extended linearly to all the formal sums in the chain group $C_i(\Delta,R)$.

Example 11 (Boundary operator)**.** With reference to Figure 7.1 we can identify two 2-dimensional faces $E_1 = \{1,2,3\}$ and $E_2 = \{2,4,3\}$. The application of the boundary

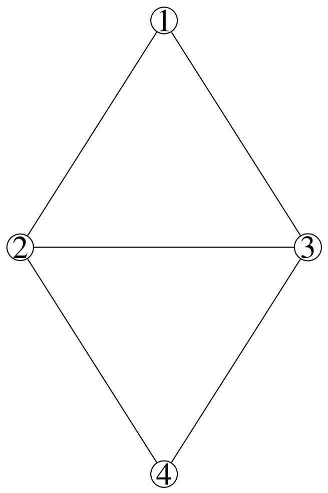


Figure 7.1: A simplicial complex made of the two simplices $\{1,2,3\}$ and $\{2,4,3\}$. Note that we have given an orientation to the simplices by ordering the vertices counter-clockwise.

operator to E_1 produces the chain $\partial_2(E_1) = 1\cdot\{2,3\} - 1\cdot\{1,3\} + 1\cdot\{1,2\}$, and similarly the boundary of E_2 is $\partial_2(E_2) = 1\cdot\{4,3\} - 1\cdot\{2,3\} + 1\cdot\{2,4\}$. Let us consider the chain C made of the sum of E_1 and E_2: $C = E_1 + E_2 = 1\cdot\{1,2,3\} + 1\cdot\{2,3,4\}$. According to Definition 76 we can extend by linearity the operator ∂_2 to the chain C:

$$\begin{aligned}
\partial_2(C) &= \partial_2(1\cdot\{1,2,3\} + 1\cdot\{2,4,3\}) && (7.1)\\
&= 1\cdot\partial_2(\{1,2,3\}) + 1\cdot\partial_2(\{2,4,3\}) && (7.2)\\
&= 1\cdot\{2,3\} - 1\cdot\{1,3\} + 1\cdot\{1,2\} + 1\cdot\{4,3\} - 1\cdot\{2,3\} + 1\cdot\{2,4\} && (7.3)\\
&= -1\cdot\{1,3\} + 1\cdot\{1,2\} + 1\cdot\{4,3\} + 1\cdot\{2,4\} && (7.4)\\
&= 1\cdot\{1,2\} + 1\cdot\{2,4\} + 1\cdot\{4,3\} - 1\cdot\{1,3\} && (7.5)\\
&= 1\cdot\{1,2\} + 1\cdot\{2,4\} + 1\cdot\{4,3\} + 1\cdot\{3,1\} && (7.6)
\end{aligned}$$

Note that with the counter-clockwise ordering of the two simplices the boundary operator applied to the formal combination produces the boundary of the whole simplicial complex (the "sum" of E_1 and E_2), the edge in common "cancels out". In our calculations we will use $\mathbb{F}_2$ as the underlying ring, so we will not have to define a specific orientation for the simplices produced by the MFCF.

Remark 77 (Meaning of ∂_i)**.** With the help of Example 11 we see that the intuitive meaning of the $i-th$ differential is to produce the chain of the $i-1$-dimensional faces.

The alternating sign takes into account the orientation of the faces so that common faces, lying in the internal of the simplicial complex, can be cancelled out so that only the external faces can be included in the boundary. This meaning justifies the term “boundary”: it takes the faces that delimit higher dimensional chains. Elements that “cancel out” are in reality internal, since are the boundary of neighbouring simplices.

The boundary operator has the important characteristic that successive applications produce the empty set. Intuitively “the boundary of a boundary is empty”. This fact is made more formal in the following Theorem 78.

Theorem 78 (Munkres (1984)). Let $\partial_i : C_i(\Delta,R) \to C_{i-1}(\Delta,R)$ and $\partial_{i-1} : C_{i-1}(\Delta,R) \to C_{i-2}(\Delta,R)$, then $\partial_{i-1} \circ \partial_i = 0$.

Example 12 (Boundary of a boundary). Let us carry on from Example 11. The calculation of the boundary of the boundary of the chain C is as follows:

$$\begin{aligned}
\partial_1 \circ \partial_2(C) &= \partial_1(1\cdot\{1,2\}+1\cdot\{2,4\}+1\cdot\{4,3\}+1\cdot\{3,1\}) && (7.7)\\
&= 1\cdot\partial_1(\{1,2\})+1\cdot\partial_1(\{2,4\})+1\cdot\partial_1(\{4,3\})+1\cdot\partial_1(\{3,1\}) && (7.8)\\
&= \{2\}-\{1\}+\{4\}-\{2\}+\{3\}-\{4\}+\{1\}-\{3\} && (7.9)\\
&= \emptyset && (7.10)
\end{aligned}$$

Definition 79 (i-th homology group (Munkres, 1984)). The kernel of ∂_i is called the group of i-cycles (often denoted with $Z_i(\Delta)$) and the image of ∂_{i+1} is called the group of i-boundaries (and is denoted by $B_i(\Delta)$). By Theorem 78 every i-boundary is also an i-cycle and therefore $B_i(\Delta) \subset Z_i(\Delta)$. The quotient group $H_i(\Delta) = Z_i(R)/B_i(R)$ is the i-th homology group of the simplicial complex Δ.

The intuitive meaning of the i-th homology group is to evidence the cycles that are not the boundaries of a simplicial complex. For instance, if we take a simplicial complex generated by the three segments $\{1,2\},\{2,3\},\{3,1\}$ we see that the three segments generate the only cycle, but since they are not the boundary of another higher dimensional simplicial complex (that is, the group of the 1-boundaries is empty) the dimension of the group H_1 is 1.

Now let us assume that the underlying ring R is a field F. It can be shown that the dimension of $H_i(\Delta)$ as an F-vector space is $\beta_i^F = \mathrm{Dim}\,\mathrm{Ker}(\partial_i) - \mathrm{Dim}\,\mathrm{Im}(\partial_{i+1})$. Since the boundary operators are linear operators between vector spaces they can be represented by matrices with values in F. The calculation of β_i^F therefore is reduced to a problem in linear algebra which is generally solved by reducing the boundary operator matrices to the so-called *Smith normal form* (Dumas et al., 2003). β_i^F is called the i-th Betti number and roughly represents the number of i-dimensional holes present in the simplicial complex Δ. The analysis of the Betti numbers is important because it allows to draw conclusions on the shape of the complex. Recently this idea has been coupled with the idea of a *filtered complex* to analyse which feature of a given geometrical object persist at different scales.

Definition 80 (Filtered complex (Zomorodian, 2009))**.** A filtration of complex K is a sequence of sub-complexes $\varnothing = K_0 \subseteq K_1 \subseteq \cdots \subseteq K_n \subseteq K$. A complex with a filtration is called a *filtered complex*.

The idea of persistent homology is to study the homology groups of a filtered complex to see if there are topological features that persist over a reasonable span of the filtration parameter. Features of interest could be the number of connected components, the number of cycles (or one dimensional holes) or the number of two-dimensional holes. These attributes are roughly measured by the first three Betti numbers. If the dataset is multidimensional, then the higher dimensional Betti numbers might be of interest.

We have already seen that the calculation of the Betti numbers, especially for higher dimension, involves linear algebra calculations in spaces with a very high number of objects in many dimensions (Dumas et al., 2003). If we want to analyse the shape of data under the assumption that the data generating process is low-dimensional we need to build simplicial complexes that are amenable to linear algebra routines, and this means in practice that they need to be relatively sparse. Is is therefore necessary to be able to build a simplicial complex that is sparse and with a configurable, but controllable, dimension. From this consideration our proposal to use the clique forest produced by the MFCF.

7.1.2 The Simplicial Complexes built by the MFCF

In our proposed approach we build a family of clique forests (and, therefore, simplicial complexes) by filtering a data set with the MFCF using a statistically validated gain function[3]. Every value of the validation parameter (e.g. the p-value) will produce a clique forest. For instance, in the case of the p-value, the value of 0 would then correspond to the null hypothesis where all the variables underlying the data set are independent, and the value 1 would correspond to the case where all the variables are fully correlated. Strictly speaking, this family of simplicial complexes is not a filtration, since the inclusion relationship of Definition 80 do not necessarily hold and we will therefore use the term of *quasi-filtration* of clique forests.

In this approach we study the *boundaries* of the clique forests produced by the MFCF and not the clique forests themselves. This choice is explained in the Remark below, which shows that the topology of a clique forest is somewhat trivial.

Remark 81 (Clique forest as a simplicial complex)**.** Since the clique forests produced by the MFCF are made by gluing "full" simplexes together, it is easy to show that all the connected components can be collapsed to a single point. In this case the only Betti number of interest would be the first one (the 0-th Betti number), that counts the number of connected components, with all the subsequent ones being trivially zero.

Besides, if one wanted to calculated the connected components, this calculation could be performed much more easily from the counts of cliques and separators.

Remark 82 (Calculation of the connected components for clique forest)**.** From the iterative construction of the MFCF described in Chapter 3, it easy to see how the calculation of the connected components of a clique forest results from the number of cliques minus the number of separators; the separators need to be counted with their multiplicity, that is, if a separator appears more than once, it needs to be counted for as many times. It is easy to show this by taking into account the three cases of the clique expansion operator: if a new clique is purely expanded, the number of cliques and separators does not change and the number of connected components is unchanged as expected. If a new clique is created by attaching to a parent clique then the number of cliques grows by one, the number of separators grows by one and the difference

[3]To fix the ideas we can think of the one described in Section 3.3.3.4.

between number of cliques and number of separators does not change; also in this case the number of connected components is unchanged. Finally, if the new vertex is added by creating a new clique disconnected from any tree, the number of cliques grows by one but the number of separators does not grow since there is no new separator; the number of connected components grows by one.

Because of the two remarks above we observe how it is not advisable to study the clique forests as they are, but it is useful to study the boundary of the clique forest. This object is made by gluing together "empty" simplices and it is therefore much more amenable to topological analysis.

In the next sections we show some results where we prove that the boundaries simplicial complexes produces by the MFCF are amenable to Topological Data Analysis, that there is a significant overlap between the information provided by the Betti numbers and the statistics of cliques and separators and finally, we design a gain function that allows to estimate the number of factors in a factor model.

7.1.3 Examples and applications

In the next three sections we show some applications of TDA to the simplicial complexes produced by the MFCF. The conclusions are less developed than the results of the previous chapters, and the discussion concerns some examples, rather than a theoretical framework developed in full generality; most of the topics discussed here should be considered as a programme for further research. A summary of the findings of Section 7.1.3.1 and 7.1.3.2 is that, when it comes to the identification of geometrical features, the counts of cliques and separators provide essentially the same information as the Betti numbers. The advantage of the analysis of cliques and separators is mainly in the better performance, since the calculation of Betti numbers requires the calculation of many boundary operators and Smith normal forms.

In Section 7.1.3.3 we suggest an approach to the estimation of the number of factors underlying a factor model partially inspired by Independent Factor Analysis (ICA) (Comon, 1994; Hyvärinen and Oja, 2000; Bach and Jordan, 2002).

7.1.3.1 Identification of cliques

In the first application we show how both the Betti numbers and the statistics of the clique sizes can be analysed at different scales and how they can provide useful information on the number and size of the cliques underlying a model.

In this example the "sliding" factor is the p-value of the model. The size of the cliques is limited by the significance level of the hypothesis that the variables are correlated.

Example 13 (Identification of clique sizes and connected components)**.** We start by studying a matrix built out of cliques of a known structure and precisely build from 8 cliques of size ranging from 3 to 12 ([1-3], [4-7], [8-12], [13-18], [19-25], [26-33], [34-42], [43-52]). A pictorial representation of such a matrix is in Figure 7.2. We run the MFCF with the gain function based on the multivariate normal likelihood statistically validated as in chapter 6 and we study the size of the cliques and the Betti numbers for a set of p-values. We perform 20 simulations by creating 20 instances of the matrix with the prescribed structure and collect the average statistics. We allow the maximum clique size to be 10. We observe that as the p-values become more and more significant the clique sizes and the Betti numbers stabilise to a persistent value and are in very good accord. There are two observations to bear in mind in this comparison: 1. the Betti numbers are one dimension lower than the clique sizes, as they are created from the boundary of the simplicial complex, 2. the 0-th Betti number correctly counts the number of connected components, and that correspond to the eight cliques that are statistically independent. The progression from high p-values to low p-values shows that effective filtering is crucial to capture the dependency structure of the matrix.

Figure 7.3 shows the results. We see that, in accord with the clique sizes of the example, the cliques of size one and two (which are not present in the block structured matrix) converge to zero, the cliques of size 3 to 9 achieve a value close to the theoretical exact value of one (as there is exactly one clique of size 3 to 10 in the example matrix), and also the number of cliques of size 10 decreases from very high average numbers to one (note that the y-axis for the clique size of 10 has a different range).

Figure 7.4 shows a consistent picture for the Betti numbers. Also in this case the Betti numbers exhibit convergence to their theoretical value as the p-values become significant. Note that the 0-th Betti number correctly converges to the number of connected components.

Remark 83 (Connected components)**.** Example 13 shows that the use of topological

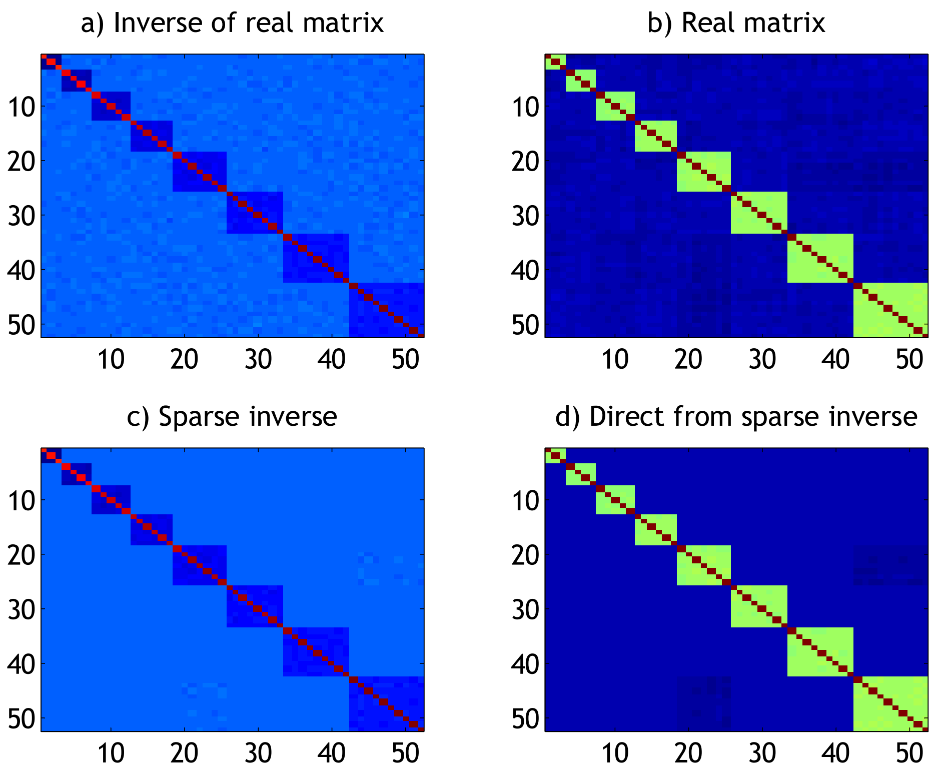


Figure 7.2: A block structured matrix composed of 8 cliques of increasing size (3 to 10). Variables in different cliques are independent, but exhibit small spurious correlation due to noise in figures a) and b). Figures c) and d) show the filtered version of the same matrix using MFCF with a very high p-value for validation (0.0005). The filtering removes the spurious noise and fully reveals the geometric structure of the matrix.

tools such as homology can give more insight into the structure of the variables than the study of the clique structure alone. However, in the specific case of a clique forest, the number of connected components can be calculated as the number of cliques minus the number of separators. On the whole, we can say that the structure of the clique forest (cliques and separators) contains as much information as the simplicial structure produced by the MFCF.

