THE UNIVERSITY OF SYDNEY

COPYRIGHT AND USE OF THIS THESIS

This thesis must be used in accordance with the provisions of the Copyright Act 1968.

Reproduction of material protected by copyright may be an infringement of copyright and copyright owners may be entitled to take legal action against persons who infringe their copyright.

Section 51 (2) of the Copyright Act permits an authorized officer of a university library or archives to provide a copy (by communication or otherwise) of an unpublished thesis kept in the library or archives, to a person who satisfies the authorized officer that he or she requires the reproduction for the purposes of research or study.

The Copyright Act grants the creator of a work a number of moral rights, specifically the right of attribution, the right against false attribution and the right of integrity.

You may infringe the author's moral rights if you:

- fail to acknowledge the author of this thesis if you quote sections from the work
- attribute this thesis to another author
- subject this thesis to derogatory treatment which may prejudice the author's reputation

For further information contact the University's Director of Copyright Services

sydney.edu.au/copyright

Conservative Sparsification for Efficient Approximate Estimation

John Vial

A thesis submitted in fulfillment
of the requirements of the degree of
Doctor of Philosophy

THE UNIVERSITY OF
SYDNEY

Australian Centre for Field Robotics
School of Aerospace, Mechanical and Mechatronic Engineering
The University of Sydney

March 2013

Declaration

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the University or other institute of higher learning, except where due acknowledgement has been made in the text.

John Vial

31 March 2013

Abstract

John Vial
The University of Sydney

Doctor of Philosophy
March 2013

Conservative Sparsification for Efficient Approximate Estimation

Linear Gaussian systems often exhibit sparse structures. For systems which grow as a function of time, marginalisation of past states will eventually introduce extra non-zero elements into the information matrix of the Gaussian distribution. These extra non-zeros can lead to dense problems as these systems progress through time.

This thesis proposes a method that can delete elements of the information matrix while maintaining guarantees about the conservativeness of the resulting estimate with a computational complexity that is a function of the connectivity of the graph rather than the problem dimension. This sparsification can be performed iteratively and minimises the Kullback Leibler Divergence (KLD) between the original and approximate distributions. This new technique is called Conservative Sparsification (CS).

For large sparse graphs employing a Junction Tree (JT) for estimation, efficiency is related to the size of the largest clique. Conservative Sparsification can be applied to clique splitting in JTs, enabling approximate and efficient estimation in JTs with the same conservative guarantees as CS for information matrices.

In distributed estimation scenarios which use JTs, CS can be performed in parallel and asynchronously on JT cliques. This approach usually results in a larger KLD compared with the optimal CS approach, but an upper bound on this increased divergence can be calculated with information locally available to each clique.

This work has applications in large scale distributed linear estimation problems where the size of the problem or communication overheads make optimal linear estimation difficult.

Acknowledgements

I would first like to thank my supervisors Tim Bailey and Hugh Durrant-Whyte for their patience and guidance. I would also like to thank Ian Manchester for help with solidifying and eventually simplifying my reduction proof. I would also like to thank the Australian Centre for Field Robotics for giving me the opportunity to pursue this thesis.

I would like to thank the upstairs cubicle crew including Dan Steinberg, Ariell Friedman and Asher Bender as well as Lachlan McCalman, Mark DeDouge and Alistair Quadros for their patience as I persistently brought my thesis topic to our lunch time discussions. As well as the Tariq Abuhashim, Jen Jen Chung and Nicholas Lawrence for putting up with me in our share house, and for being a sounding board for my ideas.

I wish to thank my proof readers for their valuable feedback which includes Tim Bailey, Abhinay Mukunthanan and Alanna Vial.

I would also like to thank my parents, Peter and Kathrine Vial and grandparents Brian and Marie Bourke who have supported my through this process.

Finally I would like to thank my soon to be wife Hilary Goh, for her tireless patience and support.

Contents

Declaration i

Abstract ii

Acknowledgements iii

Contents iv

1 Approximate Estimation: Efficiency and Conservativeness 1

1.1 Example: Sparsifying the 'Smart Grid' 2

1.2 Problem Statement 5

1.3 Thesis Outline 6

1.4 Contributions 6

2 Gaussian Estimation 7

2.1 Gaussian Estimation 7

2.2 Canonical Gaussian Parametrisation 9

2.2.1 Graphical Models 10

2.2.2 Updates and Augmentation 11

2.2.3 Marginalisation 12

2.2.4 Matrix Inversion and Fill-in 14

2.2.5 Efficient Mean and Covariance Recovery using Cholesky . . . 16

2.3 Estimation with Junction Trees 19

2.3.1 Fundamentals 20

2.3.2 Obtaining Mean and Covariance Estimates 23
2.3.3 Forming a Junction Tree from an Information Matrix 25
2.3.4 Incremental Estimation 29
2.3.5 Cost of the JT algorithm 35
2.3.6 Special cases of Junction Tree Inference 35
2.4 Sparse Problems 36
2.4.1 SLAM 36
2.4.2 Distributed Sensor Networks 38
2.4.3 Cooperative Localisation 40
2.5 Approximate Estimation 44
2.5.1 Approximation Quality 44
2.5.2 Improving Efficiency of Information Form 51
2.5.3 Improving Efficiency of Junction Tree Estimation 54
2.5.4 Summary 55

3 Conservative Sparsification for Information Matrices 57
3.1 Conservative Optimisation 57
3.1.1 Defining the Problem 58
3.1.2 Conservative Sparsification (CS) 60
3.2 Efficient Implementation of CS 61
3.2.1 Efficient Conservative Sparsification 61
3.2.2 Feasible Set for Optimisation 67
3.2.3 Chordal Conservative Sparsification 69
3.2.4 Sparsification and Semi-Definite Matrices 76
3.2.5 CS for Disjoint Markov Blankets 77
3.3 Experiments 80
3.3.1 Conservative Sparsification for Incremental Estimation 80
3.3.2 Linear SLAM 83
3.3.3 Smoothing SLAM example 87
3.3.4 Benchmark matrices 89

4 Conservative Sparsification for Junction Trees 92

4.1 Conservative Sparsification in Junction Trees 92

4.1.1 Conservative Sparsification 93

4.1.2 Efficient Clique Splitting 94

4.1.3 Computational Cost 98

4.1.4 Multiple Cliques 99

4.2 Parallel Sparsification 101

4.2.1 Key Assumption 101

4.2.2 Exact Clique Split 103

4.2.3 Individual Clique Splitting 104

4.2.4 Separator Splitting 108

4.2.5 Locally Computable Error Bound 110

4.2.6 Parallel Sparsification Computational Cost 110

4.3 Comparison Between JT and Information Based Sparsification 111

4.4 Linear SLAM Simulation 111

5 Conclusion and Future Work 114

5.1 Future Work 115

Bibliography 117

A Proofs 123

A.1 Useful Identities 123

A.2 Alternative Proof Approaches 125

B Details of Previous Work in Sparse Approximations 136

B.1 Approximate Sparsification for Data Fusion 136

B.1.1 Covariance Intersection and Related Approaches 136

B.1.2 Approximately Marginalised Sparse Filtering 138

B.1.3 Approximate Sparsification for SLAM 144

B.1.4 Short overview of SLAM 144
B.2 Previous Work in Sparsification for Junction Trees 146
B.2.1 Thin Junction Trees (TJT) 146
B.2.2 TreeMap 152
B.2.3 CS-JT Comparison to Chordal Embedding 155
B.2.4 CS-JT Comparison to Treemap 155
B.3 Matrix Approximation 156
B.3.1 Optimisation Based Approaches 156

C Results **158**

C.1 Extra Benchmark Matrices 158
C.2 Suite Sparse matrices, Chordal CS vs. CS comparisons 163

Glossary

clique width Also known as tree width, this is the size of the largest clique in a junction tree . 20, 34

distributed A system is considered distributed if its state is spread over multiple platforms such that no one platform contains all the information about every state. 66

fill-in Extra edges or non-zeros induced in a graph or matrix by the marginalisation of a state. 2, 143–145

Schur complement A marginal projection. Consider a partitioned matrix, the Schur complement of $\mathbf{A}_{22}$ on to $\mathbf{A}_{11}$ is $\mathbf{A}_{11} - \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{12}^{T}$. 14, 142

sparsity Amount of zero elements in a representation (graphical model or matrix). A 'high' level of sparsity implies a large number of zero valued elements of the matrix or a graphical model with a very low number of edges. 2

Acronyms

AM-SEIF Approximately Marginalised - Sparse Extended Information Filter. 52–54, 84–87, 136, 138–144, 146

CCS Chordal Conservative Sparsification. 70–75, 164–167
CI Covariance Intersection. 53, 136–138
CS Conservative Sparsification. ii, 2, 61–74, 79, 80, 84, 92–94, 98–101, 110, 111, 115, 116, 155, 158, 164–167
cspy color sparsity. 10, 90
CU Covariance Union. 136, 137

DD-SEIF Data Discarding Sparse Extended Information Filters. 53, 144, 145
DSN Distributed Sensor Network. 1, 9, 11, 38–40, 61

JT Junction Tree. ii, 6, 19–21, 23–26, 29, 30, 35, 36, 54–56, 63, 92–99, 101, 103, 108, 110–112, 114, 146, 149

KLD Kullback Leibler Divergence. ii, 44, 45, 47, 55, 58–60, 63, 71–73, 79, 80, 84–90, 92, 93, 100, 101, 105, 110–112, 114, 115, 164

MAXDET Maximum Determinant. 60, 137

SDP Semi-Definite Program. 137
SLAM Simultaneous Localisation and Mapping. 1, 9, 11, 36–40, 52, 57, 61, 80–84, 86, 87, 111, 114, 136–138, 140, 144, 146, 149, 152, 154, 155
split-CI split Covariance Intersection. 136, 137

TJT Thin Junction Tree. 36, 54, 56, 146, 148–150

Chapter 1

Approximate Estimation: Efficiency and Conservativeness

Gaussian estimation is a simple technique for inferring the value of uncertain variables known as states. It is a convenient representation as it is the most entropic two parameter distribution, while its low number of parameters and self-conjugate property make it useful for estimation of large problems [8].

Despite having few parameters there are scenarios for which Gaussian estimation can become intractable, for example, scenarios that exhibit state space growth as a function of time. Typical examples such as Simultaneous Localisation and Mappings (SLAMs) and Distributed Sensor Networks (DSNs) are described in Section 2.4. The intractability of these problems is due to increase in density (number of non-zero entries of the information matrix or edges in the graph) of the estimate through marginalisation, which is described in Section 2.2.

Given finite computer resources, these system can never be fully approximated by a multi-dimensional Gaussian distribution due to the unbounded growth of time. Designers of solutions in these settings have to make trade-offs to correctly estimate these systems. The choices available at the moment are;

- completely forget states and observations that are old,

- perform marginalisation and suffer reduced efficiency due to fill-in,
- remove links and risk a distribution that underestimates the uncertainty (i.e., an overconfident estimate), or
- place an upper bound on the system running time.

These choices are undesirable because;

- Marginalisation can quickly reduce sparsity, increasing the computational resources necessary for performing matrix solves and inversion.
- Some safety critical control tasks require conservative estimates (that is estimates which have larger uncertainty than the true uncertainty).
- Placing an upper bound on the system running time limits the long-term autonomy of the robot.

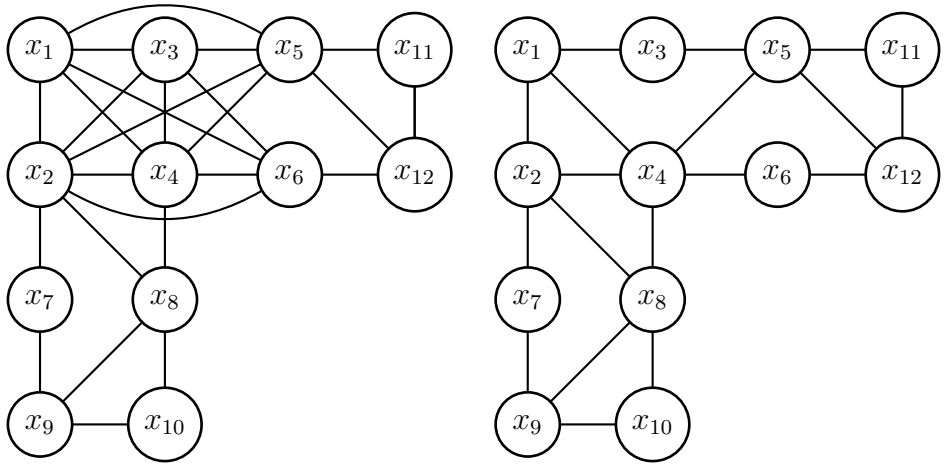
This thesis suggests an alternative to these approaches, described in Chapter 3, which is called Conservative Sparsification (CS). CS allows the removal of any edge in the graph representing the information matrix of a Gaussian distribution and guarantees that the resulting distribution will be conservative and representative.

An important setting where computational constraints are observed is distributed sensor networks. In this scenario a large number of small sensors with limited computer resources, communication bandwidth and battery power must attempt to estimate some dynamic quantity. Chapter 4 describes an approach to Conservative Sparsification that uses only local information, requires no additional message passing to be performed and can be done in an entirely asynchronous manner, albeit at the cost of higher divergence from the true distribution.

1.1 Example: Sparsifying the 'Smart Grid'

The smart grid [3] is a futuristic vision of electrical power distribution where distributed generators (using energy sources such as solar panels or wind farms) control

the power applied to the electricity network in their local area to ensure a required level of power quality. This is a useful example as it combines the estimation problem with a control problem that has strict constraints.



(a) Prior knowledge of conditional dependency structure between generators.

(b) Approximate sparse structure.

Figure 1.1 – In this distributed generation example the dependency structure between loads near each distributed generator is shown in (a). It is desired to represent this using the estimation structure shown in (b). Conservative Sparsification can be applied to (a) to transform it to the structure in (b).

For a network the size of a city or region, consider the problem of estimating the power that is required to be output by a distributed generator. Let the city in question contain 12 generators that are located in different parts of the city. Assume that some prior knowledge is available which describes the conditional dependency structure of the network loads. Figure 1.1(a) shows the structure of the conditional dependencies between loads near[1] each generator. It is desired to use this as prior knowledge so that the estimates of the voltages and currents in the network are more accurate. Also assume that the number of links in this graph make the problem difficult[2]. Upon inspection of the prior structure many of the link weights are "close"

[1] A path exists through the network from loads to every generator, therefore, the loads are shared by all the generators. However, it is more costly (in terms of line losses) for a distant generator to supply power to a local load, therefore it is desired that each generator provides enough power to supply its nearby loads.

[2] For illustration purposes this section considers a simple 12 node graph, which is not difficult for modern computers to solve. However, the techniques described in this thesis are applicable to graphs of arbitrary size.

to zero, indicating that the dependence between some loads is weak. Maintaining only the strong links results in the structure in Figure 1.1(b). This structure is much simpler and will be easier to solve given due to the increased sparsity. As the new structure maintains only strong links, it will hopefully be a close approximation to the original system. The desire to remove links from the graph could have been driven by restrictions in communication between the generators, limited computing power at each generator or a low latency requirement in computation.

Many methods exist to enforce the sparsity constraints required to transform Figure 1.1(a) to Figure 1.1(b) (see Section 2.5.2) while also minimizing the divergence between them but most fail to ensure conservatism. Conservatism is important as the generator must ensure that the power it applies to the grid will not damage users of the grid, it is discussed in Section 2.5.1.

A simple performance measure of power quality is voltage. A power company may require that the line voltage remain between 216V and 255V to prevent damage to devices on the network. Assume that generator x_1 is estimating the effect of not changing its present power output. The probability density function for the predicted line voltage from this is shown in Figure 1.2. When assessing the action, the generator needs to consider the probability that the network voltage will rise over the limit, and possibly damage devices connected to the network[3]. Given an overconfident, conservative and true estimate, the probability that the voltage will rise above 255 V is given in Table 1.1. This table shows that the overconfident estimate will underreport the probability of an over voltage, while the conservative estimate will inflate it. Thus, if the controller used this over-confident estimate it would have a 2% chance of failure but would believe that its chance was only 0.6%. Thus a control system which bases its observations on overconfident estimators is more likely to make poor decisions. Chapter 3 describes a method for performing sparsification such that the conservativeness of estimates can be enforced.

[3]It is also important to ensure an under voltage does not occur, but to keep the discussion brief under-voltages will not be considered.

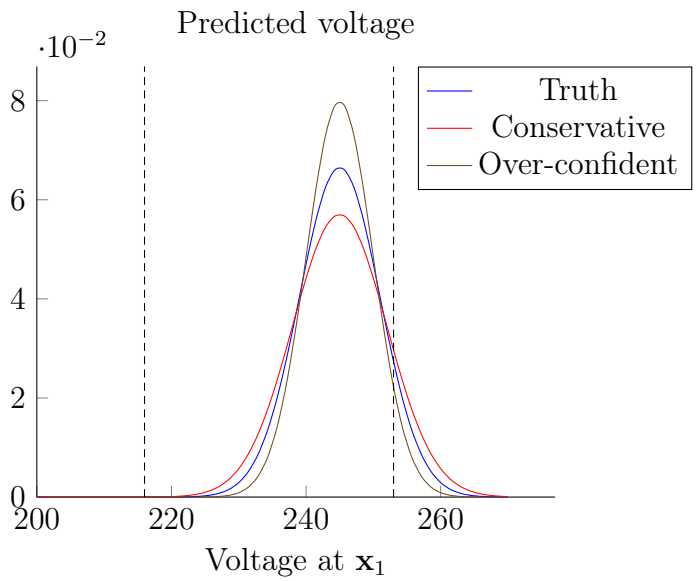


Figure 1.2 – This shows the three predicted voltage distributions that are based upon a conservative, overconfident and true estimate of the system. The two dashed lines indicate the maximum and minimum line voltages of 255V and 216V respectively. The distributions share the same mean (245V) and vary only with the standard deviation. The true standard deviation is 5V, while the conservative is 6V and the over-confident is 4V.

Name	Prob(Voltage $>$ 255)	Ratio to Truth ($\frac{p}{p_{tr}}$)
Truth	0.023	-
Conservative	0.048	2.09
Over-confident	0.006	0.27

Table 1.1 – Probability that the action proposed by generator x_1 will push the voltage over the 255V limit. In this case the conservative estimate has inflated the probability of an over-voltage by 2 times the actual, while the overconfident estimate reports only 1/4 of the actual probability.

1.2 Problem Statement

This work investigates methods of increasing the sparsity of linear Gaussians systems. This will result in approximate estimates that are:

- close to the original distribution,
- have increased sparsity, and,

- are conservative with respect to the original distribution.

The increased sparsity will allow for faster estimation and easier distributed estimation.

1.3 Thesis Outline

The outline of the thesis is as follows:

Chapter 2: Estimation This chapter describes the basic principles required to understand this work and describe some prior approaches to this problem.

Chapter 3: Conservative Sparsification for Information Matrices This chapter introduces the Conservative Sparsification approach and shows an important complexity reduction technique.

Chapter 4: Conservative Sparsification for Junction Trees This chapter describes the use of Junction Trees as a way to enable immediate marginal recovery and allow for approximate parallel sparsification.

Chapter 5: Conclusion The work is summarised.

1.4 Contributions

The key contributions of this thesis are:

- Conservative Sparsification, an approach to link removal and a complexity reduction theorem.
- Conservative Sparsification applied to a Junction Tree (JT).
- Parallel CS using Junction Trees, a formulation that allows for completely decentralised sparsification but trades off accuracy.

Chapter 2

Gaussian Estimation

This thesis explores problems that exhibit a particular structure, these problems:

- consist of large number of unknown states to be estimated;
- operate over a very long time horizon;
- are well approximated by a Gaussian distribution;
- continually observe new information;
- have a dynamic state space size (i.e., it can grow or shrink); and
- have states that are conditionally dependent on a small number of other states.

There are many examples of problems that exhibit this structure, some of which are described in more detail in Section 2.4. The following sections will describe computationally efficient approaches for the estimation of these systems and consider methods for the approximate estimation of these systems.

2.1 Gaussian Estimation

This section presents a brief overview of the fundamentals of estimation. For a more complete treatment see [8][39]. Let the vector $\mathbf{x}$ contain a set of states of interest

(e.g. Cartesian coordinates of the location of objects, temperature in a room, noisy observations of these quantities), which have an inherent uncertainty about them. This uncertainty is expressed with a probability density function (pdf), i.e. $p(\mathbf{x})$. The pdf $p(\mathbf{x})$ has the property

$$\int_{-\infty}^{\infty} p(\mathbf{x})dx = 1. \tag{2.1}$$

The vector $\mathbf{x}$ can be split into two vectors $\mathbf{x}_1$ and $\mathbf{z}_1$ and the joint distribution over these can be expressed as

$$p(\mathbf{x}) = p(\mathbf{x}_1, \mathbf{z}_1). \tag{2.2}$$

The marginal distribution $p(\mathbf{x}_1)$ is obtained through the use of the sum rule on the joint and allows the removal of states $\mathbf{z}_1$ from the estimate, i.e.

$$p(\mathbf{x}_1) = \int_{-\infty}^{\infty} p(\mathbf{x}_1, \mathbf{z}_1)d\mathbf{z}_1. \tag{2.3}$$

The product or chain rule also allows the joint to be decomposed in terms of a marginal and a conditional distribution, that is

$$p(\mathbf{x}_1, \mathbf{z}_1) = p(\mathbf{x}_1|\mathbf{z}_1)p(\mathbf{z}_1). \tag{2.4}$$

The product rule can be expressed using the alternate factorisation

$$p(\mathbf{x}_1, \mathbf{z}_1) = p(\mathbf{z}_1|\mathbf{x}_1)p(\mathbf{x}_1). \tag{2.5}$$

Rearranging this equation gives the definition of a conditional distribution

$$p(\mathbf{x}_1|\mathbf{z}_1) = \frac{p(\mathbf{x}_1, \mathbf{z}_1)}{p(\mathbf{z}_1)}. \tag{2.6}$$

Combining both variants of the product rule results in Bayes theorem,

$$p(\mathbf{x}_1|\mathbf{z}_1) = \frac{p(\mathbf{z}_1|\mathbf{x}_1)p(\mathbf{x}_1)}{p(\mathbf{z}_1)}. \tag{2.7}$$

Bayes theorem can be used to combine a likelihood function $p(\mathbf{z}_1|\mathbf{x})$ with a prior estimate $p(\mathbf{x})$, and a normalising distribution $p(\mathbf{z}_1)$. This thesis only considers Gaussian probability density functions.

A multi-dimensional Gaussian distribution is a probability density function over a vector of unknown states $\mathbf{x}$, $p(\mathbf{x}) = \mathcal{N}(\mathbf{x}; \hat{\mathbf{x}}, \mathbf{\Sigma})$. The covariance form of the distribution has two sufficient statistics, a vector of mean estimates $\hat{\mathbf{x}}$ (of length D) and a covariance matrix $\mathbf{\Sigma}$ (of size $D \times D$). The equation for a Gaussian distribution is shown below:

$$\mathcal{N}(\mathbf{x}; \hat{\mathbf{x}}, \mathbf{\Sigma}) = \frac{1}{(2\pi)^{D/2}} \frac{1}{\det(\mathbf{\Sigma})^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \hat{\mathbf{x}})^T \mathbf{\Sigma}^{-1} (\mathbf{x} - \hat{\mathbf{x}})\right). \tag{2.8}$$

2.2 Canonical Gaussian Parametrisation

An alternative representation for the Gaussian distribution is the canonical or information form. This is parametrised by a real valued $D \times D$ positive definite information matrix $\mathbf{Y}$ and an information (row) vector $\mathbf{y}$ also of size D.

$$\mathcal{N}^{-1}(\mathbf{x}; \mathbf{y}, \mathbf{Y}) = \frac{1}{(2\pi)^{D/2}} \frac{1}{\det(\mathbf{Y})^{-1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{Y}^{-1}\mathbf{y})^T \mathbf{Y} (\mathbf{x} - \mathbf{Y}^{-1}\mathbf{y})\right). \tag{2.9}$$

The information and covariance representations are related through the following equations:

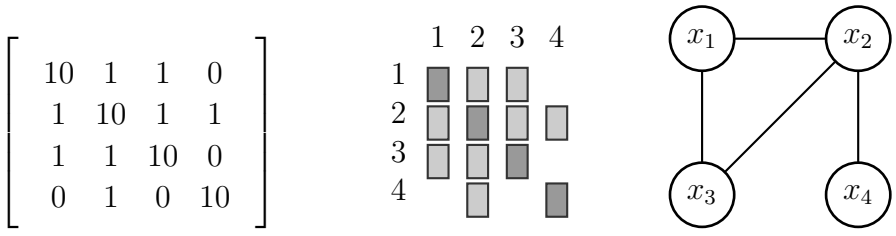
$$\mathbf{\Sigma} = \mathbf{Y}^{-1}, \tag{2.10}$$

$$\mathbf{x} = \mathbf{Y}^{-1}\mathbf{y}. \tag{2.11}$$

For estimation problems such as the SLAM Problem (Section 2.4.1) and DSN (Section 2.4.2), the information matrix is sparse (that is a high number of zero valued entries) while the equivalent covariance matrix is dense. The following subsections illustrate a graphical approach to describing the structure of sparse matrices, and explain methods of exploiting sparse structures to increase efficiency.

2.2.1 Graphical Models

A sparse matrix is a matrix that has many zero valued elements. This can be depicted graphically using a color sparsity (cspy) plot [14], as seen in Figure 2.1(a). This cspy plot highlights the location of non-zero elements in the matrix using colored squares. Darker colours indicate larger magnitude elements. Figure 2.1(b) demonstrates an alternate representation using an undirected graphical model (or Markov random field). The diagonal elements of the matrix are represented by labeled vertexes while any non-zero off-diagonal elements are shown by edges between vertexes. If a graph does not contain a vertex, then the row and column of the information matrix associated with that state is zero. The graphs used in this thesis will not represent the magnitude of an edge, rather just its existence. If any set of states are connected such that they form a complete sub-graph[1], that set is known as a clique. Figure 2.1(a) contains a clique between the states x_1, x_2 and x_3. The *cardinality* of a vertex is the number of edges attached to it. For example, in Figure 2.1(b), vertex x_2 has a cardinality of three. In this thesis cspy plots and graphs represent the sparsity pattern of information matrices.



(a) Information Matrix **(b)** CSPY Plot **(c)** Graphical Model

Figure 2.1 – Example of mapping from an information matrix (a), to a cspy plot (b), to equivalent graphical model (c), a light grey square indicates nonzero elements in the information matrix which correspond to conditional dependency links in the graph.

[1] A graph is called *complete* if no extra edges can be added to it.

2.2.2 Updates and Augmentation

New observations are of the form $\mathbf{z} = \mathbf{H}\mathbf{x} + \mathbf{r}$, where $\mathbf{z}$ is the actual observation, $\mathbf{H}$ is a linear model which describes a mapping between the states $\mathbf{x}$ and the observation $\mathbf{z}$, $\mathbf{r}$ is the zero mean additive Gaussian noise with covariance $\mathbf{R}$ and $\mathbf{x}$ is the true value of the states from which the observation is derived. $\hat{\mathbf{x}}$ is the mean estimate of the Gaussian distribution. For large problems, observations are usually dependent upon a small subset of the total states $\mathbf{x}$.

Since the observations are affected by noise, their actual value is uncertain and a joint distribution $p(\mathbf{x}, \mathbf{z})$ can be formed by augmenting $p(\mathbf{x})$. Consider the following example parameters for $p(\mathbf{x})$,

$$\mathbf{Y} = \begin{bmatrix} \mathbf{Y}_{11} & \mathbf{Y}_{12} & \mathbf{0} \\ \mathbf{Y}_{12}^T & \mathbf{Y}_{22} & \mathbf{Y}_{23} \\ \mathbf{0} & \mathbf{Y}_{23}^T & \mathbf{Y}_{33} \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \mathbf{y}_3 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \end{bmatrix}. \tag{2.12}$$

If the observation was only a function of the states in $\mathbf{x}_3$, then $p(\mathbf{x}, \mathbf{z})$ will have the following information matrix and vector [5]:

$$\mathbf{Y} = \begin{bmatrix} \mathbf{Y}_{11} & \mathbf{Y}_{12} & \mathbf{0} & \mathbf{0} \\ \mathbf{Y}_{12}^T & \mathbf{Y}_{22} & \mathbf{Y}_{23} & \mathbf{0} \\ \mathbf{0} & \mathbf{Y}_{23}^T & \mathbf{Y}_{33} + \mathbf{H}^T\mathbf{R}^{-1}\mathbf{H} & -\mathbf{H}^T\mathbf{R}^{-1} \\ \mathbf{0} & \mathbf{0} & -\mathbf{R}^{-1}\mathbf{H} & \mathbf{R}^{-1} \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \mathbf{y}_3 + \mathbf{H}^T\mathbf{R}^{-1}\mathbf{z} \\ \mathbf{R}^{-1}\mathbf{z} \end{bmatrix}. \tag{2.13}$$

Observe that this operation has increased the dimension of the problem and only affected values in the information matrix and vector associated with $\mathbf{x}_3$. An example of augmentation can be found in Figure 2.3. In many practical scenarios such as SLAM and DSN (Section 2.4), the joint over $\mathbf{x}$ and $\mathbf{z}$ is not maintained and the conditional distribution $p(\mathbf{x}|\mathbf{z})$ can be used instead.

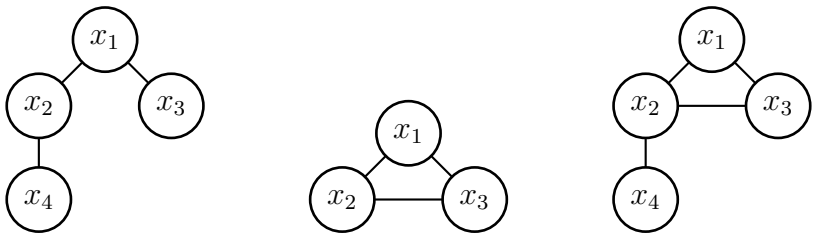
A useful property of the canonical form is that conditional distributions can be formed by simply removing the row and column associated with the conditioned variable [8].

For graphical models this corresponds to removing the vertex being conditioned upon and any connecting edges. Therefore, $p(\mathbf{x}|\mathbf{z})$ has the following parameters:

$$\mathbf{Y} = \begin{bmatrix} \mathbf{Y}_{11} & \mathbf{Y}_{12} & \mathbf{0} \\ \mathbf{Y}_{12}^T & \mathbf{Y}_{22} & \mathbf{Y}_{23} \\ \mathbf{0} & \mathbf{Y}_{23}^T & \mathbf{Y}_{33} + \mathbf{H}^T\mathbf{R}^{-1}\mathbf{H} \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \mathbf{y}_3 + \mathbf{H}^T\mathbf{R}^{-1}\mathbf{z} \end{bmatrix}. \tag{2.14}$$

Examining this result, it can be seen that forming the conditional distribution based on an observation is an additive step for an information matrix and vector. An example of the update process is shown in Figure 2.2, where the update is treated as an additive term related to a particular set of states.

The augmentation operation described in Equation 2.13 is not specific to observations and can be used to add new states of interest to the estimation problem [5]. Figure 2.3 demonstrates a scenario where a new state (x_5) is augmented but not conditioned upon and therefore remains in the network for future updates.

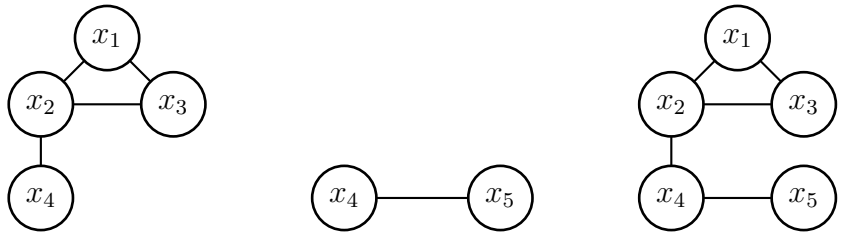


(a) Prior Graph **(b)** Structure of Update **(c)** Resultant Graph

Figure 2.2 – Graphical Model of an update step (Equation 2.14). (a) shows the original graph structure. (b) is the structure of the observation, it connects the states (x_1, x_2, x_3). (c) displays the final structure, observe that pre-exisiting links are not removed, but new links can be induced by this operation.