Remark 84 (Computational considerations)**.** In performing the tests described in this Chapter, we have been mostly constrained by the performance of the algorithm for the calculations of the Betti numbers. Not only this algorithm accounts for most of the computation time, but is also the most demanding in terms of memory usage, thus limiting our study to matrices of size 60 or less. While there is space for future research in optimising the calculation of the homology groups for simplicial complexes associated with clique forests, it appears that the MFCF allows much more flexibility in the

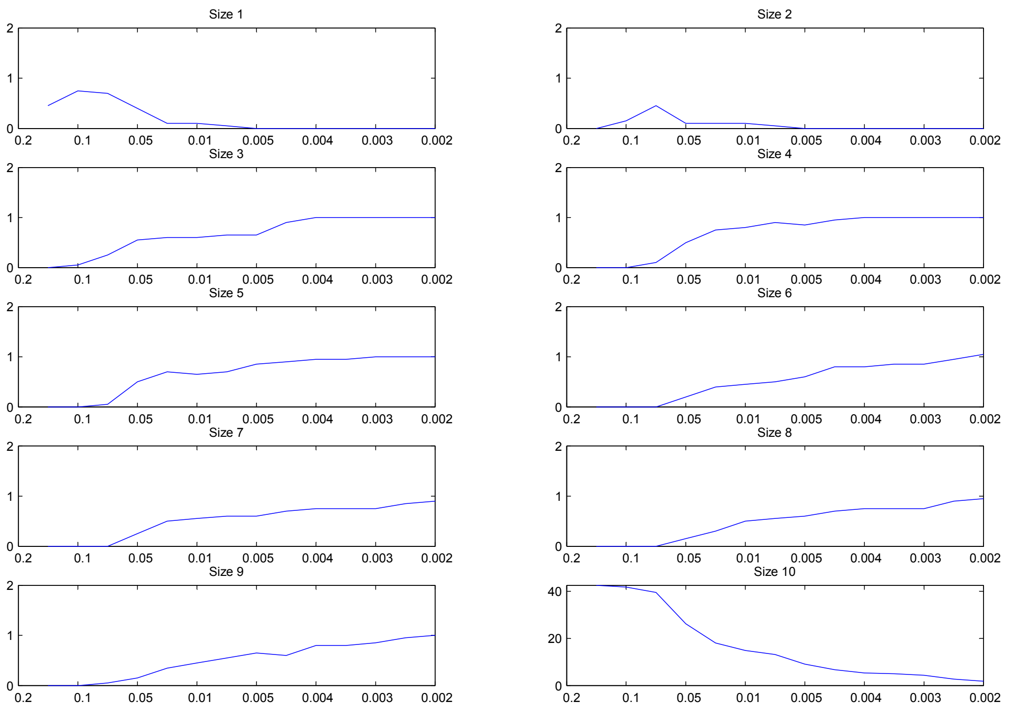


Figure 7.3: Results of the simulation of 20 different matrices with similar clique structure. We show one panel for each clique size between 1 and 10. On the x-axis we report the p-value used for the validation, on the y-axis we report the average number of cliques of the given size across the 20 simulations. We expect that the average number of cliques goes to zero for cliques of size 1 and 2 and that it converges to 1 for all the clique sizes between 3 and 10.

analysis and is less limited.

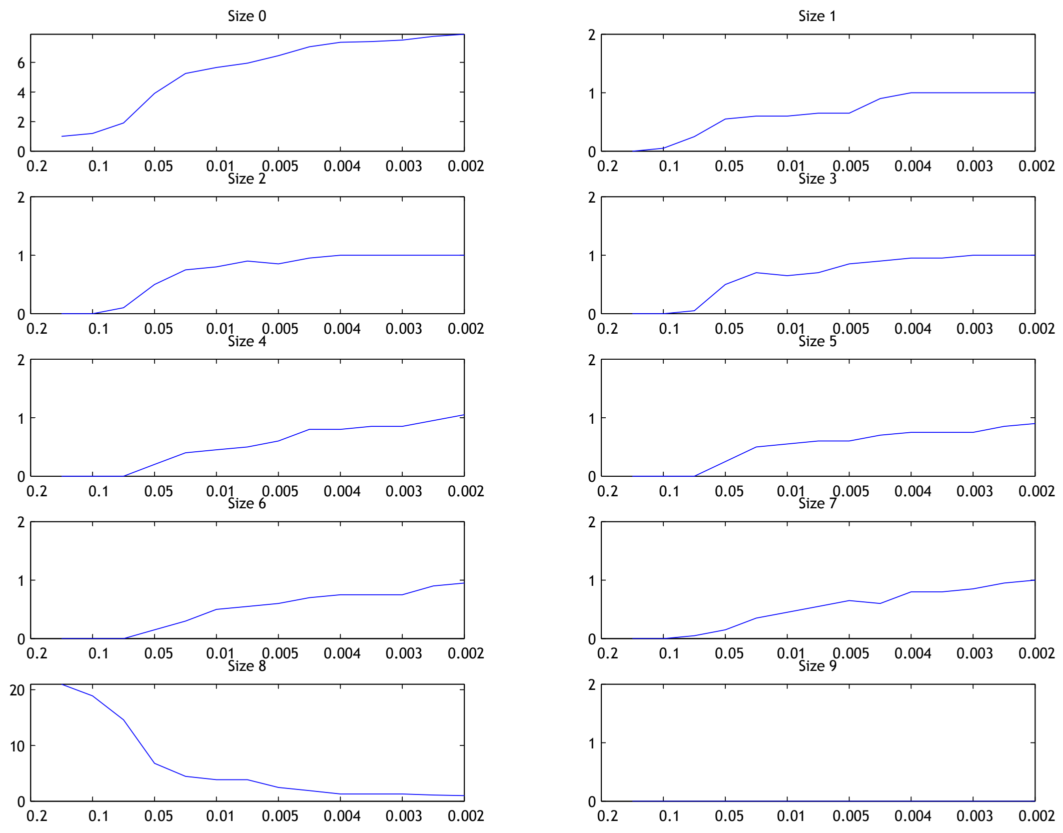


Figure 7.4: Results of the simulation of 20 different matrices with similar clique structure. We show one panel for every Betti number between 0 and 9. On the x-axis we report the p-value used for the validation, on the y-axis we report the average Betti number across the 20 simulations.

7.1.3.2 Identification of large dimensional blocks

In the next example we show an interesting consequence of the geometry of clique forests. Specifically, we limit the maximum size of the cliques to 10, but we show how it is possible to identify geometric features that have a larger dimensionality. In this example we are able to identify three clusters of correlated variables that have a higher dimensionality than the maximum allowed clique size.

Example 14 (Large connected components)**.** In this example we discuss a correlation matrix built out of three relatively large cliques ([1-20], [21-40]; [41-60]); the matrix is shown in Figure 7.5. Specifically, the cliques are larger than the maximum allowed clique size. We run the MFCF with the same settings as in Example 13. Also in this case we observe that, as the p-values become more and more significant, the average values of the clique sizes (Figure 7.6 and of the Betti numbers (Figure 7.7) converges to the theoretically correct values. Similarly to Section 7.1.3.1, we have built 20 matrices with the same structure and performed the analysis for a range of p-values.

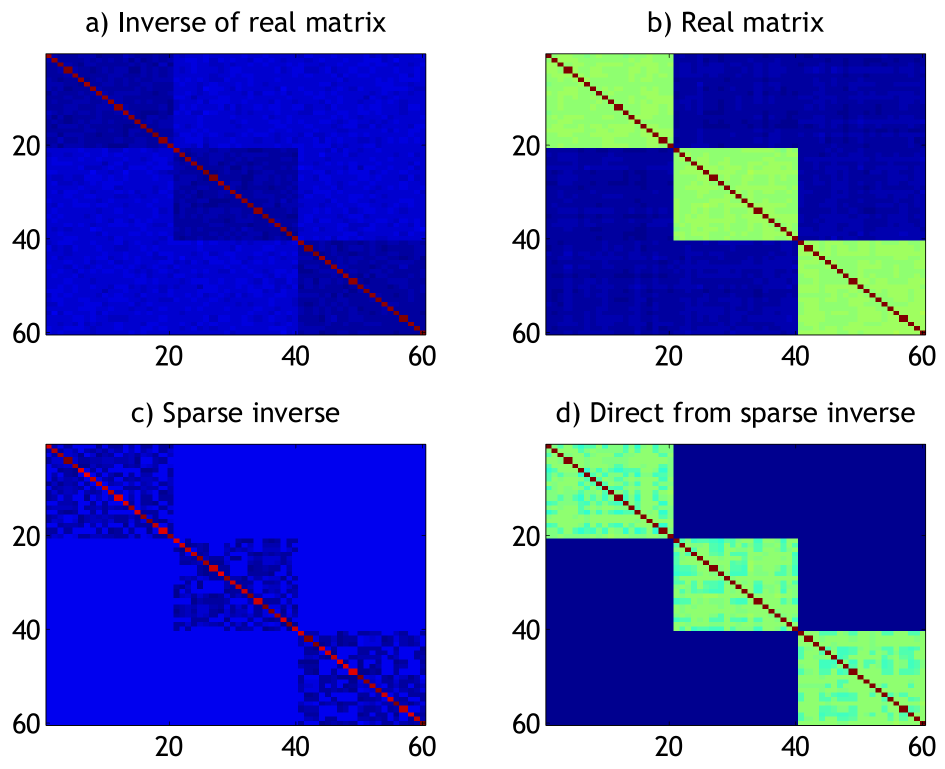


Figure 7.5: A block structured matrix composed of 3 cliques of size larger than the maximum allowed clique size. Variables in different cliques are independent, but exhibit small spurious correlation due to noise in figures a) and b). Figures c) and d) show the filtered version of the same matrix using MFCF with a very high p-value for validation (0.0005).

Remark 85 (Identification of large connected components)**.** The analysis of Figures 7.6 and 7.7 shows a similar feature: the persistence, even at high significance level, of high-dimensional cliques (or Betti numbers), in fact, of the maximum size allowed by the algorithm. This suggests that the variables underneath the matrix are correlated in cliques that are of a higher dimension than the MFCF would be able to detect, due to the constraint on the clique size; however, interestingly, the Betti numbers show the correct number of connected components even if they are of a higher size than the reach of the algorithm.

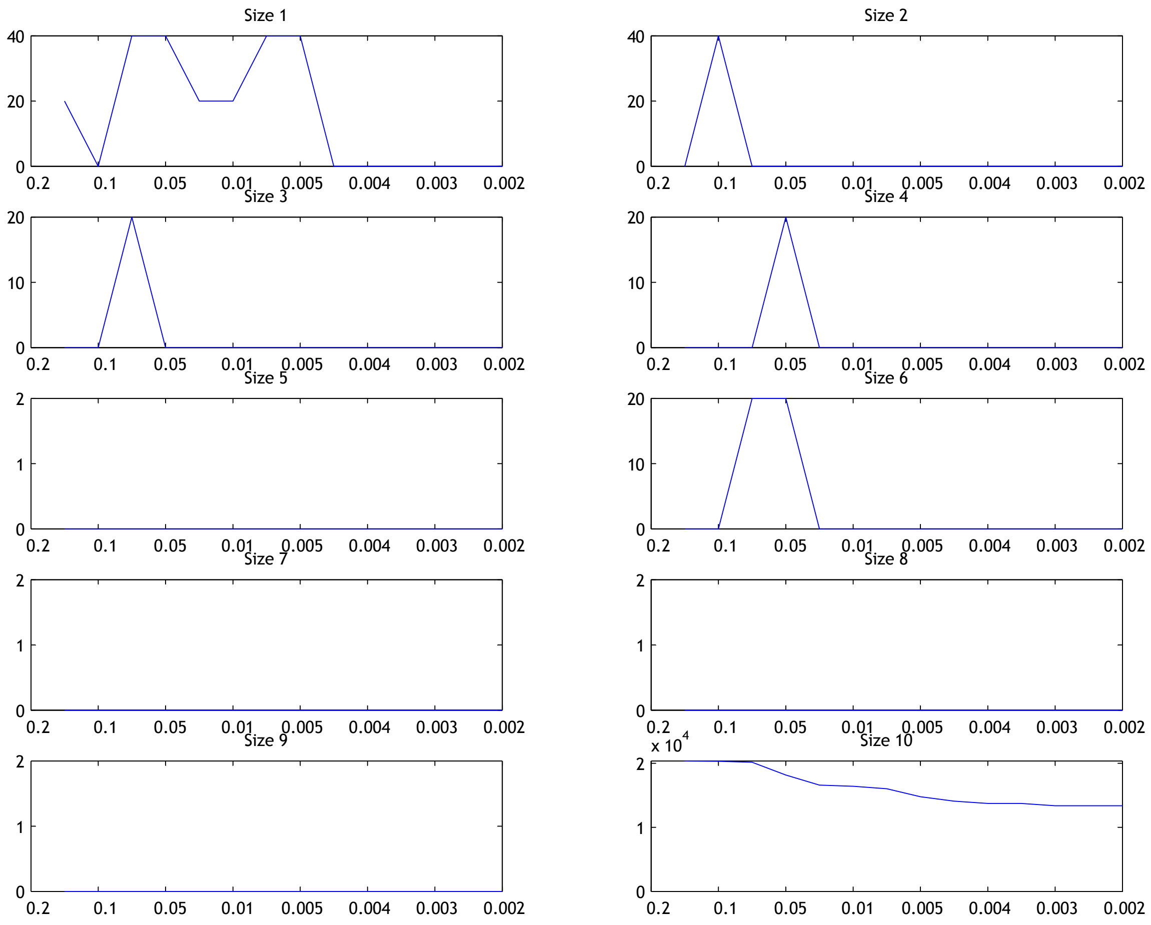


Figure 7.6: On the x-axis we have the p-values (from less significant to more significant) and on the y-axis we have the average clique sizes across 20 simulations of the matrix. Note how the cliques of size less than 10 go quickly to zero as the p-values become significant and how the cliques of maximum allowed size stabilise at a relatively high level. This is the same phenomenon discussed in Chapter 6, where a model that is denser than the maximum clique size leads to an exclusive representation made of cliques of maximum size.

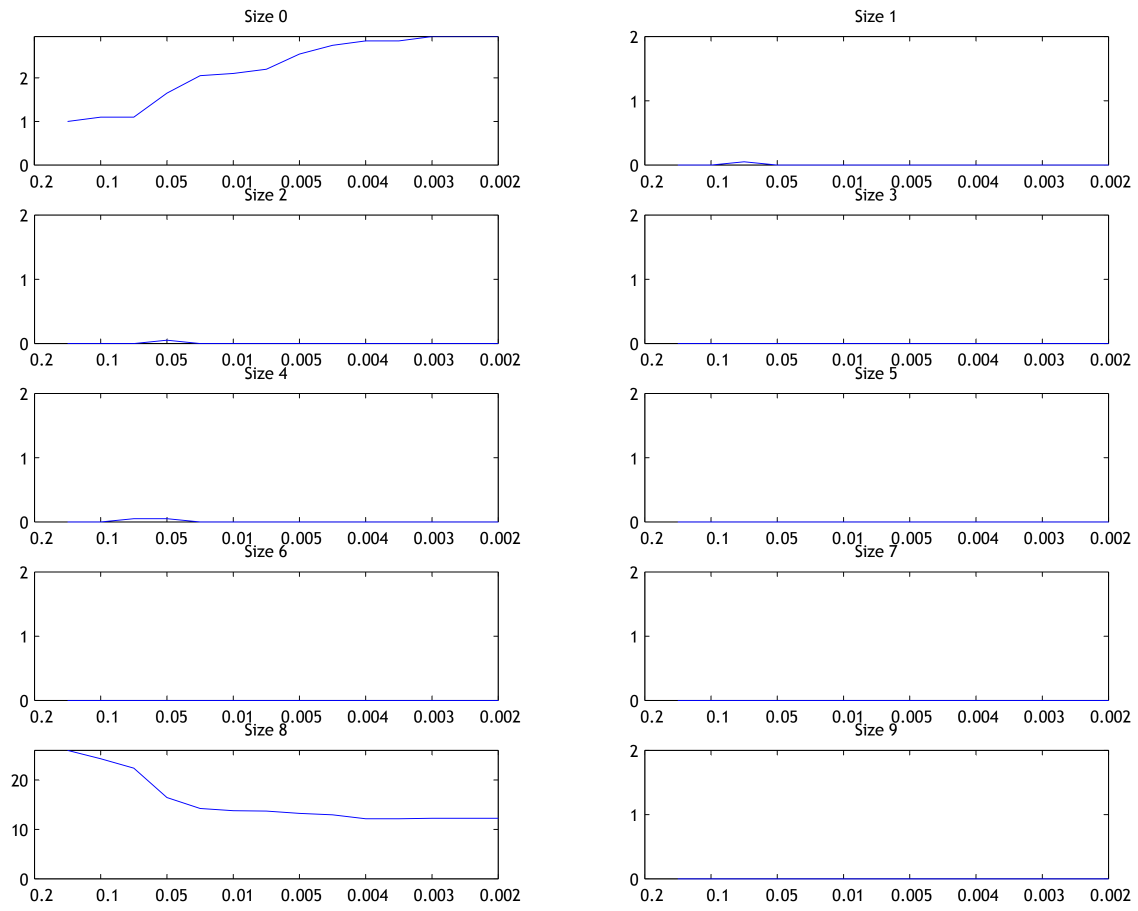


Figure 7.7: On the x-axis we have the p-values (from less significant to more significant) and on the y-axis we have the average Betti numbers across 20 simulations of the matrix. Note how the 0-th Betti number goes to three (the number of independent cliques) while all the others, excluding the one of maximum allowed size, go to zero.

7.1.3.3 Structure of factor models

In our study of the factor models in Chapter 6 we have observed that there is not a significant difference between the clique structure of the clique forests built by the MFCF when the underlying factor is a factor model, irrespective of the number of factors. This has been confirmed also in Examples 13 and 14. In this section we introduce a new gain function that allows to perform an analysis of the number of factors in a factor model. The key change is a shift in focus from the study of dependence to the study of independence. This approach is somewhat reminiscent of some of the techniques used in Independent Factor Analysis (Comon, 1994; Bach and Jordan, 2002; Hyvärinen and Oja, 2000) where the identification of independent factors requires the minimisation of mutual information (or of other related quantities such as *negentropy* as described for example in Hyvärinen and Oja (2000)).

In all of the examples and applications discussed so far we have built cliques of variables that are highly dependent; in this section we invert the perspective and try to build cliques of variables that are as independent as possible.

In the linear case it is easy as we can analyse the correlation matrix of a clique of variables and state that they are reasonably independent if the determinant, restricted to the clique, is close to one. Similarly, we can assess whether a new variable can be added to a clique by examining the relative change of the determinant after the addition of the variable: if the determinant reduces below a given relative threshold we argue that the variable is dependent from the variables in the clique, otherwise we conclude that the new variable is independent from the remaining variables in the clique and we can add the variable to the clique. The reasoning behind this approach is that the MFCF will try to expand the clique in a direction that is orthogonal to the subspace spanned by the other variables in the cliques; when this expansion is no longer possible the clique is considered to be complete and the algorithm will move to a different clique. If the data is not sparse, that is all the variables have non-zero loadings on every factor, the next clique will also tend to expand to the maximum size allowed by the underlying factors. We investigate, therefore, whether the maximum dimension of a clique in a clique forest is a good indicator of the dimensionality of the underlying factor model. Further to that, the more cliques in the clique forest achieve the maximum dimension, the higher the possibility that the underlying factor model contains exactly as many factors as the maximum size of the cliques.

Example 15 (Maximum determinant gain function)**.** This gain function tries to maximise the minor (determinant of a minor matrix) of the correlation matrix made of the new vertex and the separators in a given clique. This is the opposite of what we have done so far: by maximising the direct determinant (as opposed to the determinant of the inverse) the function tries to pick up variables that are as independent as possible. As a validation we use a threshold and we stipulate that adding a new vertex should not cause the determinant to shrink more than a given quantity: for instance adding a vertex to the separator should not allow the determinant to decrease by more than a factor of 0.2. This validation is aimed at preventing the inclusion of a variable that is close to a linear combination of the others added so far.

In this case we reverse what we have done in Chapter 3 (Equation 3.16). and define the gain function with the opposite signs. We look for the variable with the maximum KL-divergence (minimal mutual information) from the separators[4].

$$G(\tilde{c},c) = \frac{1}{2}\left(-\ln|J_{\tilde{c}}| + \ln|J_c|\right) = \frac{1}{2}\left(+\ln|\Sigma_{\tilde{c}}| - \ln|\Sigma_c|\right) \tag{7.11}$$

Remark 86 (Role of separators)**.** In this example not all the information related to the geometry of the clique forest is readily interpretable; for instance the separators do not seem to carry a particular meaning, but the study of the underlying probability distribution, defined by the Lauritzen formula, could be a topic for future research.

Remark 87 (Application of ICA estimations)**.** One question that is clearly open is whether a clique of maximal size can provide a representation of the latent factors. It is certainly possible to apply known ICA estimation methods (Hyvärinen and Oja, 2000) to the space spanned by a maximal clique to see if it is possible to reproduce the latent factors. It is a known result (Hyvärinen and Oja, 2000) that when the factors are normally distributed the result is known up to a rotation of the factors. Interestingly, when the factors are not normally distributed, the results can be identified much better.

We have tested this idea by generating 20 factor models of 50 variables and 25 factors. The factors are normally distributed and the factor loading are randomly assigned

[4]As in Chapter 3, $\tilde{c}$ is the clique after the expansion of the clique c.

to the variables. Initially we have generated time series with 500 data points: this setting allows to compare the results of our algorithm with with commonly used heuristics such as the analysis of the eigenvalues of the correlation matrix. For every simulation we generate a clique forest for a range of parameters (relative decrease int determinant value) and we collect the statistic of the cliques. In this Section we have limited our analysis to the statistic of clique sizes because the calculation of Betti numbers for simplices of dimension 25 would have been prohibitive. The results are shown in Figure 7.8: both our method and the heuristic based on the eigenvalues indicate that 25 is the most likely number of factors. This is shown by the brightest spot in the heatmap in the top panel, and by the sharp drop in the bottom panel.

As a next experiment, and to show how our method is preferable in some cases, we have generated 20 factor models with the same parameters, excepting for the length of the time series, which we have set to 50. The results are shown in Figure 7.9. In this case it is impossible to use the correlation matrix for eigenvalue analysis, as it is almost surely not positive definite and poorly conditioned, but our method still shows the brightest spot in the region of 24 and 23 factors, with a rather sharp distribution, which is a very good approximation to the correct value 25.

Remark 88 (Extension to other distributions)**.** We have just demonstrated a possible benefit of the use of the MFCF in the individuation of the number of factors. As the idea is based on the notion of "independence" of random variables, it is almost immediate to generalise our proposed methodology to other distributions, whenever a statistical test of multivariate independence is available. For instance in the case of multivariate distributions a χ^2 could be used, instead of testing the determinant, to identify the cliques of independent variables. This would open the possibility of testing the number of factors for multivariate discrete distributions. We observe, as an hint to further research, that the extension of kernel-based methods to ICA is already established (Bach and Jordan, 2002).

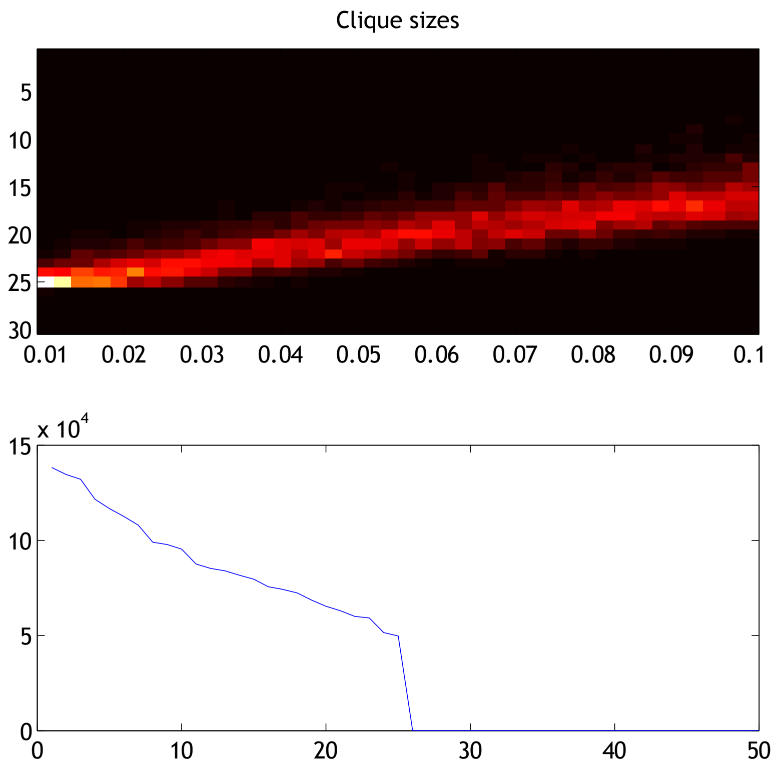


Figure 7.8: The top panel shows in the x-axis the relative decrease in the value of the clique determinant as described in Example 15; this value is used as an adjustable parameters to produce a quasi-filtration. In the y-axis we show the dimension of the cliques produced in the simulation. The color is a heatmap where white id the hottest spot (many cliques) and black is the coldest. The bottom panel shows the ordered eigenvalues of the correlation matrix of one of the 20 factor models. It is plain to see that the eigenvalues drop almost to zero between 25 and 26.

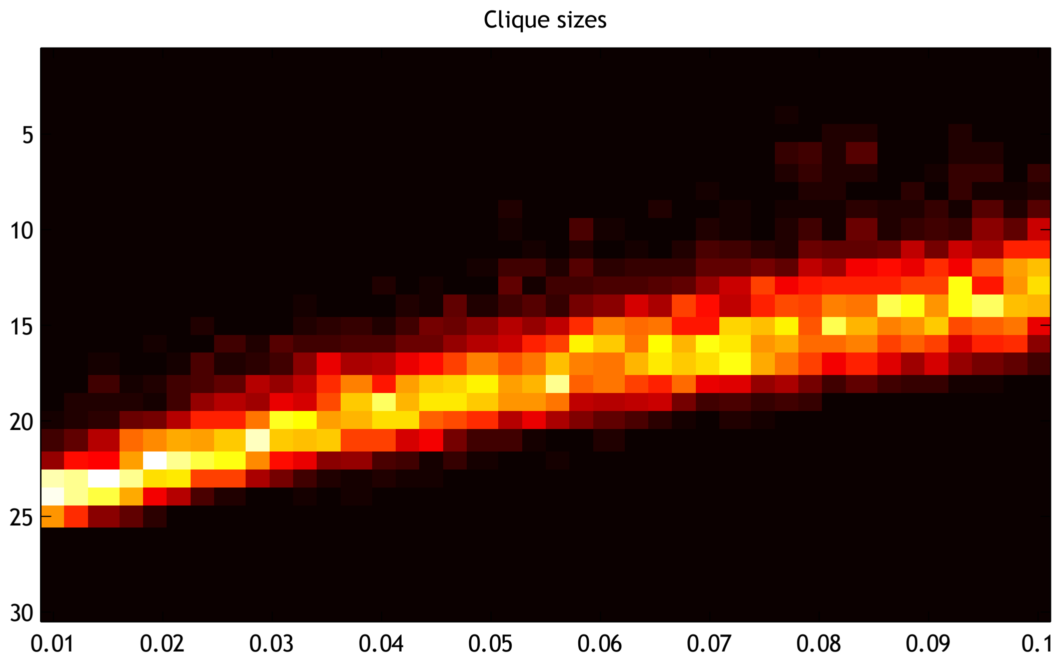


Figure 7.9: The highest Betti number for a single factor model is higher than the highest Betti number for a multifactor model. This figure shows the statistics of 20 simulations, where every simulation consists of the generation and analysis of factor models of dimension 1 to p, the number of variables (in this case $p = 50$).

Chapter 8

Conclusions and further research

8.1 Conclusions

In this thesis we have presented the case for building sparse networks using topological operators. In the approach we have presented, sparsity is enforced in the network by means of a *topological penalisation*, whereby the allowed configurations of the network are limited by constraints on the topology of the connections. In this thesis we have mainly examined a variant of the preferential attachment scheme; vertices are added one by one and are connected to the network through a subset of the existing nodes, the attachment points. We have shown how it is possible to associate a score function, and the related gain function, to the interaction between the new vertices and the attachment points so that it is possible to apply a greedy approach to the optimisation of the total score. We have argued for the use of well defined topological operators that preserve the desired topological properties so that said properties are invariant with respect to the attachment process. We have described in detail two families of topologically invariant operators: CT-invariant operators and planarity-invariant operators. A second mean to introduce sparsity in the networks produced is to add a validation step in the gain functions: only the operations that pass the validation are considered for network expansion.

Motivated by the theory of decomposable graphical models and by the convenient computational properties of chordal graphs, we have designed a generic algorithm, the MFCF, that uses our preferential attachment scheme in building clique forests by using CT-invariant operators. We have shown how the MFCF can be considered as a generalisation of two well-known algorithms: the Minimum Spanning Tree and the Maximum

Cardinality Search. The MFCF has been designed with flexibility in mind; the user can specify an initial clique forest to seed the algorithm, the topological constraints are enforced by defining the maximum and minimum size of the cliques, and the gain functions can be fully tailored to the problem at hand. Following an approach inspired by generic programming, we have specified the contract that any gain function must fulfil in order to be used with the MFCF; this provides a clear separation of concerns between the learning algorithm and the modelling of the gains, and allows a researcher to design new gain functions without any need to change the MFCF algorithm. The main application is the sparsification of networks from the point of view of structure learning. As a lead into further research, we have briefly studied some extensions that do not fit into the preferential attachment scheme, such as the direct join and the pruning of clique forests.

As a further contribution to the existing line of research into the planarization of networks, we have applied the MFCF to the Maximum Weight Planar Subgraph problem. We have studied variants of the algorithm that use planarity-invariant operators that are not necessarily CT-invariant and we have compared the result with the Planar Maximally Filtered Graph. We have studied a number of different cases and we have demonstrated a range of improvements with respect to the Planar Maximally Filtered Graph. While much of the geometric intuition behind the PMFG is still available, or even improved, in the networks produced by our proposed methodology, the probabilistic perspective brought to bear by the MFCF is a definite element of novelty.

We have applied the MFCF to the problem of Covariance Selection and we have compared our results with the state-of-the-art algorithm for such problem: the Graphical Lasso. We have shown that is many cases we obtain results that are comparable, or better, than the Graphical Lasso. We highlight how the purely geometrical approach adopted in the MFCF allows to decouple the inference of the structure and the estimation of the model parameters. We point out that the MFCF approach can be adopted in settings that are more general than the multivariate Gaussian setting which is the original environment where the Graphical Lasso was initially formulated. In particular we want to single out for future research the application of the methodology to general exponential family distributions, such as discrete and extreme values distributions.

We have found that the rich probabilistic structure allowed for by the language of graphical models, and in particular the tool-set of inference algorithms available for Gaussian Markov Random Fields lends itself very well to the modelling of sets of random variables such as the ones found in financial risk management and econometrics modelling. We have provided example applications to the conditional allocation of financial risk and to the extraction of networks with heterogeneous financial and econometrics time series.

Finally, we have looked at some examples where we show that the clique forests produced by the MFCF can be analysed with the tools developed by the Topological Data Analysis community and we have proposed a multi-scale analysis procedure for clique forests analogous to Persistent Homology. We highlight how much of information obtained by means of computational homology can be obtained more easily form the statistics of cliques and separators of the associated clique forest. We have designed a specific gain function to be applied to the study of the dimensionality of factor models and we have shown that it works well in the case of datasets with more variables than observations, where most classical methods do not work.

8.2 Further research

We have, hopefully, exposed the results of our research in a way that is logical and linear, but that does not reflect in the least the real historical development of the ideas contained in this work: many topics of great personal interest have been left behind because in some cases it would not have been possible to include all the material in a single organic exposition, lest it dilutes the exposition of the main subject. We would like, however, to go back ideally to the notes of these last few years to at least provide an idea of the developments that we had to set aside and that, in our view, constitute valid ideas for further research.

8.2.1 Non linear interactions and fat-tailed distributions

We would like to study the modelling of non-linear interactions by using new gain functions based on the MFCF. Some preliminary work has been done on kernels (Gretton et al., 2005; Shawe-Taylor and Cristianini, 2004; Bach and Jordan, 2003). We would also like to extend the study to distance-correlation (Székely and Rizzo, 2013) and to fat tailed distibutions such as the Student-t and in general to elliptically distributionsAnderson (1962). The main challenge in this field is to develop a coherent probabilistic theory which reuses as much as possible the theory of graphical models for exponential families. Especially for the student-t distributions, this might be possible through the use of non-extensive entropy theories.

8.2.2 Financial applications

The inclusion of non-linear and fat-tailed interactions would prove extremely useful in the development of financial risk management applications. If the formulation in terms of decomposable graphical models were to be successful, the use of the junction tree algorithm would become the work horse for all sorts of financial applications, allowing to extend and generalise the modern portfolio theory of Markowitz beyond multivariate normal interactions in a coherent and computationally practical way. In our working experience we have witnessed the effectiveness of saddle-point approximations in risk allocation and importance sampling: the dependence setting for this development is still based on multi-factor models: we would like to extend the theory to general decomposable models.

8.2.3 Geometry of clique forests

With the last chapter we have barely scratched the surface of the wealth of tools that are available in the fields of computational topology and discrete differential geometry. We believe that there are at least two paths worth investigating: one is the study of the differential geometry of clique forests to see if the discrete extensions of concepts like curvature and differential forms can help in understanding the structure of the data. A second path is the application of discrete Morse theory to the study of changes in correlation regimes though the changes over time of critical points.

8.2.4 Confirmatory factor analysis

An important feature of the MFCF is the possibility to specify an initial clique forest, maybe generated from previous experiments or from expert judgement, that can be grown by adding further vertices. This feature can be used to test hypotheses, such as in the psychological theory of personality, that can be formulated in terms of clique forest reflecting the assumed closed associations between factors.

8.2.5 Geometry of factor models

The study of the geometry of independent cliques carried out in the last chapter opens the possibility to the study of the dimensionality of factor models and, possibly, to the identification of latent factors. The use of ICA methodology on subspaces identified by the cliques seems to be a promising field of research.