2.2.3 Marginalisation

Marginalisation is the process of removing states from the system. Consider Equation 2.12. Let the states in $\mathbf{x}_3$ be selected for marginalisation. The marginal $\mathbf{Y}_m, \mathbf{y}_m$ has



(a) Prior Graph **(b)** Structure of Update **(c)** Resultant Graph

Figure 2.3 – Graphical model demonstrating augmentation (Equation 2.13). (a) shows the prior graph, while (b) shows an update which references a state that is not in (a). The resultant graph is shown in (c). Observe that a new vertex has been added and that no other part of the graph was affected.

the following form [6]:

$$\mathbf{Y}_m = \begin{bmatrix} \mathbf{Y}_{11} & \mathbf{Y}_{12} \\ \mathbf{Y}_{12}^T & \mathbf{Y}_{22} - \mathbf{Y}_{23}\mathbf{Y}_{33}^{-1}\mathbf{Y}_{23}^T \end{bmatrix}, \quad \mathbf{y}_m = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 - \mathbf{Y}_{23}\mathbf{Y}_{33}^{-1}\mathbf{y}_3 \end{bmatrix}, \tag{2.15}$$

where the expression $\mathbf{Y}_{22} - \mathbf{Y}_{23}\mathbf{Y}_{33}^{-1}\mathbf{Y}_{23}^T$ is known as the Schur complement of $\mathbf{x}_3$ [51]. Another example of this process is depicted in Figure 2.4. Observe that marginalisation only affects states that are connected to the removed states. This thesis uses a shorthand for marginalisation:

$$\mathbf{Y}_m = \mathbf{Y}/\ \mathbf{Y}^{*3}, \tag{2.16}$$

$$\mathbf{y}_m = \mathbf{y}/\ \mathbf{y}^{*3}. \tag{2.17}$$

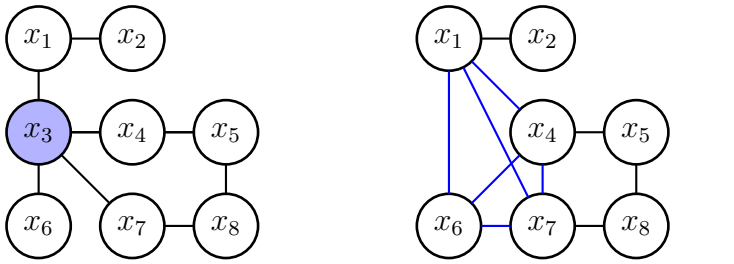
States after the * in the superscript indicate the states to be marginalised.

Matrix inversion and marginalisation are related problems, this can be shown by considering the 2×2 matrix inversion lemma (Eq, 14.11(1) [51]):

Lemma 2.1. *Let* $\mathbf{Y}$ *be partitioned as* $\mathbf{Y} = \begin{bmatrix} \mathbf{Y}_{11} & \mathbf{Y}_{12} \\ \mathbf{Y}_{12}^T & \mathbf{Y}_{22} \end{bmatrix}$, *then* $\mathbf{Y}^{-1}$ *is defined as:*

$$\mathbf{Y}^{-1} = \begin{bmatrix} (\mathbf{Y}/\ \mathbf{Y}^{*2})^{-1} & -\mathbf{Y}_{11}^{-1}\mathbf{Y}_{12}(\mathbf{Y}_{22}^{-1} - \mathbf{Y}_{21}\mathbf{Y}_{11}^{-}1\mathbf{Y}_{21}^{T}) \\ (-\mathbf{Y}_{11}^{-1}\mathbf{Y}_{12}(\mathbf{Y}_{22}^{-1} - \mathbf{Y}_{21}\mathbf{Y}_{11}^{-}1\mathbf{Y}_{21}^{T}))^{T} & \mathbf{Y}_{22}^{-1} + \mathbf{Y}_{22}^{-1}\mathbf{Y}_{12}^{T}(\mathbf{Y}/\ \mathbf{Y}^{*2})^{-1}\mathbf{Y}_{12}\mathbf{Y}_{22}^{-1} \end{bmatrix} \tag{2.18}$$

In the top left of the partitioned matrix in Equation 2.18, the inverse is equal to the inverse of the Schur complement. Thus, the inverse of this smaller marginal provides all the information required to find the covariance over the states in $\mathbf{x}_1$. For large matrices the marginalisation can be performed several times until the top left block contains very few or even a single element, for which inversion is trivial. This illustrates the relationship between marginalisation, the inverse and the Schur complement. That is, a large matrix inversion problem can be reduced to a series of matrix marginalisation problems.



(a) Prior Graph **(b)** Marginalised Graph

Figure 2.4 – Marginalisation example, showing the structure of removing state x_3 (indicated in blue). (a) shows the graph prior to marginalisation, (b) shows the graph after marginalisation. Blue edges indicate fill-in. Observe that only neighbours of x_3 are effected by this operation.

2.2.4 Matrix Inversion and Fill-in

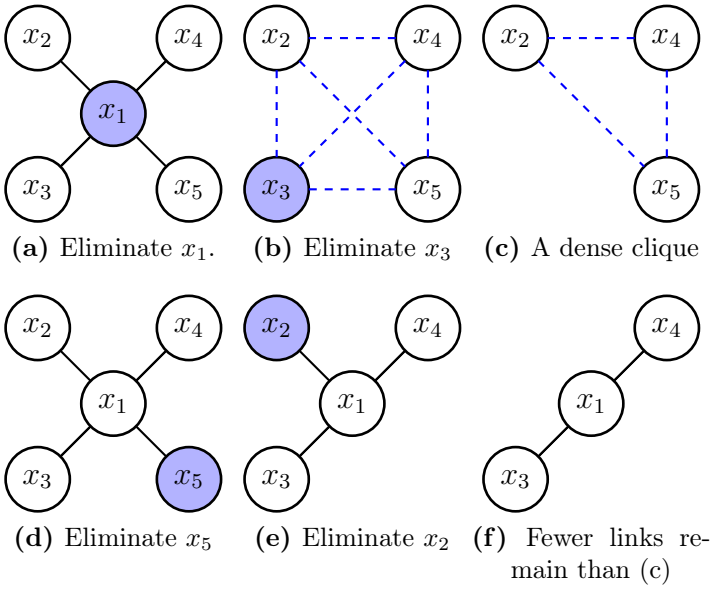
A significant disadvantage of using the information form is that the mean and covariance must be derived from the information matrix and vector using Equation 2.10.

This requires solving the linear equation $\mathbf{y} = \mathbf{Y}\mathbf{x}$ for $\mathbf{x}$ and inverting the matrix $\mathbf{Y}$ to form the covariance, which has a worst case time complexity of $\mathcal{O}(n^3)$ when the matrix is dense (no or few zero valued entries).

If $\mathbf{Y}$ is sparse, then sparse matrix techniques can be used to greatly improve upon $\mathcal{O}(n^3)$[14]. Matrix inversion on sparse matrices is equivalent to incremental matrix marginalisation (Section 2.2.3) and marginalisation results in fill-in as shown in Figure 2.4. This fill-in increases the number of non-zero entries in the graph, making it less sparse.

Fill-in is dependent upon the order that vertices are removed from the graph. This order is called the elimination ordering. Figure 2.5 shows two different elimination ordering choices, and their respective graphs. Ordering choices can greatly affect the amount of fill-in. For example in Figure 2.5(b) eight fill-in links have been introduced, while in Figure 2.5(e) zero fill-in links have been introduced. For any graph, there exists an elimination ordering that results in the minimum amount of fill-in. However, finding such an ordering is an NP-Complete problem for general graphs [65]. There exist good heuristics that can find fill-reducing matrix orderings. An example is the approximate minimum degree (amd) ordering [2].

For a special class of graphs, known as chordal graphs, a perfect elimination ordering (i.e. one that induces no fill-in) can be found in $\mathcal{O}(n+e)$ time (where e is the number of edges in the graph, and n is the number of vertices)[50]. A chordal graph has an edge between every cycle of size 4 or greater in the graph, where a 4-cycle is a loop of 4 vertices. Figure 2.6 demonstrates a chordal graph. Any graph can be transformed to a chordal graph by introducing the edges induced through fill-in into the original graph. This process is known as triangulation. Such a graph is known as a triangulated graph, and an example is illustrated in Figure 2.7.



(a) Eliminate x_1. **(b)** Eliminate x_3 **(c)** A dense clique

(d) Eliminate x_5 **(e)** Eliminate x_2 **(f)** Fewer links remain than (c)

Figure 2.5 – In this example the marginal of x_4 is desired. The first two steps of the elimination process is shown for two different elimination orderings is shown. Fill-in is shown by dashed blue links, a blue shaded node indicates the next node to be eliminated. (a-c) demonstrate a poor ordering choice that results in a large amount of fill-in. (d-f) demonstrate a much better ordering that results in zero fill-in edges. This can be seen by comparing the number of edges in (b) to (e) and (c) to (f).

2.2.5 Efficient Mean and Covariance Recovery using Cholesky

A Cholesky factor is a unique lower triangular matrix $\mathbf{L}$ which satisfies the following equation:

$$\mathbf{Y} = \mathbf{L}\mathbf{L}^T \tag{2.19}$$

Given a Cholesky factor, a linear system can be decomposed:

$$\mathbf{x} = \mathbf{Y}^{-1}\mathbf{y} \tag{2.20}$$

$$= (\mathbf{L}\mathbf{L}^T)^{-1}\mathbf{y} \tag{2.21}$$

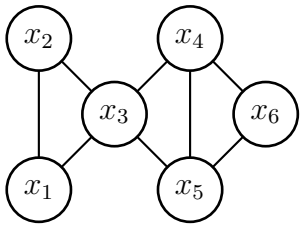
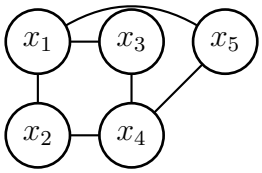
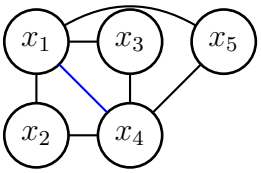


Figure 2.6 – Example of a chordal graph. This graph has a perfect elimination ordering, that is, an ordering which introduces no fill-in. For this graph a perfect elimination ordering is $x_1, x_2, x_3, x_4, x_5, x_6$. Observe the 4-cycle between x_3, x_4, x_6 and x_5 has a chord between x_4 and x_5, a requirement of a chordal graph.



(a) Non-chordal graph



(b) Triangulated graph

Figure 2.7 – Illustration of transformation from a non-chordal graph (a) to a chordal graph (b) through triangulation. The elimination ordering used was x_5, x_1, x_2, x_3, x_4. The blue edge in (b) was induced by this elimination ordering, by adding it to the graph in (a), a chordal graph has been created. Since the graph was made chordal by using an elimination ordering, it is also called the triangulated graph of (a).

With this decomposition, the solution can be found by solving two simpler systems of equations. The first system is (solve for $\mathbf{t}$):

$$\mathbf{L}\mathbf{t} = \mathbf{y}, \tag{2.22}$$

while the second linear system is (solve for $\mathbf{x}$):

$$\mathbf{L}^T\mathbf{x} = \mathbf{t}. \tag{2.23}$$

Since both matrices are triangular, each solve has cost $\mathcal{O}(|\mathbf{L}|)$ [2][14]. The sparsity of the Cholesky factor is related to the fill-in of the elimination ordering. Figure 2.8 shows the fill-in pattern of an example Cholesky decomposition. Observe that fill-in edges appear as extra non-zeros in the Cholesky factor. The columns of the Cholesky

[2]Here $|\mathbf{A}|$ refers to the number of non-zeros in A.

factor also represent cliques in the triangulated form of the underlying graph.

To calculate the Cholesky decomposition consider (c.f. [14]):

$$\begin{bmatrix} \mathbf{L}_{11} & \mathbf{0} \\ \mathbf{l}_{12}^T & l_{22} \end{bmatrix} \begin{bmatrix} \mathbf{L}_{11} & \mathbf{l}_{12} \\ \mathbf{0} & l_{22} \end{bmatrix} = \begin{bmatrix} \mathbf{Y}_{11} & \mathbf{m}_{12} \\ \mathbf{m}_{12}^T & m_{22} \end{bmatrix}, \tag{2.24}$$

where $\mathbf{m}_{12}$ and $\mathbf{l}_{12}$ are a vectors[3], while m_{22} and l_{22} are scalars. This leads to the following equations:

$$\mathbf{L}_{11}\mathbf{L}_{11}^T = \mathbf{Y}_{11} \tag{2.25}$$

$$\mathbf{L}_{11}\mathbf{l}_{12} = \mathbf{m}_{12} \tag{2.26}$$

$$\mathbf{l}_{12}^T\mathbf{l}_{12} + l_{22}^2 = \mathbf{m}_{22} \tag{2.27}$$

The Cholesky factorisation can be found by iteratively solving the above equations as shown in Algorithm 2.1. Note that Algorithm 2.1 makes use of MATLAB style matrix indexing[4].

Algorithm 2.1: Up-looking Cholesky factorisation, c.f. [14].

input : $\mathbf{Y}$: Matrix to be factored.
output: $\mathbf{L}$: Cholesky factor of $\mathbf{Y}$.
n := dimension of $\mathbf{Y}$
for *k = 1 to n* **do**
  $\mathbf{L}(k, 1:k-1) = \mathbf{L}(1:k-1, 1:k-1)^{-1}\mathbf{Y}(1:k-1, k)^T$
  $\mathbf{L}(k,k) = \sqrt{\mathbf{Y}(k,k) - \mathbf{L}(k, 1:k-1)\mathbf{L}(k, 1:k-1)^T}$

Computation of the Cholesky factor can be done in $\mathcal{O}(\sum_{k=1}^{n} l_k^2)$, where l_k is the number of non-zeros in row k [14]. Thus the total time complexity to solve the system in Equation 2.21 is $\mathcal{O}(2|\mathbf{L}| + \sum_{k=1}^{n} l_k^2)$, which for sparse matrices is dominated by $\mathcal{O}(\sum_{k=1}^{n} l_k^2)$.

[3]The vector $\mathbf{m}$ was introduced to avoid confusion with the information vector $\mathbf{y}$.

[4]Matlab indexing uses a colon ':' to indicate a range, while the ranges in brackets after the matrix name indicate which rows and columns are indexed. For example, $\mathbf{Y}(1:k, k)$ accesses the rows 1 to k of $\mathbf{Y}$, while only accessing column k. Resulting in a column vector of length k.

These complexities assume a very sparse matrix and the time complexity increases as $\mathbf{Y}$ becomes more dense. This can be shown by considering the Cholesky factorisation cost if $\mathbf{Y}$ is dense. Begin by expanding the sum of the row counts:

$$\mathcal{O}(\sum_{k=1}^{n} l_k^2) = \mathcal{O}(n^2 + (n-1)^2 + \ldots + (n-(n-1))^2), \tag{2.28}$$

$$= \mathcal{O}(nn^2 - 2n(1 + 2 + \ldots + (n-1)) + 1^2 + 2^2 + \ldots + (n-1)^2) \tag{2.29}$$

$$= \mathcal{O}(n^3), \tag{2.30}$$

which is the same time complexity as performing a matrix inverse.

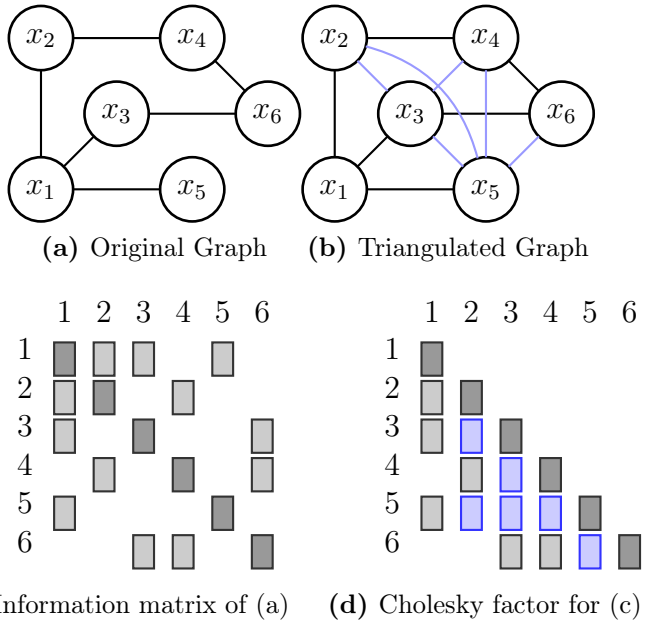
For sparse Choleksy factors Section 6.7.4 of [9] describes an algorithm that can compute the value of elements in the covariance matrix in $\mathcal{O}(nl^2)$, where l is the maximum number of elements on any row of $\mathbf{L}$. This algorithm is restricted to only elements which correspond to non-zeros in the information matrix. This is also known as the Takahashi equations and is described in [46][53][40].

If the updates are of low rank (i.e, are with respect to a small number of states), then a low-rank update can be applied directly to the Cholesky factor such that the information matrix need not be formed [14]. This was implemented in a decentralised estimation scenario in [44][6].

Many software packages that perform sparse linear algebra exist including MATLAB[43], the SciPy package for Python [27] and the Eigen library for C++ [24].

2.3 Estimation with Junction Trees

An equivalent approach to estimation is to distribute the information matrix and vector using a Junction Tree (JT). Rather than employing large sparse linear algebra to find the mean and covariance, Junction Trees employ the message passing algorithm. This algorithm is a standard algorithm for inference in graphical models and is also easily decentralised [42].



(a) Original Graph **(b)** Triangulated Graph

(c) Information matrix of (a) **(d)** Cholesky factor for (c)

Figure 2.8 – This figure shows the relationship between the fill-in and the Cholesky factor. Blue links indicate a fill in link between nodes, while a blue square indicates a non-zero element due to fill-in. The elimination ordering for the graph in (a) is $x_1, x_2, x_3, x_4, x_5, x_6$. The fill-in induced by this ordering is apparent in the Cholesky factor (d).

2.3.1 Fundamentals

Junction Trees (JTs) are collections of cliques connected in a tree structure. These cliques represent densely connected components of the underlying triangulated graph of the information matrix. To construct a JT from an information matrix, see Section 2.3.3. The size of the largest clique is known as the clique width or tree width (w). The cliques must be connected such that the running intersection property is maintained. This property is defined below:

Definition 2.1 (Running Intersection Property [28])**.** *A clique tree possesses the running intersection property if for every pair of cliques $\mathcal{C}_i$ and $\mathcal{C}_j$, all cliques on the unique path between $\mathcal{C}_i$ and $\mathcal{C}_j$ contain $\mathcal{C}_i \cap \mathcal{C}_j$.*

Given the set of cliques, a maximal spanning tree can be constructed that uses the size of the intersection between clique sets as the link weight. Using this approach

will always result in a clique tree that satisfies the running intersection property (see Ch. 17 Th. 5 [28]).

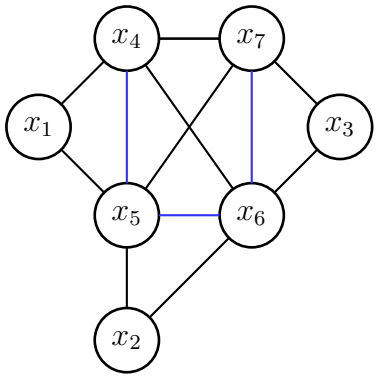
Two cliques are neighbours if they share an edge in the JT. Neighbouring cliques must have at least one overlapping state. The set of states that are shared by neighbouring cliques $\mathcal{C}_i$ and $\mathcal{C}_j$ is known as the separator set $\mathcal{S}_{ij}$, and is defined $\mathcal{S}_{ij} = \mathcal{C}_i \cap \mathcal{C}_j$. Each clique $\mathcal{C}_i$ must maintain a potential over its states, denoted Ψ_i. This potential is not strictly a probability, but all the potentials of a JT must satisfy the following property:

$$p(\mathbf{x}) \propto \prod_i \Psi_i, \tag{2.31}$$

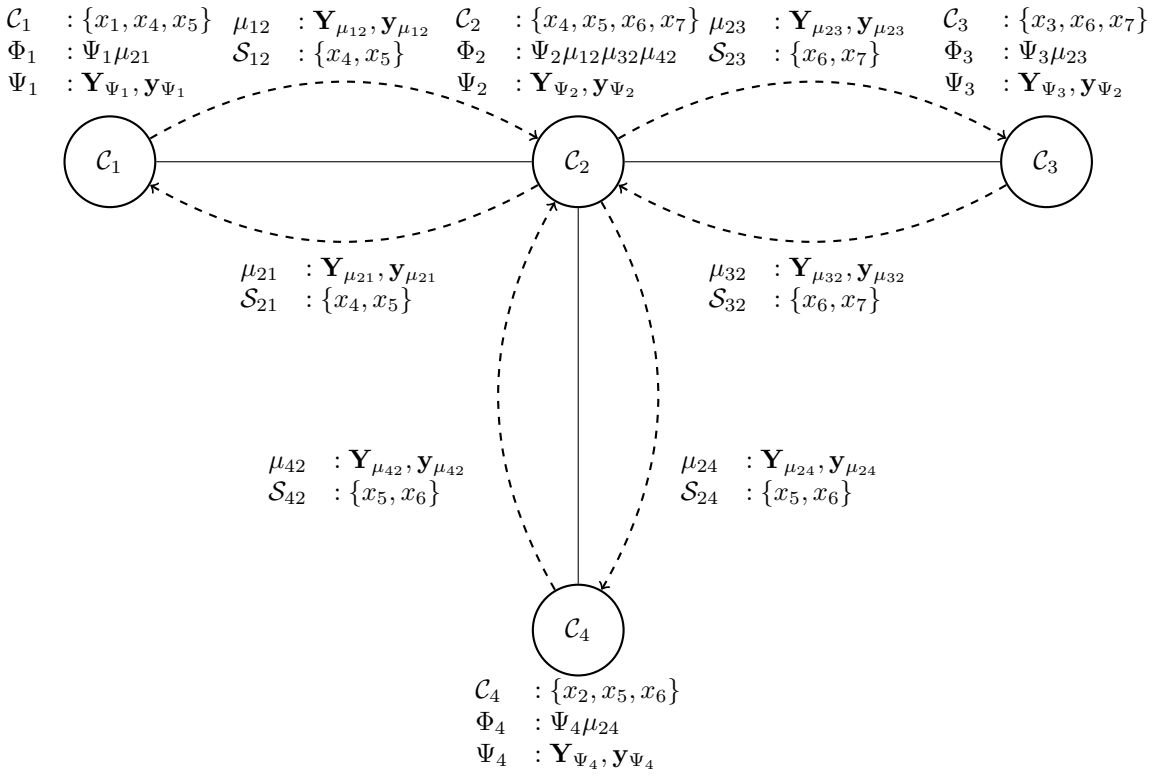
that is the joint pdf over all states must equal the product of the clique potentials. If the joint is Gaussian and each potential Ψ_i is parametrised by an information matrix ($\mathbf{Y}_{\Psi_i}$) and vector ($\mathbf{y}_{\Psi_i}$), then Equation 2.31 can be written in terms of the joint information matrix $\mathbf{Y}$ and vector $\mathbf{y}$:

$$\mathbf{Y} = \sum_i \mathbf{Y}_{\Psi_i}, \quad \mathbf{y} = \sum_i \mathbf{y}_{\Psi_i}. \tag{2.32}$$

Figure 2.9 shows an example JT that demonstrates the notation associated with cliques. Note that the quantity Φ_i displayed in this graph is the marginal over the clique states $\mathcal{C}_i$, and is described in Section 2.3.2. The μ subscript refers to message potentials described in Section 2.3.2, while the $\mathcal{S}$ notation refers to clique separators which is described above. Note that it is *not* a requirement that a particular matrix/vector pair $(\mathbf{Y}_{\Psi_i}, \mathbf{y}_{\Psi_i})$ is valid for a Gaussian distribution. This also means that the matrix $\mathbf{Y}_{\Psi_i}$ is not necessarily positive definite.



(a) Triangulated Graph, showing fill-in.



(b) Annotated JT

Figure 2.9 – Annotated transformation between a graphical model (a) and a junction tree (b). The junction tree is formed using the elimination ordering $\{x_1, x_2, x_3, x_4, x_5, x_6, x_7\}$, which induces fill-in (blue edges) shown in (a). Information from this graph is spread into the four clique potentials, each potential information matrix and vector is only over the states in the cliques. For example $\mathbf{Y}_{\Psi_1}$ is a 3×3 matrix over $\{x_1, x_4, x_5\}$. The messages μ and separator sets $\mathcal{S}$ between cliques are also shown. Similarly to the clique potentials, these matrices and vectors are of the size of the seperator sets. The direction of each message is also indicated.

2.3.2 Obtaining Mean and Covariance Estimates

To find the mean and covariance, a JT uses the message passing algorithm. Messages are sent between cliques in a directed fashion. The message from $\mathcal{C}_i$ to $\mathcal{C}_j$ is:

$$\mu_{ij} = \int_{-\infty}^{\infty} \Psi_i \prod_k \mu_{ki} d(\mathcal{C}_i \notin \mathcal{S}_{ij}), \tag{2.33}$$

where k includes any messages already received by clique i excluding any sent from j. That is, a message is formed by marginalising the product of the clique potential and any other messages to only the states that are in the destination clique. In the case of cliques that are leaves of the tree (i.e., only 1 neighbour) there are no other messages. For the Gaussian case, the messages can be parametrised by an information matrix $\mathbf{Y}_{\mu_{ij}}$ and vector $\mathbf{y}_{\mu_{ij}}$, such that Equation 2.33 is equivalent to:

$$\mathbf{Y}_{\mu_{ij}} = (\mathbf{Y}_{\Psi_i} + \sum_k \mathbf{Y}_{\mu_{ki}})^{*\mathcal{C}_i \notin \mathcal{S}_{ij}}, \quad \mathbf{y}_{\mu_{ij}} = (\mathbf{y}_{\Psi_i} + \sum_k \mathbf{y}_{\mu_{ki}})^{*\mathcal{C}_i \notin \mathcal{S}_{ij}}, \tag{2.34}$$

which uses the shorthand for marginalisation described in Section 2.2.3, and k is as defined in Equation 2.33.

Algorithm 2.2 demonstrates the message passing algorithm at a high level. It begins by identifying a particular clique to be designated as root. Then, for each leaf it performs the DistributeMessages function shown in Algorithm 2.3. This algorithm sends a message to a neighbouring clique only if it has at least 1 neighbour that has not sent it a message. These messages eventually propagate to the root node. For example, in Figure 2.9 if $\mathcal{C}_1$ is the root node, $\mathcal{C}_3$ and $\mathcal{C}_4$ are leaves. DistributeMessages is first run on $\mathcal{C}_3$. This sends a message to $\mathcal{C}_2$. However, since $\mathcal{C}_2$ has yet to receive two other messages from its neighbours, DistributeMessages ends. DistributeMessages is then called on $\mathcal{C}_4$, which sends another message to $\mathcal{C}_2$. $\mathcal{C}_2$ has now received messages from all but one neighbour, so it sends a message to clique $\mathcal{C}_1$. The final call to Distribute Messages on the root node transmits the final message in the other direction to the other cliques. Once this occurs, a message is sent to $\mathcal{C}_2$, which has now received messages from every neighbour. It sends messages to each of its neighbours with the

exception of the parent (in this case $\mathcal{C}_1$). These messages arrive at $\mathcal{C}_3$ and $\mathcal{C}_4$ and, as they have no neighbours other than their parent ($\mathcal{C}_2$), DistributeMessages ends. After this process, every clique has received a message from every one of its neighbours.

Any clique $\mathcal{C}_i$ is now capable of calculating the marginal ($\Phi_i = p(\mathbf{x}_{\mathcal{C}_i})$) over its states using information available locally. This is done using the following Equation:

$$\Phi_i = \Psi_i \prod_k \mu_{ki}, \tag{2.35}$$

where k contains all the neighbours of $\mathcal{C}_i$. That is, the marginal over the states in $\mathcal{C}_i$ can be constructed using the clique potential and the messages sent from neighbouring cliques. For Gaussian distributions, the marginal is parametrised by the information matrix $\mathbf{Y}_{\Phi_i}$ and vector $\mathbf{y}_{\Phi_i}$, such that Equation 2.35 becomes:

$$\mathbf{Y}_{\Phi_i} = \mathbf{Y}_{\Psi_i} + \sum_k \mathbf{Y}_{\mu_{ki}}, \quad \mathbf{y}_{\Phi_i} = \mathbf{y}_{\Psi_i} + \sum_k \mathbf{y}_{\mu_{ki}}. \tag{2.36}$$

Figure 2.9(b) shows the form of the messages and marginals on an example Junction Tree.

The message passing algorithm can also be used in decentralised estimation. In this case, Algorithm 2.3 can be continuously performed as new updates occur to the system. This allows information to propagate through the JT without requiring any node to be chosen as root [42].

Algorithm 2.2: MessagePassing

input : A junction tree, $\mathcal{T}$
output: A junction tree, $\mathcal{T}$, with marginals available for every clique
begin
  $\mathcal{L}$ = all leaves of tree $\mathcal{T}$
  Choose one leaf to be root r
  Remove r from $\mathcal{L}$
  foreach $i \in \mathcal{L}$ **do**
    DistributeMessages($\mathcal{L}_i$)
  DistributeMessages(r)

Algorithm 2.3: Distribute Messages, c.f. [28]

input: A clique of a junction tree, $\mathcal{C}_i$
input: parent clique, $\mathcal{C}_p$
if ≤ 1 *neighbours have not yet sent a message to* $\mathcal{C}_i$ **then**
  foreach *Neighbour* $j \neq p$ **do**
    $\mu_{ij} = \int_{\mathcal{C}_i \notin \mathcal{S}_{ij}} \Psi_i \prod_k \mu_{ki}, \forall k \neq j$
    Send message to neighbour i
    DistributeMessages($\mathcal{C}_j$)

2.3.3 Forming a Junction Tree from an Information Matrix

The process of converting an information matrix and vector to a JT is given in Algorithm 2.4. This requires the determination of an elimination ordering, which can be done using the amd heuristic (see Section 2.2.4). Algorithm 2.5 applies the elimination ordering to the matrix $\mathbf{Y}$ and records the cliques that are formed as vertices are marginalised from the graph. In the case of Figure 2.9, the first two cliques recorded would be $\{x_1, x_4, x_5\}$ and $\{x_2, x_5, x_6\}$. These cliques are equivalent to the subscripts of non-zeros in the rows of the Cholesky factor of $\mathbf{Y}$ for this particular ordering. It is likely that many cliques will be subsets of larger cliques and Algorithm 2.6 is used to remove any non-maximal cliques from the clique set. With reference to Figure 2.9, the cliques $\{x_5, x_6, x_7\}$ and $\{x_6, x_7\}$ are subsets of $\mathcal{C}_2 = \{x_4, x_5, x_6, x_7\}$ and would be removed from the clique set. Once the cliques are formed they need to be connected as a tree such that the running intersection property holds. This is performed by finding a maximal spanning tree between the cliques where the link weight is the size of the separator sets between cliques. Figure 2.10 demonstrates the graph of cliques and their connectivity weight based on Figure 2.9(b). Edges are labeled with the size of the separator and a maximal weight tree is shown with blue edges. The final step in Algorithm 2.4 is to spread $\mathbf{Y}$ and $\mathbf{y}$ across the clique potentials of the JT. This uses Algorithm 2.7, which iterates through the cliques assigning the values contained in $\mathbf{Y}$ and $\mathbf{y}$ to the clique potentials. Afterwards, the elements of $\mathbf{Y}$ and $\mathbf{y}$ that were assigned to clique potentials are set to zero. These matrix and

vector elements are zeroed so that any neighbouring clique that is visited afterwards does not have those values assigned to it. This is done to ensure that Equation 2.32 is satisfied. Upon completion of Algorithm 2.4 the JT is ready for message passing to be performed. Note that Algorithm 2.7 could be replaced by any other decomposition approach that satisfies Equation 2.32. For example Algorithm 2.8 will result in clique potentials that are also positive definite.

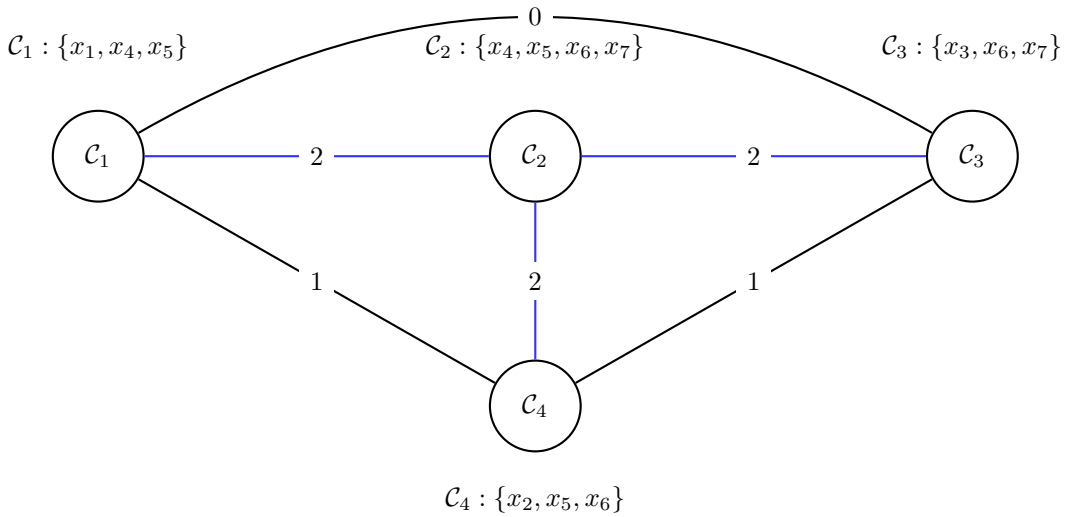


Figure 2.10 – Weighted Graph for finding the JT of Figure 2.9. The maximal weight tree is shown in blue, edge weights that are equal to the clique seperator sizes are labeled on each edge. Note that in this case the tree is also the maximum spanning tree for the graph. It is not a requirement that the maximum spanning tree be found, rather only the maximal.