Chapter 9

Appendix A

9.1 Computer Codes

9.1.1 TMFG

```
function [ P, cliques, separators, rev_peo ] = TMFG(W)
% Basic TMFG Algorithm, uses only $T_2$, builds a clique tree (chordal graph).
% The gain function is simply the sum of the new edges introduced by the $T_2$
%
% INPUT:
% - $W$: a weighted network
%
% OUTPUT:
% - $P$: the filtered TMFG graph
% - $cliques$: the list of 4-cliques
% - $separators$: the list of 3-cliques that are clique separators
% - $rev\_peo$: reverse of perfect elimination ordering

n = size(W,1);    % $n = |V|$
P = sparse(n,n); % sparse matrix
max_clique_gains = zeros(3*n - 6, 1);
best_vertex = zeros(3*n - 6, 1);

% $cliques$ contains the $K_4$ cliques of the clique forest,
% $separators$ contains the triangles that are \glcf{\textbf{not}} triangular faces,
% $triangles$ contains the triangular faces of the planar graph.
% At any given time $triangles$ constitutes a basis for the cycle
% space of the planar graph.
cliques = [];     % $cliques \leftarrow \varnothing$
separators = []; % $separators \leftarrow \varnothing$
triangles = [];  % $triangles \leftarrow \varnothing$

% Get first simplex using $max\_clique$
% $cliques \leftarrow max\_clique$
% $vertex\_list \leftarrow vertex\_list \setminus vertices(max\_clique)$
% $t_1, t_2, t_3, t_4 = faces(max\_clique)$
% $triangles \leftarrow t_1 \cup t_2 \cup t_3 \cup t_4$
cliques(1, :) = max_clique(W);
vertex_list = setdiff(1:n, cliques(1,:));
triangles(1,:) = cliques(1, [1 2 3]);
triangles(2,:) = cliques(1, [1 2 4]);
triangles(3,:) = cliques(1, [1 3 4]);
triangles(4,:) = cliques(1, [2 3 4]);

% $rev\_peo \leftarrow vertices(max\_clique)$
rev_peo = cliques(1, :);

W(1:(n+1):n^2) = 0; % $\operatorname{diag} W = 0$

% add the edges between the vertices to the filtered matrix
```

```
P(rev_peo, rev_peo) = W(rev_peo, rev_peo);

% $\forall t \in \{t_1,t_2,t_3,t_4\}$:
%       choose $v \in vertex\_list$ with best gain for $t$ and corresponding gain
for t = 1:4
    [max_clique_gains(t) best_vertex(t)] = ...
        get_best_gain(vertex_list, triangles(t,:), W);
end

% there are $n-4$ outstanding vertices to be added
for i = 1:(n-4)
    % get maximum local gain and corresponding triangular face $nt$ and vertex $nv$
    [~, nt] = max(max_clique_gains);
    nv = best_vertex(nt);

    % $rev\_peo \leftarrow concatenate(rev\_peo, nv)$
    rev_peo(end + 1) = nv;

    % $new\_clique \leftarrow nt \cup nv$
    % $cliques \leftarrow cliques \cup new\_clique$
    cliques(end+1, :) = [nv triangles(nt,:)];

    % $nt$ becomes a separator
    newsep = triangles(nt, :);

    % add edges in $new\_clique$ to $P$
    P([nv newsep], [nv newsep]) = W([nv newsep], [nv newsep]);

    % $separators \leftarrow separators \cup nt$
    separators(end+1, :) = newsep;

    % here we are applying the $T_2$ operator
    % $t_a, t_b, t_c \leftarrow T_2(nt, nv)$
    % $nt$ disappears from the triangular basis, replaced by one
    % of the three new triangles ...
    % $triangles \leftarrow triangles \setminus nt$
    % $triangles \leftarrow triangles \cup t_a \cup t_b \cup t_c$
    triangles(nt, :) = [newsep(1) newsep(2) nv];
    triangles(end+1, :) = [newsep(1) newsep(3) nv];
    triangles(end+1, :) = [newsep(2) newsep(3) nv];

    % $vertex\_list \leftarrow vertex\_list \setminus vertices(new\_clique)$
    vertex_list = setdiff(vertex_list, nv);

    % update the gain table by updating the gains corresponding to the now
    % unfeasible vertex $nv$
    if length(vertex_list) > 0
        for t = find(best_vertex == nv).'
            [max_clique_gains(t) best_vertex(t)] = ...
                get_best_gain(vertex_list, triangles(t,:), W);
        end
    end

    % update the gain table by calculating the gains for $t_a, t_b, t_c$
    max_clique_gains(nt) = 0;
    ct = size(triangles, 1);
    if length(vertex_list) > 0
        for t = [nt (ct-1) ct]
            [max_clique_gains(t) best_vertex(t)] = ...
                get_best_gain(vertex_list, triangles(t,:), W);
        end
    end
end

end

function [gain vertex] = get_best_gain(vertex_list, triangle, W)
% the gain function in this case is the sum of the edges introduced by
% the $T_2$.
% it cas also be seen as the sum of the edges of the new clique minus the
% sum of edges of the separator..
```

```
    gvec(vertex_list) = W(vertex_list, triangle(1)) + W(vertex_list, triangle(2)) + W(
    vertex_list, triangle(3));
    [gain vertex] = max(gvec);
end

function cl = max_clique(W)
% return the first four vertices in order of sum where the weights
% exceed the mean of the weight matrix.
    v = sum(W.*(W>mean(W(:))),2);
    [~, sortindex] = sort(v, 'descend');
    cl = sortindex(1:4);
end
```

./Code/TMFG_matlab/TMFG.m

9.1.2 TMFG-T1

```
function [ P, cliques, triangles, tc  ] = TMFGT1(W)
% TMFG variant with use of $T_2$ and $T_1$.
% It first evaluates the best $T_2$ move, and after that examines
% any possible additional gain to be achieved using the $T_1$
% operator on the newly introduced triangular faces.
% The output is not chordal, hence the algorithm returns the
% triangular faces, instead of the separators, and does not return a PEO.
% It also returns a sparse matrix $tc$ containing the triangular
% faces contact structure, recording the triangles that share an edge.
% The list of cliques only reflects the cliques created with the $T_2$
% operator, but since the graph is not chordal, it does not represent the
% clique structure of the filtered matrix after the $T_1$ operator
% is applied.
%
% INPUT:
% - $W$: a weighted network
%
% OUTPUT:
% - $P$: the filtered TMFG graph
% - $cliques$: the list of 4-cliques produced by the $T_2$
% - $triangles$: the list of triangular faces
% - $tc$: triangles contact structure

n = size(W,1);    % $n = |V|$
P = sparse(n,n); % sparse matrix
max_clique_gains = zeros(3*n - 6, 1);
best_vertex      = zeros(3*n - 6, 1);
tc = sparse(n,n); % holds the contact structure between triangular faces

% $cliques$ contains the $K_4$ cliques of the clique forest,
% $triangles$ contains the triangular faces of the planar graph.
% At any given time $triangles$ constitutes a basis for the cycle
% space of the planar graph.
cliques = [];    % $cliques \leftarrow \varnothing$
triangles = []; %v*\color{mygreen}$triangles \leftarrow \varnothing$*)

% Get first simplex using $max\_clique$
% $cliques \leftarrow max\_clique$
% $vertex\_list \leftarrow vertex\_list \setminus vertices(max\_clique)$
% $t_1, t_2, t_3, t_4 = faces(max\_clique)$
% $triangles \leftarrow t_1 \cup t_2 \cup t_3 \cup t_4$
cliques(1, :) = max_clique(W);
vertex_list = setdiff(1:n, cliques(1,:));
triangles(1,:) = cliques(1, [1 2 3]);
triangles(2,:) = cliques(1, [1 2 4]);
triangles(3,:) = cliques(1, [1 3 4]);
triangles(4,:) = cliques(1, [2 3 4]);

% Populate the contact structure. Since they are part of a 4-clique all the
% triangles share an edge. The $tc$ structure is indexed by the
% identifiers of the triangles.
% $tc_{12}, tc_{13}, tc_{1,4}, tc_{23}, tc_{34} \leftarrow 1$
tc([1 2 3 4], [1 2 3 4]) = 1;
tc(1:(n+1):n^2) = 0;

% $startv \leftarrow vertices(max\_clique)$
startv = cliques(1, :);
W(1:(n+1):n^2) = 0; % $\mathrm{diag}\,W = 0$

% add the edges between the vertices to the filtered matrix
P(startv, startv) = W(startv, startv);

% $\forall t \in \{t_1, t_2, t_3, t_4\}$ :
%        choose $v \in vertex\_list$ with best gain for $t$ and corresponding gain
for t = 1:4
    [max_clique_gains(t) best_vertex(t)] = get_best_gain(vertex_list, triangles(t,:),
    W);
end

% there are $n-4$ outstanding vertices to be added
for i = 1:(n-4)

    % get maximum local gain and corresponding triangular face $nt$ and vertex $nv$
    [~, nt] = max(max_clique_gains);
    nv = best_vertex(nt);

    % $new\_clique \leftarrow nt \cup nv$
    % $cliques \leftarrow cliques \cup new\_clique$
    cliques(end+1, :) = [nv triangles(nt,:)];

    % Use the triangular face as a temporary separator to build the new
    % triangular faces.
    newsep = triangles(nt, :);

    % add edges in $new\_clique$ to $P$
    P([nv newsep], [nv newsep]) = W([nv newsep], [nv newsep]);

    % here we are applying the $T_2$ operator
    % $t_a, t_b, t_c \leftarrow T_2(nt, nv)$
    % $nt$ disappears from the triangular basis, replaced by one
    % of the three new triangles ...
    % $triangles \leftarrow triangles \setminus nt$
    % $triangles \leftarrow triangles \cup t_a \cup t_b \cup t_c$
    triangles(nt, :) = [newsep(1) newsep(2) nv];
    triangles(end+1, :) = [newsep(1) newsep(3) nv];
    triangles(end+1, :) = [newsep(2) newsep(3) nv];

    % $vertex\_list \leftarrow vertex\_list \setminus vertices(new\_clique)$
    vertex_list = setdiff(vertex_list, nv);

    % update the gain table by updating the gains corresponding to the now
    % unfeasible vertex $nv$
    if length(vertex_list) > 0
        for t = find(best_vertex == nv).'
            [max_clique_gains(t) best_vertex(t)] = ...
                get_best_gain(vertex_list, triangles(t,:), W);
        end
    end

    % start assessing if there are possible gains from $T_1$
    max_clique_gains(nt) = 0;

    % find in $tc$ any neighbour of $nt$
    possible_neighbours = find(tc(nt,:));

    % since $nt$ goes out of the basis (it is no longer a
    % triangular face) we have to remove it from the contact structure.
    tc(nt, :) = 0; tc(:,nt) = 0;
    ct = size(triangles, 1);

    % update the gain table by calculating the gains for $t_a, t_b, t_c$
    if length(vertex_list) > 0
        for t = [nt (ct-1) ct]
            [max_clique_gains(t) best_vertex(t)] = ...
                get_best_gain(vertex_list, triangles(t,:), W);
        end
    end

    % $t_a$,$t_b$,$t_c$ share an edge and hence they need to be
    % marked as adiacent in the contact structure.
    % $tc_{ab}$,$tc_{ac}$,$tc_{bc} \leftarrow 1$
    new_triangles = [nt (ct-1) ct];
    tc(new_triangles, new_triangles) = 1;
    tc(new_triangles(1), new_triangles(1)) = 0;
    tc(new_triangles(2), new_triangles(2)) = 0;
    tc(new_triangles(3), new_triangles(3)) = 0;

    triangles_to_update = new_triangles;

    % for all the new triangles and all the possible neighbours of the new
    % triangles, check if a $T_1$ is convenient and perform it.
    % the function% $flip$ is the implem,entation of the $T_1$
    % and updates the filtered matrix and the set of triangular faces.
    for t1 = possible_neighbours
        for t2 = new_triangles
            tc(t1, t2) = is_neighbour(triangles, t1, t2);
            tc(t2, t1) = tc(t1, t2);
            if tc(t1, t2)
                [triangles P flipped] =  flip(t1, t2, triangles, W, P);
                % if $T_1$ executed add the triangles to the set
                % to be updated
                if flipped
                    triangles_to_update = unique([triangles_to_update, t1]);
                end
            end
        end
    end

    % update the gain table by calculating the gains for $t_a$,$t_b$,$t_c$
    if ~isempty(vertex_list)
        for t = triangles_to_update
            [max_clique_gains(t) best_vertex(t)] = get_best_gain(vertex_list,
    triangles(t,:), W);
        end
    end

end

end

function [gain vertex] = get_best_gain(vertex_list, triangle, W)
    gvec(vertex_list) = W(vertex_list, triangle(1)) + W(vertex_list, triangle(2)) + W(
    vertex_list, triangle(3));
    [gain vertex] = max(gvec);
end

function cl = max_clique(W)
% return the first four vertices in order of sum where the weights
% exceed the mean of the weight matrix.
    v = sum(W.*(W>mean(W(:))),2);
    [~, sortindex] = sort(v, 'descend');
    cl = sortindex(1:4);
end

function [triangles_out P_out flipped] = flip(t1, t2, triangles, W, P)
    flipped = false;
    C = intersect(triangles(t1, :), triangles(t2, :));
    v1 = C(1); v3 = C(2);
    v2 = setdiff(triangles(t1, :), [v1 v3]);
    v4 = setdiff(triangles(t2, :), [v1 v3]);
    if (~isempty(v2) && ~isempty(v4) && ...
        ~isempty(v1) && ~isempty(v3) && ...
        (P(v2, v4) + 0) == 0 && (W(v2, v4) > W(v1, v3)))
        triangles_out = triangles;
        triangles_out(t1, :) = [v1 v2 v4];
        triangles_out(t2, :) = [v2 v3 v4];
        P_out = P;
        P_out(v2, v4) = W(v2, v4);
        P_out(v4, v2) = W(v4, v2);
        P_out(v1, v3) = 0;
        P_out(v3, v1) = 0;
        flipped = true;
    else
        P_out = P;
        triangles_out = triangles;
    end
end

% support function returns true is two triangles are neighbours, i.e. share
% an edge.
function isn = is_neighbour(triangles, t1, t2)
    cnt = length(intersect(triangles(t1, :), triangles(t2, :)));
    if cnt == 2 ; isn = true; else isn = 0 ;end
end
```