Algorithm 2.4: FormJT

input : $\mathbf{Y}$, Information Matrix
input : $\mathbf{y}$, Information Vector
output: $\mathcal{T}$, Junction Tree
$\mathcal{I}$:= find an elimination ordering for $\mathbf{Y}$
$\mathcal{R}$:= FormCliqueSet($\mathbf{Y}, \mathcal{I}$)
$\mathcal{T}$:= connect $\mathcal{R}$ such running intersection property is true
$\mathcal{T}$:= SimpleDecomposeInformationToCliques($\mathbf{Y}, \mathbf{y}, \mathcal{T}$)

Algorithm 2.5: FormCliquesSet

input : $\mathbf{Y}$, Information Matrix
input : $\mathcal{I}$, elimination ordering
output: $\mathcal{R}$, set of cliques
foreach *i in* $\mathcal{I}$ **do**
    $\mathcal{N}$:= neighbours of i in $\mathbf{Y}$.
    Add $\mathcal{N}$ into $\mathcal{R}$
    Eliminate(i,$\mathbf{Y}$)
$\mathcal{R}$:= RemoveNonMaximalCliques($\mathcal{R}$)

Algorithm 2.6: RemoveNonMaximalCliques

input : $\mathcal{R}$, set of cliques
output: $\mathcal{R}$, set of cliques
foreach $\mathcal{C}_i$ *in* $\mathcal{R}$ **do**
    foreach $\mathcal{C}_j$ *in* $\mathcal{R}$ **do**
        if $\mathcal{C}_i \subset \mathcal{C}_j$ **then**
            Remove $\mathcal{C}_i$ from $\mathcal{R}$
            break

Algorithm 2.7: SimpleDecomposeInformationToCliques

input : Information Matrix, $\mathbf{Y}$, information vector $\mathbf{y}$
input : Junction Tree, $\mathcal{T}$
output: Junction Tree, $\mathcal{T}$
foreach $\mathcal{C}_i$ *in* $\mathcal{T}$ **do**
    $\mathbf{Y}_{\Psi_i} = \mathbf{Y}(\mathcal{C}_i, \mathcal{C}_i)$
    $\mathbf{y}_{\Psi_i} = \mathbf{y}(\mathcal{C}_i)$
    $\mathbf{y}(\mathcal{C}_i) = 0$
    $\mathbf{Y}(\mathcal{C}_i, \mathcal{C}_i) = 0$

Algorithm 2.8: PSD-DecomposeInformationToCliques

input : Information Matrix, $\mathbf{Y}$, information vector $\mathbf{y}$
input : Junction Tree, $\mathcal{T}$
output: Junction Tree, $\mathcal{T}$
$\mathcal{V} :=$ leaves of $\mathcal{T}$
$\mathcal{D} := \emptyset$
while $\mathcal{V} \neq \emptyset$ **do**
  $\mathcal{C}_i :=$ any single clique in $\mathcal{V}$
  add $\mathcal{C}_i$ to $\mathcal{D}$
  remove $\mathcal{C}_i$ from $\mathcal{V}$
  $\mathcal{K} := \notin \mathcal{C}_i$
  $\mathbf{S} = \mathbf{Y}(\mathcal{K}, \mathcal{C}_i)\mathbf{Y}(\mathcal{C}_i, \mathcal{C}_i)^{-1}\mathbf{Y}(\mathcal{C}_i, \mathcal{K})$
  $\mathbf{Y}(\mathcal{K}, \mathcal{K}) = \mathbf{Y}(\mathcal{K}, \mathcal{K}) - \mathbf{S}$
  $\mathbf{y}(\mathcal{K}) = \mathbf{y}(\mathcal{K}) - \mathbf{Y}(\mathcal{K}, \mathcal{C}_i)\mathbf{y}_{\mathcal{C}_i}$
  $\mathbf{Y}_{\Psi_i} = \mathcal{S}$
  $\mathbf{y}_{\Psi_i} = \mathbf{Y}(\mathcal{K}, \mathcal{C}_i)\mathbf{y}_{\mathcal{C}_i}$
  $\mathbf{y}(\mathcal{C}_i) = 0$
  $\mathbf{Y}(\mathcal{C}_i, \mathcal{C}_i) = 0$
  foreach *neighbour* $\mathcal{N}_i$ *of* $\mathcal{C}_i$ **do**
    if *size*$(\mathcal{N}_i \setminus \mathcal{D}) \leq 1$ **then**
      add $\mathcal{N}_i$ to $\mathcal{V}$

2.3.4 Incremental Estimation

Junction Trees can be used for incremental estimation in a manner similar to incremental estimation with information matrices (see Section 2.2.2). There are 3 cases to consider when adding a new observation into the system:

1. Observed states are all members of a single clique.
2. Some observed states are not in any clique.
3. No single clique contains all the states.

For case 1, the approach is analogous to an information matrix update. Algorithm 2.10 can be used, that simply fuses the observation with the potential of the clique which contains every state of the observation. Note that this approach only changes states within a single clique.

Case 2 is equivalent to augmenting the information matrix. Algorithm 2.9 can be used to fuse the new observations. This requires finding a clique that contains all of the states that are in the observation and also exist in the JT and then adding the previously unobserved states to it. This is followed by applying case 1 to the newly enlarged clique. This is demonstrated in Figure 2.11. Also observe that this requires the addition of information to only a single clique.

Case 3 is more difficult, since it requires joining all the cliques which lie in the path between the cliques that share the states contained in the observation. The approach is described in Algorithm 2.11. Figure 2.12 also demonstrates an example of this approach. In this example, a new observation connects states x_2 and x_4, which appear separately in $\mathcal{C}_2$ and $\mathcal{C}_3$. In this case, the shortest path between these cliques contains only $\mathcal{C}_2$ and $\mathcal{C}_3$ but in general there can be many cliques on this path. All the cliques on the path must be merged. This is done by combining the clique potentials (equivalent to adding the parameters $\mathbf{Y}_{\Psi_i}, \mathbf{y}_{\Psi_i}$ for each clique) and finding the union of the clique states. When finished, the new observation can be fused into the new

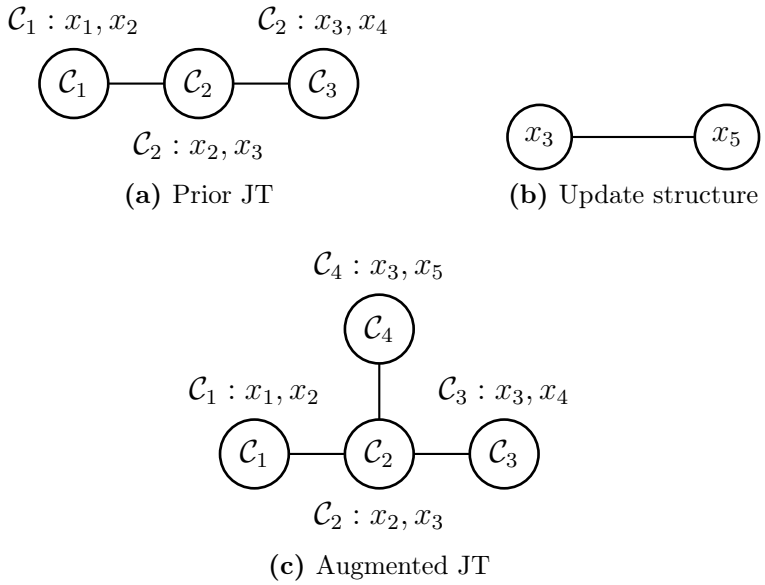
clique potential in a manner similar to case 1. This operation has formed a large clique ($\mathcal{C}_2$).

The final step of Algorithm 2.11 is to perform Algorithm 2.12, which attempts to exploit any intra-clique sparsity through reordering. Consider Figure 2.13 which shows an example of reordering using Algorithm 2.12. In this example the shortest path between the cliques containing the observed states includes two other cliques, and consequently the potential of the large clique contains some sparsity, as seen in Figure 2.13(d). Algorithm 2.12 begins by adding any state in a separator of $\mathcal{C}_2$ to the beginning of the elimination ordering, ensuring that they form a connected clique and that the structure $\mathcal{T}_{new}$ can be connected to the remaining components of the prior JT $\mathcal{T}$. A new ordering is found for the potential in $\mathcal{C}_2$ that allows the separation of the clique into three smaller cliques, as seen in Figure 2.13(e). By reordering this potential, the maximum clique size (or clique width) of the JT reduced from eight in Figure 2.13(c) to four in Figure 2.13(e).

Note that it is possible for cases 2 and 3 to occur simultaneously, that is, a state must be augmented but the preexisting states do not exist in a single clique. This can be addressed by using Algorithm 2.9 when fusing the new observation in Algorithm 2.11.

Algorithm 2.9: Augment and add new observation

input : Junction Tree $\mathcal{T}$
input : New potential Ψ_{obs}
output: Updated Junction Tree $\mathcal{T}$
$\mathcal{C}_i$ = Find a clique in $\mathcal{T}$ that contains a state from $\mathcal{C}_{obs}$
$\mathcal{C}_i = \mathcal{C}_i \cup \mathcal{C}_{obs}$
$\Psi_i = \Psi_i \Psi_{obs}$



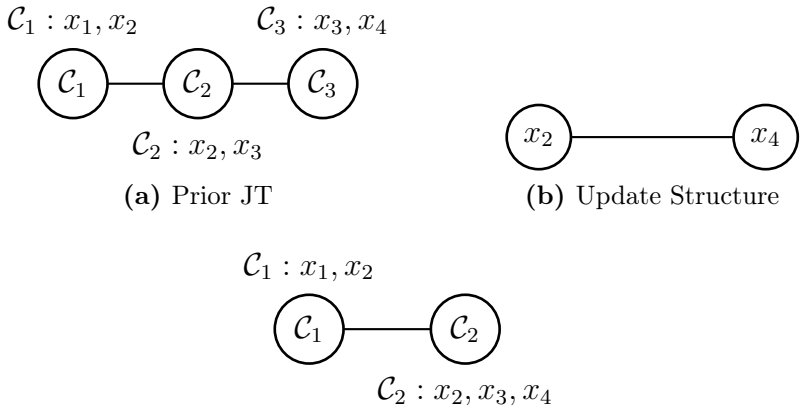
(a) Prior JT **(b)** Update structure

(c) Augmented JT

Figure 2.11 – JT update case 2. The prior junction tree is given in (a), the structure of the update is shown in (b). Since the state x_5 does not exist in any cliques, a new clique $\mathcal{C}_4$ is created to contain this new connection as shown in (c). The clique is connected to any other clique such that the running intersection property is maintained. In this example either $\mathcal{C}_2$ or $\mathcal{C}_3$ could become neighbours with $\mathcal{C}_4$ to satisfy this property.

Algorithm 2.10: FuseUpdateIntoJT

input : New potential Ψ_{obs}
input : Junction Tree $\mathcal{T}$
input : New potential Ψ_{obs}
output: Updated Junction Tree $\mathcal{T}$
$\mathcal{C}_i$ = Find a clique in $\mathcal{T}$ for which $\mathcal{C}_i \cap \mathcal{C}_{obs}$
$\Psi_i = \Psi_i \Psi_{obs}$



(a) Prior JT **(b)** Update Structure

(c) JT after Augment

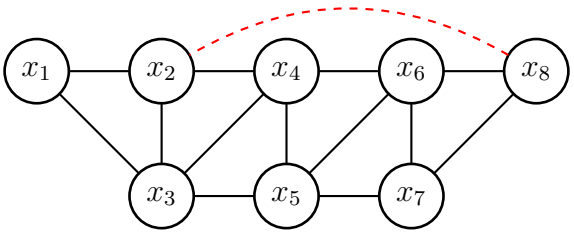
Figure 2.12 – JT update case 3. (a) shows the original junction tree, while (b) shows the structure of the update. Since this update contains two states that both exist in the junction tree, cliques must be merged such that a single clique contains both states. In this case $\mathcal{C}_2$ and $\mathcal{C}_3$ are merged to form (c).

Algorithm 2.11: Merge Cliques and Fuse

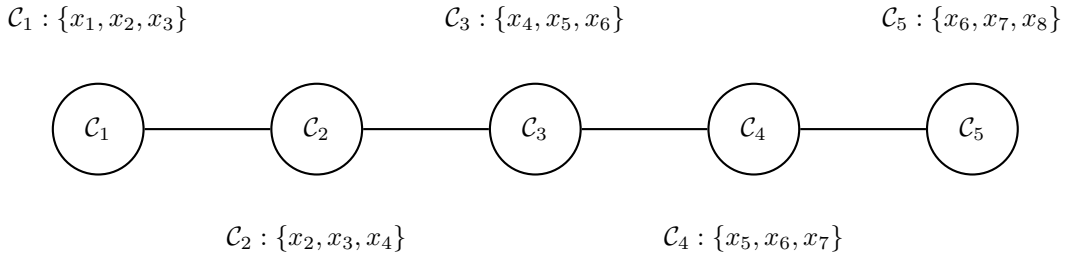
input : New potential Ψ_{obs}
input : Junction Tree $\mathcal{T}$
output: Junction Tree $\mathcal{T}$
$\mathcal{P}$ = shortest path between all states of $\mathcal{C}_{obs}$
`// Make large single clique` $\mathcal{C}_m$
foreach $\mathcal{P}_i$ *in* $\mathcal{P}$ **do**
  $\mathcal{C}_m = \mathcal{C}_m \cup \mathcal{C}_i$
  $\Psi_m = \Psi_m \Psi_i$
`// Fuse into new clique`
$\Psi_m = \Psi_m \Psi_{obs}$
$\mathcal{U}$ = Reorder $\mathcal{C}_m$
replace $\mathcal{C}_m$ from $\mathcal{T}$ with $\mathcal{U}$

Algorithm 2.12: Reorder

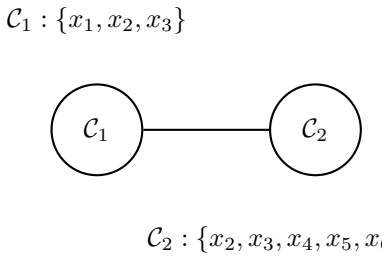
input : A Large clique $\mathcal{C}_m$
input : Junction Tree $\mathcal{T}$
output: Junction Tree $\mathcal{T}$
$\mathcal{I}$:= Find an elimination ordering for $\mathbf{Y}_{\Psi_m}$
`// Ensure that separators of` $\mathcal{C}_m$ `are also cliques in` $\mathcal{T}_{new}$
Move all states in $\mathcal{S}_{*m}$ to beginning of $\mathcal{I}$
$\mathcal{R}$:= FormCliqueSet($\mathbf{Y}_{\Psi_m}$,$\mathcal{I}$)
$\mathcal{T}_{new}$:= connect $\mathcal{R}$ such that the running intersection property is true.
$\mathcal{T}_{new}$:= SimpleDecomposeInformationToCliques($\mathbf{Y}_{\Psi_m}, \mathbf{y}_{\Psi_m}, \mathcal{T}_{new}$)
$\mathcal{T}$:= Connect $\mathcal{T}_{new}$ to previous neighbours of $\mathcal{C}_m$ in $\mathcal{T}$.



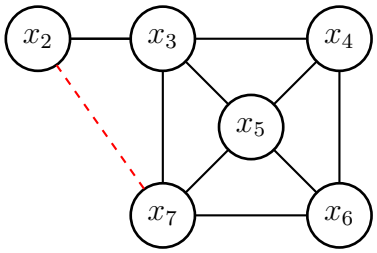
(a) Graph of estimate, with new observed edge in dashed red



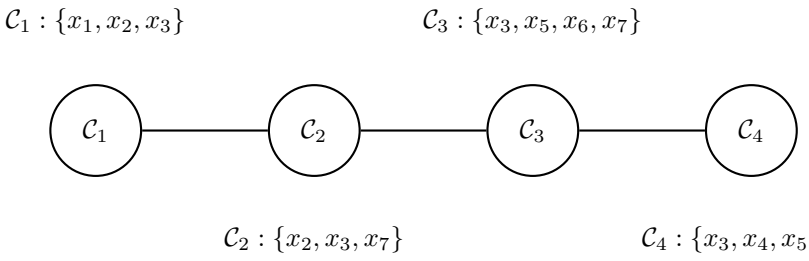
(b) JT before new observation.



(c) JT after new observation.



(d) New structure of $\mathbf{Y}_{\Psi_2}$



(e) Structure after re-ordering $\mathcal{C}_2$

Figure 2.13 – Example of reordering. (a) shows the equivalent information matrix before a new edge is introduced, while (b) shows the equivalent JT before this new edge. Since the edge connecting states x_8 and x_2 does not exist in any single clique, this is a case 3 update. The shortest path contains $\mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4, \mathcal{C}_5$, these are merged to produce the JT in (c). However, the structure of the clique potential in $\mathcal{C}_2$ is shown in (d) and is clearly sparse (the new link is shown in dashed red). Reordering (Algorithm 2.12) can be used to exploit this sparsity resulting in the JT in (e), which has a much smaller clique width.

2.3.5 Cost of the JT algorithm

A significant cost is incurred each time a message is formed as this requires inverting a matrix of dimension $|\mathcal{C}_i| - |\mathcal{S}_{ij}|$. Each clique transmits one message per neighbour, or edge in the JT. Let $\mathcal{E}$ contain the size of the required inversion per edge, such that a single element $e_k \in \mathcal{E}$ is equal to $e_k = |\mathcal{C}_i| - |\mathcal{S}_{ij}|$ for some i, j in the JT $\mathcal{T}$. Then the cost of message passing is:

$$\mathcal{O}\left(\sum_k e_k^3\right), \forall e_k \in \mathcal{E} \tag{2.37}$$

After message passing, the mean and covariance over a particular clique set can be found by inverting the marginal matrix $\mathbf{Y}_{\Phi_i}$. The cost of performing this for every clique is:

$$\mathcal{O}\left(\sum_i |\mathcal{C}_i|^3\right). \tag{2.38}$$

For sparse systems, these costs are much less than $\mathcal{O}(n^3)$. Compare this to the cost of the Cholesky solve $\mathcal{O}(\sum_k |l_k|^2)$ and the cost of finding the elements of the covariance $\mathcal{O}(nl^2)$ described in Section 2.2.5. Although the number of cliques is usually less than the number of rows in $\mathbf{L}$, the cubic cost still dominates, resulting in the Cholesky factorisation being more efficient. Note that this analysis is valid only for the particular formulation of JT message passing used in this thesis.

2.3.6 Special cases of Junction Tree Inference

The JT algorithm presented in this section is a special case of general inference using JTs. This work chooses a particular approach to message passing known as the 'Shafer-Shenoy' method[28]. An equivalent approach for inference is the 'Hugin' method. In this method, potentials are maintained over each separator set as well as the clique set, and message passing is performed by updating the separator potentials

of neighbouring cliques. After message passing is performed the clique potential becomes a marginal estimate over the clique set. This method is employed by the Thin Junction Tree (TJT) filter described in Section 2.5.3 and Appendix B.2.1.

Other variants of JT inference can use the Cholesky or other matrix decompositions as their internal representation. A good example of this is the Bayes Tree [33][34], which uses a QR factorisation of the clique potentials and messages.

2.4 Sparse Problems

This section discusses some example problems and their associated graphs. The aim is to show the types of problems that can be addressed by a sparse estimation approach and to highlight areas where these problems may benefit from sparsity.

2.4.1 SLAM

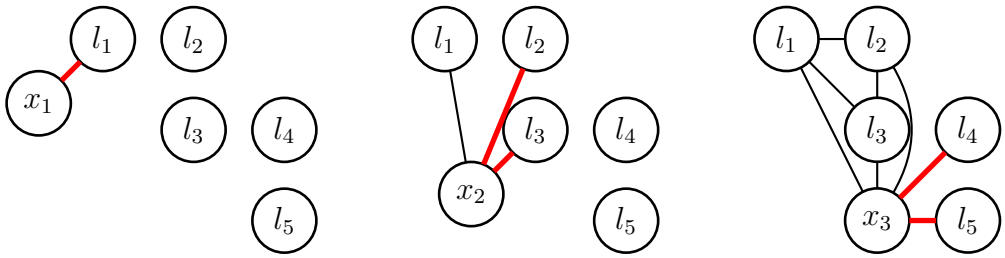
Simultaneous Localisation and Mapping (SLAM) has become a standard problem in robotic applications [5][17]. SLAM is was initially applied to indoor robots, but has been successfully applied in outdoor mapping[34], undersea exploration [40][18], ship hull inspection [60], mining robot navigation [47] and aerial surveying [11]. While there exist many variants of SLAM, the problems considered in this thesis are restricted to simplified linear systems, and also ignores the data association problem in typical SLAM implementations.

This thesis considers a simple formulation of the SLAM problem where there are two types of observations; observations of landmarks made by the robot; and estimates of the robots trajectory based on internal observations such as wheel movement (from a wheel encoder) or forces acting on the robot (observed through an accelerometer). It will also consider two types of states; robot locations at a particular time $\mathbf{x}_t$; and stationary landmark locations l_i. Modern approaches to SLAM can extend this by making landmarks implicit [41], or attempting to estimate biases in sensors such as accelerometers.

Early work on the SLAM problem focused on the filtering variant, whereby the robot state would be marginalised after every time step. This resulted in a problem that was dense in both the covariance and information form, but whose state vector only grew with the observation of new landmarks. The form of the state vector is:

$$\mathbf{x} = \begin{bmatrix} x_t & l_1 & \dots & l_n \end{bmatrix}^T, \tag{2.39}$$

Figure 2.14 shows the structure of a filtering SLAM problem. The robot moves through a field of landmarks, marginalising at every time step and slowly connecting the landmarks to form a dense clique. As can be seen in Figure 2.14(c) the state estimate gradually increases in density as more landmarks become connected to the robot state and other landmarks.



(a) time = 1 **(b)** time = 2 **(c)** time = 3

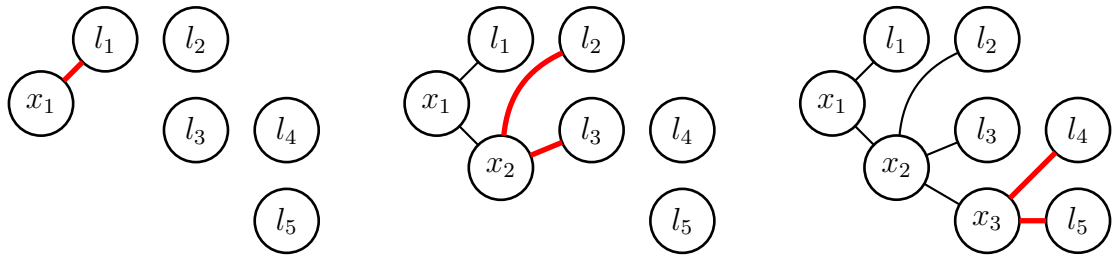
Figure 2.14 – Structure of the filtering SLAM problem. Red links indicate observations from the current time set.

The SLAM problem can be reformulated as a sparse problem by maintaining the robot state estimates rather than marginalising them. The state vector for this is:

$$\mathbf{x} = \begin{bmatrix} x_1 & \dots & x_t & l_1 & \dots & l_n \end{bmatrix}^T, \tag{2.40}$$

The structure of the problem can be seen in Figure 2.15. This shows the landmarks being connected through the observations of the robot states. Over time, smoothing SLAM will remain sparse, while an equivalent filtering SLAM solution will be dense. A significant disadvantage is that the size of the problem grows linearly with time, and eventually old robot states will need to be marginalised. As old states are marginalised, smoothing SLAM begins to show the same sparsity problem exhibited

by filtering SLAM.



(a) time = 1 **(b)** time = 2 **(c)** time = 3

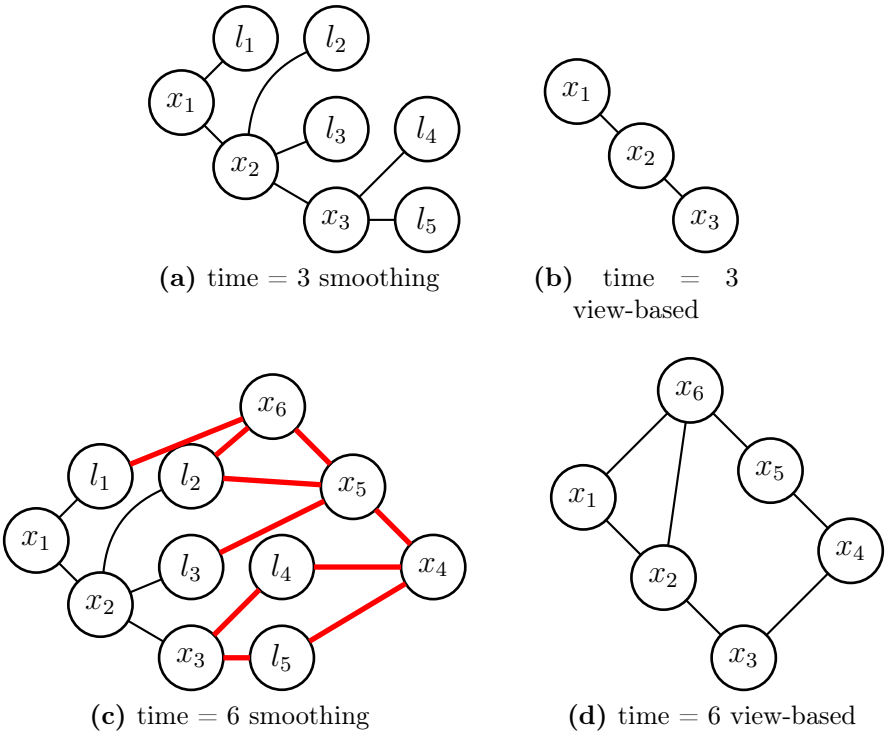
Figure 2.15 – Smoothing SLAM approach. The landmarks are conditionally dependent upon the robot states. Red links indicate observations collected at the current time step.

Another variant of SLAM is view-based SLAM [20]. In this variant only the state history is maintained. While this results in a large sparse chain of states, links can be induced between robot states through re-observing a view. Figure 2.16 shows a comparison between smoothing and view-based SLAM. Since the state space of view-based SLAM grows linearly with time, robot states will eventually need to be marginalised. This will result initially in fill-in and eventually in a dense problem. Thus both smoothing and view based SLAM approaches will eventually become dense if operated for a sufficiently long time period.

2.4.2 Distributed Sensor Networks

Distributed Sensor Networks (DSNs) are useful for gathering observations of a particular unknown parameter over a wide geographical area. DSNs have also widely been used for water and ocean monitoring in Australia [55] and the world [32], localisation of animals through sound [63], monitoring temperature changes throughout a house [49] and in distributed power generation [3][1] and vehicular sensor networks [4].

Although each application of a DSN is different, a simple structure can be applied to all models. Each platform is concerned with observing a quantity at its current location. For example the water temperature at a particular depth. These quantities



(a) time = 3 smoothing **(b)** time = 3 view-based

(c) time = 6 smoothing **(d)** time = 6 view-based

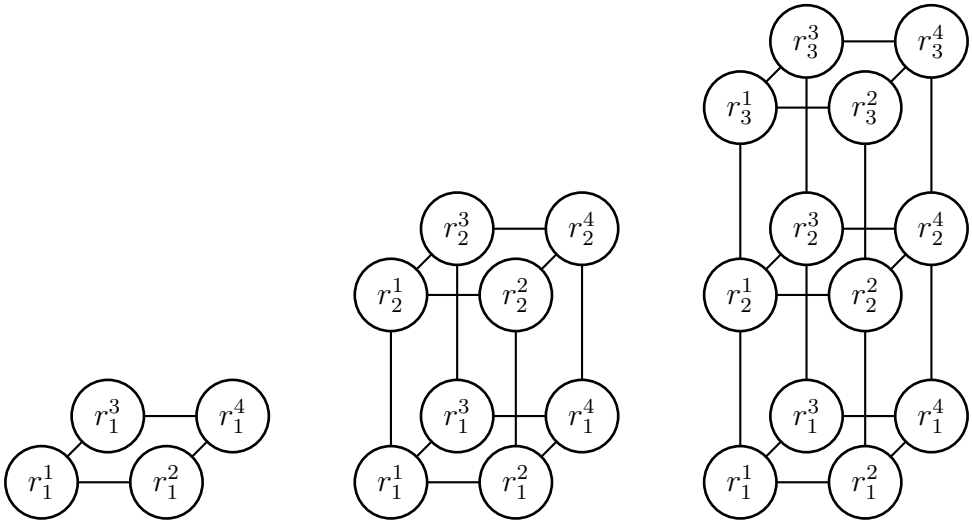
Figure 2.16 – Comparison between smoothing and view-based approaches. Observe that the view based system (b) is simply a chain of states. (c) shows a possible smoothing SLAM system after 6 time steps, with the robot re-observing the landmarks. The view-based SLAM solution for (c) is shown in (d). Observe that a loop has formed, and that x_6 is now connected to the earlier states x_1 and x_2. Re-observing landmarks can create loops between states and lead to a dense system.

can change over time, and also with the platforms location. However, since each quantity is local to the platform, they can be considered as a single quantity r_i^j for platform i at time j. Therefore the state space for this problem is:

$$\mathbf{x} = \begin{bmatrix} r_1^1 & \dots & r_1^n & \dots & r_t^1 & \dots & r_t^n \end{bmatrix}^T. \tag{2.41}$$

The system is concerned with how the quantities are related to other nearby nodes, and also how they change through time. The growth of a DSN can be seen in Figure 2.17. Note that past states eventually need to be forgotten and marginalisation will quickly generate a dense problem. In fact, simply marginalising all states at time i will result in dense connection of all states at time $i + 1$. DSNs also require communication between nodes, which is related to how many links exist between

nodes. See Example 2.4.1 for a discussion of how this affects the related problem of Cooperative Localisation.



(a) time = 1 **(b)** time = 2 **(c)** time = 3

Figure 2.17 – Structure of the Distributed Sensor Network Problem

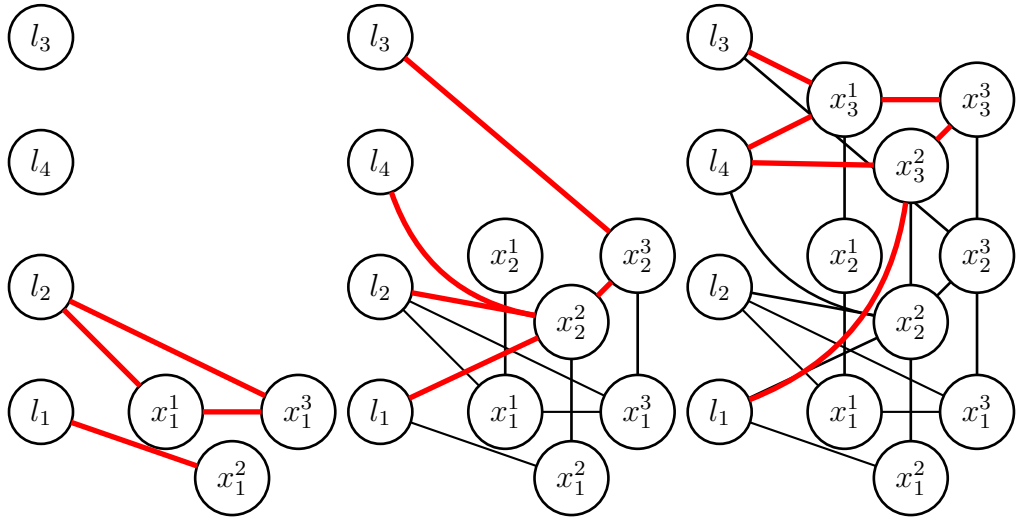
2.4.3 Cooperative Localisation

Cooperative Localisation combines the DSN and SLAM problems [6]. It estimates the location of a set of mobile robot states (x_i^k for robot i at time k). These robots observe stationary landmarks in the environment (l_i) as well as other robots. The state space for this system is:

$$\mathbf{x} = \begin{bmatrix} x_1^1 & \dots & x_1^n & \dots & x_t^1 & \dots & x_t^n & l_1 & \dots & l_n \end{bmatrix}^T, \tag{2.42}$$

The structure of the cooperative localisation problem can be seen in Figure 2.18. Although the structure looks dense, robots do not observe past robot states, and the structure is thus similar to the DSN problem in Figure 2.17. Robots also need to communicate their information to each other. This cost is considered in Example

2.4.1. As with all other problems in this section, marginalisation of past states will lead to a dense graph, which is demonstrated in Example 2.4.2.

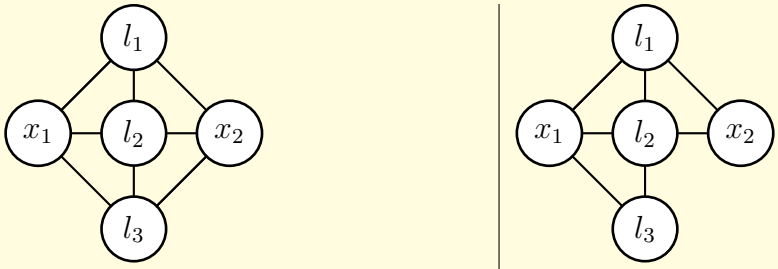


(a) time = 1 **(b)** time = 2 **(c)** time = 3

Figure 2.18 – Structure of the Cooperative Localisation Problem. Red links indicate observations made in the current time step.

Example 2.4.1. *An important factor when designing distributed cooperative localisation estimation schemes is the message size. Message passing can be used to allow each robot to maintain a consistent[a] estimate with respect to all the other robots.*

In this example consider a simple two robot problem in the following scenarios:



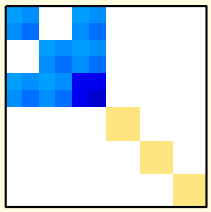
Scenario A *Scenario B*

In scenario A, robot x_1 needs to communicate complete information to robot

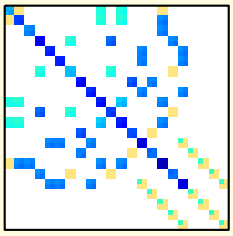
x_2 about the location of landmarks l_1, l_2 and l_3. This is because x_2 is directly related to each of these landmarks, and needs to know its full state to fuse its internal information. However in scenario B, x_2 is not directly related to l_3. This allows robot x_1 to marginalise the estimate of l_3 and send a message which only contains the states of l_1 and l_2. Thus increased sparsity has reduced communication costs.

[a] In this setting, consistent means that the distribution found on every robot for each identical state will be the same [25][39]. This is different to the definition of conservative (Section 2.5.1), where an estimate is 'less certain' than some other estimate.

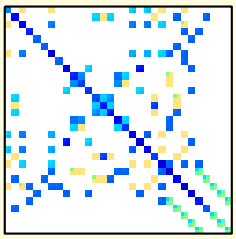
Example 2.4.2. *Marginalisation is a significant source of non-zeros in a dynamically updating estimation system. This example shows the progression of a cooperative localisation problem where states are marginalised when they are 5 time-steps old. The state space grows in size as new landmarks are observed. Darker colors indicate larger numbers. The dimension of each matrix is listed below it, along with time step.*



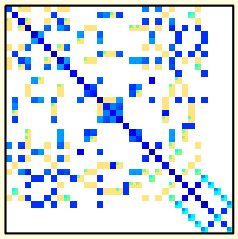
t=1 (dim=12)



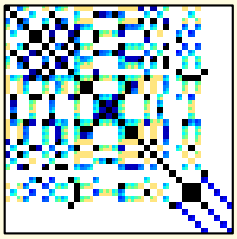
t=50 (dim=112)



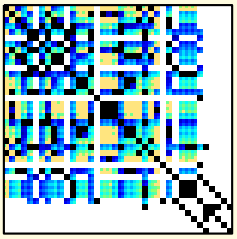
t=100 (dim=212)



t=150 (dim=312)



t=200 (dim=412)



t=250 (dim=512)

At time 250 nearly all of the landmarks are now connected through marginalisation induced fill-in. Note that even at t=250 many of the links are of low weight (light yellow). These are good candidates for sparsification as their removal will only have a small affect on the overall estimate quality.

2.5 Approximate Estimation

The previous section describes several sparse problems, which under certain conditions can become dense, and therefore intractable for large problems. This section begins with an exploration of the cost functions and the requirement for conservative estimates. Following this, methods for reducing the computational complexity of information form and Junction Tree based estimates through approximation are explored.

2.5.1 Approximation Quality

The most obvious quality that an approximation must have is that it is in some way 'close' to the original estimate. There are many possible cost functions that can be used, such as an l_p norm between the matrices (i.e., difference between values in the information matrix of the approximate and true taken to the power p). Since the estimates represent probabilities, it is more appropriate to use a measure that is commonly used for probabilities. The Kullback Leibler Divergence (KLD) is chosen as it is a commonly used measure of divergence between probability distributions [8][39]. The equation for the KLD is shown below:

$$\mathrm{D_{KL}}\left(p(\mathbf{x})||q(\mathbf{x})\right) = \int_{-\infty}^{+\infty} p(\mathbf{x}) \log\left(\frac{p(\mathbf{x})}{q(\mathbf{x})}\right) d\mathbf{x}, \tag{2.43}$$

If $p(\mathbf{x})$ and $q(\mathbf{x})$ are Gaussian, this can then be computed as [8]

$$\mathrm{D_{KL}}\left(p(\mathbf{x})||q(\mathbf{x})\right) = \frac{1}{2}\left(\mathrm{tr}(\Sigma_q^{-1}\Sigma_p) + (\mu_p - \mu_q)^T\Sigma_q^{-1}(\mu_p - \mu_q) - \log\frac{\det(\Sigma_p)}{\det(\Sigma_q)} - n\right), \tag{2.44}$$

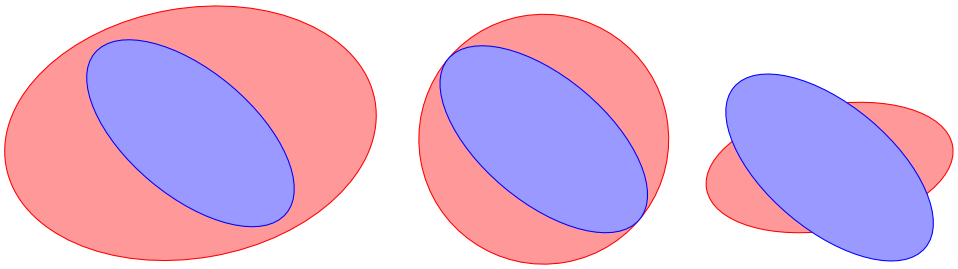
where n is the number of states. This cost function is advantageous since it is also convex in the elements of Σ_q^{-1}. The KLD is chosen as the measure of divergence in this thesis.

Another important property for an approximate estimate is conservativeness. A mo-

tivation for conservativeness was given in Chapter 1. Assuming the distributions are Gaussian with coincident means, an approximate estimate is conservative if its covariance is larger than or equal to the covariance of the true estimate. The covariance of an n-dimensional Gaussian can be plotted using the following equation where c is a constant:

$$\varepsilon_A = \{\mathbf{x} | \mathbf{x}^T \mathbf{Y} \mathbf{x} = c\} \tag{2.45}$$

The set of points in ε_A contain the locus of a hyper-ellipse. The covariances of two Gaussians can be compared by generating the locus for the same value of c. For a simple 2-dimensional case, consider Figure 2.19. In this figure, the red ellipse represents the approximate covariance and the blue, the original covariance. Observe that Figure 2.19(a) and Figure 2.19(b) are conservative estimates while Figure 2.19(c) is not. If an estimate is not conservative it is called over-confident. Unfortunately,



(a) $\mathbf{Y}_1^{-1} \succ \mathbf{Y}_2^{-1}$ **(b)** $\mathbf{Y}_1^{-1} \succeq \mathbf{Y}_2^{-1}$ **(c)** Indefinite

Figure 2.19 – The possible ellipsoids for positive semi-definite matrices centred on the origin are shown. $\mathbf{Y}_1^{-1}$ is represented by a red ellipse, while $\mathbf{Y}_2^{-1}$ is represented by a blue ellipse. (c) Shows the ellipses where neither $\mathbf{Y}_1^{-1} \succeq \mathbf{Y}_2^{-1}$ nor $\mathbf{Y}_1^{-1} \preceq \mathbf{Y}_2^{-1}$ are true.

minimising the KLD does not guarantee a conservative estimate. Example 2.5.1 demonstrates a scenario where an overconfident estimate has a smaller KLD than a conservative estimate.

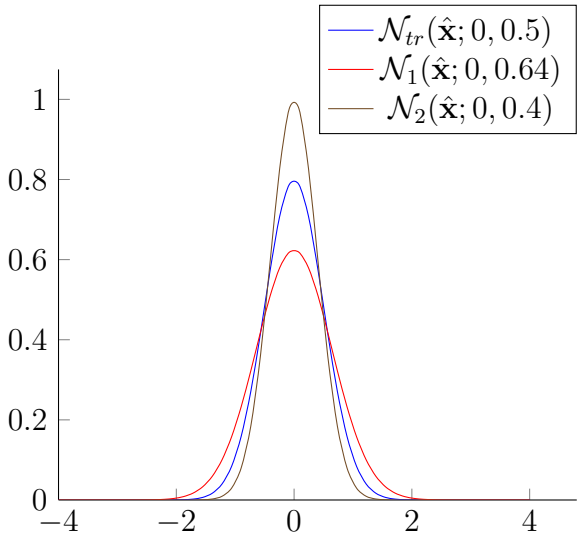


Figure 2.20 – Probability density functions for distributions described in Example 2.5.1. Note that the Gaussian distributions have been parametrised in covariance form ($\mathcal{N}(\hat{\mathbf{x}}; \mathbf{x}, \sigma)$). Although the variance of $\mathcal{N}_2$ is closer to that of $\mathcal{N}_{tr}$ than the variance of $\mathcal{N}_1$, $\mathcal{N}_2$ it expresses more certainty than $\mathcal{N}_{tr}$ and is therefore an over-confident estimate.

Example 2.5.1. *Consider three single dimension Gaussians as shown in Figure 2.20 (where the Gaussians distributions are parametrised using the covariance form $\mathcal{N}(\hat{\mathbf{x}}; \mathbf{x}, \sigma)$):*

$$\mathcal{N}_{tr}(\hat{\mathbf{x}}; 0, 0.5), \quad \mathcal{N}_1(\hat{\mathbf{x}}; 0, 0.64), \quad \mathcal{N}_2(\hat{\mathbf{x}}; 0, 0.4),$$

$\mathcal{N}_1$ is a conservative estimate of $\mathcal{N}_{tr}$, while $\mathcal{N}_2$ is an overconfident estimate. The KLD between these estimates and $\mathcal{N}_{tr}$ is given by:

$$\begin{aligned}
\mathrm{D}_{\mathrm{KL}}\left(\mathcal{N}_{tr} || \mathcal{N}_1\right) &= 0.5 * \left(\frac{\sigma_{tr}}{\sigma_1} - ln(\frac{\sigma_{tr}}{\sigma_1}) - 1\right) \\
&= 0.5 * \left(\frac{0.5}{0.64} - ln(\frac{0.5}{0.64}) - 1\right) \\
&= 1.41 \times 10^{-2} \\
\mathrm{D}_{\mathrm{KL}}\left(\mathcal{N}_{tr} || \mathcal{N}_2\right) &= 0.5 * (\frac{0.5}{0.4} - ln(\frac{0.5}{0.4}) - 1) \\
&= 1.34 \times 10^{-2}
\end{aligned}$$

Even though $\mathcal{N}_2$ has a smaller KLD and has a closer variance to $\mathcal{N}_{tr}$, preferring $\mathcal{N}_2$ over $\mathcal{N}_1$ could be undesirable because $\mathcal{N}_1$ is not conservative.

For n-dimensional Gaussians, comparing the covariance for every state to check for conservatism could be a time consuming process. Fortunately, the set of positive definite matrices which include covariance and information matrices have an ordering (known as the Loewner partial ordering [10]), that if satisfied also implies that the covariances of every state are greater or equal. This relation is ensured by Theorem 2.1.

Theorem 2.1. $\mathbf{Y}_1^{-1} \succeq \mathbf{Y}_2^{-1}$ *implies that every covariance ellipsoid of* $\mathcal{N}_1(\hat{\mathbf{x}}; \mathbf{x}, \mathbf{Y}_1^{-1})$ *is contained by the equivalent ellipsoid of* $\mathcal{N}_2(\hat{\mathbf{x}}; \mathbf{x}, \mathbf{Y}_2^{-1})$.

Proof. Let ε_1 and ε_2 be the covariance ellipsoid for $\mathcal{N}_1$ and $\mathcal{N}_2$ found using equation 2.53 for any fixed value of c so that the ellipsoids represent equivalent probability contours. Let the sets $\mathcal{S}_1$ and $\mathcal{S}_2$ contain all points that are contained in ε_1 and ε_2 respectively. That is

$$\mathcal{S}_1 = \{\mathbf{x} | \mathbf{x}^T \mathbf{Y}_1 \mathbf{x} \leq c\} \tag{2.46}$$

$$\mathcal{S}_2 = \{\mathbf{x} | \mathbf{x}^T \mathbf{Y}_2 \mathbf{x} \leq c\} \tag{2.47}$$

Now consider the property: [51, Eq. 10.46]:

$$\mathbf{A} \succeq \mathbf{B} \iff \mathbf{x}^T \mathbf{A} \mathbf{x} \geq \mathbf{x}^T \mathbf{B} \mathbf{x}, \quad \forall \mathbf{x} \in \mathbb{R}^n \tag{2.48}$$

Since $\mathbf{Y}_1 \succeq \mathbf{Y}_2$ then the above property implies that:

$$\mathbf{x}^T \mathbf{Y}_1 \mathbf{x} \geq \mathbf{x}^T \mathbf{Y}_2 \mathbf{x}, \quad \forall \mathbf{x} \in \mathcal{S}_2 \tag{2.49}$$

but $\mathbf{x}^T \mathbf{Y}_2 \mathbf{x} \geq c$, $\forall \mathbf{x} \in \mathcal{S}_2$. Therefore

$$\mathbf{x}^T \mathbf{Y}_1 \mathbf{x} \geq \mathbf{x}^T \mathbf{Y}_2 \mathbf{x} \geq c, \quad \forall \mathbf{x} \in \mathcal{S}_2 \tag{2.50}$$

but since

$$\mathbf{x}^T \mathbf{Y}_1 \mathbf{x} \geq c, \quad \forall \mathbf{x} \in \mathcal{S}_1 \tag{2.51}$$

it follows that $\mathcal{S}_2 \subseteq \mathcal{S}_1$. That is, all the points contained in ε_2 are a subset of the points contained in ε_1. Therefore ε_2 is contained by ε_1. □

Therefore conservatism between an approximate distribution $\mathcal{N}^{-1}_{app}(\hat{\mathbf{x}}; \mathbf{y}_{app}, \mathbf{Y}_{app})$ and original distribution $\mathcal{N}^{-1}_{tr}(\hat{\mathbf{x}}; \mathbf{y}_{tr}, \mathbf{Y}_{tr})$ can be checked by ensuring that the relation

$$\mathbf{Y}^{-1}_{app} \succeq \mathbf{Y}^{-1}_{tr} \tag{2.52}$$

holds, where $\mathbf{A} \succeq \mathbf{B}$ implies that the difference $\mathbf{A} - \mathbf{B}$ is a positive semi-definite matrix.

For a Gaussian parametrised using the information matrix, the requirement to invert to check for conservatism is also quite costly. Fortunately, another property of the Loewner ordering is:

$$\mathbf{A} \succeq \mathbf{B} \iff \mathbf{A}^{-1} \preceq \mathbf{B}^{-1}. \tag{2.53}$$

Therefore Equation 2.52 can be expressed as

$$\mathbf{Y}_{tr} \succeq \mathbf{Y}_{app}, \tag{2.54}$$

which is also a convex constraint [10].

Another property of the Loewner partial ordering is that if the estimate which is being approximated is not the 'true' estimate, as long as the estimate being approximated is conservative then the new approximate distribution will also be conservative. This is expressed as:

$$\mathbf{Y}^{-1}_{app1} \succeq \mathbf{Y}^{-1}_{tr}, \mathbf{Y}^{-1}_{app2} \succeq \mathbf{Y}^{-1}_{app1} \implies \mathbf{Y}^{-1}_{app2} \succeq \mathbf{Y}^{-1}_{tr}, \tag{2.55}$$

and is demonstrated for a simple 2-D case in Figure 2.21. This is useful since it allows a conservative estimate to certify that another estimate is also conservative. This enables systems to iteratively approximate conservatively.

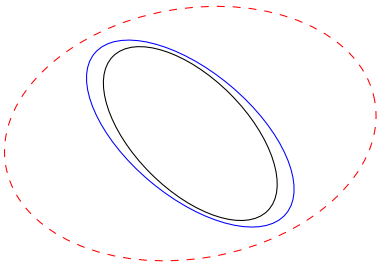


Figure 2.21 – The true distribution's ($\mathbf{Y}_{tr}$) locus is given by the black ellipse; the prior ($\tilde{\mathbf{Y}}$) is given by the blue ellipse; the red dashed ellipse represents the $\mathbf{Y}_{app}$ which is the conservative approximation of $\tilde{\mathbf{Y}}$. This picture graphically demonstrates the expression $\mathbf{Y}_{app}^{-1} \succeq \tilde{\mathbf{Y}}^{-1} \succeq \mathbf{Y}_{tr}^{-1}$. This shows that if the prior estimate is a conservative approximation of the true distribution, then new conservative estimates based on the prior will also be conservative with respect to the true distribution.

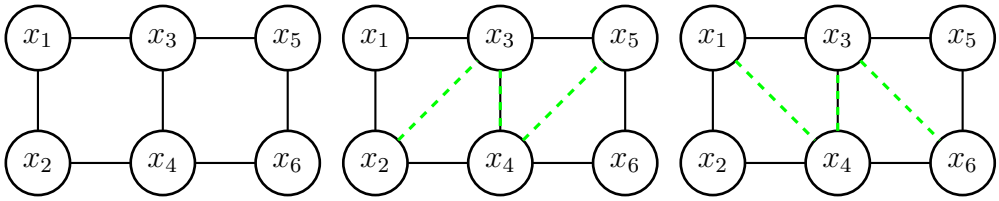
2.5.2 Improving Efficiency of Information Form

Section 2.2.5 describes the Cholesky factorisation for efficient solving. This algorithm has time complexity dominated by $\mathcal{O}(\sum_k l_k^2)$, where l_k is the number of non-zeros on row k. Considering this cost function, a simple approach to improve efficiency is to reduce the number of non-zeros in the largest row of the Cholesky factor.

Attempting to remove non-zeros from a particular row of a Cholesky factor is not simple. Any non-zero element may be a product of fill-in and this fill-in is dependent upon a particular choice of elimination ordering. Consider Figure 2.22 and recall that the Cholesky factor has the same sparsity as the triangulated graph. Observe that in Figure 2.22(b) the fill-in between vertices x_1 and x_3 can be removed by changing the elimination ordering to the one used in Figure 2.22(c). However, this change in elimination has not improved the sparsity of the Cholesky factor and has moved the non-zero to another location. One way to remove this fill-in is to delete the edge between x_1 and x_2. However if this edge is removed and the elimination ordering of Figure 2.22(c) is used, then the equivalent fill-in between x_3 and x_4 remains. Thus, the particular choice of elimination ordering needs to be taken into account when removing links. The best approach is to choose a new elimination ordering for the graph after sparsification. For large graphs it will be costly to reorder the matrix after every sparsification event.

Some links will be subject to fill-in even if they are removed from the graph. Consider the edge between x_3 and x_4 in Figure 2.22(b) and Figure 2.22(c). Regardless of elimination ordering the edge will be subject to fill-in and requires the removal of another existing edge before it can be deleted from the chordal graph (such as the edge between x_1 and x_2). However, care must be taken to ensure that the correct elimination ordering is used, as using the elimination ordering form Figure 2.22(c) and deleting the edge between x_1 and x_2 will still result in fill-in between x_3 and x_4.

In addition to choosing an edge that minimises fill-in, it is also desirable to reduce the error between the true system and the approximate system. [35] and [36] suggest a simple cost function that attempts to trade the cost of approximation with the



(a) Original Graph **(b)** Triangulated with ordering O_1 **(c)** Triangulated with ordering O_2

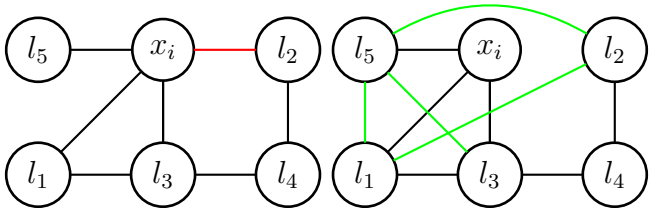
Figure 2.22 – Inducing sparsity in the Cholesky factor by modifying the information matrix. Dashed green lines indicate fill-in. (a) shows the original graph. (b) shows the triangulated graph for an elimination ordering $O_1 = \{x_1, x_2, x_3, x_4, x_5, x_6\}$, while (c) shows the graph for ordering $O_2 = \{x_5, x_6, x_4, x_3, x_1, x_2\}$. Observe that removing the edge x_3, x_4 in the graph will not change the triangulated graph structure due to fill-in.

efficiency gains of sparsification. However, the exact form of the function is left undefined. The approaches used in this thesis assume that removal of *any* link in the information matrix results in increased efficiency gains and does not consider the impact of different elimination orderings on the efficiency of the final result.

There have been many previous approaches to link removal in information matrices. It is observed in [35] and [54] that elements of the information matrix can approach zero in some situations (for example in filtering SLAM between the robot state and distant landmarks). This observation has generated many approaches that set small valued elements in the information matrix to zero.

Approximately Marginalised - Sparse Extended Information Filter (AM-SEIF) [54] is a technique for the filtering SLAM problem, which removes links between the robot and nearby landmarks to reduce landmarks connected to the robot state. AM-SEIF selects links whose normalised weight is below a particular threshold, and only when the number of landmarks connected to the robot state rises above a particular threshold. AM-SEIF uses marginalisation to create and then combine distributions such that a particular set of states are conditionally independent. Unfortunately, this marginalisation results in fill-in between landmarks connected to the robot and consequently adds many more links than it removes, resulting in a matrix with more non-zeros. This is illustrated in Figure 2.23. However, for the filtering SLAM scenario,

this is acceptable as the landmarks are assumed to be densely connected by previous robot marginalisation. This work is proved in [61] to result in over-confident estimates. Appendix B.1 describes AM-SEIF in more detail.



(a) Original graph **(b)** Sparsified graph

Figure 2.23 – Removing the link between x_i and l_2 using AM-SEIF approach (shown in red). This link is introduced through marginalisation of a prior state. Green links are induced by AM-SEIF. After this operation, the robot state has 3 rather than 4 edges, and is therefore more sparse. However, the overall graph sparsity has been reduced (there are 10 edges in (b) but only 7 edges in (a)). See Appendix B.1 for more details.

Another important approximate technique is Covariance Intersection (CI). This technique allows for de-correlation between states in the estimate while guaranteeing that the approximation will be conservative. CI is typically used in data fusion scenarios where two pieces of information need to be combined without knowledge of the cross-correlation between them. Appendix B.1 also describes CI and its variants in more detail.

Convex optimisation and other matrix optimisation approaches such as Covariance Selection [16] can also be used to introduce sparsity into an information matrix. These approaches are not used in incremental estimation as they have a high cost of $\mathcal{O}(n^{6.5})$ for dense matrices, and $\mathcal{O}(n^{2.5}s^2)$ for sparse matrices, where s is the number of non-zeros in the matrix. Appendix B.3.1.1 describes this problem in more detail.

Data Discarding Sparse Extended Information Filters (DD-SEIF)[61] uses a heuristic to ignore observations that connect each robot state through time and instead use observations of known landmarks to estimate the robot's new location. While this has the effect of maintaining a conservative approximation, the accuracy of this approach depends upon the strength of the link being discarded. In the case of strong links

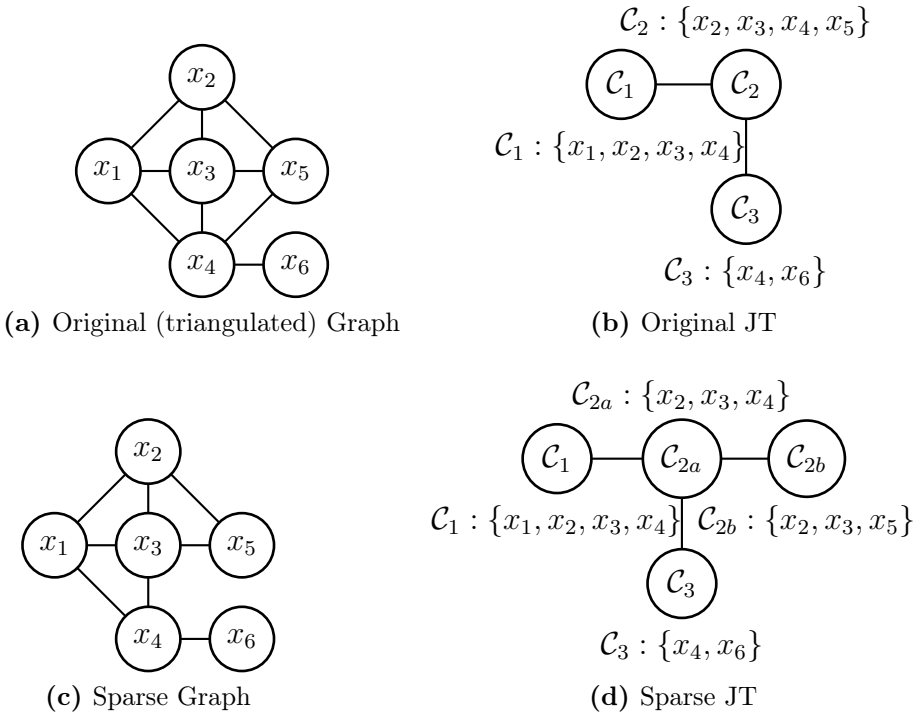
being discarded, large divergences can occur. This was observed in [59].

2.5.3 Improving Efficiency of Junction Tree Estimation

Section 2.3 describes the JT formulation for linear estimation. The largest cost of this approach is finding the marginal of each clique by inverting the sum of the potentials and received messages. This has cost $\mathcal{O}(\sum_i |\mathcal{C}_i|^3)$, which suggests that the best way to increase efficiency is to reduce the size of the clique sets. Additionally, reducing the clique size will also reduce the cost of message passing.

Some states cannot be removed from a clique without violating the running intersection property. Consider Figure 2.24(b). If state x_4 was removed from $\mathcal{C}_2$, then the running intersection property for x_4 between cliques $\mathcal{C}_1$ and $\mathcal{C}_3$ would be violated. Removal of states from a clique can also be considered as a clique splitting operation. A clique split occurs when a clique is approximated by dividing it into two smaller cliques, where the union of the states of the two cliques is equal to the states of the previous clique. As this is an approximation the sum of the clique potentials will not equal the original clique's potential. For example, in Figure 2.24(d), $\mathcal{C}_2$ is split into two cliques $\mathcal{C}_{2a}$ and $\mathcal{C}_{2b}$ such that $\mathcal{C}_2 = \mathcal{C}_{2a} \cup \mathcal{C}_{2b}$. The new clique sets also need to be chosen such that the running intersection property still holds. In Figure 2.24(d), $\mathcal{C}_{2a}$ contains x_4 such that the property is maintained while reducing the clique sizes.

Another approach to clique reduction in JTs is to remove states from the edges of the sub-trees relating to a particular state. This approach is taken by the TJT [48]. This approach works by approximately marginalising the state from the target clique, an approach which is similar to AM-SEIF. Figure 2.25 demonstrates the basic principle. Appendix B.2.1 shows that this technique is not conservative and provides more details.



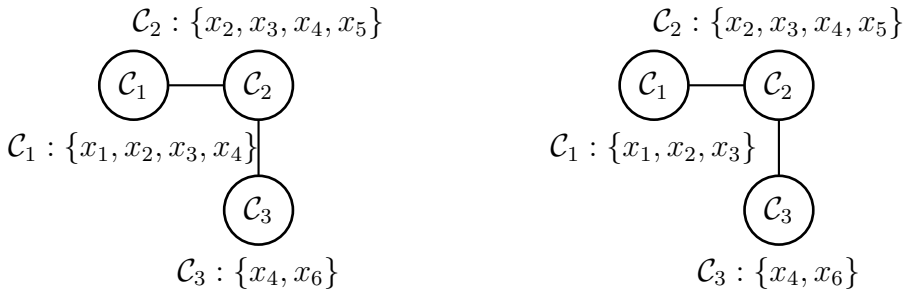
(a) Original (triangulated) Graph **(b)** Original JT

(c) Sparse Graph **(d)** Sparse JT

Figure 2.24 – Relationship between link removal and clique splitting. Note that there may be more than one valid JT for any graph, so link removal can affect the different JTs differently. (a) and (b) show the original graph and a corresponding Junction Tree. The clique $\mathcal{C}_2$ is then split, into two cliques where neither clique contains both x_5 and x_4. This is equivalent to removing the link between x_5 and x_4 in the underlying triangulated graph. This is an approximation such that $\Psi_{\mathcal{C}_2} \neq \Psi_{\mathcal{C}_{2a}}\Psi_{\mathcal{C}_{2b}}$. Note that the split results in $\mathcal{C}_{2a}$ becoming a subset of $\mathcal{C}_1$. This clique can now be merged with $\mathcal{C}_1$.

2.5.4 Summary

This section has described methods for link removal and clique splitting to produce more efficient, yet approximate estimates of linear systems. It has briefly discussed existing techniques, and highlighted the need for approximate solutions which can maintain existing sparsity, minimise the KLD and ensure the conservativeness of estimates.



(a) Original JT **(b)** Approximate JT

Figure 2.25 – Simple example of TJT clique thinning, (a) shows a simple JT. Observe that x_4 is a member of all cliques. (b) shows the JT structure with $\mathcal{C}_1$ thinned by removing x_4. Note that x_4 cannot be removed from $\mathcal{C}_2$ as this will violate the running intersection property.

Chapter 3

Conservative Sparsification for Information Matrices

This chapter describes an approach to sparsification that can remove links in an information matrix while maintaining guarantees on the conservativeness of the estimates. This is a general approach that is only computationally feasible for small problems, and but it will be shown that large problems can often be reduced to a computationally feasible problem. The chapter will also consider the effect of performing conservative sparsification in a sequential manner, and finally show demonstrate the approach using a simulated linear SLAM problem.

3.1 Conservative Optimisation

This section describes the methodology used to construct a conservative sparsification problem. It is necessary to first define the cost function and ideal sparsity constraints, these constraints will be relaxed to form a convex problem which becomes a starting point for developing the theory of conservative sparsification.

3.1.1 Defining the Problem

Given a multidimensional Gaussian distribution whose information matrix is sparse, the aim is to find a new information matrix that has fewer non-zero elements and is conservative with respect to the original distribution.

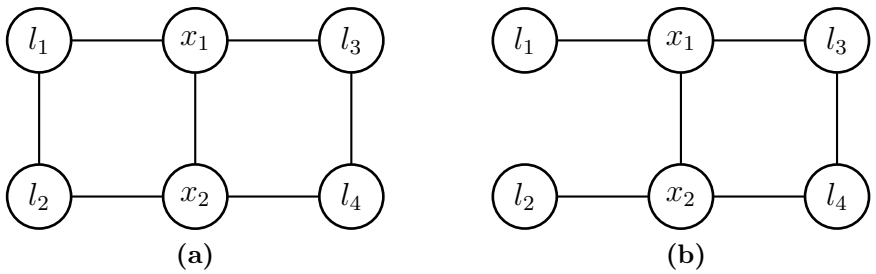


Figure 3.1 – The aim is to make a graph which is already sparse (a) more sparse (fewer edges or non-zero elements) by the removal of links (b).

Let the true distribution be $\mathcal{N}_{tr}^{-1}(\mathbf{x}; \mathbf{y}_{tr}, \mathbf{Y}_{tr})$, and let the approximating distribution be $\mathcal{N}_{app}^{-1}(\mathbf{x}; \mathbf{y}_{app}, \mathbf{Y}_{app})$. It is required that $\mathcal{N}_{app}^{-1}$ be a conservative estimate of $\mathcal{N}_{tr}^{-1}$, which is equivalent to:

$$\mathbf{Y}_{tr} - \mathbf{Y}_{app} \succeq \mathbf{0} \tag{3.1}$$

$$\mathbf{Y}_{tr}^{-1}\mathbf{y}_{tr} = \mathbf{Y}_{app}^{-1}\mathbf{y}_{app} \tag{3.2}$$

Equation 3.2 enforces that the mean estimates will be equal. Rearranging 3.2 gives:

$$\mathbf{y}_{app} = \mathbf{Y}_{app}\mathbf{Y}_{tr}^{-1}\mathbf{y}_{tr} \tag{3.3}$$

Therefore, the parameter $\mathbf{y}_{app}$ is determined by the matrix variable $\mathbf{Y}_{app}$. Thus, $\mathbf{Y}_{app}$ is the optimisation parameter.