./Code/TMFG_matlab/TMFGT1.m

9.1.3 TMFG-S

```
function [P, cliques, separators, rev_peo] = TMFGS(W)
% Optimised TMFG Algorithm, uses T2 and S.
% builds a clique tree (chordal graph). Initially builds a clique tree by
% introducing simplicial vertices using T2 and then examinse
% the most convenient swap (S) operator among the vertices of
% the new clique.
% The gain function is simply the sum of the new edges introduced by the T2
%
% INPUT:
% - W: a weighted network
%
% OUTPUT:
% - P: the filtered TMFG graph
% - cliques: the list of 4-cliques
% - separators: the list of 3-cliques that are clique separators
% - rev_peo: reverse of perfect elimination ordering
%

INITIAL_TRIANGLES = 4;

n = size(W,1);    % n = |V|
P = sparse(n,n); % sparse output matrix
max_clique_gains = zeros(3*n - 6, 1);
best_vertex = zeros(3*n - 6, 1);

% cliques contains the K4 cliques of the clique forest,
% separators contains the triangles that are \glcf{\textbf{not}} triangular faces,
% triangles contains the triangular faces of the planar graph.
% At any given time triangles constitutes a basis for the cycle
% space of the planar graph.
cliques = [];     % cliques ← ∅
separators = []; % separators ← ∅
triangles = [];  % triangles ← ∅

% Get first simplex using max_clique
% cliques ← max_clique
% vertex_list ← vertex_list \ vertices(max_clique)
% t1,t2,t3,t4 = faces(max_clique)
% triangles ← t1 ∪ t2 ∪ t3 ∪ t4
cliques(1, :) = max_clique(W);
vertex_list = setdiff(1:n, cliques(1,:));
triangles(1,:) = cliques(1, [1 2 3]);
triangles(2,:) = cliques(1, [1 2 4]);
triangles(3,:) = cliques(1, [1 3 4]);
triangles(4,:) = cliques(1, [2 3 4]);

% rev_peo ← vertices(max_clique)
rev_peo = cliques(1, :);
```

```
W(1:(n+1):n^2) = 0; % diag W = 0

% add the edges between the vertices to the filtered matrix
P(rev_peo, rev_peo) = W(rev_peo, rev_peo);

% ∀t ∈ {t1,t2,t3,t4}:
%        choose v ∈ vertex_list with best gain for t and corresponding gain
for t = 1:INITIAL_TRIANGLES
    [max_clique_gains(t) best_vertex(t)] = get_best_gain(vertex_list, triangles(t,:),
    W);
end

% there are n−4 outstanding vertices to be added
for i = (INITIAL_TRIANGLES+1):n

    % get maximum local gain and corresponding triangular face nt and vertex nv
    [~, nt] = max(max_clique_gains);
    nv = best_vertex(nt);

    % rev_peo ← concatenate(rev_peo, nv)
    rev_peo(end + 1) = nv;

    % new_clique ← nt ∪ nv
    % cliques ← cliques ∪ new_clique
    this_clique = [triangles(nt,:) nv];

    % Find locally optimal orientation of the new clique
    % First of all let's find the vertices in P that are connected to the
    % clique, excluding the vertices belonging to the triangle being
    % extended.
    tmp = find(P(triangles(nt,1), :) ~= 0);
    tmp = tmp(tmp ~= triangles(nt,1) & tmp ~= triangles(nt,2) & tmp ~= triangles(nt,3)
    );
    neighbours_1 = tmp; % neighbours of vertex 1
    tmp = find(P(triangles(nt,2), :) ~= 0);
    tmp = tmp(tmp ~= triangles(nt,1) & tmp ~= triangles(nt,2) & tmp ~= triangles(nt,3)
    );
    neighbours_2 = tmp; % neighbours of vertex 2
    tmp = find(P(triangles(nt,3), :) ~= 0);
    tmp = tmp(tmp ~= triangles(nt,1) & tmp ~= triangles(nt,2) & tmp ~= triangles(nt,3)
    );
    neighbours_3 = tmp; % neighbours of vertex 3

    % Now let's set to zero the values in the filtered matrix, next we
    % identify the best combination and restore the links.
    P(neighbours_1, triangles(nt,1)) = 0;
    P(neighbours_2, triangles(nt,2)) = 0;
    P(neighbours_3, triangles(nt,3)) = 0;
    P(triangles(nt,1), neighbours_1) = 0;
    P(triangles(nt,2), neighbours_2) = 0;
    P(triangles(nt,3), neighbours_3) = 0;
    P(triangles(nt,:), triangles(nt,:)) = 0;

    local_max = 0.0;
    newsep = triangles(nt,:).';
    last_added_v = nv;

    % evaluate the gain function for every swap of the veritices of the
    % clique.
    for perm_clique = perms(this_clique).'
        % Assume that the last vertex is the internal one and the external
        % triangle vertices are the first three.
        int_vertex = perm_clique(4);
        ext_triangle = perm_clique(1:3);
        running_max = sum(W(neighbours_1, ext_triangle(1))) + ...
                      sum(W(neighbours_2, ext_triangle(2))) + ...
                      sum(W(neighbours_3, ext_triangle(3)));
        if running_max > local_max
            local_max = running_max;
            nv = int_vertex;
            newsep = ext_triangle;
        end
    end

    % $new\_clique \leftarrow nt \cup nv$
    % $cliques \leftarrow cliques \cup new\_clique$
    cliques(end + 1, :) = [newsep' nv];

    % Update the filtered matrix
    P(neighbours_1, newsep(1)) = W(neighbours_1, newsep(1));
    P(newsep(1), neighbours_1) = W(newsep(1), neighbours_1);
    P(neighbours_2, newsep(2)) = W(neighbours_2, newsep(2));
    P(newsep(2), neighbours_2) = W(newsep(2), neighbours_2);
    P(neighbours_3, newsep(3)) = W(neighbours_3, newsep(3));
    P(newsep(3), neighbours_3) = W(newsep(3), neighbours_3);
    P([nv; newsep], [nv; newsep]) = W([nv; newsep], [nv; newsep]);

    % $separators \leftarrow separators \cup nt$
    separators(end+1, :) = newsep;

    % replace triangles where some of the vertices were changed
    new_clique = [newsep' nv];
    triangles_updated = triangles;
    changed_triangles = [];
    for iv = 1:length(new_clique)
        if (new_clique(iv) ~= this_clique(iv))
            idx_t = find(triangles == this_clique(iv));
            triangles_updated(idx_t) = new_clique(iv);
            [tt ~] = ind2sub(size(triangles), idx_t);
            changed_triangles = unique([changed_triangles' tt'].');
        end
    end

    triangles = triangles_updated;

    % here we are applying the $T_2$ operator
    % $t_a, t_b, t_c \leftarrow T_2(nt, nv)$
    % $nt$ disappears from the triangular basis, replaced by one
    % of the three new triangles ...
    % $triangles \leftarrow triangles \setminus nt$
    % $triangles \leftarrow triangles \cup t_a \cup t_b \cup t_c$
    triangles(nt, :) = [newsep(1) newsep(2) nv];
    triangles(end+1, :) = [newsep(1) newsep(3) nv];
    triangles(end+1, :) = [newsep(2) newsep(3) nv];



    % $vertex\_list \leftarrow vertex\_list \setminus vertices(new\_clique)$
    vertex_list = setdiff(vertex_list, last_added_v);

    % update the gain table by updating the gains corresponding to the now
    % unfeasible vertex $nv$
    if ~isempty(vertex_list)
        for t = find(best_vertex == last_added_v).'
            [max_clique_gains(t) best_vertex(t)] = get_best_gain(vertex_list,
    triangles(t,:), W);
        end
    end

    % update the gain table by calculating the gains for $t_a, t_b, t_c$
    max_clique_gains(nt) = 0.0;
    best_vertex(nt) = 0;
    ct = size(triangles, 1);
    if ~isempty(vertex_list)
        for t = [nt (ct-1) ct]
            [max_clique_gains(t) best_vertex(t)] = get_best_gain(vertex_list,
    triangles(t,:), W);
        end
    end

    % update the gain table by calculating the gains for the traingles that
    % have been changed
    if ~isempty(vertex_list)
```

```
        for t = changed_triangles'
            if max_clique_gains(t) ~= 0
                [max_clique_gains(t) best_vertex(t)] = get_best_gain(vertex_list,
    triangles(t,:), W);
            end
        end
    end
end
end

function [gain vertex] = get_best_gain(vertex_list, triangle, W)
% the gain function in this case is the sum of the edges introduced by
% the $T_2$.
% it cas also be seen as the sum of the edges of the new clique minus the
% sum of edges of the separator.
    gvec(vertex_list) = W(vertex_list, triangle(1)) + W(vertex_list, triangle(2)) + W(
    vertex_list, triangle(3));
    [gain vertex] = max(gvec);
end

function cl = max_clique(W)
% return the first four vertices in order of sum where the weights
% exceed the mean of the weight matrix.
    v = sum(W.*(W>mean(W(:))),2);
    [~, sortindex] = sort(v, 'descend');
    cl = sortindex(1:4);
end
```

./Code/TMFG_matlab/TMFGS.m

9.1.4 TMFG-A

```
function [ P, triangles, tc  ] = TMFGA(w, num_swaps)
% A variant of the TMFG algorithm, characterised by two features:
% - uses three operators: $T_2$, $T_1$ and $A$.
% - It works on pairs of triangles $(t_1,t_2)$, rather than a single triangle
%   according to 5 possible operations:
%   1) $T_2$ on $t_1$
%   2) $T_2$ on $t_2$
%   3) $A$ on $t_1$ and $t_2$
%   4) $T_2$ on $t_1$ followed by $T_1$
%   5) $T_2$ on $t_2$ followed by $T_1$
% It does not build a chordal graph and therefore does not return cliques
% nor separators.
% Uses a sparse matrix to track the pairs of adjacent triangles
% upon which the operators act.
% The gain function is simply the sum of the new edges introduced by the
% chosen operation. The gain table in this case, differently from the
% traditional TMFG implementation has the cardinality of the pairs of
% adjacent triangles.
%
% INPUT:
% - $W$: a weighted network
% - $num\_swaps$: how many $T_1$ optimisation steps can
% be done
%
% OUTPUT:
% - $P$: the filtered TMFG graph
% - $triangles$: the list of triangular faces
% - $tc$: triangles contact structure

n = size(w,1);    % $n = |V|$
P = sparse(n,n); % sparse matrix

% the gain table now is indexed by the ids of two triangles. The entry is
% not empty only if the triangles are adjacent. Differently from the TMFG
% we have three structures: the conatiner of the gains, the container of
% the vertices that attain the best gains, and -new- the operation that
% attains the best gain.
max_tc_gains  = sparse(3*n,3*n); % gains
```

```
best_vertex    = sparse(3*n,3*n); % best vertices
best_op        = sparse(3*n,3*n); % best operation (1 to 5)

tc = sparse(false(3*n,3*n)); % holds the contact structure between triangular faces
triangles =  sparse(false(3*n, n)); % triangular faces
triangles_changed = []; % triangles that have been changed and require an update

% Get first simplex using $max\_clique$
% $cliques \leftarrow max\_clique$
% $vertex\_list \leftarrow vertex\_list \setminus vertices(max\_clique)$
% $t_1,t_2,t_3,t_4 = faces(max\_clique)$
% $triangles \leftarrow t_1 \cup t_2 \cup t_3 \cup t_4$
K_4 = max_clique(w);
vertex_list = setdiff(1:n, K_4);
triangles(K_4([1 2 3]), 1) = true;
triangles(K_4([1 2 4]), 2) = true;
triangles(K_4([1 3 4]), 3) = true;
triangles(K_4([2 3 4]), 4) = true;
% Flag triangles changed so that the gain is calculated for them
triangles_changed([1 2 3 4]) = 1;

% Populate the contact structure, all triangles share an edge
tc([1 2 3 4], [1 2 3 4]) = 1;
tc = tril(tc, -1);
w(1:(n+1):n^2) = 0;

P(K_4, K_4) = w(K_4, K_4);

% init gain matrix
% Lower triangular
% Loop over adjacent triangles and find best operation and best gain
% $\forall (t_a,t_b), t_a,t_b \in \{t_1,t_2,t_3,t_4\} t_a \neq t_b$ :
%        choose $v \in vertex\_list$ with best gain for $(t_a,t_b)$ and corresponding gain and operation
[row, col, ~] = find(tril(tc));
for idx = 1:numel(row)
    [max_tc_gains(row(idx), col(idx)) best_vertex(row(idx), col(idx)) best_op(row(idx)
    , col(idx))] = ...
            get_loz_gains(w, triangles, row(idx), col(idx), vertex_list, P);
end

% there are $n-4$ outstanding vertices to be added
for i = 1:(n-4)
     % get maximum local gain and corresponding plaquette $nt$ and vertex $nv$
    [~, nt] = max(max_tc_gains(:)); % this is a matrix maximum
    nv = best_vertex(nt); % nv is the vertex to insert
    op = best_op(nt); % op is the operation to apply

    % get triangles $nt = (t_a,t_b)$ that make up the plaquette
    [t1 t2] = ind2sub(size(max_tc_gains), nt);

    % apply the operation to the plaquette
    % this can change the structure of triangles and plaquettes
    [P triangles tc triangles_changed] = apply_op(op, w, nv, P, t1, t2, triangles, tc)
    ;

    % $vertex\_list \leftarrow vertex\_list \setminus vertices(new\_clique)$
    vertex_list = setdiff(vertex_list, nv);

    % delete gains where a plaquette was deleted
    max_tc_gains = max_tc_gains .* (tc ~= 0);
    best_vertex = best_vertex .* (tc ~= 0);
    best_op = best_op .* (tc ~= 0);

    % update max gains where the vertex nv was involved
    if ~isempty(vertex_list)
        [row col val] = find(best_vertex == nv);
        for idx = 1:numel(row)
            max_tc_gains(row(idx), col(idx))= 0;
            best_vertex(row(idx), col(idx))= 0;
            best_op(row(idx), col(idx))= 0;
            if tc(row(idx), col(idx)) == true
```

```
                [max_tc_gains(row(idx), col(idx)) best_vertex(row(idx), col(idx))
    best_op(row(idx), col(idx))] = ...
                    get_loz_gains(w, triangles, row(idx), col(idx), vertex_list, P);
            end
        end
        for it = 1:num_swaps
            [P triangles tc triangles_changed] = apply_swaps(w, P, triangles, tc);
        end

        % recalculate gains for plaquettes where triangles were changed
        [row col val] = find(tril(tc));
        for idx = 1:numel(row)
            if triangles_changed(row(idx)) || triangles_changed(col(idx)) ||
    max_tc_gains(row(idx), col(idx)) == 0
                [max_tc_gains(row(idx), col(idx)) best_vertex(row(idx), col(idx))
    best_op(row(idx), col(idx))] = ...
                    get_loz_gains(w, triangles, row(idx), col(idx), vertex_list, P);
                triangles_changed(row(idx)) = 0;
                triangles_changed(col(idx)) = 0;
            end
        end
    end
end

end

function cl = max_clique(W)
% return the first four vertices in order of sum where the weights
% exceed the mean of the weight matrix.
    v = sum(W.*(W>mean(W(:))),2);
    % v = sum(W .* W);
    [~, sortindex] = sort(v, 'descend');
    cl = sortindex(1:4);
end

% support function returns true is two triangles are neighbours, i.e. share
% an edge.
function isn = is_neighbour(triangles, t1, t2)
    cnt = length(find(triangles(:, t1) & triangles(:, t2)));
    if cnt == 2 ; isn = true; else isn = 0 ;end
end

function [loz_gain vertex op] =  get_loz_gains(w, triangles, t1, t2, vertex_list, P)
    gvec = zeros(numel(vertex_list), 6);

    N = tsetdiff(triangles, t1, t2);
    S = tsetdiff(triangles, t2, t1);

    tmp = tintersect(triangles, t1, t2);
    if (numel(tmp) ~= 2)
        fprintf('numel %d\n', numel(tmp));
    end
    W = tmp(1);
    E = tmp(2);
    % Different gain functions for the different operations. Below the
    % layout for the plaquette with N(orth), S(outh), W(est), E(ast) nodes.
    %                   N
    %                  / \
    %                 /   \
    %                W --- E
    %                 \   /
    %                  \ /
    %                   S
    % operation 1
    gvec(vertex_list, 1) = w(vertex_list, N) + w(vertex_list, W) + w(vertex_list, E);
    % operation 2
    gvec(vertex_list, 2) = w(vertex_list, N) + w(vertex_list, S) + w(vertex_list, E) +
     w(vertex_list, W) - w(W,E);
    % operation 3
    gvec(vertex_list, 3) = w(vertex_list, S) + w(vertex_list, W) + w(vertex_list, E);
    if (P(N,S) == 0)
        % operation 4
        gvec(vertex_list, 4) = w(vertex_list, N) + w(vertex_list, S) + w(vertex_list,
    W) + w(N,S) - w(W,E);
        % operation 5
        gvec(vertex_list, 5) = w(vertex_list, N) + w(vertex_list, S) + w(vertex_list,
    E) + w(N,S) - w(W,E);
    else
        % operation 4
        gvec(vertex_list, 4) = 0;
        % operation 5
        gvec(vertex_list, 5) = 0;
    end

    idx = find(gvec == max(max(gvec)));
    loz_gain = gvec(idx);
    [vertex op] = ind2sub(size(gvec), idx);
end

function [P triangles tc triangles_changed] = apply_op(op, w, nv, P, t1, t2, triangles
    , tc)
    triangles_changed = [];
    N = tsetdiff(triangles, t1, t2);
    S = tsetdiff(triangles, t2, t1);
    tmp = tintersect(triangles, t1, t2);
    if (numel(tmp) <2)
        fprintf('hhh\n')
    end
    W = tmp(1);
    E = tmp(2);
    C = nv;
    switch op
        case 1
            % adjust triangles
            triangles( :, t1) = false;
            triangles([W E C], t1) = true;
            triangles_changed(t1) = 1;
            [~, y] = find(triangles) ; t3 = max(y) +1;
            t4 = t3 + 1;
            triangles( :, t3) = false;
            triangles([W C N], t3) = true;
            triangles_changed(t3) = 1;
            triangles( :, t4) = false;
            triangles([E C N], t4) = true;
            triangles_changed(t4) = 1;
            % adjust tc externally
            %   find triangles bordering with t1
            for tn = find_neib(tc, t1)
                if tn == t2
                    % do nothing because t1 and t2 are neighbours anyway by
                    % construction
                end
                if is_neighbour(triangles, tn, t3)
                    tc = connecttril(tc, tn, t3);
                    tc(max(tn, t1), min(tn,t1)) = 0;
                    triangles_changed(tn) = 1;
                end
                if is_neighbour(triangles, tn, t4)
                    tc = connecttril(tc, tn, t4);
                    tc(max(tn, t1), min(tn,t1)) = 0;
                    triangles_changed(tn) = 1;
                end
            end
            % adjust tc internally
            tc(t1,:) =0; tc(:,t1)= 0;
            tc = connecttril(tc, t1, t2);
            tc = connecttril(tc, t1, t3);
            tc = connecttril(tc, t1, t4);
            tc = connecttril(tc, t3, t4);
            % adjust P
            P(W, C) = w(W, C); P(C, W) = w(C, W);
            P(E, C) = w(E, C); P(C, E) = w(C, E);
            P(N, C) = w(N, C); P(C, N) = w(C, N);
        case 2
            triangles( :, t1) = false;
            triangles([W C N], t1) = true;
            triangles_changed(t1) = 1;
            triangles( :, t2) = false;
            triangles([W C S], t2) = true;
            triangles_changed(t2) = 1;
            [~, y] = find(triangles) ; t3 = max(y) +1;
            t4 = t3 + 1;
            triangles( :, t3) = false;
            triangles([N C E], t3) = true;
            triangles_changed(t3) = 1;
            triangles( :, t4) = false;
            triangles([S C E], t4) = true;
            triangles_changed(t4) = 1;
            % adjust tc externally
            % boundary triangles
            tb1 =  find_neib(tc, t1);
            tb2 =  find_neib(tc, t2);
            tb = union(tb1, tb2);
            tc(t1, :) = 0; tc(:, t1) = 0;
            tc(t2, :) = 0; tc(:, t2) = 0;
            for tn = tb
                if is_neighbour(triangles, tn, t1)
                    tc(max(tn, t1), min(tn,t1)) = 1;
                    triangles_changed(tn) = 1;
                end
                if is_neighbour(triangles, tn, t2)
                    tc(max(tn, t2), min(tn,t2)) = 1;
                    triangles_changed(tn) = 1;
                end
                if is_neighbour(triangles, tn, t3)
                    tc(max(tn, t3), min(tn,t3)) = 1;
                    triangles_changed(tn) = 1;
                end
                if is_neighbour(triangles, tn, t4)
                    tc(max(tn, t4), min(tn,t4)) = 1;
                    triangles_changed(tn) = 1;
                end
            end
            % adjust tc internally
            tc = connecttril(tc, t1, t3);
            tc = connecttril(tc, t1, t2);
            tc = connecttril(tc, t2, t4);
            tc = connecttril(tc, t3, t4);
            % adjust P
            P(W, E) = 0; P(E, W) = 0;
            P(W, C) = w(W, C); P(C, W) = w(C, W);
            P(E, C) = w(E, C); P(C, E) = w(C, E);
            P(S, C) = w(S, C); P(C, S) = w(C, S);
            P(N, C) = w(N, C); P(C, N) = w(C, N);
        case 3
            % adjust triangles
            triangles( :, t2) = false;
            triangles([W E C], t2) = true;
            triangles_changed(t2) = 1;
            [~, y] = find(triangles) ; t3 = max(y) +1;
            t4 = t3 + 1;
            triangles( :, t3) = false;
            triangles([W C S], t3) = true;
            triangles_changed(t3) = 1;
            triangles( :, t4) = false;
            triangles([E C S], t4) = true;
            triangles_changed(t4) = 1;
            % adjust tc externally
            %   find triangles bordering with t1
            for tn = find_neib(tc, t2)
                if tn == t1
                    % do nothing
                end
                if is_neighbour(triangles, tn, t3)
                    tc(max(tn, t3), min(tn,t3)) = 1;
                    tc(max(tn, t2), min(tn,t2)) = 0;
                    triangles_changed(tn) = 1;
                end
                if is_neighbour(triangles, tn, t4)
                    tc(max(tn, t4), min(tn,t4)) = 1;
                    tc(max(tn, t2), min(tn,t2)) = 0;
                    triangles_changed(tn) = 1;
                end
            end
            % adjust tc internally
            tc(t2,:) =0; tc(:,t2)= 0;
            tc = connecttril(tc, t1, t2);
            tc = connecttril(tc, t2, t3);
            tc = connecttril(tc, t2, t4);
            tc = connecttril(tc, t3, t4);
            % adjust P
            P(W, C) = w(W, C);
            P(C, W) = w(C, W);
            P(E, C) = w(E, C);
            P(C, E) = w(C, E);
            P(S, C) = w(S, C);
            P(C, S) = w(C, S);
        case 4
            % t1
            triangles( :, t1) = false;
            triangles([W C N], t1) = true;
            triangles_changed(t1) = 1;
            % t2
            triangles( :, t2) = false;
            triangles([W C S], t2) = true;
            triangles_changed(t2) = 1;
            % increment triangles count
            [~, y] = find(triangles) ; t3 = max(y) +1;
            t4 = t3 + 1;
            % t3
            triangles( :, t3) = false;
            triangles([N S C], t3) = true;
            triangles_changed(t3) = 1;
            % t4
            triangles( :, t4) = false;
            triangles([N S E], t4) = true;
            triangles_changed(t4) = 1;
            % process neighboring triangles
            tb1 = find_neib(tc, t1);
            tb2 = find_neib(tc, t2);
            tb = union(tb1, tb2);
            % disconnect neighboring triangles
            tc(t1, :) = 0; tc(:, t1) = 0;
            tc(t2, :) = 0; tc(:, t2) = 0;
            % reconnect external triangles
            for tn = tb
                if is_neighbour(triangles, tn, t1)
                    tc = connecttril(tc, tn, t1);
                    triangles_changed(tn) = 1;
                end
                if is_neighbour(triangles, tn, t2)
                    tc = connecttril(tc, tn, t2);
                    triangles_changed(tn) = 1;
                end
                if is_neighbour(triangles, tn, t3)
                    tc = connecttril(tc, tn, t3);
                    triangles_changed(tn) = 1;
                end
                if is_neighbour(triangles, tn, t4)
                    tc = connecttril(tc, tn, t4);
                    triangles_changed(tn) = 1;
                end
            end
            % adjust tc internally
            tc = connecttril(tc, t1, t2);
            tc = connecttril(tc, t1, t3);
            tc = connecttril(tc, t2, t3);
            tc = connecttril(tc, t3, t4);
```

```
            % adjust P
            P(W, E) = 0; P(E, W) = 0;
            P(N, S) = w(N, S); P(S, N) = w(S, N);
            P(S, C) = w(S, C); P(C, S) = w(C, S);
            P(N, C) = w(N, C); P(C, N) = w(C, N);
            P(W, C) = w(W, C); P(C, W) = w(C, W);
        case 5
            % t1
            triangles( :, t1) = false;
            triangles([N E C], t1) = true;
            triangles_changed(t1) = 1;
            % t2
            triangles( :, t2) = false;
            triangles([E C S], t2) = true;
            triangles_changed(t2) = 1;
            % increment triangles count
            [~, y] = find(triangles) ; t3 = max(y) +1;
            t4 = t3 + 1;
            % t3
            triangles( :, t3) = false;
            triangles([N S C], t3) = true;
            triangles_changed(t3) = 1;
            % t4
            triangles( :, t4) = false;
            triangles([N S W], t4) = true;
            triangles_changed(t4) = 1;
            % process neighboring triangles
            tb1 = find_neib(tc, t1);
            tb2 = find_neib(tc, t2);
            tb = union(tb1, tb2);
            % disconnect neighboring triangles
            tc(t1, :) = 0; tc(:, t1) = 0;
            tc(t2, :) = 0; tc(:, t2) = 0;
            % reconnect external triangles
            for tn = tb
                if is_neighbour(triangles, tn, t1)
                    tc = connecttril(tc, tn, t1);
                    triangles_changed(tn) = 1;
                end
                if is_neighbour(triangles, tn, t2)
                    tc = connecttril(tc, tn, t2);
                    triangles_changed(tn) = 1;
                end
                if is_neighbour(triangles, tn, t3)
                    tc = connecttril(tc, tn, t3);
                    triangles_changed(tn) = 1;
                end
                if is_neighbour(triangles, tn, t4)
                    tc = connecttril(tc, tn, t4);
                    triangles_changed(tn) = 1;
                end
            end
            % adjust tc internally
            tc = connecttril(tc, t1, t2);
            tc = connecttril(tc, t1, t3);
            tc = connecttril(tc, t2, t3);
            tc = connecttril(tc, t3, t4);
            % adjust P
            P(W, E) = 0; P(E, W) = 0;
            P(N, S) = w(N, S); P(S, N) = w(S, N);
            P(S, C) = w(S, C); P(C, S) = w(C, S);
            P(N, C) = w(N, C); P(C, N) = w(C, N);
            P(E, C) = w(E, C); P(C, E) = w(C, E);
        otherwise
            fprintf('some serious error here');
            return;
    end
end

function [P triangles tc triangles_changed] = apply_swaps(w, P, triangles, tc)
    [row col] = find(tril(tc));
    for idx = 1:numel(row)
```

```
        t1 = row(idx); t2 = col(idx);
        if ~tc(t1, t2) % the contact might have changed in the loop
            continue;
        end
        N = tsetdiff(triangles, t1, t2);
        S = tsetdiff(triangles, t2, t1);
        tmp = tintersect(triangles, t1, t2);
        W = tmp(1);
        E = tmp(2);
        triangles_changed(t1) = 0; triangles_changed(t2) = 0;
        if w(N, S) > w(W, E) && P(N, S) == 0
            P(W, E) = 0; P(E, W) = 0;
            P(N, S) = w(N, S); P(S, N) = w(S, N);
            triangles(:, t1) = false;
            triangles([N W S], t1) = true;
            triangles_changed(t1) = true;
            triangles(:, t2) = false;
            triangles([N S E], t2) = true;
            triangles_changed(t2) = true;

            tb1 = find_neib(tc, t1);
            tb2 = find_neib(tc, t2);
            tb = union(tb1, tb2);
            % disconnect neighboring triangles
            tc(t1, :) = 0; tc(:, t1) = 0;
            tc(t2, :) = 0; tc(:, t2) = 0;
            % reconnect external triangles
            for tn = tb
                if is_neighbour(triangles, tn, t1)
                    tc = connecttril(tc, tn, t1);
                    triangles_changed(tn) = 1;
                end
                if is_neighbour(triangles, tn, t2)
                    tc = connecttril(tc, tn, t2);
                    triangles_changed(tn) = 1;
                end
            end
            tc = connecttril(tc, t1, t2);
        end
    end
end

function tsd = tsetdiff(triangles, t1, t2)
    tsd = find(triangles(:,t1) & ~triangles(:, t2));
end

function tc = connecttril(tc, u,v)
  tc(max(u, v), min(u, v)) = 1;
end

function ti = tintersect(triangles, t1, t2)
    ti = find(triangles(:, t1) & triangles(:, t2));
end

function ti = txor(triangles, t1, t2)
    ti = find(xor(triangles(:, t1),triangles(:, t2)));
end

function tn = find_neib(tc, t1)
  tn = union(find(tc(t1,:)), find(tc(:,t1)));
  if size(tn, 1) ~= 1, tn = tn'; end
end
```

./Code/TMFG_matlab/TMFGA.m

Bibliography

H. Akaike. Information theory and an extension of the maximum likelihood principle. In B. N. Petrov and F. Csaki, editors, *Second International Symposium on Information Theory*, pages 267–281, Budapest, 1973. Akadémiai Kiado.

Hirotogu Akaike. Information theory and an extension of the maximum likelihood principle. In *Selected Papers of Hirotugu Akaike*, pages 199–213. Springer, 1998.

Hirotugu Akaike. A new look at the statistical model identification. *Automatic Control, IEEE Transactions on*, 19(6):716–723, 1974.

Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(1):47, 2002.

J. W. Alexander. The combinatorial theory of complexes. *Ann. of Math.*, 31:294–322, 1930.

Theodore Wilbur Anderson. *An introduction to multivariate statistical analysis*. 1962.

J. S. Andrade Jr., H. J. Herrmann, R. F. S. Andrade, and L. R. da Silva. Apollonian networks: Simultaneously scale-free, small world, euclidean, space-filling and with matching graphs. *Phys. Rev. Lett.*, 94:018702, 2005. e-print: cond-mat/0406295.

T. Aste. An algorithm to compute maximally filtered planar graphs. http://www.mathworks.com/ matlabcentral/fileexchange/27360, 2014. URL `http://www.mathworks.com/matlabcentral/fileexchange/27360`.

T. Aste and D. Sherrington. Glass transition in self organizing cellular patterns. *J. Phys. A: Math. Gen.*, 32:7049–56, 1999.

T. Aste, T. Di Matteo, and S.T. Hyde. Complex networks on hyperbolic surfaces. *Physica A*, 346:20–26, 2005a.

T Aste, T Di Matteo, and ST Hyde. Complex networks on hyperbolic surfaces. *Physica A: Statistical Mechanics and its Applications*, 346(1):20–26, 2005b.

T Aste, W Shaw, and T Di Matteo. Correlation structure and dynamics in volatile markets. *New Journal of Physics*, 12(8):085009, 2010a. URL `http://stacks.iop.org/1367-2630/12/i=8/a=085009`.

Tomaso Aste, W Shaw, and T Di Matteo. Correlation structure and dynamics in volatile markets. *New Journal of Physics*, 12(8):085009, 2010b.

Tomaso Aste, Ruggero Gramatica, and T. Di Matteo. Random and frozen states in complex triangulations. *Philosophical Magazine*, 92:246–254, 2012a. doi: 10.1080/14786435.2011.613861. URL `http://www.tandfonline.com/doi/abs/10.1080/14786435.2011.613861`.

Tomaso Aste, Ruggero Gramatica, and T Di Matteo. Exploring complex networks via topological embedding on surfaces. *Phys. Rev. E*, 86(3):036109, 2012b.

Kunihiro Baba, Ritei Shibata, and Masaaki Sibuya. Partial correlation and conditional correlation as measures of conditional independence. *Australian & New Zealand Journal of Statistics*, 46(4):657–664, 2004. ISSN 1467-842X. doi: 10.1111/j.1467-842X.2004.00360.x. URL `http://dx.doi.org/10.1111/j.1467-842X.2004.00360.x`.

Francis R Bach and Michael I Jordan. Thin junction trees. In *Advances in Neural Information Processing Systems*, pages 569–576, 2001.

Francis R Bach and Michael I Jordan. Kernel independent component analysis. *Journal of machine learning research*, 3(Jul):1–48, 2002.

Francis R Bach and Michael I Jordan. Learning graphical models with mercer kernels. In *Advances in Neural Information Processing Systems*, pages 1033–1040, 2003.

Jushan Bai, Serena Ng, et al. Large dimensional factor analysis. *Foundations and Trends® in Econometrics*, 3(2):89–163, 2008.

Hans-Jurgen Bandelt and Victor Chepoi. Metric graph theory and geometry: a survey. *Contemporary Mathematics*, 453:49–86, 2008.

Onureena Banerjee, Laurent El Ghaoui, Alexandre d'Aspremont, and Georges Natsoulis. Convex optimization techniques for fitting sparse gaussian graphical models. In *Proceedings of the 23rd international conference on Machine learning*, pages 89–96. ACM, 2006.

Onureena Banerjee, Laurent El Ghaoui, and Alexandre d'Aspremont. Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data. *The Journal of Machine Learning Research*, 9:485–516, 2008.

Albert-László Barabási. Emergence of scaling in complex networks. *Handbook of graphs and networks: from the genome to the internet*, pages 69–84, 2002.

Albert-László Barabási. The network takeover. *Nature Physics*, 8(1):14, 2011.

Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

David Barber. *Bayesian reasoning and machine learning*. Cambridge University Press, 2012.

Wolfram Barfuss, Guido Previde Massara, Tiziana Di Matteo, and Tomaso Aste. Parsimonious modeling with information filtering networks. *Physical Review E*, 94(6): 062306, 2016.

Baruch Barzel, Amitabh Sharma, and Albert-Lszl Barabsi. Graph theory properties of cellular networks, 2013.

Christopher M Bishop et al. *Neural networks for pattern recognition*. Oxford university press, 1995.

Jean RS Blair and Barry Peyton. An introduction to chordal graphs and clique trees. In *Graph theory and sparse matrix computation*, pages 1–29. Springer, 1993.

Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. Recommender systems survey. *Knowledge-based systems*, 46:109–132, 2013.

Andrej Bogdanov, Elchanan Mossel, and Salil Vadhan. The complexity of distinguishing markov random fields. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pages 331–342. Springer, 2008.

Béla Bollobás. *Modern graph theory*, volume 184. Springer Science & Business Media, 2013.

Giovanni Bonanno, Guido Caldarelli, Fabrizio Lillo, Salvatore Micciche, Nicolas Vandewalle, and Rosario Nunzio Mantegna. Networks of equities in financial markets. *The European Physical Journal B*, 38(2):363–371, 2004.

Denny Borsboom. A network theory of mental disorders. 16:5–13, 2017. ISSN 1723-8617. doi: 10.1002/wps.20375.

Otakar Boruvka. O jistém problému minimálním. *Práce Mor. Prırodved. Spol. v Brne (Acta Societ. Scienc. Natur. Moravicae)*, 3(3):37–58, 1926.

Raouf Boutaba, Mohammad A. Salahuddin, Noura Limam, Sara Ayoubi, Nashid Shahriar, Felipe Estrada-Solano, and Oscar M. Caicedo. A comprehensive survey on machine learning for networking: evolution, applications and research opportunities. *Journal of Internet Services and Applications*, 9(1):16, Jun 2018. ISSN 1869-0238. doi: 10.1186/s13174-018-0087-2. URL `https://doi.org/10.1186/s13174-018-0087-2`.

John M Boyer and Wendy Myrvold. Simplified o(n) planarity algorithms. 2001.