It is important that the new system be a close approximation of the original system. The KLD is a standard measure of divergence between two distributions. Since $\mathrm{D}_{\mathrm{KL}}\left(\mathcal{N}_{tr}^{-1} || \mathcal{N}_{app}^{-1}\right)$ is convex in $\mathbf{Y}_{app}$ (Section 2.5.1), this leads to the following convex

cost function[1]:

$$\min_{\mathbf{Y}_{app}} \text{tr}(\mathbf{Y}_{tr}^{-1}\mathbf{Y}_{app}) - \text{logdet}(\mathbf{Y}_{app}) \tag{3.4}$$

Combining the cost function and constraints results in:

$$\begin{aligned} &\min_{\mathbf{Y}_{app}} \text{tr}(\mathbf{Y}_{tr}^{-1}\mathbf{Y}_{app}) - \text{logdet}(\mathbf{Y}_{app}) \\ &s.t. \\ &\quad ||\mathbf{Y}_{app}||_0 \leq k, \\ &\quad \mathbf{Y}_{app} \succeq \mathbf{0}, \\ &\quad \mathbf{Y}_{tr} - \mathbf{Y}_{app} \succeq \mathbf{0}, \end{aligned} \tag{3.5}$$

where $||\mathbf{Y}_{app}||_0$ is the L0 norm of the matrix, or a count of the number of non-zero elements in the matrix. This problem can be stated *find a conservative approximation to* $\mathbf{Y}_{tr}$ *such that the number of non-zeros is less than* k*, and the KLD between* $\mathbf{Y}_{tr}$ *and* $\mathbf{Y}_{app}$ *is minimised.* where m is the number of nonzero values in $\mathbf{Y}_{tr}$ and $k < m$. Unfortunately, the use of the L0 norm makes finding an optimal solution to this problem N.P. Hard [45].

Covariance Selection approaches[7][23][16] relax the L0 constraint by adding a L1 term to the cost function. This yields a convex problem as the L1 norm is convex. Applying this technique to Equation 3.5 yields a new problem called Conservative Covariance Selection.

$$\begin{aligned} &\min_{\mathbf{Y}_{app}} \text{tr}(\mathbf{Y}_{tr}^{-1}\mathbf{Y}_{app}) - \text{logdet}(\mathbf{Y}_{app}) + \rho||\mathbf{Y}_{app}||_1, \\ &s.t. \\ &\quad \mathbf{Y}_{app} \succeq \mathbf{0}, \\ &\quad \mathbf{Y}_{tr} - \mathbf{Y}_{app} \succeq \mathbf{0}. \end{aligned} \tag{3.6}$$

[1]Note that optimising $\text{D}_{\text{KL}}\left(\mathcal{N}_{app}^{-1}||\mathcal{N}_{tr}^{-1}\right)$ will yield the cost function $\min_{\mathbf{Y}_{app}} \text{tr}(\mathbf{Y}_{app}^{-1}\mathbf{Y}_{tr}) + \text{logdet}(\mathbf{Y}_{app})$. The solutions found using this cost function will differ from those which minimise $\text{D}_{\text{KL}}\left(\mathcal{N}_{tr}^{-1}||\mathcal{N}_{app}^{-1}\right)$. However, since this alternate form is not convex in $\mathbf{Y}_{app}$ it is not pursued further.

A significant disadvantage of this approach is the requirement to choose the weight ρ of the L1 norm. This problem also inherits Covariance Selections time complexity properties making it prohibitive for large matrices. The computational complexity is $\mathcal{O}(n^{2.5}k^2)$, where n is the matrix dimension (see Appendix B.3.1.1). For large n this will become intractable.

3.1.2 Conservative Sparsification (CS)

Rather than adding an L0 term to the cost function, a sparsity pattern is chosen apriori. This removes the $\rho||\mathbf{Y}_{app}||_1$ term from the cost function of Equation 3.6, but requires extra equality constraints to express the desired sparsity. Let E_{app} contain the index pairs (i, j) that index the non-zeros to be removed, and let E_{tr} contain the index of zero valued entries in $\mathbf{Y}_{tr}$, the CS problem is defined as a convex semidefinite program:

$$\begin{aligned} &\min_{\mathbf{Y}_{app}} \operatorname{tr}(\mathbf{Y}_{tr}^{-1}\mathbf{Y}_{app}) - \operatorname{logdet}(\mathbf{Y}_{app}) \\ &s.t. \\ &\quad \mathbf{Y}_{app} \succeq \mathbf{0} \\ &\quad (\mathbf{Y}_{app})_{ij} = 0, \quad \forall\{i,j\} \in E_{app} \\ &\quad (\mathbf{Y}_{tr})_{ij} = 0, \quad \{i,j\}\forall \in E_{tr} \\ &\quad \mathbf{Y}_{tr} - \mathbf{Y}_{app} \succeq \mathbf{0} \end{aligned} \tag{3.7}$$

The problem statement is: *Find a matrix $\mathbf{Y}_{app}$ that minimises the KLD from $\mathbf{Y}_{tr}$, is conservative with respect to $\mathbf{Y}_{tr}$, and has a predetermined sparsity pattern.* This is a Maximum Determinant (MAXDET) problem as described in [64] and is readily solved using an interior point method such as the solver SDPT3 ([56][57]), or a Newton-conjugate gradient method such as logDetPPA [62]. Thus it still has a time complexity of $\mathcal{O}(n^{2.5}k^2)$, and is in feasible for large problems.

3.2 Efficient Implementation of CS

By using a predefined sparsity pattern rather than a L0 norm, CS has removed a parameter from the problem, but the time complexity to solve the problem remains very high, at $\mathcal{O}(n^{2.5}k^2)$. This section describes a method of reducing this complexity. This method requires that the matrix $\mathbf{Y}_{tr}$ and the links chosen for removal, satisfy certain conditions. One important caveat is that the analysis will not include the one-off cost inverting $\mathbf{Y}_{tr}$, which is required to compute the term $\text{tr}(\mathbf{Y}_{tr}^{-1}\mathbf{Y}_{app})$ in Equation 3.7. As marginalisation and matrix inversion are related (Section 2.2.3), Chapter 4 will discuss the use of the Junction Tree algorithm to provide an efficient approach to marginalisation and local matrix inversion.

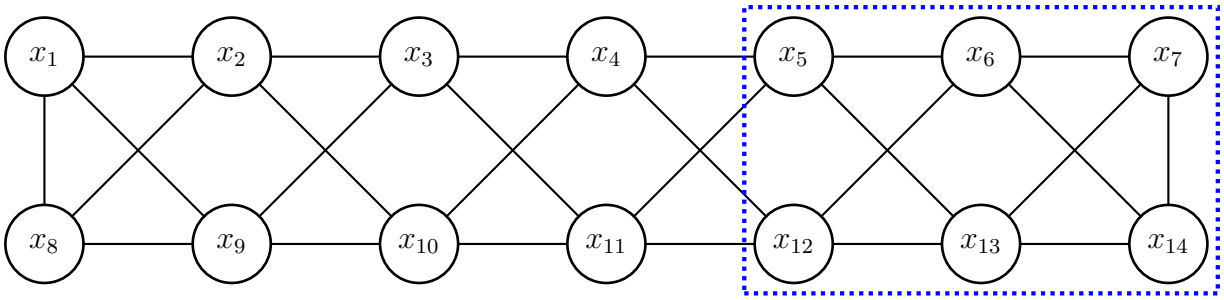
3.2.1 Efficient Conservative Sparsification

The current formulation of CS applies to all parts of the graph, and for large graphs will lead to computationally intractable problems. To be computationally feasible for removing fill-in, a sparsification algorithm needs to also have a local cost. That is, the cost must not be a function of the global state size, but rather some property which is localised to the new sparsity. This section demonstrates that CS can be reformulated to have this property. In problems such as SLAM or DSN, the graph may be already sparse, but marginalisation can induce fill-in. Marginalisation is local to the Markov blanket around the node that was removed. Figure 3.2 shows an example graph where marginalisation does not involve large parts of the graph. If many of these links have low weight they will be good candidates for removal [35][54].

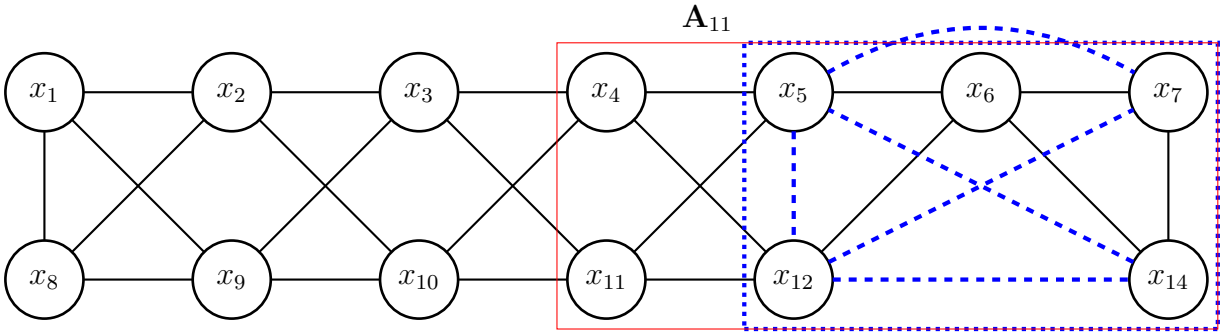
Let the problem variables be decomposed as follows:

$$\mathbf{Y}_{tr} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{12}^T & \mathbf{A}_{22} \end{bmatrix}, \quad \mathbf{Y}_{app} = \begin{bmatrix} \mathbf{B}_{11} & \mathbf{B}_{12} \\ \mathbf{B}_{12}^T & \mathbf{B}_{22} \end{bmatrix}, \tag{3.8}$$

where the matrix sizes are equivalent, that is $dim(\mathbf{A}_{11}) = dim(\mathbf{B}_{11})$. Now assume that all the new equality constraints E_{app} apply only to elements in $\mathbf{A}_{11}$. Also, let



(a) Original Graph



(b) Large graph after marginalisation of x_{13}

Figure 3.2 – Large sparse graph before and after marginalisation. The Markov blanket around x_{13} is indicated as a dotted blue box, and fill-in shown as dashed blue edges. Observe that this marginalisation does not effect every node, thus a sparsification routine for removing fill-in would benefit from also having a local formulation. The red box in (b) shows a particular choice of partition for $\mathbf{A}_{11}$ in the problem decomposition from Equation 3.8. If a solution is found using this partition, then only vertexes and edges inside this box will be altered by the solution to CS.

E_{tr11} be the set of edges that are zero in $\mathbf{A}_{11}$ (that is E_{tr11} represents the prior sparsity of $\mathbf{A}_{11}$).

Therefore a new cost function could be written in terms of sub-matrices:

$$\min_{\mathbf{B}_{11}} \operatorname{tr}((\mathbf{A}_{11} - \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{12}^{T})^{-1}\mathbf{B}_{11}) - \operatorname{logdet}(\mathbf{B}_{11} - \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{12}^{T}) \tag{3.9}$$

$s.t.$

$$\mathbf{B}_{11} \succ \mathbf{0}$$

$$\mathbf{B}_{11} - \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{12}^{T} \succ \mathbf{0}$$

$$\mathbf{A}_{11} - \mathbf{B}_{11} \succeq \mathbf{0}$$

$$(\mathbf{B}_{11})_{i,j} = 0, \forall \{i,j\} \in E_{app} \cup E_{tr11}.$$

Observe that this problem now modifies only parts of the graph that correspond to non-zeros within $\mathbf{A}_{11}$, and the problem complexity is a function of the dimension of $\mathbf{A}_{11}$ rather than the full problem dimension[2]. Therefore, this efficient approach to CS is has a local cost. However, it is required to show that this new problem is equivalent to the full CS problem in Equation 3.7. To do this the following lemma and corollary are required:

Lemma 3.1. *If* $p(\mathbf{x}_1, \mathbf{x}_2) = p(\mathbf{x}_1)p(\mathbf{x}_2|\mathbf{x}_1)$, $q(\mathbf{x}_1, \mathbf{x}_2) = q(\mathbf{x}_1)q(\mathbf{x}_2|\mathbf{x}_1)$ *and* $p(\mathbf{x}_1) \neq q(\mathbf{x}_1)$, *then the* $\mathrm{D}_{\mathrm{KL}}(p||q)$ *is minimised when* $p(\mathbf{x}_2|\mathbf{x}_1) = q(\mathbf{x}_2|\mathbf{x}_1)$.

Proof. The KLD between p and q is:

$$\min_q \mathrm{D}_{\mathrm{KL}}(p||q) = \min_q \iint p(\mathbf{x}_1)p(\mathbf{x}_2|\mathbf{x}_1) \log \frac{p(\mathbf{x}_1)p(\mathbf{x}_2|\mathbf{x}_1)}{q(\mathbf{x}_1)q(\mathbf{x}_2|\mathbf{x}_1)} d\mathbf{x}_2 d\mathbf{x}_1 \tag{3.10}$$

$$= \min_q \iint p(\mathbf{x}_1)p(x_2|x_1) \log \frac{p(\mathbf{x}_1)}{q(\mathbf{x}_1)}) d\mathbf{x}_2 d\mathbf{x}_1$$

$$+ \min_q \iint p(\mathbf{x}_1)p(\mathbf{x}_2|\mathbf{x}_1) \log \frac{p(\mathbf{x}_2|\mathbf{x}_1)}{q(\mathbf{x}_1|\mathbf{x}_2)} d\mathbf{x}_2 d\mathbf{x}_1 \tag{3.11}$$

Considering only the second term:

$$\min_q \iint p(\mathbf{x}_1)p(\mathbf{x}_2|\mathbf{x}_1) \log \frac{p(\mathbf{x}_2|\mathbf{x}_1)}{q(\mathbf{x}_2|\mathbf{x}_1)}) d\mathbf{x}_2 d\mathbf{x}_1 \tag{3.12}$$

$$= \min_q \int p(\mathbf{x}_1) \int p(\mathbf{x}_2|\mathbf{x}_1) \log \frac{p(\mathbf{x}_2|\mathbf{x}_1)}{q(\mathbf{x}_2|\mathbf{x}_1)} d\mathbf{x}_2 d\mathbf{x}_1 \tag{3.13}$$

$$= \min_q \int p(\mathbf{x}_1) \mathrm{D}_{\mathrm{KL}}(p(\mathbf{x}_2|\mathbf{x}_1)||q(\mathbf{x}_2|\mathbf{x}_1)) d\mathbf{x}_1 \tag{3.14}$$

Notice that $\mathrm{D}_{\mathrm{KL}}(p(\mathbf{x}_2|\mathbf{x}_1)||q(\mathbf{x}_2|\mathbf{x}_1))$ term is uniquely minimised when

$$p(\mathbf{x}_2|\mathbf{x}_1) = q(\mathbf{x}_2|\mathbf{x}_1). \tag{3.15}$$

□

The following corollary applies Lemma 1 to Gaussian distributions.

[2] $\mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{12}^T$ is a marginal projection onto $\mathbf{A}_{11}$, and is a local quantity for some estimation approaches such as JT, see Section 2.3.

Corollary 1. *For two Gaussian distributions using the information form* $p(\mathbf{x}) = \mathcal{N}_1^{-1}(\mathbf{x}; \mathbf{y}_1, \mathbf{Y}_{tr})$, $q(\mathbf{x}) = \mathcal{N}_2^{-1}(\mathbf{x}; \mathbf{y}_2, \mathbf{Y}_{app})$, *with the following partitions:*

$$\mathbf{Y}_{tr} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{12}^T & \mathbf{A}_{22} \end{bmatrix}, \quad \mathbf{Y}_{app} = \begin{bmatrix} \mathbf{B}_{11} & \mathbf{B}_{12} \\ \mathbf{B}_{12}^T & \mathbf{B}_{22} \end{bmatrix}, \tag{3.16}$$

then the $\mathrm{D}_{\mathrm{KL}}(p(\mathbf{x})||q(\mathbf{x}))$ *is minimised when* $\mathbf{A}_{12} = \mathbf{B}_{12}$ *and* $\mathbf{A}_{22} = \mathbf{B}_{22}$.

Proof. Consider the factorisation $q(x_1, x_2) = q(x_1)q(x_2|x_1)$. The information matrix for this is:

$$\mathbf{Y}_{app} = \underbrace{\begin{bmatrix} \mathbf{B}_{11} - \mathbf{B}_{12}\mathbf{B}_{22}^{-1}\mathbf{B}_{12}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}}_{q(x_1)} + \underbrace{\begin{bmatrix} \mathbf{B}_{12}\mathbf{B}_{22}^{-1}\mathbf{B}_{12}^T & \mathbf{B}_{12} \\ \mathbf{B}_{12}^T & \mathbf{B}_{22} \end{bmatrix}}_{q(x_2|x_1)} \tag{3.17}$$

similarly for $p(x_1, x_2) = p(x_1)p(x_2|x_1)$:

$$\mathbf{Y}_{tr} = \underbrace{\begin{bmatrix} \mathbf{A}_{11} - \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{12}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}}_{p(x_1)} + \underbrace{\begin{bmatrix} \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{12}^T & \mathbf{A}_{12} \\ \mathbf{A}_{12}^T & \mathbf{A}_{22} \end{bmatrix}}_{p(x_2|x_1)} \tag{3.18}$$

Using Lemma 3.1, then $\mathrm{D}_{\mathrm{KL}}(p||q)$ is minimised when $p(x_1|x_2) = q(x_1|x_2)$, which is equivalent to requiring that $\mathbf{A}_{12} = \mathbf{B}_{12}$ and $\mathbf{A}_{22} = \mathbf{B}_{22}$ □

Note that this corollary is true regardless of the value of $\mathbf{A}_{11}$. The following theorem shows that the problem described by Equation 3.9 is equivalent to CS, and that only the elements of $\mathbf{A}_{11}$ are modified. With reference to Figure 3.2(b), assuming that a solution was found, then all vertices and edges within the red square (i.e., states $\{x_4, x_5, x_6, x_6, x_{11}, x_{12}, x_{14}\}$) are modified, while all other vertices and edges (i.e., states $\{x_1, x_2, x_3, x_8, x_9, x_{10}\}$) remain unchanged by the solution to CS.

Theorem 3.1. *Let the optimal solution to the problem in Equation 3.9 (if it exists)*

be $\mathbf{B}_{11}^*$*, then the solution to the problem in Equation 3.7 is*

$$\mathbf{Y}_{app} = \begin{bmatrix} \mathbf{B}_{11}^* & \mathbf{A}_{12} \\ \mathbf{A}_{12}^T & \mathbf{A}_{22} \end{bmatrix}. \tag{3.19}$$

Proof outline: This proof will show that the problem in Equation 3.9 can be derived from the CS problem by analysing each of the additional constraints introduced by the problem in Equation 3.9.

Proof. Let $p(\mathbf{x}) = \mathcal{N}^{-1}(\mathbf{x}; \mathbf{y}_{tr}, \mathbf{Y}_{tr})$ and $q(\mathbf{x}) = \mathcal{N}^{-1}(\mathbf{x}; \mathbf{y}_{app}, \mathbf{Y}_{app})$ be Gaussian distributions with co-incident means and information matrices $\mathbf{Y}_{tr}$ and $\mathbf{Y}_{app}$ partitioned in the following way (let $\mathbf{A}_{11}$ be the sub-block associated with the states $\mathbf{x}_1$, and $\mathbf{A}_{22}$ be the sub-block and $\mathbf{x}_2$):

$$\mathbf{Y}_{tr} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{12}^T & \mathbf{A}_{22} \end{bmatrix}, \quad \mathbf{Y}_{app} = \begin{bmatrix} \mathbf{B}_{11} & \mathbf{B}_{12} \\ \mathbf{B}_{12}^T & \mathbf{B}_{22} \end{bmatrix}. \tag{3.20}$$

Applying Corollary 1, the optimal values for $\mathbf{B}_{12}$ and $\mathbf{B}_{22}$ are $\mathbf{A}_{12}$ and $\mathbf{A}_{22}$ respectively. This results in the following:

$$\mathbf{Y}_{app} = \begin{bmatrix} \mathbf{B}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{12}^T & \mathbf{A}_{22} \end{bmatrix}. \tag{3.21}$$

Using this substitution, the cost function can be decomposed into:

$$\min_{\mathbf{Y}_{app}} \mathrm{D}_{\mathrm{KL}}\left(\mathbf{Y}_{tr} || \mathbf{Y}_{app}\right) \tag{3.22}$$

$$= \min_{\mathbf{Y}_{app}} \mathrm{tr}(\mathbf{Y}_{tr}^{-1}\mathbf{Y}_{app}) - \mathrm{logdet}(\mathbf{B}_{11} - \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{12}^T) - \mathrm{logdet}(\mathbf{A}_{22}) \tag{3.23}$$

$$= \min_{\mathbf{B}_{11}} \mathrm{tr}((\mathbf{A}_{11} - \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{12}^T)^{-1}\mathbf{B}_{11}) - \mathrm{logdet}(\mathbf{B}_{11} - \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{12}^T) + C, \tag{3.24}$$

where C is a constant containing the terms not dependent upon $\mathbf{B}_{11}$. This proof will now consider each matrix constraint and show that they can be expressed in

terms only related to $\mathbf{B}_{11}$. The constraints are $\mathbf{Y}_{app} \succ \mathbf{0}$, $\mathbf{Y}_{tr} \succeq \mathbf{Y}_{app}$ and sparsity constraints on sub-block $\mathbf{B}_{11}$.
Consider the constraint $\mathbf{Y}_{app} \succ \mathbf{0}$. Using Equation 3.21 and Equation A.9 this is equivalent to,

$$\begin{aligned}&\mathbf{B}_{11} \succ \mathbf{0},\\ &\mathbf{B}_{11} - \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{12}^{T} \succ \mathbf{0}.\end{aligned}$$

Now consider:

$$\mathbf{Y}_{tr} \succeq \mathbf{Y}_{app} \tag{3.25}$$

Using the substitution in Equation 3.21 simplifies this constraint to,

$$\mathbf{A}_{11} - \mathbf{B}_{11} \succeq \mathbf{0}. \tag{3.26}$$

As the global equality constraints are only specified for the sub-block $\mathbf{B}_{11}$, their specification does not need to be changed. The problem in Equation 3.7 can now be written with respect to $\mathbf{B}_{11}$ only.

$$\begin{aligned}&\min_{\mathbf{B}_{11}} \operatorname{tr}((\mathbf{A}_{11} - \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{12}^{T})^{-1}\mathbf{B}_{11}) - \operatorname{logdet}(\mathbf{B}_{11} - \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{12}^{T}) \qquad (3.27)\\ &s.t.\\ &\quad \mathbf{B}_{11} \succ \mathbf{0}\\ &\quad \mathbf{B}_{11} - \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{12}^{T} \succ \mathbf{0}\\ &\quad \mathbf{A}_{11} - \mathbf{B}_{11} \succeq \mathbf{0}\\ &\quad (\mathbf{B}_{11})_{i,j} = 0, \forall \{i,j\} \in E_{app} \cup E_{tr11}.\end{aligned}$$

which is identical to the problem in Equation 3.9. □

The following corollary is useful for large distributed systems.

Corollary 2. *To solve a CS problem the full matrix* $\mathbf{Y}_{tr}$ *is not required. Only* $\mathbf{A}_{11}$

and the result $\mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{12}^{T}$ *is required.*

Proof. This is true as only the terms $\mathbf{A}_{11}$ and $\mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{12}^{T}$ are present in equation 3.27. □

The reduction in the problem size has significantly reduced the cost. The cost to solve a CS problem is now $\mathcal{O}(a^{2.5}|\mathbf{B}_{11}|^2)$, where a is the dimension of the sub-matrix $\mathbf{A}_{11}$ and $|\mathbf{B}_{11}|$ is the number of non-zeros in the approximate solution.

3.2.2 Feasible Set for Optimisation

A caveat of Theorem 3.1 is that the reduced problem must be feasible. This relates to the states that are selected for membership of $\mathbf{A}_{11}$. Since full CS is a convex problem, Slaters condition states that it is sufficient to find a feasible point to determine if a problem has an optimal solution [10]. For full CS it is clear that $\mathbf{Y}_{app} = \epsilon\mathbf{I}$ is always a feasible solution (for small ϵ). However for the reduced CS problem $\mathbf{B}_{11} = \epsilon\mathbf{I}$ is not in general a feasible solution, and for some choices of $\mathbf{A}_{11}$ a feasible solution may not exist. This section describes a simple heuristic for choosing the states within $\mathbf{A}_{11}$ that can be used to ensure feasibility for a particular class of problems.

The following definitions are now required:

Definition 3.1. *A CS problem is locally chordal if the non-zero pattern of the expression* $\mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{12}^{T}$ *is a subset of the non-zero pattern of* $\mathbf{B}_{11}$*.*

Definition 3.2. *The Markov blanket around an edge is defined as the union of the Markov blankets of the two vertices.*

These definitions allow the following theorem.

Theorem 3.2. *For a locally chordal CS problem, reducing to the Markov blanket around the new edge* E_{app} *will always result in a feasible problem.*

Proof outline: This proof will show that for a chordal graph, when the partition is chosen by selecting the Markov blanket, that there always exists a feasible solution to efficient CS with that partition.

Proof. Assume that our problem has the following chordal sparsity pattern[3].

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \mathbf{0} \\ \mathbf{A}_{12}^T & \mathbf{A}_{22} & \mathbf{A}_{23} \\ \mathbf{0} & \mathbf{A}_{23}^T & \mathbf{A}_{33} \end{bmatrix} \succ \mathbf{0}, \tag{3.28}$$

where additional sparsity is only introduced into the matrix represented by $\mathbf{A}_{11}$. A feasible solution $\mathbf{B}$ as can be constructed follows (where ϵ is a small positive value such that $\mathbf{A}_{11} - \epsilon\mathbf{I} \succ \mathbf{0}$ and $\mathbf{A}_{22} - \mathbf{A}_{12}^T(\mathbf{A}_{11} - \epsilon\mathbf{I})^{-1}\mathbf{A}_{12} \succ \mathbf{0}$):

$$\mathbf{B} = \begin{bmatrix} \epsilon\mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{22} - \mathbf{A}_{12}^T(\mathbf{A}_{11} - \epsilon\mathbf{I})^{-1}\mathbf{A}_{12} & \mathbf{A}_{23} \\ \mathbf{0} & \mathbf{A}_{23}^T & \mathbf{A}_{33} \end{bmatrix}. \tag{3.29}$$

Observe that $\mathbf{B} \succ \mathbf{0} \iff \mathbf{A} \succ \mathbf{0}$. Also considering the inequality $\mathbf{A} - \mathbf{B} \succeq \mathbf{0}$,

$$\begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \mathbf{0} \\ \mathbf{A}_{12}^T & \mathbf{A}_{22} & \mathbf{A}_{23} \\ \mathbf{0} & \mathbf{A}_{23} & \mathbf{A}_{33} \end{bmatrix} - \begin{bmatrix} \epsilon\mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{22} - \mathbf{A}_{12}^T(\mathbf{A}_{11} - \epsilon\mathbf{I})^{-1}\mathbf{A}_{12} & \mathbf{A}_{23} \\ \mathbf{0} & \mathbf{A}_{23} & \mathbf{A}_{33} \end{bmatrix} \succeq \mathbf{0} \tag{3.30}$$

$$\begin{bmatrix} \mathbf{A}_{11} - \epsilon\mathbf{I} & \mathbf{A}_{12} & \mathbf{0} \\ \mathbf{A}_{12}^T & \mathbf{A}_{12}^T(\mathbf{A}_{11} - \epsilon\mathbf{I})^{-1}\mathbf{A}_{12} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \succeq \mathbf{0} \tag{3.31}$$

Equation 3.31 is true $i.f.f. \mathbf{A} \succ \mathbf{0}$.

Finally we must check that $\mathbf{B}_{22} - \mathbf{A}_{23}\mathbf{A}_{33}^{-1}\mathbf{A}_{23}^T \succeq \mathbf{0}$.

[3]This is a general pattern found in sparse chordal problems, the partition of the states into $\mathbf{A}_{11}$ and $\mathbf{A}_{22}$ is quite flexible.

$$\mathbf{B}_{22} - \mathbf{A}_{23}\mathbf{A}_{33}^{-1}\mathbf{A}_{23}^{T} = \mathbf{A}_{22} - \mathbf{A}_{12}^{T}(\mathbf{A}_{11} - \epsilon\mathbf{I})^{-1}\mathbf{A}_{12} - \mathbf{A}_{23}\mathbf{A}_{33}^{-1}\mathbf{A}_{23}^{T} \tag{3.32}$$

$$\succ \mathbf{0}, \text{ if } \mathbf{A} \succ \mathbf{0} \tag{3.33}$$

Therefore $\mathbf{B}$ is a feasible solution. □

This proof is significant as it bounds the dimensions of $\mathbf{A}_{11}$ to be a function of the cardinality of the vertices. Note that in the case of non-chordal graphs, choosing $\mathbf{B}_{22} = \mathbf{A}_{22} - \mathbf{A}_{12}^{T}(\mathbf{A}_{11} - \epsilon\mathbf{I})^{-1}\mathbf{A}_{12}$ may result in fill-in in $\mathbf{B}_{22}$, thus violating the constraint to maintain all pre-existing zero terms and thus is not a feasible solution, Section 3.2.3 describes a solution to this problem.

Lemma 3.2. *Let the maximum cardinality of any vertex in a graph be k. Then the number of vertices in the Markov blanket around a single edge is less than or equal to $2k$.*

Proof. By the definition of cardinality, each vertex will have at most k neighbours in its Markov blanket, thus the union of their individual Markov blankets is at most $2k$. □

This lemma is demonstrated in Figure 3.3. Using this lemma a bound on the cost of sparsification can be determined for locally chordal problems. The cost is $\mathcal{O}((2k)^{2.5}(2kk)^2) = \mathcal{O}(k^{6.5})$. Note that for sparse graphs (where $k << n$) computationally hard problems for Conservative Covariance Selection (Equation 3.6) should be tractable for locally chordal CS problems.

3.2.3 Chordal Conservative Sparsification

There are two solutions to the infeasibility of a general reduced CS problem, expanding the Markov blanket and approximating the sparsity with a chordal pattern.

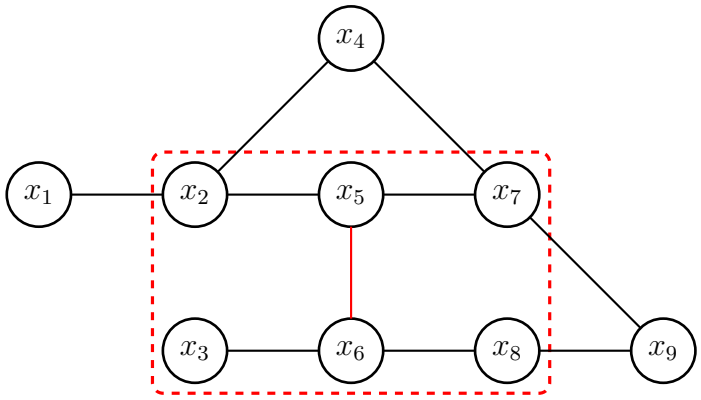


Figure 3.3 – This figure demonstrates Lemma 3.2 for the case $k = 3$. The link to be removed is in red. The Markov blanket is indicated using red dashed, it contains 6 nodes, which is equal to $2k$

In some cases it may be possible to increase the size of $\mathbf{A}_{11}$ such that it is locally chordal. However there is no guarantee that such a situation will exist until the dimensions of $\mathbf{A}_{11}$ coincide with the dimensions of $\mathbf{Y}_{tr}$. Figure 3.5 demonstrates a sparsification problem where expanding the Markov blanket will not result in a locally chordal graph until the entire graph is contained in $\mathbf{A}_{11}$.

A CS problem can be relaxed to a locally chordal problem by removing the edges from E_{tr}, which correspond to fill-in locations from the marginal of all other states in the system. This new problem, called Chordal Conservative Sparsification (CCS), has the same form of the reduced CS but is always guaranteed to have a feasible solution. A significant drawback is that allowing fill-in may increase the number of non-zeros in the system. This also has the effect of restricting the elimination ordering that can be used (as the fill-in is dependent upon the particular elimination ordering).

This technique is similar to the Chordal Embedding approach to Covariance Selection [13]. CCS differs in that only the edges that interface between the Markov blanket and the external network are required to be filled in. Figure 3.4 demonstrates the differences between a local and global chordal embedding.