Jol Bun, Jean-Philippe Bouchaud, and Marc Potters. Cleaning large correlation matrices: Tools from random matrix theory. *Physics Reports*, 666:1 – 109, 2017. ISSN 0370-1573. doi: https://doi.org/10.1016/j.physrep.2016.10.005. URL `http://www.sciencedirect.com/science/article/pii/S0370157316303337`.

James R Bunch and Donald J Rose. *Sparse matrix computations*. Academic Press, 2014.

Gunnar Carlsson. Topology and data. *Bulletin of the American Mathematical Society*, 46(2):255–308, 2009.

Avishy Y Carmi, Lyudmila Mihaylova, and Simon J Godsill. *Compressed sensing & sparse filtering*. Springer, 2014.

Gonzalo Castañeda Ramos and Omar A Guerrero. Evaluating policy priorities under social learning and endogenous government behavior. *Available at SSRN 3257917*, 2018a.

Gonzalo Castañeda Ramos and Omar A Guerrero. The resilience of public policies in economic development. 2018b.

Gonzalo Castañeda Ramos and Omar A Guerrero. The importance of social and government learning in ex ante policy evaluation. *Journal of Policy Modeling*, 41(2): 273–293, 2019.

Gonzalo Castañeda Ramos, Florian Chávez-Juárez, and Omar A Guerrero. How do governments determine policy priorities? studying development strategies through spillover networks. *Studying Development Strategies Through Spillover Networks (January 20, 2018)*, 2018.

Diane Castonguay, Elisângela Silva Dias, and Leslie Richard Foulds. Results for the maximum weight planar subgraph problem. *arXiv preprint arXiv:1712.05711*, 2017.

Chung Chan and Tie Liu. Clustering by multivariate mutual information under chow-liu tree approximation. In *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 993–999. IEEE, 2015.

Bernard Chazelle. A minimum spanning tree algorithm with inverse-ackermann type complexity. 47:1028–1047, 2000. ISSN 0004-5411. doi: 10.1145/355541.355562.

Kwang-Cheng Chen, Mung Chiang, and H Vincent Poor. From technological networks to social networks. *IEEE Journal on Selected Areas in Communications*, 31(9):548–572, 2013.

David M Chickering, Dan Geiger, David Heckerman, et al. Learning bayesian networks is np-hard. Technical report, Citeseer, 1994.

David Maxwell Chickering. Learning bayesian networks is np-complete. In *Learning from data*, pages 121–130. Springer, 1996.

Matteo Chinazzi and Giorgio Fagiolo. Systemic risk, contagion, and financial networks: A survey. *Institute of Economics, Scuola Superiore SantAnna, Laboratory of Economics and Management (LEM) Working Paper Series*, (2013/08), 2015.

C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467, May 1968. ISSN 0018-9448. doi: 10.1109/TIT.1968.1054142.

Alexander P Christensen, Katherine Cotter, Paul Silvia, and Mathias Benedek. Scale development via network analysis: A comprehensive and concise measure of openness to experience. 2018a.

Alexander P Christensen, Katherine N Cotter, and Paul J Silvia. Reopening openness to experience: A network analysis of four openness to experience inventories. *Journal of Personality Assessment*, pages 1–15, 2018b.

Alexander P Christensen, Yoed N Kenett, Tomaso Aste, Paul J Silvia, and Thomas R Kwapil. Network structure of the wisconsin schizotypy scales–short forms: Examining psychometric network filtering approaches. *Behavior research methods*, 50(6): 2531–2550, 2018c.

Vinícius de S Coelho, Wellington S Martins, Leslie R Foulds, Elisângela S Dias, Diane Castonguay, and Humberto J Longo. Uma proposta de solução aproximada para o problema do subgrafo planar de peso máximo. *XVII Simpósio em Sistemas Computacionais de Alto Desempenho (WSCAD 2016)*, pages 16–27, 2016.

Pierre Comon. Independent component analysis, a new concept? *Signal processing*, 36(3):287–314, 1994.

Owen T Courtney and Ginestra Bianconi. Generalized network structures: The configuration model and the canonical ensemble of simplicial complexes. *Physical Review E*, 93(6):062311, 2016.

Owen T Courtney and Ginestra Bianconi. Weighted growing simplicial complexes. *Physical Review E*, 95(6):062301, 2017.

Alexandre d'Aspremont, Onureena Banerjee, and Laurent El Ghaoui. First-order methods for sparse covariance selection. *SIAM Journal on Matrix Analysis and Applications*, 30(1):56–66, 2008.

JA van Lidth de Jeude, Tomaso Aste, and Guido Caldarelli. The multilayer structure of corporate networks. *New Journal of Physics*, 21(2):025002, 2019.

Yves Colin de Verdiere. Sur un nouvel invariant des graphes et un critere de planarité. *Journal of Combinatorial Theory, Series B*, 50(1):11–21, 1990.

A. P. Dempster. Covariance selection. *Biometrics*, 28(1):157–175, 1972.

Alexander Denev. *Probabilistic Graphical Models: A New Way of Thinking in Financial Modelling*. Risk Books, 2015.

Yue Deng, Qionghai Dai, and Zengke Zhang. An overview of computational sparse models and their applications in artificial intelligence. In *Artificial Intelligence, Evolutionary Computing and Metaheuristics*, pages 345–369. Springer, 2013.

Amol Deshpande, Minos Garofalakis, and Michael I Jordan. Efficient stepwise selection in decomposable models. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pages 128–135. Morgan Kaufmann Publishers Inc., 2001.

T Di Matteo, T Aste, ST Hyde, and S Ramsden. Interest rates hierarchical structure. *Physica A: Statistical Mechanics and its Applications*, 355(1):21–33, 2005.

Tiziana Di Matteo and Tomaso Aste. How does the eurodollar interest rate behave? *International Journal of Theoretical and Applied Finance*, 5(01):107–122, 2002.

Reinhard Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer, Heidelberg, 2010.

Gabriel Andrew Dirac. On rigid circuit graphs. In *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, volume 25, pages 71–76. Springer, 1961.

Sergei N Dorogovtsev. *Lectures on complex networks*, volume 24. Oxford University Press Oxford, 2010.

M. Drton and M. H. Maathuis. Structure Learning in Graphical Modeling. *Annual Review of Statistics and Its Application*, 4:365–393, March 2017. doi: 10.1146/annurev-statistics-060116-053803.

Mathias Drton, Steffen Lilholt Lauritzen, Marloes Maathuis, and Martin Wainwright. *Handbook of Graphical Models*. CRC Press, 2018.

B. Dubertret, T. Aste, H. M. Ohlenbusch, and N. Rivier. Two-dimensional froths and the dynamics of biological tissues. *Phys. Rev. E*, 58(5):6368–6378, Nov 1998. doi: 10.1103/PhysRevE.58.6368.

Jean-Guillaume Dumas, Frank Heckenbach, David Saunders, and Volkmar Welker. Computing simplicial homology based on efficient smith normal form algorithms. In *Algebra, Geometry and Software Systems*, pages 177–206. Springer, 2003.

ME Dyer, Leslie Richard Foulds, and AM Frieze. Analysis of heuristics for finding a maximum weight planar subgraph. *European Journal of Operational Research*, 20 (1):102–114, 1985.

David Easley and Jon Kleinberg. *Networks, crowds, and markets: Reasoning about a highly connected world*. Cambridge University Press, 2010.

Daniel Eaton and Kevin Murphy. Bayesian structure learning using dynamic programming and mcmc. *arXiv preprint arXiv:1206.5247*, 2012.

Robert Engle. *Anticipating correlations: a new paradigm for risk management*. Princeton University Press, 2009.

Robert Engle and Riccardo Colacito. Testing and valuing dynamic correlations for asset allocation. *Journal of Business & Economic Statistics*, 2012.

Jianqing Fan, Jinchi Lv, and Lei Qi. Sparse high-dimensional models in economics. *Annu. Rev. Econ.*, 3(1):291–317, 2011.

Paweł Fiedor. Networks in financial markets based on the mutual information rate. *Physical Review E*, 89(5):052801, 2014.

Ismail Onur Filiz, Xin Guo, Jason Morton, and Bernd Sturmfels. Graphical models for correlated defaults. 22:621–644, 2012. ISSN 0960-1627. doi: 10.1111/j.1467-9965.2011.00499.x.

Marcelo Fiori, Pablo Musé, and Guillermo Sapiro. Topology constraints in graphical models. In *Advances in Neural Information Processing Systems*, pages 791–799, 2012.

LR Foulds and David F Robinson. Graph theoretic heuristics for the plant layout problem. *THE INTERNATIONAL JOURNAL OF PRODUCTION RESEARCH*, 16(1):27–37, 1978.

LR Foulds and DF Robinson. A strategy for solving the plant layout problem. *Operational Research Quarterly*, pages 845–855, 1976.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics (Oxford, England)*, 9(3):432–441, 2008.

Prasanna Gai and Sujit Kapadia. Contagion in financial networks. 466:2401–2423, 2010. ISSN 1364-5021. doi: 10.1098/rspa.2009.0410.

Philippe Galinier, Michel Habib, and Christophe Paul. Chordal graphs and their clique graphs. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 358–371. Springer, 1995.

Robert Ghrist. Barcodes: the persistent topology of data. *Bulletin of the American Mathematical Society*, 45(1):61–75, 2008.

Marian Gidea. Topological data analysis of critical transitions in financial networks. In *International Conference and School on Network Science*, pages 47–59. Springer, 2017.

John Winston Giffin. *Graph theoretic techniques for facilities layout*. PhD thesis, University of Canterbury, 1984.

P Giudici and PJ Green. Decomposable graphical gaussian model determination. *Biometrika*, 86(4):785–801, 1999. doi: 10.1093/biomet/86.4.785. URL +http://dx.doi.org/10.1093/biomet/86.4.785.

Paolo Giudici and Alessandro Spelta. Graphical network models for international financial flows. *Journal of Business & Economic Statistics*, 34(1):128–138, 2016.

Chad Giusti, Eva Pastalkova, Carina Curto, and Vladimir Itskov. Clique topology reveals intrinsic geometric structure in neural correlations. *Proceedings of the National Academy of Sciences*, 112(44):13455–13460, 2015.

Bornali Gogoi and Bichitra Kalita. Algorithm for designing vlsi floorplan using planar triangulated graph. *International Journal of Information and Communication Technology Research*, 2(7), 2012.

Hudson Golino, Dingjing Shi, Luis Eduardo Garrido, AP Christensen, María Dolores Nieto, Ritu Sadana, and Jotheeswaran Amuthavalli Thiyagarajan. Investigating the performance of exploratory graph analysis and traditional techniques to identify the number of latent factors: A simulation and tutorial, 2018.

Hudson F Golino and Sacha Epskamp. Exploratory graph analysis: A new approach for estimating the number of dimensions in psychological research. *PloS one*, 12(6): e0174035, 2017.

Martin Charles Golumbic. *Algorithmic graph theory and perfect graphs*, volume 57. Elsevier, 2004.

RH Green and LAR Al-Hakim. A heuristic for facilities layout planning. *Omega*, 13 (5):469–474, 1985.

Arthur Gretton, Ralf Herbrich, Alexander Smola, Olivier Bousquet, and Bernhard Schölkopf. Kernel methods for measuring independence. *Journal of Machine Learning Research*, 6(Dec):2075–2129, 2005.

Elizabeth Gross, Seth Sullivant, et al. The maximum likelihood threshold of a graph. *Bernoulli*, 24(1):386–407, 2018.

Omar A Guerrero and Gonzalo Castañeda Ramos. Quantifying the coherence of development policy priorities. *Available at SSRN 3264005*, 2018.

Frank Harary and Edgar M Palmer. *Graphical enumeration*. Elsevier, 2014.

Trevor Hastie, Robert Tibshirani, and Martin Wainwright. *Statistical learning with sparsity: the lasso and generalizations*. CRC press, 2015.

Jose Sergio Hleap and Christian Blouin. Inferring meaningful communities from topology-constrained correlation networks. *PLOS ONE*, 9(11):1–9, 11 2014. doi: 10.1371/journal.pone.0113438. URL `https://doi.org/10.1371/journal.pone.0113438`.

Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

John Hopcroft and Robert Tarjan. Efficient planarity testing. *Journal of the ACM (JACM)*, 21(4):549–568, 1974.

Anne-Caroline Hser. Too interconnected to fail: A survey of the interbank networks literature. SAFE Working Paper Series 91, Research Center SAFE - Sustainable Architecture for Finance in Europe, Goethe University Frankfurt, 2015. URL `https://ideas.repec.org/p/zbw/safewp/91.html`.

Cho-jui Hsieh, Inderjit S. Dhillon, Pradeep K. Ravikumar, and Mátyás A. Sustik. Sparse inverse covariance matrix estimation using quadratic approximation. In J. Shawe-Taylor, R.S. Zemel, P.L. Bartlett, F. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2330–2338. Curran Associates, Inc., 2011. URL `http://papers.nips.cc/paper/4266-sparse-inverse-covariance-matrix-estimation-using-quadratic-a` `pdf`.

Kaizhu Huang, Irwin King, and Michael R Lyu. Constructing a large node chow-liu tree based on frequent itemsets. In *Neural Information Processing, 2002. ICONIP'02. Proceedings of the 9th International Conference on*, volume 1, pages 498–502. IEEE, 2002.

Ke Huang and Selin Aviyente. Sparse representation for signal classification. In *Advances in neural information processing systems*, pages 609–616, 2007.

Aapo Hyvärinen and Erkki Oja. Independent component analysis: algorithms and applications. *Neural networks*, 13(4-5):411–430, 2000.

Giovanni Iacobello, Stefania Scarsoglio, Hans Kuerten, and Luca Ridolfi. Spatial network investigation of wall turbulence. In *BOOK OF ABSTRACTS*, page 300, 2017.

Trey Ideker and Ruth Nussinov. Network approaches and applications in biology. *PLOS Computational Biology*, 13(10):1–3, 10 2017. doi: 10.1371/journal.pcbi.1005771. URL `https://doi.org/10.1371/journal.pcbi.1005771`.

Amir Globerson Tommi Jaakkola. Approximate inference using planar graph decomposition. In *Advances in Neural Information Processing Systems 19: Proceedings of the 2006 Conference*, volume 19, page 473. MIT Press, 2007.

Matthew O Jackson. *Social and economic networks*. Princeton university press, 2010.

Matthew O. Jackson. An overview of social networks and economic applications. In *Handbook of social economics*, volume 1, pages 511–585. Elsevier, 2011. doi: 10.1016/b978-0-444-53187-2.00012-7.

Beatrix Jones, Carlos Carvalho, Adrian Dobra, Chris Hans, Chris Carter, Mike West, et al. Experiments in stochastic computation for high-dimensional graphical models. *Statistical Science*, 20(4):388–400, 2005.

Michael Jünger and Petra Mutzel. Solving the maximum weight planar subgraph problem by branch-and-cut. 1993.

David Karger and Nathan Srebro. Learning markov networks: Maximum bounded tree-width graphs. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 392–401. Society for Industrial and Applied Mathematics, 2001.

Ross Kindermann, James Laurie Snell, et al. *Markov random fields and their applications*, volume 1. American Mathematical Society Providence, RI, 1980.

Alexander Kirpich, Elizabeth A Ainsworth, Jessica M Wedow, Jeremy RB Newman, George Michailidis, and Lauren M McIntyre. Variable selection in omics data: A practical evaluation of small sample sizes. *PloS one*, 13(6):e0197910, 2018.

Sergey Kirshner, Padhraic Smyth, and Andrew W Robertson. Conditional chow-liu tree structures for modeling discrete-valued vector time series. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 317–324. AUAI Press, 2004.

Oliver Knill. A discrete gauss-bonnet type theorem. *arXiv preprint arXiv:1009.2292*, 2010.

Oliver Knill. On index expectation and curvature for networks. *arXiv preprint arXiv:1202.4514*, 2012.

Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

Timo Koski. Lectures on statistical learning theory for chow-liu trees. *The 32nd Finnish Summer School on Probability Theory*, 2010.

Timo JT Koski and John Noble. A review of bayesian networks and structure learning. *Mathematica Applicanda*, 40(1):51–103, 2012.

Edith Kovács and Tamás Szántai. Discovering the markov network structure. *CoRR*, abs/1307.0643, 2013. URL `http://arxiv.org/abs/1307.0643`.

Joost Kruis and Gunter Maris. Three representations of the ising model. 6, 2016. ISSN 2045-2322. doi: 10.1038/srep34175.

H Ku and Solomon Kullback. Approximating discrete probability distributions. *IEEE Transactions on Information Theory*, 15(4):444–447, 1969.

Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 03 1951. doi: 10.1214/aoms/1177729694. URL `http://dx.doi.org/10.1214/aoms/1177729694`.

Casimir Kuratowski. Sur le probleme des courbes gauches en topologie. *Fundamenta mathematicae*, 15(1):271–283, 1930.

Laurent Laloux, Pierre Cizeau, Jean-Philippe Bouchaud, and Marc Potters. Noise dressing of financial correlation matrices. *Physical review letters*, 83(7):1467, 1999.