Figure 3.6 shows a comparison between CCS and CS. This figure was generated by choosing links to remove from the matrix LF10 in the Oberwalfach set from the University of Florida's Sparse Matrix Collection [15]. The x-axis shows independent

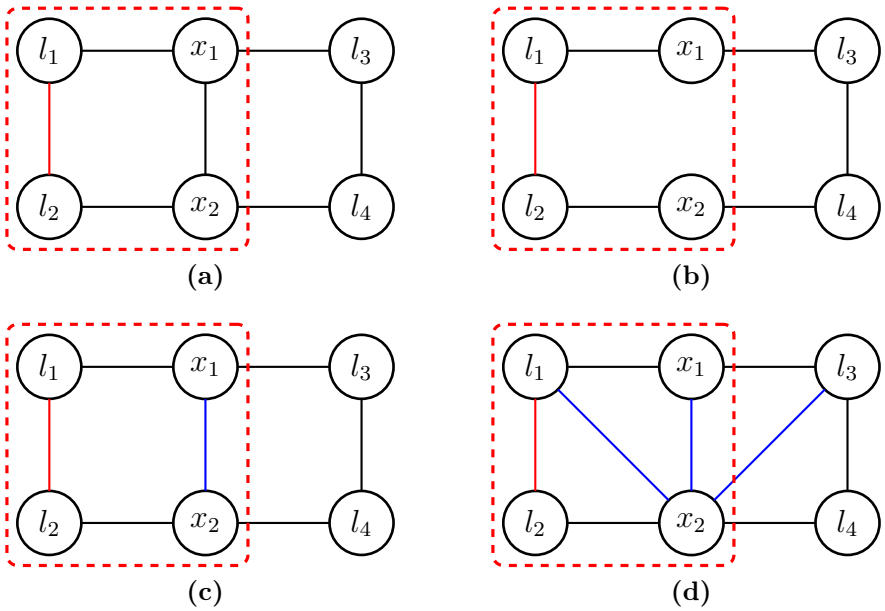


Figure 3.4 – A red edge indicates the link to be sparsified, a blue edge indicates an edge added by an embedding. The red dashed box indicates the Markov blanket around the link to be sparsified. (a) A graph that is locally chordal. (b) A graph that is not locally chordal. (c) The locally chordal embedding for graph (b). (d) The chordal embedding for graph (b). Comparing (c) to (d) it can be seen that a complete chordal embedding adds more links than a local chordal embedding.

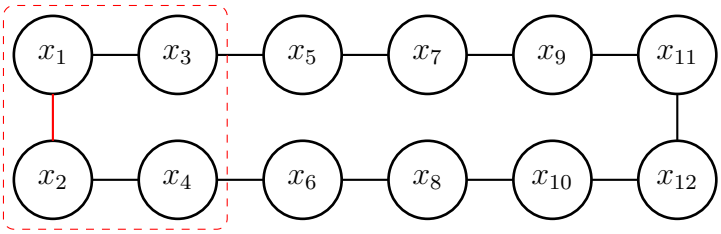


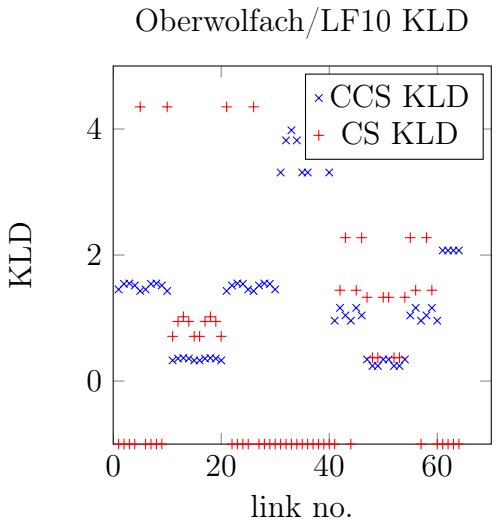
Figure 3.5 – The link between x_1 and x_2 is chosen for removal. However the Markov blanket around this is not locally chordal, expanding the Markov blanket will not result in a locally chordal graph until the entire graph is included.

runs where a single edge is chosen for removal. The runs are independent because each run is performed on the original matrix without any other induced sparsity. An edge was chosen only if its Markov blanket was not locally chordal. A KLD of -1 is plotted for links that have no solution using the CS.

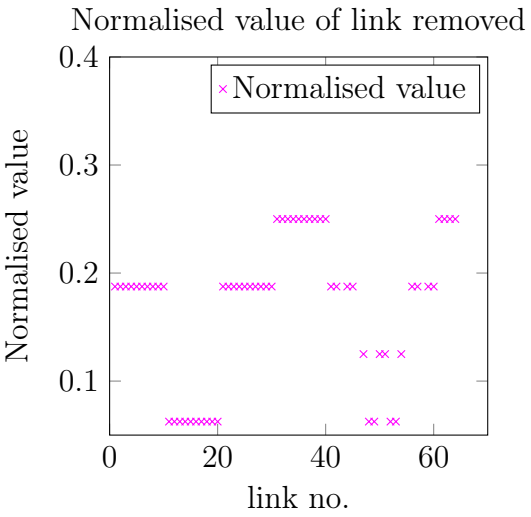
Figure 3.6(a) shows that CCS can find a solution even when CS is in-feasible. Figure 3.6(c) shows the amount of fill-in used by the CCS algorithm. Note that the key

relaxation of CCS is that some existing sparsity in the Markov blanket is allowed to be filled in to ensure feasibility. It is interesting to observe that the algorithm does not always use all of the allowed fill-in. Figure 3.6(a) shows that the KLD for CCS is a lower bound for the KLD of CS. Appendix C.2 demonstrates this approach on several other matrices from the sparse matrix library. In Figure 3.7 CCS always used all the available fill-in edges, observe that this resulted in extra fill-in of between 50 to 70 new edges, since CCS is only removing a single existing edge, this represents a large decrease in the sparsity of the matrix. Thus CCS must be used with care. In all cases CCS has a lower KLD but in some cases CCS can reduce the amount of sparsity. The lower KLD of the CCS approach compared with the CS is due to the additional variables which allow the CCS problem to more closely represent the original distribution, but at the cost of increased sparsity.

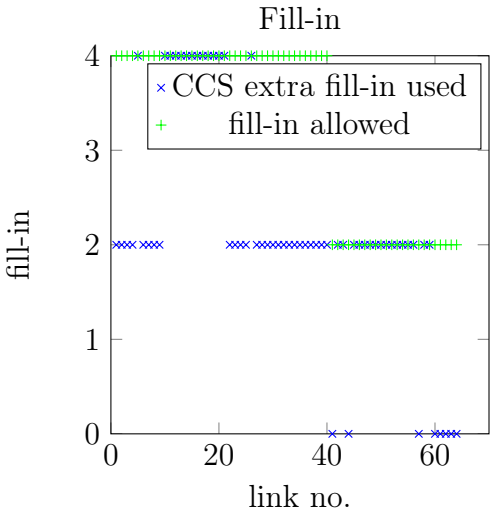
Single link sparsification can be extended to multiple link removal. For CS all that is required is to find the union of the Markov blankets of the links to be removed. Once the Markov blanket is known a convex problem can be formed using Equation 3.9, by adding the new link to the set of sparsity constraints. CCS can be used by enforcing the local chordal property of the combined Markov blanket. Figure 3.8 shows the intersection of two Markov blankets, and the edge required to use CCS. For the case where the Markov blankets of the links to be removed are disjoint, see Section 3.2.5.



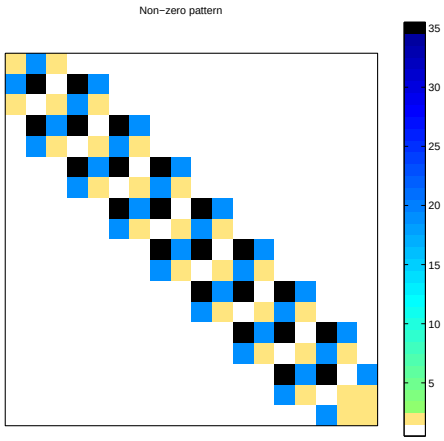
(a) KLD



(b) Normalised Link Value



(c) Extra Fill-In Used by CCS



(d) Color Spy plot of LF10

Figure 3.6 – An example of the tradeoff for CCS vs CS, on the Oberwalfach matrix LF10 from the SuiteSparse matrix library [15]. Links are selected for removal if they have a Markov blanket that is not locally chordal, they are then ordered and the effect of removing a single link independently is examined. A KLD of -1 indicates that a solution could not be found. (a) shows that in all instances CCS was able to find a solution even when CS failed, while (c) shows that CCS did not always use all of its available fill-in for every solution, and in some cases had the same sparsity as CS. (d) shows the sparsity pattern of the complete matrix, while (b) plots the normalised weight of the link which was removed.

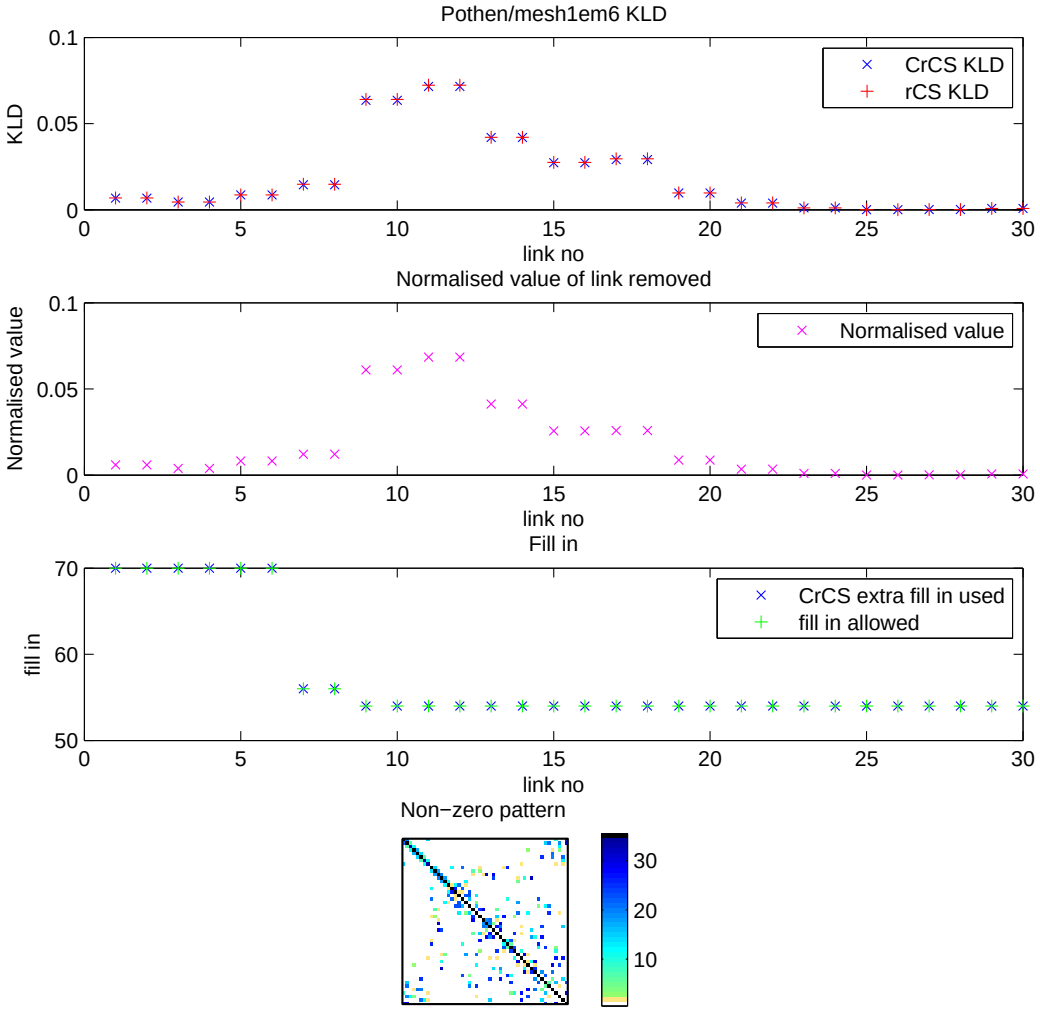


Figure 3.7 – For this matrix every single non locally chordal link removal could be solved by both CCS and CS. CCS always used all of the allowed edges. Observe that each run of CCS has added up to an extra 70 edges into the problem. This will quickly destroy any gains made through sparsification.

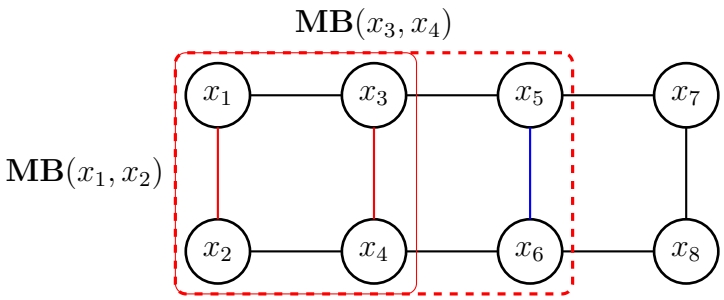


Figure 3.8 – This figure demonstrates sparsification with multiple links when the Markov blankets are intersecting. The dashed rectangle illustrates $\mathbf{MB}(x_3, x_4)$, the full stroke rectangle outlines $\mathbf{MB}(x_1, x_2)$. A blue line indicates an edge that is required if CCS is used.

3.2.4 Sparsification and Semi-Definite Matrices

There are some cases for which conservative sparsification cannot occur. Consider the matrix below, which is positive semi-definite, but has all non-zero diagonals:

$$\mathbf{Y} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^T & \mathbf{B}\mathbf{A}^{-1}\mathbf{B}^T \end{bmatrix} \succeq \mathbf{0} \tag{3.34}$$

with $\mathbf{A} \succ \mathbf{0}$.

Theorem 3.3. *There does not exist a matrix* $\tilde{\mathbf{A}} \neq \mathbf{A}$ *for which* $\tilde{\mathbf{Y}}$ *will be a conservative approximation of* $\mathbf{Y}$.

Proof. Proceed with a proof by contradiction. Assume that there does exist an $\tilde{\mathbf{A}}$ for which $\tilde{\mathbf{Y}}$ is a conservative approximation. This implies that:

$$\mathbf{A} \succeq \tilde{\mathbf{A}} \tag{3.35}$$

$$\mathbf{A}^{-1} \preceq \tilde{\mathbf{A}}^{-1} \tag{3.36}$$

$$\mathbf{B}\mathbf{A}^{-1}\mathbf{B}^T \preceq \mathbf{B}(\tilde{\mathbf{A}}^{-1})\mathbf{B}^T \tag{3.37}$$

However if $\tilde{\mathbf{Y}}$ is a conservative approximation, then $\tilde{\mathbf{Y}} \succeq \mathbf{0}$. Consider the Schur factor $\tilde{\mathbf{Y}}_{22}^{*11}$:

$$\tilde{\mathbf{Y}}_{22}^{*11} = \mathbf{B}\mathbf{A}^{-1}\mathbf{B}^T - \mathbf{B}\tilde{\mathbf{A}}^{-1}\mathbf{B}^T \succeq \mathbf{0} \tag{3.38}$$

$$\mathbf{B}\mathbf{A}^{-1}\mathbf{B}^T \succeq \mathbf{B}\tilde{\mathbf{A}}^{-1}\mathbf{B}^T, \tag{3.39}$$

which is a contradiction (using the antisymmetric property of the Loewner partial ordering and since $\tilde{\mathbf{A}} \neq \mathbf{A}$). □

Theorem 3.3 poses an interesting problem for data fusion problems, as information added to a system when performing an augment has the same structure as the positive semi-definite matrix $\mathbf{Y}$ in Equation 3.34 (see Section 2.2.2). Fortunately this does

not prevent sparsification, provided that the matrix being augmented was previously positive-definite, then the resulting matrix will remain positive definite. This is shown by the following theorem:

Theorem 3.4. *Let* $\mathbf{C} \succ \mathbf{0}$ *and*

$$\mathbf{Y} = \begin{bmatrix} \mathbf{C} + \mathbf{A} & \mathbf{B} \\ \mathbf{B}^T & \mathbf{B}\mathbf{A}^{-1}\mathbf{B}^T \end{bmatrix}$$

which implies $\mathbf{Y} \succ \mathbf{0}$.

Proof. To show $\mathbf{Y} \succ \mathbf{0}$ requires $\mathbf{C} + \mathbf{A} \succ \mathbf{0}$ and $\mathbf{C} + \mathbf{A} - \mathbf{B}(\mathbf{B}\mathbf{A}^{-1}\mathbf{B}^T)^{-1}\mathbf{B}^T \succ \mathbf{0}$ (using Equation A.8). Begin with,

$$\mathbf{C} + \mathbf{A} \succ \mathbf{0} \tag{3.40}$$

since $\mathbf{C} \succ \mathbf{0}$ and $\mathbf{A} \succ \mathbf{0}$ (using Equation A.2). Next,

$$\mathbf{C} + \mathbf{A} - \mathbf{B}^T(\mathbf{B}\mathbf{A}^{-1}\mathbf{B}^T)\mathbf{B} \succeq \mathbf{C} \succ \mathbf{0} \tag{3.41}$$

where the above has used (Equation A.4). Therefore the matrix $\mathbf{Y}$ is positive definite. □

3.2.5 CS for Disjoint Markov Blankets

A natural extension of Theorem 3.2 is to the case where the edges to be sparsified can be separated into two disjoint Markov blankets. Let **MB** be an operator that returns the Markov blanket around a set of edges. Consider the case where the following conditions are true (see Figure 3.9 for an example):

$$\begin{aligned} \mathbf{MB}(E_{app}) &= \mathbf{MB}(E_{mb1}) \cup \mathbf{MB}(E_{mb2}) \\ \emptyset &= \mathbf{MB}(E_{mb1}) \cap \mathbf{MB}(E_{mb2}) \end{aligned}$$

There are 3 possible approaches. Combining the Markov blankets and solving jointly, solve each Markov blanket sequentially or solve them synchronously. Let the partition of the variables $\mathbf{Y}_{tr}$ and $\mathbf{Y}_{app}$ as be redefined as:

$$\mathbf{Y}_{tr} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \mathbf{0} \\ \mathbf{A}_{12}^T & \mathbf{A}_{22} & \mathbf{A}_{23} \\ \mathbf{0} & \mathbf{A}_{23}^T & \mathbf{A}_{33} \end{bmatrix}, \mathbf{Y}_{app} = \begin{bmatrix} \mathbf{B}_{11} & \mathbf{A}_{12} & \mathbf{0} \\ \mathbf{A}_{12}^T & \mathbf{A}_{22} & \mathbf{A}_{23} \\ \mathbf{0} & \mathbf{A}_{23}^T & \mathbf{B}_{33} \end{bmatrix}.$$

where $\mathbf{MB}(E_{mb1}) \subset \mathbf{B}_{11}$ and $\mathbf{MB}(E_{mb2}) \subset \mathbf{B}_{33}$, and Theorem 3.1 allows the substitution $\mathbf{B}_{22} = \mathbf{A}_{22}, \mathbf{B}_{12} = \mathbf{A}_{12}$ and $\mathbf{B}_{23} = \mathbf{A}_{23}$. The CS cost function can be decomposed

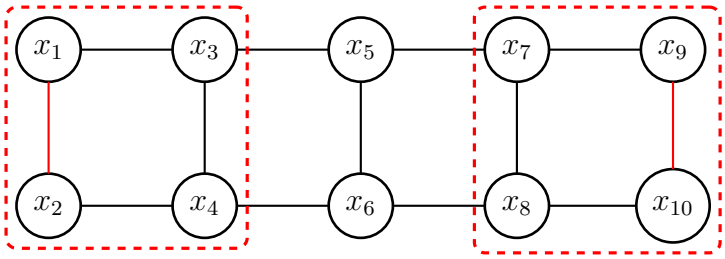


Figure 3.9 – An example where two disjoint markov blankets exist around the sparsity to be induced. Red links are to be removed, while the markov blankets are indicated by dashed red rectangles.

as follows:

$$\text{tr}(\mathbf{Y}_{tr}^{-1}\mathbf{Y}_{app}) - \text{logdet}(\mathbf{B}_{11}) - \text{logdet}(\mathbf{B}_{33}) - \text{logdet}(\mathbf{A}_{22} - \mathbf{A}_{12}^T\mathbf{B}_{11}^{-1}\mathbf{A}_{12} - \mathbf{A}_{23}^T\mathbf{B}_{33}^{-1}\mathbf{A}_{23}). \tag{3.42}$$

Using the rule $\det(\mathbf{A}) > \det(\mathbf{A}-\mathbf{B})$ for $\mathbf{A}-\mathbf{B} \succ \mathbf{0}$, $\mathbf{B} \succ \mathbf{0}$ (Equation A.3 from [51]), an upper bound on the cost function can be expressed as

$$\text{tr}(\mathbf{Y}_{tr}^{-1}\mathbf{Y}_{app}) - \text{logdet}(\mathbf{B}_{11}) - \text{logdet}(\mathbf{B}_{33}) - \text{logdet}(\mathbf{A}_{22}), \tag{3.43}$$

which is equivalent to:

$$\text{tr}((\mathbf{Y}_{tr}^{-1})_{11}\mathbf{B}_{11}) - \text{logdet}(\mathbf{B}_{11}) + \text{tr}((\mathbf{Y}_{tr}^{-1})_{33}\mathbf{B}_{33}) - \text{logdet}(\mathbf{B}_{33}) + C, \tag{3.44}$$

where the notation $(\mathbf{Y}_{tr}^{-1})_{11}$ represents the upper left block of the matrix $\mathbf{Y}_{tr}^{-1}$. Thus the cost function has been decomposed into terms relating to only $\mathbf{B}_{11}$ and $\mathbf{B}_{33}$.

Unfortunately the problem cannot be completely decoupled as the implicit constraint $\mathbf{Y}_{app} \succ \mathbf{0}$ requires that[4]:

$$\mathbf{A}_{22} - \mathbf{A}_{12}^T \mathbf{B}_{11}^{-1} \mathbf{A}_{12} - \mathbf{A}_{23}^T \mathbf{B}_{33}^{-1} \mathbf{A}_{32} \succ \mathbf{0}. \tag{3.45}$$

This section considers two out of three approaches to enabling large disjoint markov blanket sparsification. These are forming a joint and sparsifying separate problems sequentially.

A simple approach is to form the union of markov blankets. This approach will achieve optimal KLD for a CS problem, but it does not take advantage of the inherent separation in the problem, and so has cost $\mathcal{O}((k_1 + k_2)^{2.5}(r_1 + r_2)^2)$, where k_1 and k_2 are the size of the Markov blankets for $\mathbf{A}_{11}$ and $\mathbf{A}_{33}$, respectively, and r_1 and r_2 are the number of non-zeros in the respective Markov blankets. The cost function for this approach is:

$$\begin{aligned} \min_{\mathbf{B}_{11}, \mathbf{B}_{33}} \; & \operatorname{tr}((\mathbf{Y}_{tr}^{-1})_{11} \mathbf{B}_{11}) + \operatorname{tr}((\mathbf{Y}_{tr}^{-1})_{33} \mathbf{B}_{33}) \\ & - \operatorname{logdet} \left(\begin{bmatrix} \mathbf{B}_{11} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_{33} \end{bmatrix} - \begin{bmatrix} \mathbf{A}_{12} \\ \mathbf{A}_{32} \end{bmatrix} \mathbf{A}_{22}^{-1} \begin{bmatrix} \mathbf{A}_{12}^T & \mathbf{A}_{32}^T \end{bmatrix} \right) \end{aligned} \tag{3.46}$$

The complexity of this approach does not grow with an increase in the seperation between $\mathbf{A}_{11}$ and $\mathbf{A}_{22}$. However in distributed estimation settings it will require transmitting the information to a single node for processing. This is because the complete state and marginal need to be collected at a data fusion centre, and then the result transmitted back to the nodes.

An alternative approach is to approximate the solution by considering two separate CS problems. One of the Markov blankets is then chosen for sparsification and the resulting marginal is sent to the nodes of the second Markov blanket, which can then

[4]Note that section 4.2 will use a different approach to ensure that $\mathbf{Y}_{app} \succ \mathbf{0}$.

use this to solve its problem. The cost for the first is (note that it does not depend upon $\mathbf{B}_{33}$):

$$\min_{\mathbf{B}_{11}} \text{tr}((\mathbf{Y}_{tr}^{-1})_{11}\mathbf{B}_{11}) - \text{logdet}(\mathbf{B}_{11} - \mathbf{A}_{12}(\mathbf{A}_{22} - \mathbf{A}_{23}\mathbf{A}_{33}^{-1}\mathbf{A}_{23}^{T})^{-1}\mathbf{A}_{12}^{T}). \tag{3.47}$$

The cost for the second is,

$$\min_{\mathbf{B}_{33}} \text{tr}((\tilde{\mathbf{Y}}_{tr}^{-1})_{33}\mathbf{B}_{33}) - \text{logdet}(\mathbf{B}_{33} - \mathbf{A}_{32}(\mathbf{A}_{22} - \mathbf{A}_{21}\mathbf{B}_{11}^{-1}\mathbf{A}_{21}^{T})^{-1}\mathbf{A}_{32}^{T}), \tag{3.48}$$

where $\tilde{\mathbf{Y}_{tr}}$ is the resultant system after the first sparsification. The resultant KLD will be an upper bound on the KLD derived from optimising the joint, but will have the lower computation cost of $\mathcal{O}((\max(k_1, k_2))^{2.5}(\max(r_1, r_2))^2)$. The communication required between the two Markov blankets is only the marginal after sparsification, and so the problem could be solved in a local fashion.

The third approach is to iteratively perform sequential sparsification. Unfortunately this approach is not guaranteed to converge on the optimal. This approach is better posed in a Junction Tree formulation, and thus is described in more detail in Section 4.1.4.

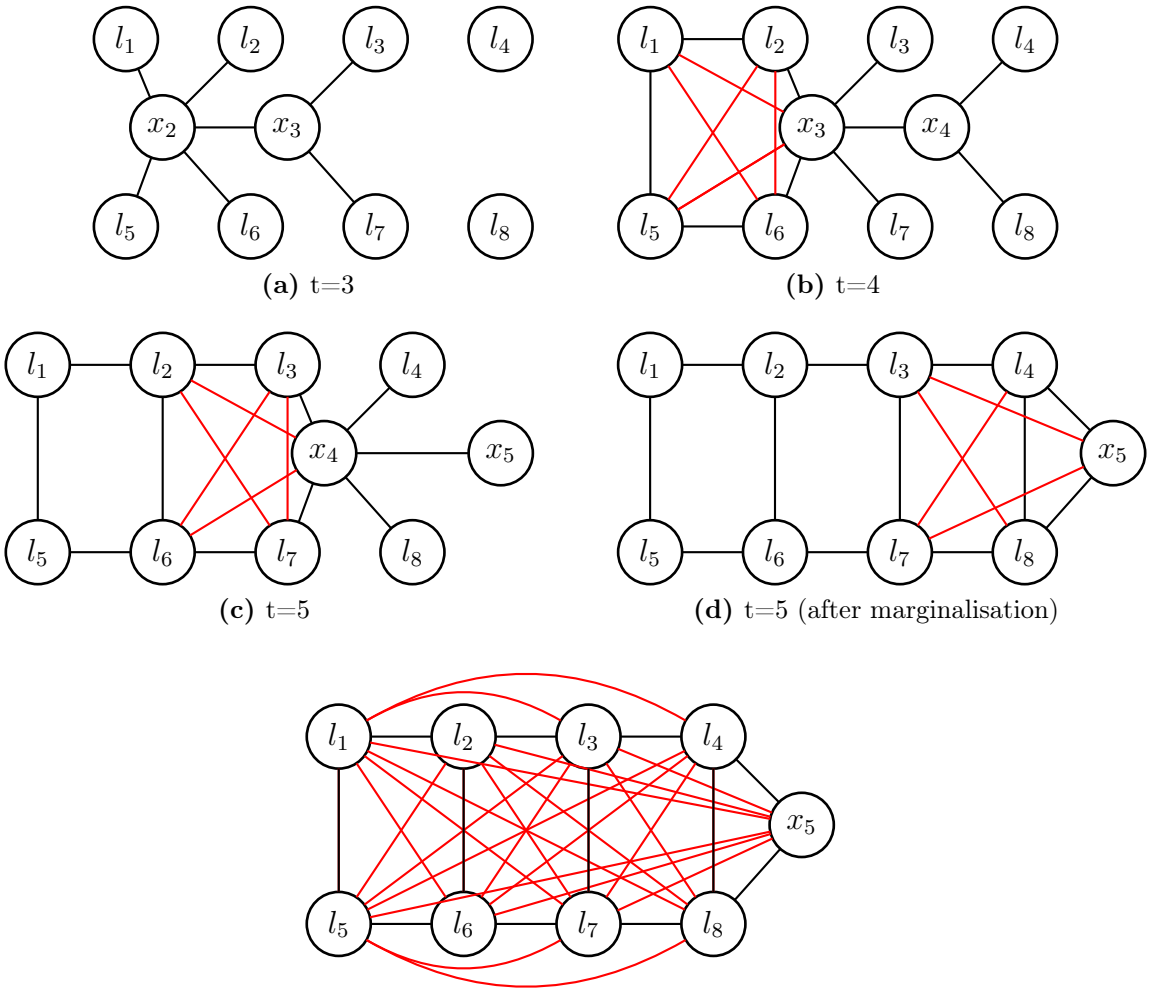
3.3 Experiments

The techniques in this chapter are assessed on a simple SLAM experiment. Section 3.3.1 describes the application CS to an incremental estimation problem. Section 3.3.2 illustrates and discusses the results. A final experiment is performed on matrices from the Suite Sparse matrix library [15] in Section 3.3.4.

3.3.1 Conservative Sparsification for Incremental Estimation

This thesis considers incremental scenarios where the state estimate grows over time and complete information is not available when sparsification is performed. This is

an approximation to a global solution where all information about states is used to sparsify. Figure 3.10 shows two approaches to sparsification for a filtering SLAM problem. In Figure 3.10(a-d) sparsification is performed after each state is marginalised, resulting in three problems of similar complexity. This can be seen by noting that the Markov blanket for Figure 3.10(b) does not include the states l_4 and l_8 and similarly Figure 3.10(d) does not include the states l_1 and l_5. Figure 3.10(e) shows a sparsification problem that is delayed until t=5. The marginalisation of states x_2, x_3 and x_4 results in a dense graph, and a single but large sparsification problem must be solved. The estimate resulting from the sparsification in Figure 3.10(e) will be a closer approximation to the true distribution, however the smaller sparsification steps in Figure 3.10(a-d) allow the system to maintain its sparse property, and has a lower computational cost.



(a) t=3 **(b)** t=4

(c) t=5 **(d)** t=5 (after marginalisation)

(e) t=5 delayed sparsification

Figure 3.10 – Difference between incremental and delayed sparsification. (a-d) show the progression of a filtering SLAM problem. Red links indicate edges that are chosen for sparsification. In (a-d) sparsification is performed after marginalising each robot state (x_t), while (e) shows the connectivity if this sparsification is delayed until t=5. In order to maintain the same sparsity of (d), a much larger problem needs to be solved in (e). However, the solution found by solving (e) will be a closer approximation to the true distribution than (d).

3.3.2 Linear SLAM

This section will demonstrate the conservative sparsification algorithm compared with the SEIF on a simulated SLAM dataset. The slam simulation data was generated using MATLAB. The world is a 50×50 unit 2-D environment, with 100 landmarks randomly distributed throughout. The robot begins at a random location and then traverses the simulated environment. The robot attempts to navigate within 3 world units of several way points in the map. These way points were chosen such that the robot would pass within range of each landmark in the simulation. The robots sensor range is 5, therefore it cannot see all the landmarks but visits every one during the mission. The mission goes for 500 time steps, at each time the robot moves in the direction of its next way point. The ground truth for the simulation is shown in Figure 3.11. The simulation was run with an observation model $\mathbf{R}$ and prediction model $\mathbf{Q}$ shown below:

$$\mathbf{R} = \begin{bmatrix} 1.3803 & 0.1730 \\ 0,1730 & 1.1108 \end{bmatrix}, \mathbf{Q} = \begin{bmatrix} 2.9242 & 0.2248 \\ 0.2248 & 2.1487 \end{bmatrix} \tag{3.49}$$

SLAM is performed using the methods as described in Section 2.4.1. State updates and augments are performed using the theory introduced in Section 2.2.2, while marginalisation is performed using the equations described in Section 2.2.3.

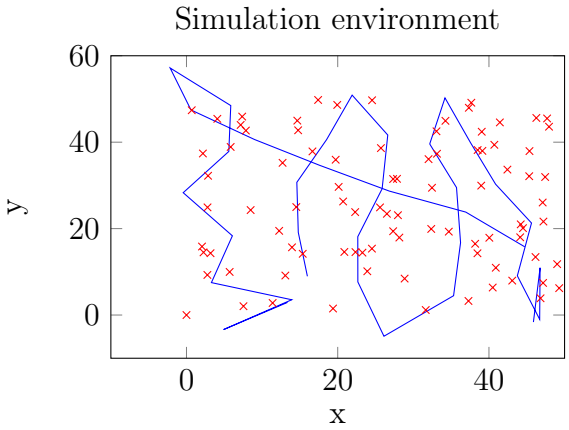


Figure 3.11 – Ground truth of simulated data. Red crosses indicate landmark positions. The blue line is the robots path.