Steffen Lauritzen. Structure estimation in graphical models. Wald Lecture, World Meeting on Probability and Statistics, 2012.

Steffen L Lauritzen. *Graphical models*. Oxford University Press, 1996.

Steffen L Lauritzen and David J Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 157–224, 1988.

Christophe Lecoutre. *Constraint Networks: Targeting Simplicity for Techniques and Algorithms*. John Wiley & Sons, 2013.

Olivier Ledoit and Michael Wolf. Improved estimation of the covariance matrix of stock returns with an application to portfolio selection. *Journal of empirical finance*, 10(5):603–621, 2003.

Olivier Ledoit and Michael Wolf. Honey, i shrunk the sample covariance matrix. *The Journal of Portfolio Management*, 30(4):110–119, 2004. ISSN 0095-4918. doi: 10.3905/jpm.2004.110. URL `http://jpm.iijournals.com/content/30/4/110`.

Thomas Lengauer. *Combinatorial algorithms for integrated circuit layout*. John Wiley & Sons, Inc., New York, NY, USA, 1990. ISBN 0-471-92838-0.

Janny Leung. A new graph-theoretic heuristic for facility layout. *Management Science*, 38(4):594–605, 1992.

Weihua Li, Tomaso Aste, Fabio Caccioli, and Giacomo Livan. Reciprocity and success in academic careers. *arXiv preprint arXiv:1808.03781*, 2018.

Weihua Li, Tomaso Aste, Fabio Caccioli, and Giacomo Livan. Early coauthorship with top scientists predicts success in academic careers. *Nature communications*, 10(1): 1–9, 2019.

Annegret Liebers. Planarizing graphs—a survey and annotated bibliography. *Journal of Graph Algorithms and Applications*, 5(1):74 p.–74 p., 2001. URL `http://eudml.org/doc/48836`.

Han Liu, Min Xu, Haijie Gu, Anupam Gupta, John Lafferty, and Larry Wasserman. Forest density estimation. *Journal of Machine Learning Research*, 12(Mar):907–951, 2011.

Giacomo Livan, Fabio Caccioli, and Tomaso Aste. Excess reciprocity distorts reputation in online social networks. *Scientific reports*, 7(1):3551, 2017.

László Lovász. Graph minor theory. *Bulletin of the American Mathematical Society*, 43(1):75–86, 2006.

Saunders Mac Lane. A combinatorial condition for planar graphs. *Fundamenta Mathematicae*, 28(1):22–32, 1937. URL `http://eudml.org/doc/212919`.

David Madigan, Jeremy York, and Denis Allard. Bayesian graphical models for discrete data. *International Statistical Review/Revue Internationale de Statistique*, pages 215–232, 1995.

Lorenzo Magnani. Reasoning through doing. epistemic mediators in scientific discovery. *Journal of Applied Logic*, 2(4):439–450, 2004.

Francesco M Malvestuto. A backward selection procedure for approximating a discrete probability distribution by decomposable models. *Kybernetika*, 48(5):825–844, 2012.

R. N. Mantegna. Hierarchical structure in financial markets. *Eur. Phys. J. B*, 11 (1):193–197, 1999. URL `http://EconPapers.repec.org/RePEc:spr:eurphb:v:11:y:1999:i:1:p:193-197`.

Dimitris Margaritis and Sebastian Thrun. Bayesian network induction via local neighborhoods. In *Advances in neural information processing systems*, pages 505–511, 2000.

Guido Previde Massara and Tomaso Aste. Learning clique forests. *arXiv preprint arXiv:1905.02266*, 2019.

Guido Previde Massara, Tiziana Di Matteo, and Tomaso Aste. Network filtering for big data: triangulated maximally filtered graph. *Journal of complex Networks*, 5(2): 161–178, 2016.

Nicolai Meinshausen and Peter Bühlmann. High dimensional graphs and variable selection with the lasso. *Annals of Statistics*, 34(3):1436–1462, 2006.

Mauro Mezzini and Marina Moscarini. Simple algorithms for minimal triangulation of a graph and backward selection of a decomposable markov network. *Theoretical Computer Science*, 411(7-9):958–966, 2010.

Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. Network motifs: simple building blocks of complex networks. *Science*, 298 (5594):824–827, 2002.

Saad Zaghloul Salem Mohammad. Biological networks: An introductory review. *Journal Of Proteomics And Genomics Research*, 2(1):41 – 111, 2018. ISSN 2326-0793. doi: https://doi.org/10.14302/issn.2326-0793.jpgr-18-2312. URL `https://openaccesspub.org/jpgr/article/869`.

J.R. Munkres. *Elements of Algebraic Topology*. Advanced book classics. Perseus Books, 1984. ISBN 9780201627282. URL `http://books.google.co.uk/books?id=Qw6m_xGryPoC`.

N Musmeci, T Aste, and T Di Matteo. Relation between financial market structure and the real economy: Comparison between clustering methods. *PLoS ONE*, 10(3): e0116201, 2015a.

Nicolo Musmeci, Tomaso Aste, and Tiziana Di Matteo. Clustering and hierarchy of financial markets data: advantages of the dbht. *arXiv preprint arXiv:1406.0496*, 2014a.

Nicolo Musmeci, Tomaso Aste, and Tiziana Di Matteo. Clustering and hierarchy of financial markets data: advantages of the dbht. *arXiv preprint arXiv:1406.0496*, 2014b.

Nicoló Musmeci, Tomaso Aste, and T Di Matteo. Risk diversification: a study of persistence with a filtered correlation-network approach. *Network Theory in Finance*, 1(1):77–98, 2015b.

Nicolò Musmeci, Vincenzo Nicosia, Tomaso Aste, Tiziana Di Matteo, and Vito Latora. The multiplex dependency structure of financial markets. *Complexity*, 2017, 2017.

David R Musser and Alexander A Stepanov. Generic programming. In *International Symposium on Symbolic and Algebraic Computation*, pages 13–25. Springer, 1988.

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

Takao Nishizeki and Norishige Chiba. *Planar graphs: Theory and algorithms*, volume 32. Elsevier, 1988.

Ibrahim H. Osman, Baydaa Al-Ayoubi, and Musbah Barake. A greedy random adaptive search procedure for the weighted maximal planar graph problem. *Computers and Industrial Engineering*, 45(4):635–651, 2003. ISSN 03608352. doi: 10.1016/j.cie.2003.09.005.

Nina Otter, Mason A Porter, Ulrike Tillmann, Peter Grindrod, and Heather A Harrington. A roadmap for the computation of persistent homology. *EPJ Data Science*, 6 (1):17, 2017.

Muge Ozman. Inter-firm networks and innovation: a survey of literature. *Economic of Innovation and New Technology*, 18(1):39–67, 2009.

Figen Oztoprak, Jorge Nocedal, Steven Rennie, and Peder A. Olsen. Newton-like methods for sparse inverse covariance estimation. In P. Bartlett, F.c.n. Pereira, C.j.c. Burges, L. Bottou, and K.q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 755–763, 2012. URL `http://books.nips.cc/papers/files/nips25/NIPS2012_0344.pdf`.

Szilard Pafka and Imre Kondor. Estimated correlation matrices and portfolio optimization. *Physica A: Statistical Mechanics and Its Applications*, 343:623–634, 2004.

Matthew Pelowski, Young-Jin Hur, Katherine N Cotter, Tomohiro Ishizu, Alexander P Christensen, Helmut Leder, and IC McManus. Quantifying the if, the when, and the what of the sublime: A survey and latent class analysis of incidence, emotions, and distinct varieties of personal sublime experiences. *Psychology of Aesthetics, Creativity, and the Arts*, 2019.

Franois Petitjean and Geoffrey I. Webb. *Scaling log-linear analysis to datasets with thousands of variables*, pages 469–477. 2015. doi: 10.1137/1.9781611974010.53. URL `http://epubs.siam.org/doi/abs/10.1137/1.9781611974010.53`.

G. Petri, P. Expert, F. Turkheimer, R. Carhart-Harris, D. Nutt, P. J. Hellyer, and F. Vaccarino. Homological scaffolds of brain functional networks. *Journal of The Royal Society Interface*, 11(101):20140873, 2014. ISSN 1742-5689. doi: 10.1098/rsif.2014.0873.

Timo Poranen. A simulated annealing algorithm for the maximum planar subgraph problem. *International Journal of Computer Mathematics*, 81(5):555–568, 2004.

F Pozzi, T Di Matteo, and T Aste. Spread of risk across financial markets: better to invest in the peripheries. *Scientific Reports*, 3:1665, 2013.

R. C. Prim. Shortest connection networks and some generalizations. *BellSystem Technical Journal*, 36:1389–1401, 1957a.

R.C. Prim. Shortest connection networks and some generalizations. *Bell System Technical Journal, The*, 36(6):1389–1401, Nov 1957b. ISSN 0005-8580. doi: 10.1002/j.1538-7305.1957.tb01515.x.

Pier Francesco Procacci and Tomaso Aste. Forecasting market states. 2018. ISSN 1556-5068. doi: 10.2139/ssrn.3215945.

Weiliang Qiu and Harry Joe. *clusterGeneration: Random Cluster Generation (with Specified Degree of Separation)*, 2015. URL `https://CRAN.R-project.org/package=clusterGeneration`. R package version 1.3.4.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2016. URL `https://www.R-project.org/`.

Pradeep Ravikumar, Martin J. Wainwright, Garvesh Raskutti, and Bin Yu. High-dimensional covariance estimation by minimizing $\ell 1$-penalized log-determinant divergence. *Electronic Journal of Statistics*, 5:935–980, 2011. doi: 10.1214/11-EJS631. URL `http://dx.doi.org/10.1214/11-EJS631`.

Riccardo Rebonato. Coherent stress testing, 2010a.

Riccardo Rebonato. *Coherent Stress Testing: a Bayesian approach to the analysis of financial stress*. John Wiley & Sons, 2010b.

Riccardo Rebonato and Alexander Denev. *Portfolio Management Under Stress: A Bayesian-net Approach to Coherent Asset Allocation*. Cambridge University Press, 2014.

Alvin C Rencher. *Methods of multivariate analysis*, volume 492. John Wiley & Sons, 2003.

Jorma Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978.

H. Rue and L. Held. *Gaussian Markov Random Fields: Theory and Applications*, volume 104 of *Monographs on Statistics and Applied Probability*. Chapman & Hall, London, 2005.

Romeil S. Sandhu, Tryphon T. Georgiou, and Allen R. Tannenbaum. Ricci curvature: An economic indicator for market fragility and systemic risk. 2:e1501495, 2016. ISSN 2375-2548. doi: 10.1126/sciadv.1501495.

Erik Schaffernicht, Volker Stephan, and Horst-Michael Groß. An efficient search strategy for feature selection using chow-liu trees. In *International Conference on Artificial Neural Networks*, pages 190–199. Springer, 2007.

Verena D. Schmittmann, Anglique O. J. Cramer, Lourens J. Waldorp, Sacha Epskamp, Rogier A. Kievit, and Denny Borsboom. Deconstructing the construct: A network perspective on psychological phenomena. 31:43–53, 2013. ISSN 0732-118X. doi: 10.1016/j.newideapsych.2011.02.007.

Walter Schnyder. Planar graphs and poset dimension. *Order*, 5(4):323–343, Dec 1989. ISSN 1572-9273. doi: 10.1007/BF00353652. URL `https://doi.org/10.1007/BF00353652`.

Gideon Schwarz et al. Estimating the dimension of a model. *The annals of statistics*, 6 (2):461–464, 1978.

Marco Scutari and Korbinian Strimmer. *Introduction to Graphical Modelling*, chapter 11, pages 235–254. Wiley-Blackwell, 2011. ISBN 9781119970606. doi: 10.1002/9781119970606.ch11. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119970606.ch11`.

John Shawe-Taylor and Nello Cristianini. *Kernel methods for pattern analysis*. Cambridge university press, 2004.

Ann E Sizemore, Chad Giusti, Ari Kahn, Jean M Vettel, Richard F Betzel, and Danielle S Bassett. Cliques and cavities in the human connectome. *Journal of computational neuroscience*, 44(1):115–145, 2018.

W.-M. Song, T. Di Matteo, and T. Aste. Nested hierarchies in planar graphs. *Discrete Applied Mathematics*, 159:2135–2146, 2011.

W.-M. Song, T. Di Matteo, and T. Aste. Hierarchical information clustering by means of topologically embedded graphs. *PLoS ONE*, 7:e31929, 2012a.

Won-Min Song, Tomaso Aste, and T Di Matteo. Correlation-based biological networks. In *Microelectronics, MEMS, and Nanotechnology*, pages 680212–680212. International Society for Optics and Photonics, 2007.

Won-Min Song, T Di Matteo, and T Aste. Building complex networks with platonic solids. *Phys. Rev. E*, 85(4):046115, 2012b.

Wataru Souma, Yoshi Fujiwara, and Hideaki Aoyama. Complex networks and economics. *Physica A: Statistical Mechanics and its Applications*, 324(1-2):396–401, 2003.

Peter Spirtes, Clark N Glymour, Richard Scheines, David Heckerman, Christopher Meek, Gregory Cooper, and Thomas Richardson. *Causation, prediction, and search*. MIT press, 2000.

G. Strang. *Introduction to Applied Mathematics*. Wellesley-Cambridge Press, 1986. ISBN 9780961408800. URL `https://books.google.co.uk/books?id=Lr9YKrCANWwC`.

Steven H Strogatz. Exploring complex networks. *nature*, 410(6825):268, 2001.

Milan Studeny. *Probabilistic conditional independence structures*. Springer Science & Business Media, 2006.

Joe Suzuki. A generalization of the chow-liu algorithm and its application to statistical learning. *arXiv preprint arXiv:1002.2240*, 2010.

Tamás Szántai and Edith Kovács. Discovering a junction tree behind a markov network by a greedy algorithm. *Optimization and Engineering*, 14(4):503–518, 2013.

Gábor J Székely and Maria L Rizzo. The distance correlation t-test of independence in high dimension. *Journal of Multivariate Analysis*, 117:193–213, 2013.

Robert Endre Tarjan. Maximum cardinality search and chordal graphs. *Unpublished Lecture Notes CS*, 259, 1976.

Robert Endre Tarjan. Data structures and network algorithms, 1983.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58(1):267–288, 1996.

Andrey Nikolayevich Tikhonov. On the stability of inverse problems. 39, 1943.

M. Tumminello, T. Aste, T. Di Matteo, and R. N. Mantegna. A tool for filtering information in complex systems. *Proceedings of the National Academy of Sciences of the United States of America*, 102(30):10421–10426, 2005. doi: 10.1073/pnas.0500298102. URL `http://www.pnas.org/content/102/30/10421.abstract`.

MICHELE Tumminello, T Di Matteo, T Aste, and Rosario Nunzio Mantegna. Correlation based networks of equity returns sampled at different time horizons. *The European Physical Journal B-Condensed Matter and Complex Systems*, 55(2):209–217, 2007.

W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer, New York, fourth edition, 2002. URL `http://www.stats.ox.ac.uk/pub/MASS4`. ISBN 0-387-95457-0.

Klaus Wagner. Über eine eigenschaft der ebenen komplexe. *Mathematische Annalen*, 114(1):570–590, 1937.

Martin J. Wainwright and Michael I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2): 1–305, 2008. ISSN 1935-8237. doi: 10.1561/2200000001. URL `http://dx.doi.org/10.1561/2200000001`.

Bao Wang and Guo-Wei Wei. Object-oriented persistent homology. *Journal of computational physics*, 305:276–299, 2016.

Larry Wasserman. *All of Statistics: A Concise Course in Statistical Inference*. Springer Publishing Company, Incorporated, 2010. ISBN 1441923225, 9781441923226.

Hassler Whitney. Non-separable and planar graphs. *Proc. Nat. Acad. Sci. U.S.A.*, 17(2):125–127, February 1931. ISSN 0027-8424. doi: 10.1073/pnas.17.2.125. JFM:57.0727.05.

Hadley Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2009. ISBN 978-0-387-98140-6. URL `http://ggplot2.org`.

Cen Wu, Fei Zhou, Jie Ren, Xiaoxi Li, Yu Jiang, and Shuangge Ma. A selective review of multi-level omics data integration using variable selection. *High-throughput*, 8 (1):4, 2019.

Tuo Zhao, Xingguo Li, Han Liu, Kathryn Roeder, John Lafferty, and Larry Wasserman. *huge: High-Dimensional Undirected Graph Estimation*, 2015. URL `https://CRAN.R-project.org/package=huge`. R package version 1.2.7.

Yang Zhou. Structure learning of probabilistic graphical models: a comprehensive survey. *arXiv preprint arXiv:1111.6925*, 2011.

Afra Zomorodian and Gunnar Carlsson. Computing persistent homology. *Discrete & Computational Geometry*, 33(2):249–274, 2005.

Afra J. Zomorodian. *Topology for Computing*. Cambridge University Press, 2009.

Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2): 301–320, 2005.