The SLAM simulation is used as ground truth for the filtering and smoothing variants of the SLAM problem. In the following simulations, conservatism is assessed by plotting the minimum eigenvalue of the conservatism constraint, that is:

$$\min(\text{eig}(\mathbf{Y}_{tr} - \mathbf{Y}_{app})). \tag{3.50}$$

If this expression is less than zero, then the following constraint is violated:

$$\mathbf{Y}_{tr} - \mathbf{Y}_{app} \succeq \mathbf{0}. \tag{3.51}$$

The timing results reported below were performed using a quad core i7 laptop computer with 16 GB of RAM.

3.3.2.1 Filtering SLAM

The first approach is AM-SEIF, that is described in Section B.1.2. AM-SEIF is an approach which can induce sparsity but does not guarantee conservatism. AM-SEIF chooses links to remove by finding the link connected to the robot state with the smallest normalised link weight. All other approaches in this experiment attempt to maintain the same sparsity pattern as AM-SEIF. The normalised link weight was calculated using the following equation:

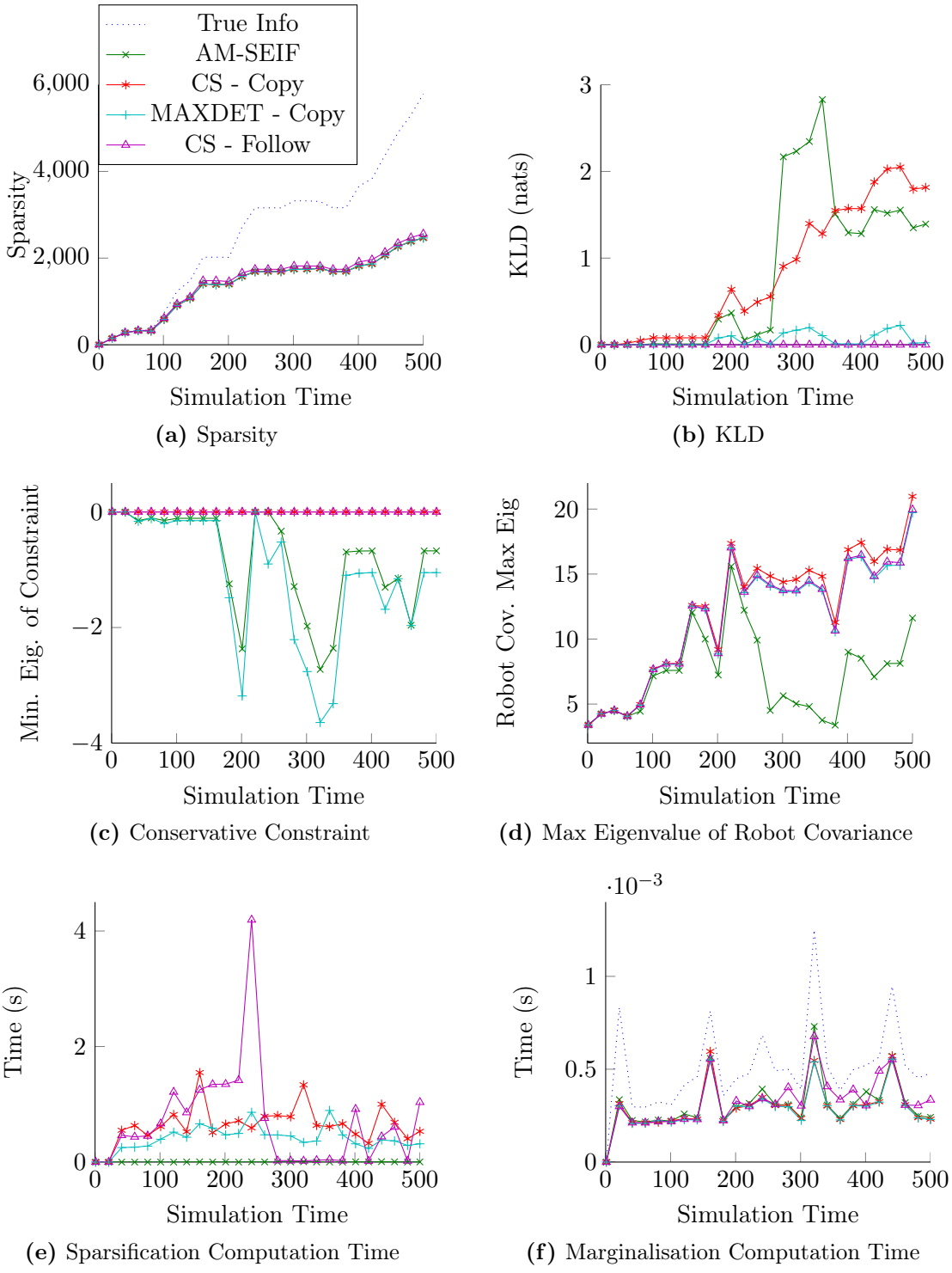
$$w = \frac{(\mathbf{Y})_{ij}^2}{(\mathbf{Y})_{ii}(\mathbf{Y})_{jj}}. \tag{3.52}$$

A new approach is introduced in this section, MAXDET - COPY, this is the same as CS but the conservative constraint $\mathbf{Y}_{tr} - \mathbf{Y}_{app} \succeq \mathbf{0}$ has been removed. MAXDET - COPY is equivalent to a Covariance Selection problem with know sparsity pattern (see Appendix B.3.1.1). It is expected that MAXDET - COPY will exhibit the best case KLD for a given sparsity pattern as it will not be constrained to be conservative. The approach CS - Copy performs Conservative Sparsification while exactly replicating a target sparsity, (which in the case of Figure 3.12 is the sparsity pattern of AM-SEIF).

The approach CS-Follow attempts to meet the same overall sparsity level but may choose any link for removal. CS-Follow uses the AM-SEIF heuristic to choose the link for removal. The final approach, CS - Threshold, chooses links for removal when their normalised link weight rises above a certain threshold (see Equation 3.52) and then applies Conservative Sparsification to the result.

The filtering solution in Figure 3.12(a) shows all approaches maintaining the same sparsity level. Observe that after time step 100 the sparsity is significantly lower than the ground-truth (shown by the True Info plot). Figure 3.12(b) shows the KLD for the approaches. It can be seen that both CS - Copy and AM-SEIF perform the worst. The poor performance of AM-SEIF as compared to MAXDET - Copy suggests that AM-SEIF may not in-fact minimise the KLD as suggested in the work [54]. The excellent performance of CS - Follow compared to CS - Copy and MAXDET - Copy suggests that the heuristic of precisely following the sparsity pattern of another approach is poor. This performance is due to CS - Follow being able to choose weaker strength links than CS - Copy or MAXDET - Copy. Figure 3.12(c) shows that the conservative approaches do indeed maintain conservativeness, while the alternate approximate methods have become inconsistent before time step 100. The maximum eigenvalue of the robot covariance is plotted in Figure 3.12(d). This illustrates the error of the robots location estimate directly, while the KLD expresses this for the entire system. Note that the plots for MAXDET, True Info and CS-Follow are overlapping. Also observe that AM-SEIF is much lower than the true value, indicating its inconsistency.

Figure 3.12(e) and (f) show the time taken to perform the sparsification and marginalisation of states. These plots show that the reduced sparsity does reduce the time taken to marginalise, and to a tiny extent fusing of observations. Its important to note however the significant difference in the scales between plots (e) and (f). As the time cost of solving the sparsification problem is not recouped in the savings in marginalisation time.



(a) Sparsity **(b)** KLD

(c) Conservative Constraint **(d)** Max Eigenvalue of Robot Covariance

(e) Sparsification Computation Time **(f)** Marginalisation Computation Time

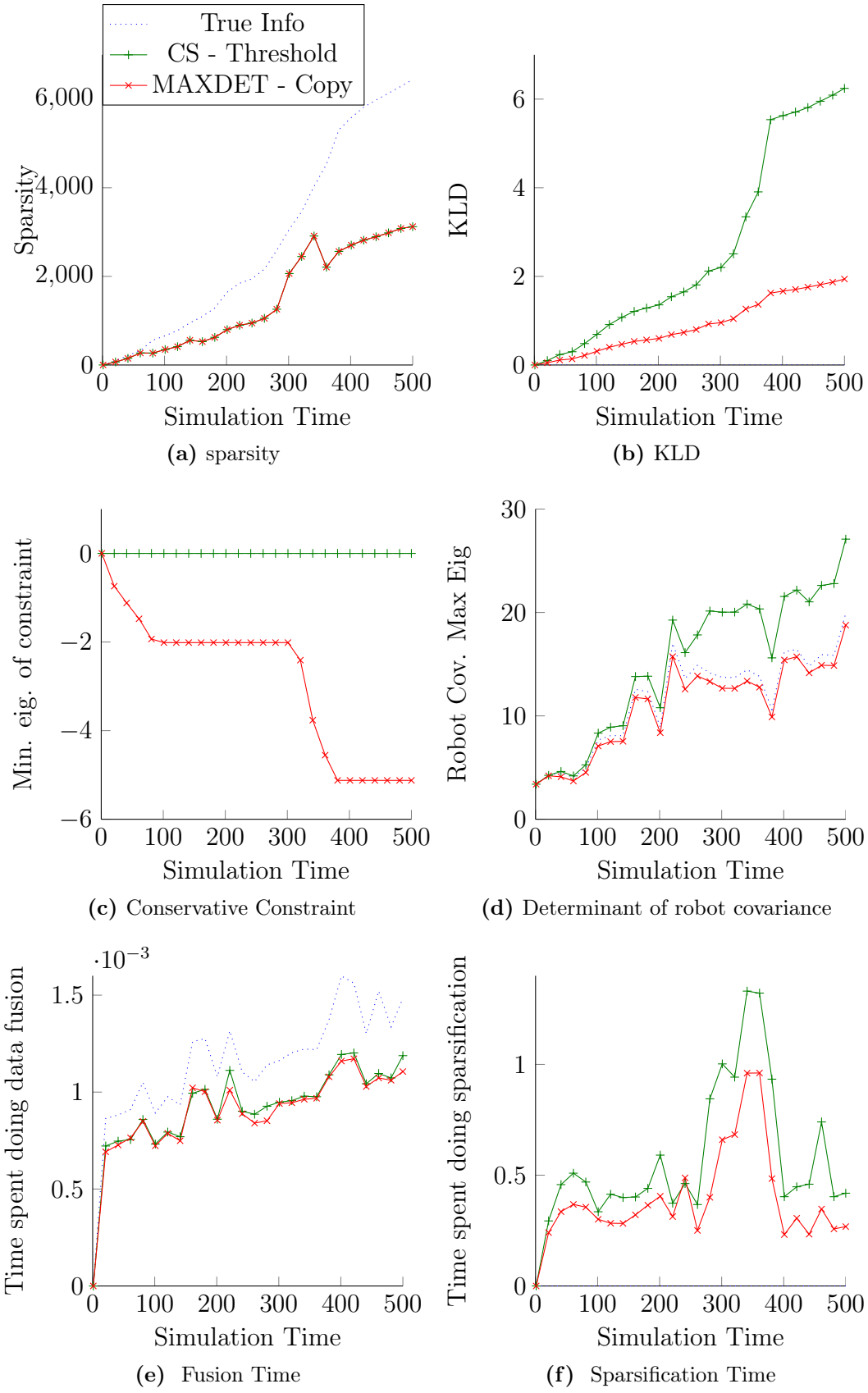
Figure 3.12 – Filtering SLAM example. Several approaches to link removal are shown. AM-SEIF chooses the sparsification and CS - Copy and KLD - Copy will copy that sparsity pattern. CS - Follow attempts to maintain a similar sparsity level but may choose different edges to remove.

3.3.3 Smoothing SLAM example

The smoothing SLAM problem is explored in Figure 3.13. In this example MAXDET - COPY is used to mimic the sparsity pattern determined by CS - Threshold. AM-SEIF was not used in this experiment because AM-SEIF is not suited to smoothing scenarios as it can induce more fill-in than it removes [59]. Figure 3.13(a) shows these approaches maintaining equivalent sparsity patterns, while Figure 3.13(c) shows that MAXDET - COPY once again becomes overconfident. Figure 3.13(b) shows the KLD, demonstrating that MAXDET - COPY is capable of outperforming CS in terms of KLD because it is not constrained to be conservative.

Figure 3.13(d) shows the maximum eigenvalue of the robot covariance, observe that the MAXDET - COPY has a slightly smaller maximum eigenvalue demonstrating its inconsistency. Also the eigenvalue is monotonically increasing as in Figure 3.12(d).

Figure 3.13(e) shows the time taken to do data fusion for the smoothing problem. It shows that the approximate approaches take slightly less time than the full solution. Figure 3.13(f) shows the time spent on marginalisation at each time step. Since the smoothing problem is naturally much more sparse than the filtering problem, the time taken to perform the required sparsification is much lower.



(a) sparsity **(b)** KLD

(c) Conservative Constraint **(d)** Determinant of robot covariance

(e) Fusion Time **(f)** Sparsification Time

Figure 3.13 – Smoothing SLAM example. Several approaches to link removal are shown, CS - Threshold chooses which links to remove and MAXDET - Copy attempts to replicate the same sparsity pattern.

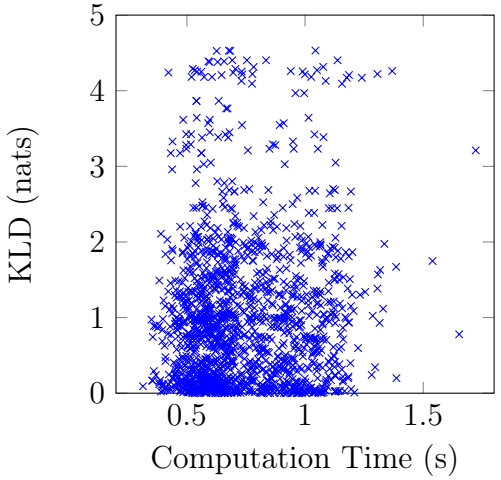
3.3.4 Benchmark matrices

Conservative Sparsification was applied to a set of matrices from the University of Florida Sparse Matrix Library [15]. Each off-diagonal was sparsified separately (i.e., starting from the original matrix), the results are ordered with respect to their normalised link weight and vertex cardinality.

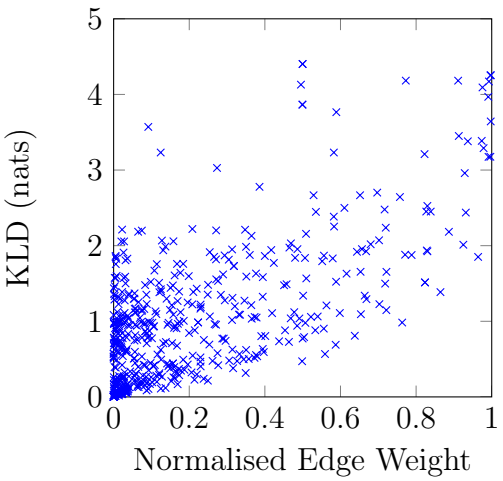
Each sparsification attempt was timed using MATLAB, and each sparsification was performed using the SDPT3 solver [57]. Thus, the timing reflects the complexity of each problem. The time for each independent sparsification was compared against the resultant KLD and the Markov blanket size.

Figure 3.14 shows the results for the 1138 Bus matrix from the HB set. This benchmark was done on other matrices, which are in Appendix C.1. The results show a loose correlation between normalised link weight and the KLD of the final result. However, this relation is clearly very weak.

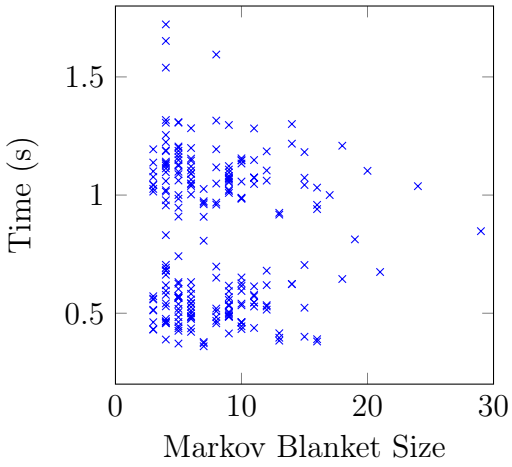
The timing results in Figure 3.14 (a) show little correlation between computation time and KLD. Similarly in Figure 3.14 (c) the Markov blanket size does not appear to effect the time taken to solve each step. This is may be because larger Markov blankets were not explored. However, in Figure 3.15(c) does show an increase in time as the Markov blanket size increases.



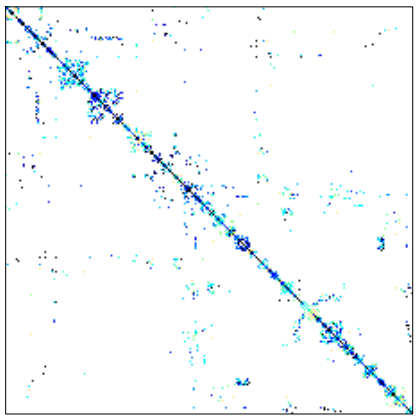
(a) KLD as Computation Time Increases



(b) KLD as a function of normalised edge weight

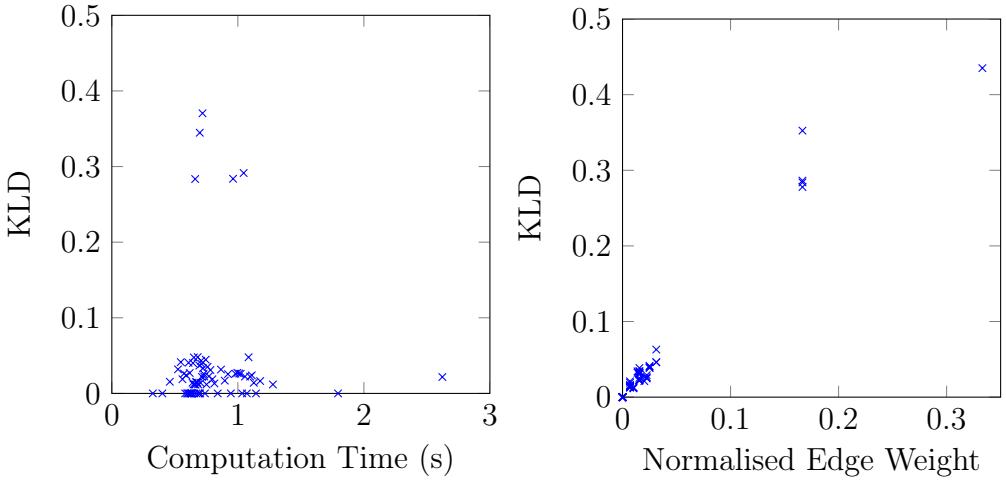


(c) Time required to solve as Markov blanket size increases

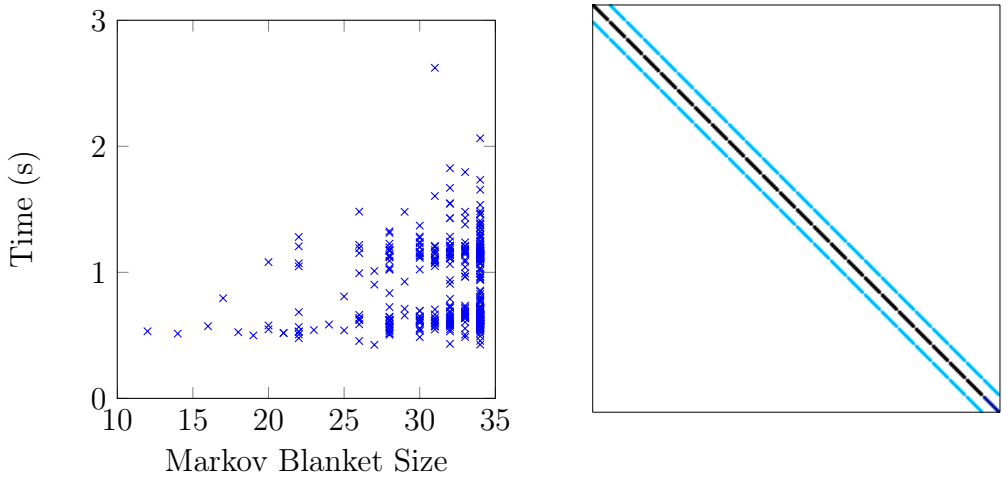


(d) cspy Plot

Figure 3.14 – HB 1138 bus Matrix sparsification result. (a) shows the KLD as the computation time increased. (b) shows the KLD for edge weights, the graph shows an upwards trend, that is, the KLD get higher as the normalised weight grows, but there are many cases where this was not significant. (c) shows the time taken to find a solution as a function of the Markov blanket size for the edge.



(a) KLD as Computation Time Increases **(b)** KLD as a function of normalised edge weight



(c) Time required to solve as Markov Blanket size increases

(d) CSPY Plot

Figure 3.15 – HB nos3 Matrix sparsification result. The solve time increases as the Markov blanket size increases.

Chapter 4

Conservative Sparsification for Junction Trees

A key assumption made by the reduction of Conservative Sparsification (CS) in Section 3.2.1 is that the marginal of nodes not in the Markov blanket can be computed in a local manner. Junction Trees (JTs) were introduced in Section 2.3 as an alternative approach to estimation that also caches the marginals at every clique. This cache can be used by CS. This chapter begins by describing how CS can be posed for JTs, and examines the properties of this new approach. This new CS approach is then extended in Section 4.2 to allow for asynchronous parallel sparsification. This approach is suited to distributed estimation schemes, but usually results in a higher Kullback Leibler Divergence (KLD) than the optimal approaches. The chapter concludes with a simple simulation.

4.1 Conservative Sparsification in Junction Trees

This section follows many of the key points in Chapter 3, as such it begins with a quick formulation of the CS problem in JTs by clique splitting rather then edge sparsification. The reduced CS problem is than considered and it is observed that the clique split problem is always locally chordal and thus always feasible. The Markov

blanket of a JT split is considered and is shown to be often larger than the equivalent Markov blanket in the information form based CS problem. Finally the cost of the algorithm is described.

4.1.1 Conservative Sparsification

To sparsify a JT it is necessary to split cliques rather then removing off-diagonals from an information matrix. Consider a JT $\mathcal{T}^{tr}$ that is to be approximated by $\mathcal{T}^{app}$ (see Figure 4.1). The KLD between this can be determined by forming the joint using each tree's clique potentials,

$$\mathrm{D_{KL}}\left(\mathcal{T}^{tr}||\mathcal{T}^{app}\right) \equiv \mathrm{D_{KL}}\left(\prod \Psi^{tr}||\prod \Psi^{app}\right). \tag{4.1}$$

Let $\mathbf{Y}_{\Psi_i^{app}}$ and $\mathbf{Y}_{\Psi_i^{tr}}$, be the information matrix associated with Ψ_i^{app} and Ψ_i^{tr} respectively, then assuming that the mean estimates of the JTs are equal[1], Equation 4.2 can be expressed using only the clique potential matrices:

$$\mathrm{D_{KL}}\left(\mathcal{T}^{tr}||\mathcal{T}^{app}\right) = \mathrm{tr}\left(\left(\sum \mathbf{Y}_{\Psi_i^{tr}}\right)\left(\sum \mathbf{Y}_{\Psi_i^{app}}\right)^{-1}\right) - \mathrm{logdet}\left(\sum \mathbf{Y}_{\Psi_i^{app}}\right). \tag{4.2}$$

The CS problem is posed in Junction Tree notation:

$$\min_{\mathbf{Y}^{app}} \mathrm{tr}\left(\left(\sum \mathbf{Y}_{\Psi_i^{app}}\right)\left(\sum \mathbf{Y}_{\Psi_i^{tr}}\right)\left(\sum \mathbf{Y}_{\Psi_i^{app}}\right)^{-1}\right) - \mathrm{logdet}\left(\sum \mathbf{Y}_{\Psi_i^{app}}\right) \tag{4.3}$$

$s.t.$

$$\sum \mathbf{Y}_{\Psi_i^{tr}} \succeq \sum \mathbf{Y}_{\Psi_i^{app}},$$

$$\sum \mathbf{Y}_{\Psi_i^{app}} = \mathbf{Y}^{app}_{\Psi_{s1}} + \mathbf{Y}^{app}_{\Psi_{s2}} + \sum_{i\neq s}^{app} \mathbf{Y}_{\Psi_i},$$

[1]This assumption was also made in Section 3.1.2, it requires the marginals to be known so the mean can be calculated before sparsification. Since the marginals are required for sparsification anyway this does not add extra complexity.

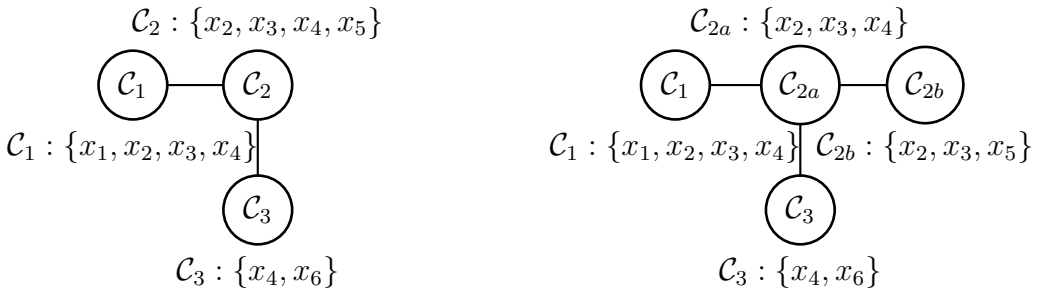
where the clique $\mathcal{C}^{tr}_{\Psi_s}$ is split into two new cliques $\mathcal{C}^{app}_{\Psi_{s1}}$ and $\mathcal{C}^{app}_{\Psi_{s2}}$ these cliques are constrained by the following set relations:

$$\mathcal{C}^{app}_{\Psi_{s1}} \cup \mathcal{C}^{app}_{\Psi_{s2}} = \mathcal{C}^{tr}_{s}, \tag{4.4}$$

$$\mathcal{C}^{app}_{\Psi_{s1}} \subset \mathcal{C}^{tr}_{\Psi_s}, \tag{4.5}$$

$$\mathcal{C}^{app}_{\Psi_{s2}} \subset \mathcal{C}^{tr}_{\Psi_s}. \tag{4.6}$$

That is, the union of the new clique states must be equal the original clique states, and the states of each new clique must be a proper subset of the original clique states. Comparing Equation 4.3 to Equation 3.7 from Section 3.1.2, the equality constraints have been replaced by adding new cliques and imposing membership constraints, thus performing CS on the clique potential matrices is equivalent to clique splitting. Figure 4.1 demonstrates clique splitting on a simple JT. Note that applying CS for clique splitting is an approximation, such that the product of the new clique potentials will not be equal to the original clique potential.



(a) Original JT **(b)** Sparse JT

Figure 4.1 – Example of clique splitting operation. (a) shows a JT before the split. $\mathcal{C}_2$ is chosen for splitting. (b) $\mathcal{C}_2$ has been split into two new cliques $\mathcal{C}_{2a}$ and $\mathcal{C}_{2b}$. The new cliques are smaller than the original clique, therefore this system will be more efficient. Section 2.5.3 discusses improving the efficiency of JTs.

4.1.2 Efficient Clique Splitting

The reduced CS approximation employed in Section 3.2.1 can also be used for sparsification of JTs. Since the reduction of CS employed in Section 3.2 requires the concept

of a Markov blanket, this concept also needs to be defined for JTs.

Definition 4.1. *Any clique $\mathcal{C}_i$ of a JT has a Markov blanket* $\mathbf{MB}(\mathcal{C}_i)$ *that is the set of all cliques that contain any of the states within that clique.* $\mathbf{MB}(\mathcal{C}_i) = \{\mathcal{C}_j\}\ \forall \mathcal{C}_j \cap \mathcal{C}_i \neq \emptyset$.

The following theorem compares the sizes of the Markov blanket in an information matrix and a JT (an example is given in Figure 4.2).

Theorem 4.1. *The maximum number of states in the Markov blanket of $\mathcal{C}_i$ is $2k$ where k is the largest clique size.*

Proof. There are k states in $\mathcal{C}_i$, each of these will have at most k neighbours. For each state $k-1$ neighbours already exist in the $\mathcal{C}_i$, so each state has one extra neighbour that does not already exist in $\mathcal{C}_i$. Therefore the total number of states is $2k$. □

It is possible to specify a JT split such that states within a separator set are involved. The Markov blanket for a separator split is larger than the Markov blanket for a clique split.

Theorem 4.2. *The maximum number of states involved in the Markov blanket of a separator split is $4k-4$, where k is the maximum clique size.*

Proof. To construct the Markov blanket of a separator, the two neighbouring cliques must be merged, let this new clique be $\mathcal{C}_{\Psi_s}$. The largest clique size will occur if the separator set is of size two[2], which results in $2k-4$ unique states in $\mathcal{C}_{\Psi_s}$. Each of these states has at most one neighbour not in $\mathcal{C}_{\Psi_s}$ (since the clique in question is of the maximum size k). Therefore, the total number of states is $2k-2+2k-2 = 4k-4$. □

Comparing this result to Theorem 3.2, the worst case size of the Markov blanket can be greater than the equivalent information form Markov blanket if a separator set is split, this is illustrated in Figure 4.2.

[2]A separator of size one cannot be split

There is an additional cost in forming the Markov blanket for a JT which involves gathering the potentials present in every clique containing the target states.

Theorem 4.3. *To gather the potentials over m states from a JT requires at most ml cliques, where l is the maximum subtree length.*

Proof. The worst case is that all m states are disjoint and their sub-trees do not share any common cliques. The conditional for a single state is found by gathering the l cliques that contain it, therefore ml cliques are required to gather all m states. □

Given a $\mathcal{C}_s$ that is to be split into $\mathcal{C}_{s1}$ and $\mathcal{C}_{s2}$ it is convenient to define the union of all other cliques in the Markov blanket, $\mathcal{B} = \{\mathbf{MB}(s)/\ s\}$. Associate with this the following:

$$\mathcal{C}_{\mathcal{B}} = \bigcup_i \mathcal{C}_i, \forall i \in \mathcal{B} \tag{4.7}$$

$$\mu_{\mathcal{B}} = \prod_i \prod_j \mu_{ij}, \forall (i,j) \in \mathcal{B} \tag{4.8}$$

$$\mathbf{Y}_{\mu_{\mathcal{B}}} = \sum_i \sum_j \mathbf{Y}_{\mu_{ij}}, \forall (i,j) \in \mathcal{B}, \tag{4.9}$$

$$\mathbf{y}_{\mu_{\mathcal{B}}} = \sum_i \sum_j \mathbf{y}_{\mu_{ij}}, \forall (i,j) \in \mathcal{B}, \tag{4.10}$$

where $\mathbf{Y}_{\mu_{ij}}$ is the information matrix and $\mathbf{y}_{\mu_{ij}}$ is the information vector associated with the message from $\mathcal{C}_i$ to $\mathcal{C}_j$. Now the problem can be posed using clique potential matrices:

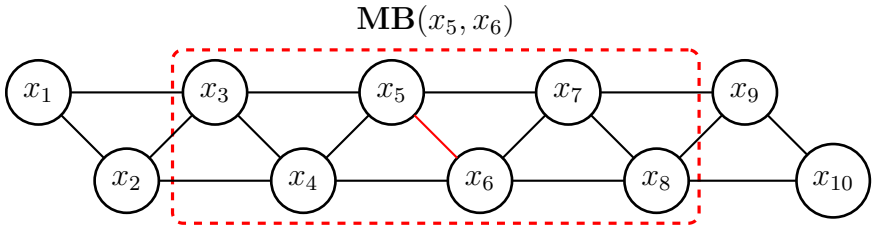
$$\min_{\mathbf{Y}_{\Psi_{s1}}, \mathbf{Y}_{\Psi_{s2}}, \mathbf{Y}^{app}_{\Psi_{\mathcal{B}}}} D_{kl}(\mathbf{Y}^{tr}_{\Psi_s} + \mathbf{Y}^{tr}_{\Psi_{\mathcal{B}}} + \mathbf{Y}^{tr}_{\mu_{\mathcal{B}}} || \mathbf{Y}_{\Psi_{s1}} + \mathbf{Y}_{\Psi_{s2}} + \mathbf{Y}^{app}_{\mathcal{B}} + \mathbf{Y}^{tr}_{\mu_{\mathcal{B}}}) \tag{4.11}$$

$$s.t.$$

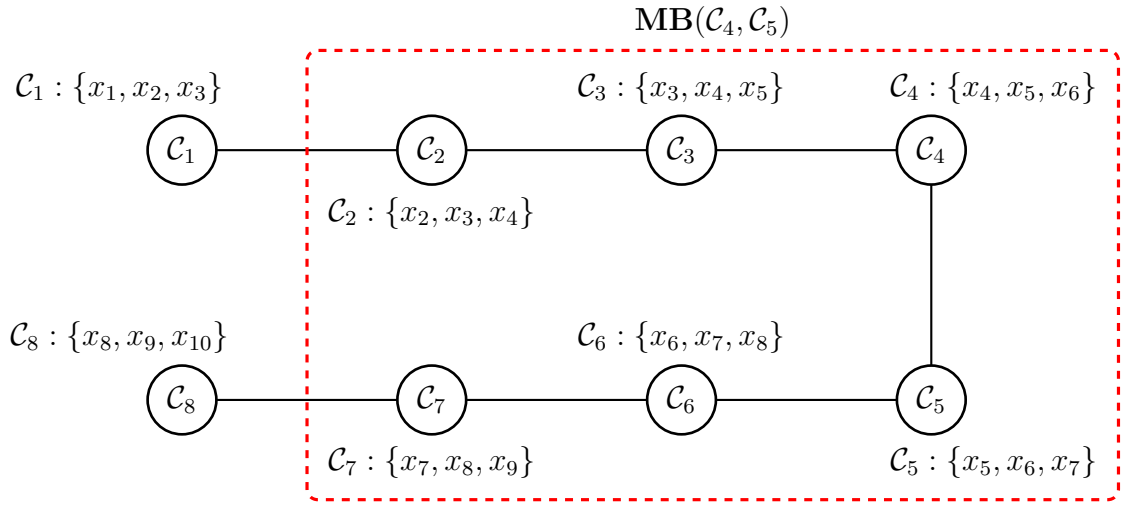
$$\mathbf{Y}_{\Psi_{s1}} + \mathbf{Y}_{\Psi_{s2}} + \mathbf{Y}^{app}_{\Psi_{\mathcal{B}}} \succ \mathbf{0}$$

$$\mathbf{Y}_{\Psi_{s1}} + \mathbf{Y}_{\Psi_{s2}} + \mathbf{Y}^{app}_{\Psi_{\mathcal{B}}} + \mathbf{Y}^{tr}_{\mu_{\mathcal{B}}} \succ \mathbf{0}$$

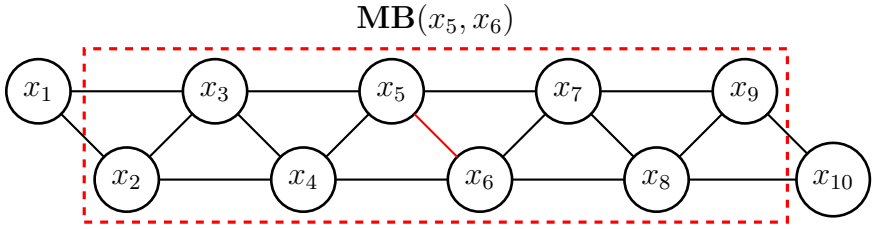
$$\mathbf{Y}_{\Psi_s} + \mathbf{Y}_{\Psi_{\mathcal{B}}} - (\mathbf{Y}_{\Psi_{s1}} + \mathbf{Y}_{\Psi_{s2}} + \mathbf{Y}^{app}_{\Psi_{\mathcal{B}}}) \succeq \mathbf{0},$$



(a) The Markov blanket around edge 5-6



(b) A JT of the graph in (a), removing the link between $\mathbf{x}_5$ and $\mathbf{x}_6$ is equivalent to splitting cliques $\mathcal{C}_4, \mathcal{C}_5$ as the states are both contained in the separator $\mathcal{S}_{45}$.



(c) The effective JT Markov blanket as seen in an information matrix

Figure 4.2 – The differences between a Markov blanket posed in a junction tree and a Markov blanket posed in an information matrix, whose sparsity pattern is represented by an undirected graphical model. The junction tree Markov blanket is larger. This is because any clique that refers to a state contained in the sparsified cliques must be gathered to perform sparsification.

where $\mathbf{Y}^{app}_{\Psi_\mathcal{B}}$ represents the summands rather than the sum of the information matrices in $\mathcal{B}$ (that is, replace $\mathbf{Y}^{app}_{\Psi_\mathcal{B}}$ with the variables $\sum_i \mathbf{Y}_{\Psi_i} \forall i \in \mathcal{B}$). Therefore, the optimisation is over all the clique potentials that are contained in the Markov blanket. Comparing this to CS for information matrices, there are several common elements. The messages from cliques outside the Markov blanket replace the marginal projection $\mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{12}^T$ (see Equation 3.9), while the clique membership constraints encode the sparsity constraints. This problem is more complex than the CS problem due to the addition of extra variables associated with optimising over the clique potentials rather than the sum of the clique potentials. These extra variables all increase the matrix dimension term in the time complexity analysis. For example, if there are m cliques in the split, each of size k, then the dimension of the optimisation matrix is $n = mk$. However, in practice the cost of the CS problem can be recovered by optimising over the sum and then performing an exact clique split (Section 4.2.2) to recover the summands (the exact split requires a feasible not optimal solution, so it can be calculated analytically).

The states inside a JT clique are densely connected, this implies a chordal structure of the underlying graph. Therefore all JT CS problems are locally chordal and therefore guaranteed to always have a feasible solution by Theorem 3.2.

4.1.3 Computational Cost

This section considers the worst case computational cost of Conservative Sparsification in a Junction Tree setting. The costs considered in this section will assume an already constructed JT. This is because JTs can be used as a fundamental representation for estimation and JTs can be built incrementally in a similar way to the information form, so the setup costs can be ignored. Costs associated with message passing and forming marginal distributions are described in Section 2.3.5.

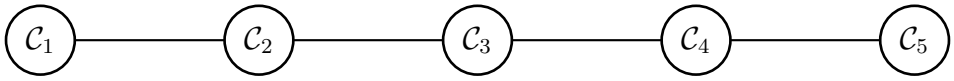
Using an interior point method for solving CS problem for dense matrices is $\mathcal{O}(n^{6.5})$, where n is the dimension of the matrix variable [64]. Let l be the length of a sub-tree

in the JT[3]. For a clique split, n is the number of states involved in the problem. For a clique split, Theorem 4.1 states that there will be $2k$ states, while Theorem 4.2 states that there will be $4k-4$ for a separator split. Taking the larger size results in a computation cost of $\mathcal{O}((4k-4)^{6.5}) = \mathcal{O}(k^{6.5})$.

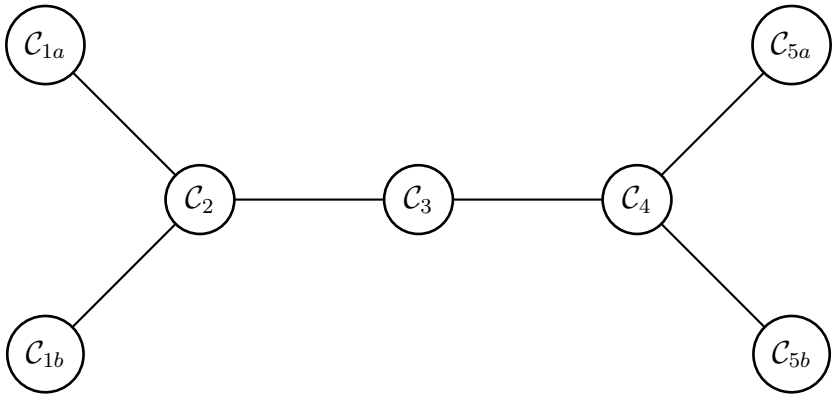
4.1.4 Multiple Cliques

A large distributed estimation system can require sparsification in two distantly related parts of the graph. This section investigates methods of splitting multiple clique sets. This problem is equivalent to the disjoint Markov blanket problem considered for the CS in Section 3.2.5.

The simplest case is a five clique chain JT, with each leaf being split, and no common states between the leaves. This structure is demonstrated in Figure 4.3.



(a) JT structure for multi-clique split examples.



(b) JT after clique splits in $\mathcal{C}_1$ and $\mathcal{C}_5$.

Figure 4.3 – In this example, cliques $\mathcal{C}_1$ and $\mathcal{C}_5$ need to be sparsified, but they do not share any common states. Unfortunately these two problems are still coupled, and so cannot be solved independently.

A simple approach is to gather all the information from cliques that belong to the

[3]A sub-tree is the set of all nodes which contain a reference to a particular state. By the running intersection property these cliques must form a connected sub-tree.

union of the Markov blanket around the cliques being split, called Joint Splitting. Let the cliques be $\mathcal{C}_i$ and $\mathcal{C}_j$, and let $\mathcal{J}_{joint} = \mathbf{MB}(i) \cup \mathbf{MB}(j)$. For Figure 4.3 these correspond to $i = 1, j = 5$ and $\mathcal{J}_{joint} = \{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_4, \mathcal{C}_5\}$ (assuming that the Markov blanket for $\mathcal{C}_1$ and $\mathcal{C}_2$ does not contain $\mathcal{C}_3$). The clique split technique described in Section 4.1.2 can be used to enforce this structure.

Joint Splitting makes no approximation in forming the CS problem, and will therefore have the optimal KLD for this problem. This is equivalent to the joint sparsification approach described in Section 3.2.5.

An alternative approach is to approximate the Joint Splitting problem by performing a clique split on each clique sequentially. That is, perform the conservative clique split $\mathcal{C}_i = \mathcal{C}_{ia} + \mathcal{C}_{ib}$ and keeping $\mathcal{C}_j$ fixed, then split $\mathcal{C}_j = \mathcal{C}_{ja} + \mathcal{C}_{jb}$ using the new values for $\mathcal{C}_{ia}, \mathcal{C}_{ib}$. This is also equivalent to the approach in Section 3.2.5, therefore this technique will provide a solution that represents an upper bound on the optimal KLD for the problem. A key benefit is that the dimension of the convex problem that is required to be solved at each step is smaller than the joint approach. This is however offset by the requirement to perform the clique splits sequentially and to transmit a message after each clique split.

An additional approach is to perform incremental sparsification iteratively until the KLD change approaches zero. That is, rather than immediately applying the result of incremental sparsification, the message associated with the final result is formed and then transmitted. The receiver also performs incremental clique splitting, caches the original result and transmits a message. This is iterated until the change in messages approaches some threshold, at which point the changes are finally applied. Unfortunately, this thesis was unable to present a satisifactory proof with regard to convergence of this technique to the optimal KLD. Finding such a proof is left for future work.

4.2 Parallel Sparsification

Junction Trees are useful because of their implicit caching of marginals. The previous section described a direct translation of the CS problem into a JT to exploit this property. Consider a decentralised estimation scenario, such as a large distributed sensor network. In this setting it may not be practical to gather information from neighbouring cliques, sparsify the result, and then transfer that information back to the original cliques. Ideally a clique split algorithm would have a computational cost that is completely local, requiring only information available at the clique level, and add no additional burden to the estimation process (i.e. by increasing the number of messages sent or using special messages). This is the motivation for the parallel sparsification approach, which allows for decentralised sparsification and local computational cost at the expense of higher KLD than the optimal CS KLD.

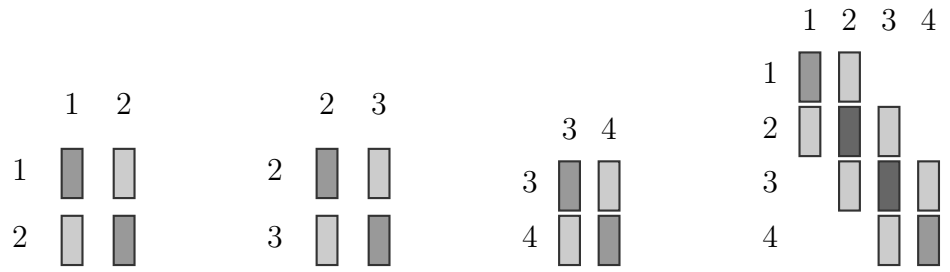
4.2.1 Key Assumption

This approach relies on one key assumption, that every clique contains a strictly positive definite clique potential. This is a reasonable assumption in Gaussian estimation, as a positive definite covariance/information matrix is required to define a multi-dimensional Gaussian. Therefore, this requires that each clique potential defines a valid Gaussian distribution. Note that the JT construction technique described in Section 2.3.3 does not have this requirement[4].

Given that each clique has a positive definite potential, the following theorem guarantees that the global potential is also positive definite, see Figure 4.4 for an example.

Theorem 4.4. *Let* $\mathbf{A}_i$ *be a positive definite matrix and let* $\mathbf{A}$ *be constructed by placing* $\mathbf{A}_1$ *to* $\mathbf{A}_m$ *in a manner such that there are no zero valued diagonal, but any amount of overlap is allowed. The matrix* $\mathbf{A}$ *will be positive definite.*

[4]Treemap (see Appendix B.2.2) also allowed positive semi-definite potentials in its bounded information blocks.



(a) $\mathbf{A}_1 \succ \mathbf{0}$ **(b)** $\mathbf{A}_2 \succ \mathbf{0}$ **(c)** $\mathbf{A}_3 \succ \mathbf{0}$ **(d)** $\mathbf{A}_1 + \mathbf{A}_2 + \mathbf{A}_3 \succ \mathbf{0}$

Figure 4.4 – Demonstration of the partition specified by Theorem 4.4. In this example three strictly positive definite matrices are summed. Each matrix has rows and columns associated with different states, such that each matrix is positive semi-definite when considered with the other states added as zeros in the respective rows and columns. The sum of these matrices is strictly positive definite. Note that it is required that at least one matrix relates to every state, otherwise a zero will occur on the diagonal.

Proof. Consider the sum $\mathbf{A} = \mathbf{A}_1^f + \ldots + \mathbf{A}_n^f$ where $\mathbf{A}_i^f$ is $\mathbf{A}_i$ padded with zeros so that it is conformable and that $\mathbf{A}$ has no zero valued diagonal elements. Each $\mathbf{A}_i^f$ is positive semi-definite since no principal sub-matrix[5] of $\mathbf{A}_i^f$ has a negative determinant[52, p.321]. (The determinants are either zero if a zero is included on the diagonal, or positive if the principal sub-matrix is $\mathbf{A}_i$ or a subset of). Recall that if $\mathbf{A}_i^f \succeq \mathbf{0} \iff \mathbf{x}\mathbf{A}_i^f\mathbf{x} \geq 0 \, \forall \mathbf{x} \neq \mathbf{0}$.

Consider $\mathbf{x}^T\mathbf{A}_i^f\mathbf{x}$, where column vector $\mathbf{x} \in \mathbb{R}^n$ and n is the dimension of $\mathbf{A}$. If at least one nonzero element of $\mathbf{x}$ corresponds a non-zero component of $\mathbf{A}_i^f$, then $\mathbf{x}^T\mathbf{A}_i^f\mathbf{x} > 0$ since $\mathbf{A}_i$ is positive definite. Therefore if $\sum_i^m \mathbf{A}_i^f$ spans the diagonal such that there is no zero valued diagonal, then if $\mathbf{x} \neq \mathbf{0}$ then $\mathbf{x}^T\mathbf{A}_i^f\mathbf{x} > 0$ will be satisfied by at least one $\mathbf{A}_i^f$. The expression $\mathbf{x}^T\mathbf{A}\mathbf{x}$ can be decomposed:

$$\mathbf{x}^T\mathbf{A}\mathbf{x} = \sum_i^m \mathbf{x}^T\mathbf{A}_i^f\mathbf{x},$$

since at least one $\mathbf{x}^T\mathbf{A}_i^f\mathbf{x}$ will be positive, and all other i satisfy $\mathbf{x}^T\mathbf{A}_i^f\mathbf{x} \geq 0$, therefore $\mathbf{x}^T\mathbf{A}\mathbf{x} > 0$ which is necessary and sufficient for $\mathbf{A} \succ \mathbf{0}$. □

[5] A principal sub-matrix is any sub-matrix that is attained by removing the same rows and columns.

Theorem 4.4 reduces the problem of maintaining a large positive definite matrix to maintaining a property on a set of much smaller and independent matrices, the information based approaches used the positive definiteness of the Schur complement to achieve the same property. This useful property will be exploited in Sections 4.2.3 and 4.2.4 to enable a parallel approach to sparsification.

4.2.2 Exact Clique Split

To allow a large information matrix to be transformed to a JT with positive definite cliques, a technique for decomposing it into a sum of small positive-definite matricies is required. This technique is called an *exact clique split* because it makes no approximations. It can be posed as a convex problem. Consider:

$$\mathbf{Y} = \begin{bmatrix} \mathbf{A} & \mathbf{B} & \mathbf{0} \\ \mathbf{B}^T & \mathbf{C} & \mathbf{D} \\ \mathbf{0} & \mathbf{D}^T & \mathbf{E} \end{bmatrix}, \tag{4.12}$$

it is desired to split $\mathbf{Y} = \mathbf{Y}_1 + \mathbf{Y}_2$ such that

$$\mathbf{Y}_1 = \begin{bmatrix} \mathbf{A} & \mathbf{B} & \mathbf{0} \\ \mathbf{B}^T & \mathbf{C}_1^* & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}, \mathbf{Y}_2 = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_2^* & \mathbf{D} \\ \mathbf{0} & \mathbf{D}^T & \mathbf{E} \end{bmatrix} \tag{4.13}$$

and ensuring that

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^T & \mathbf{C}_1^* \end{bmatrix} \succ \mathbf{0}, \begin{bmatrix} \mathbf{C}_2^* & \mathbf{D} \\ \mathbf{D}^T & \mathbf{E} \end{bmatrix} \succ \mathbf{0}. \tag{4.14}$$

This poses the following matrix splitting problem:

$$\max_{\mathbf{C}_1^*, \mathbf{C}_2^*} \operatorname{tr}(\mathbf{C}_1^* + \mathbf{C}_2^*) \tag{4.15}$$

$$s.t.$$

$$\mathbf{C}_1^* \succ \mathbf{0}, \quad \mathbf{C}_2^* \succ \mathbf{0}$$
$$\mathbf{C}_1^* \succ \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}$$
$$\mathbf{C}_2^* \succ \mathbf{D}^T\mathbf{E}^{-1}\mathbf{D}$$
$$\mathbf{C}_1^* + \mathbf{C}_2^* = \mathbf{C}$$

To ensure that every matrix is positive definite, only a feasible solution, rather than an optimal solution is required. Analytically a feasible solution can be found by defining:

$$\Delta_{\mathbf{C}} = \mathbf{C} - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B} - \mathbf{D}^T\mathbf{E}^{-1}\mathbf{D} \tag{4.16}$$

If $\mathbf{Y} \succ \mathbf{0}$ then $\Delta_{\mathbf{C}} \succ \mathbf{0}$. Let $0 < \alpha < 1$, then a feasible solution is:

$$C_1^* = \alpha\Delta_{\mathbf{C}} + \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B} \tag{4.17}$$
$$C_2^* = (1-\alpha)\Delta_{\mathbf{C}} + \mathbf{D}^T\mathbf{E}^{-1}\mathbf{D} \tag{4.18}$$

However, this analytic approach encounters numerical errors when $\mathbf{Y}$ is close to being positive semi-definite or poorly conditioned. In these cases a convex optimisation package may be capable of finding a feasible solution to the problem in Equation 4.15. If the matrix needs to be split into more than two cliques, the above process can be performed multiple times.

4.2.3 Individual Clique Splitting

This approach treats each sparsification step as an optimisation on a particular $\mathbf{Y}_i$. Therefore, as long as the sparsification maintains $\mathbf{Y}_i^{app} \succ \mathbf{0}$ then the sum of the whole system will remain positive definite. This can be done on any number of cliques asynchronously, all that is required is that the result is positive definite.

The simplest case is where a clique contains the only reference to a link between states. That is, the union of the two states is not contained by any of the separators

of that clique. Let the states of interest be $\{x_1, x_2\}$, then the following is required:

$$\{x_1, x_2\} \not\subset \mathcal{S}_j \forall j \in \mathcal{C}_i. \tag{4.19}$$

This clique ($\mathcal{C}_i$) is then split into two cliques $\mathcal{C}_{i1}$ and $\mathcal{C}_{i2}$ with messages at $\mathcal{C}_i$ given by μ_i, the problem is defined below:

$$\begin{aligned} &\min_{\mathbf{Y}_{i1}, \mathbf{Y}_{i2}} D_{kl}(\mathbf{Y}_i + \mathbf{Y}_{\mu_i} || \mathbf{Y}_{i1} + \mathbf{Y}_{i1} + \mathbf{Y}_{\mu_i}) \\ &\quad s.t. \\ &\qquad \mathbf{Y}_{i1} \succ \mathbf{0}, \mathbf{Y}_{i2} \succ \mathbf{0} \\ &\qquad \mathbf{Y}_i \succeq \mathbf{Y}_{i1} + \mathbf{Y}_{i2} \end{aligned} \tag{4.20}$$

Observe that the trivial solution $\mathbf{Y}_{i1} = \epsilon\mathbf{I}, \mathbf{Y}_{i2} = \epsilon\mathbf{I}$ for some small positive ϵ and $\mathbf{I}$ the identity matrix, is a feasible solution for this problem. Therefore, using Slaters Condition this problem has an optimal solution [10]. However, the solution found using this method will always be an upper bound of the solution found using the technique in Section 4.1.2. This is because there may exist a $\mathcal{C}_k$ that is in the Markov blanket of $\mathcal{C}_i$ and therefore contains information about some states in $\mathcal{C}_i$, but the potential Ψ_k will not be changed by this approach. This is easily shown by the following theorem, which analyses the logdet term of the KLD cost. Consider the following partitioned information matrix:

$$\mathbf{Y} = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{B} & \mathbf{C}^T \\ \mathbf{C} & \mathbf{D} \end{bmatrix}. \tag{4.21}$$

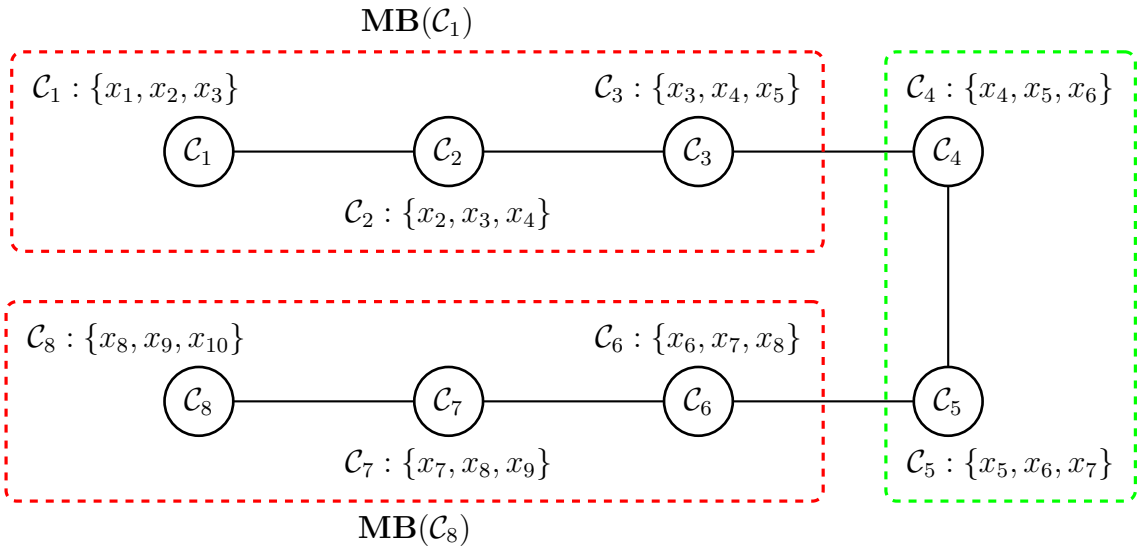
Theorem 4.5. *The cost function* $\text{logdet}(\mathbf{Y})$*, where the optimisation variable is* $\mathbf{A}$*, is bounded from below by* $\text{logdet}(\mathbf{B} - \mathbf{C}\mathbf{D}^{-1}\mathbf{C}^T) + \text{logdet}(\mathbf{D})$

Proof. Taking $\text{logdet}(\mathbf{Y})$ and using Equation A.6 we have

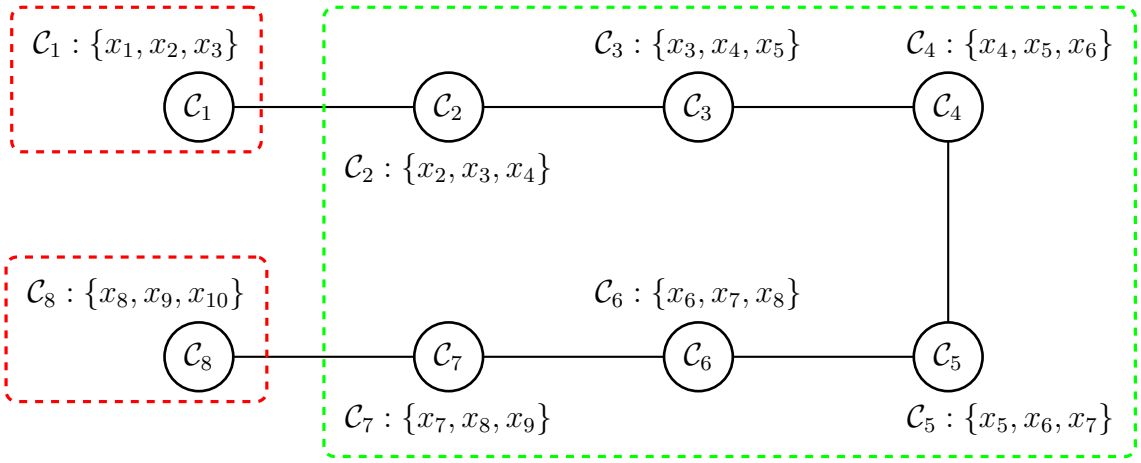
$$\text{logdet}(\mathbf{Y}) = \text{logdet}(\mathbf{A} + \mathbf{B} - \mathbf{C}\mathbf{D}^{-1}\mathbf{C}^T) + \text{logdet}(\mathbf{D}). \tag{4.22}$$

Taking the $\lim_{\mathbf{A}\to\mathbf{0}} \text{logdet}(\mathbf{Y})$, gives the desired result. □

The matrix $\mathbf{B}$ represents the information present in other cliques regarding the Markov blanket around the clique split. Thus the accuracy of the solution is affected by the magnitude of information in those cliques that are part of the Markov blanket but cannot be altered by this approach (that is they are not part of the clique potential which is modified by the problem). The cliques required for sparsification using the parallel approach, and the optimal approaches are shown in Figure 4.5.



(a) Markov blanket for the joint, sequential and distributed cases



(b) Required cliques for parallel sparsification

Figure 4.5 – Illustration of required cliques to perform two clique splits. $\mathcal{C}_1$ is to be split into $\mathcal{C}_{1a} = \{\mathbf{x}_1, \mathbf{x}_3\}$, $\mathcal{C}_{1b} = \{\mathbf{x}_2, \mathbf{x}_3\}$, while $\mathcal{C}_8$ is to be split into $\mathcal{C}_{8a} = \{\mathbf{x}_8, \mathbf{x}_{10}\}$, $\mathcal{C}_{8b} = \{\mathbf{x}_8, \mathbf{x}_9\}$. The clique required when using the approaches described in Section 4.1.2 is shown in (a), while the cliques required when using the parallel approach of Section 4.2 is shown in (b). A red box encloses cliques that contain information required for the optimisation, it can be seen that the parallel approach requires fewer cliques than the sequential approaches.

4.2.4 Separator Splitting

It may be desired to disconnect states which appear in more than one clique. This generalisation of a clique split is called a separator split, and is demonstrated in Figure 4.6. This will involve all cliques that are associated with the separator tree of links to be removed. A separator tree is defined below:

Definition 4.2. *A separator tree is a set of separators that contain a specific group of states as a subset. The running intersection property enforces that this must be a tree.*

If a link is part of a separator tree then all the cliques which are associated with that tree must be gathered, resulting in multiple cliques being split. To simplify the specification of this problem, define the original collection of cliques $\mathcal{C}_{\mathcal{O}}$ to represent all of the cliques to be modified and their associated potentials and message sets. Also let $\mathcal{C}_{\mathcal{S}}$ represent the new clique split set and its associated potentials and messages. The separator splitting problem is expressed as[6],

$$\min_{\mathbf{Y}_{\mathcal{S}}} D_{kl}(\mathbf{Y}_{\mathcal{O}} + \mathbf{Y}_{\mu_{\mathcal{O}}} || \mathbf{Y}_{\mathcal{S}} + \mathbf{Y}_{\mu_{\mathcal{O}}}) \tag{4.23}$$

$$s.t.$$

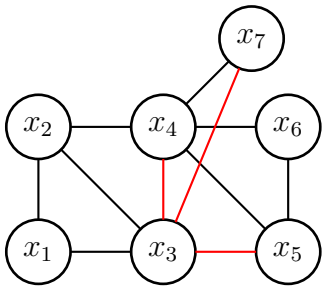
$$\mathbf{Y}_i \succ \mathbf{0}, \forall i \in \mathcal{S}$$

$$\mathbf{Y}_{\mathcal{O}} \succeq \mathbf{Y}_{\mathcal{S}}.$$

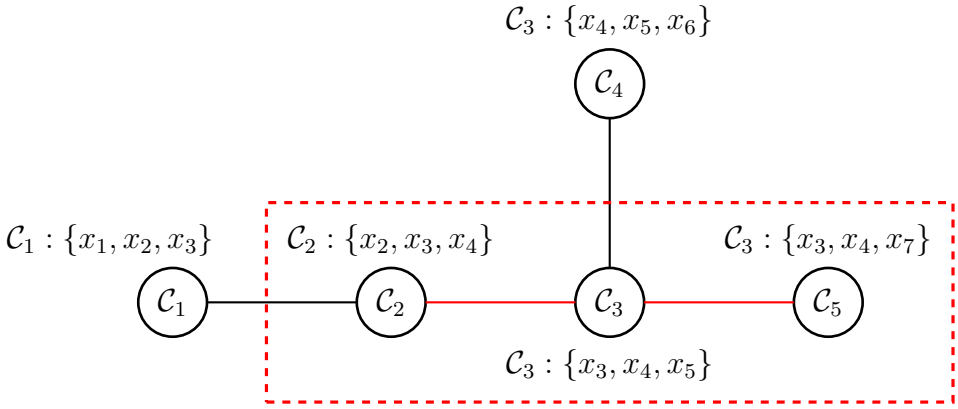
This multiple clique scenario will result in a much smaller set of altered cliques compared to the equivalent separator split in Section 4.1.2, but its solution represents an upper bound to the problem.

Since Theorem 4.4 ensures that the information matrix over all states is positive definite, the parallel variant of JT sparsification can be applied with no regard for violating the positive definiteness of the system. This allows any number of parallel sparsification events to occur asynchronously and in parallel, thus achieving a completely local cost.

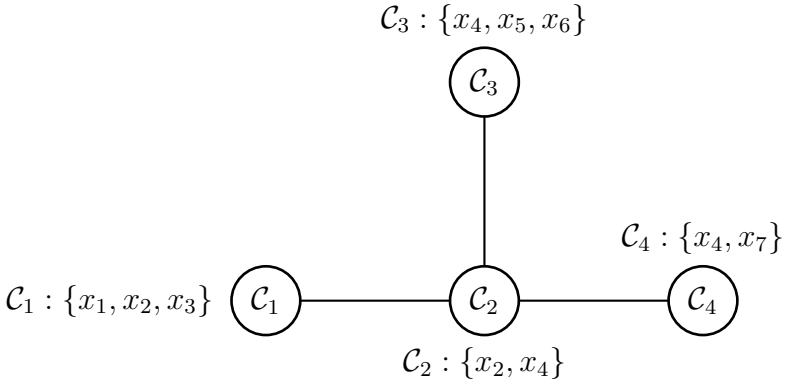
[6] $\mathbf{Y}_{\mu_{\mathcal{O}}}$ is the messages on $\mathcal{S}$ onto the states $\mathcal{O}$.



(a) Original graph. Equivalent links for sparsification indicated in red.



(b) Junction Tree. Red link indicates separator to be split. Dashed red box indicates cliques potentials to be modified.



(c) Junction tree after splitting $\mathcal{S}_{23}$ and $\mathcal{S}_{35}$

Figure 4.6 – In this example, the states to be split exist in two separators ($\mathcal{S}_{23}$ and $\mathcal{S}_{35}$). (a) shows the underlying triangulated graph, and the equivalent links which are removed by the clique split. (b) shows the junction tree, the separator tree to be split is highlighted in red, and the cliques required for this split (enclosed by a red box). (c) shows the final junction tree structure, the maximum path length has been reduced, and the size of all cliques except $\mathcal{C}_1$ and $\mathcal{C}_3$ is smaller.

4.2.5 Locally Computable Error Bound

The true error between the optimal and the parallel approaches cannot be determined without performing the optimisation. However, an upper bound on the error induced by the parallel variant can be computed by calculating the KLD without the effect of the messages. The maximum error is:

$$\Delta D_{kl} = D_{kl} < \text{With messages} > - D_{kl} < \text{Without messages} > \tag{4.24}$$

$$\Delta D_{kl} = D_{kl}(\Psi_i^{tr} - \mu_i^{tr} || \Psi_i^{app} - \mu_i^{tr}) - D_{kl}(\Psi_i^{tr} - \mu_i^{tr} || \Psi_i^{app}) \tag{4.25}$$

That is, if the messages from other cliques are ignored, then the KLD found at the clique coincides with the true KLD (which is the minimum). If the messages are included, then this represents a maximum KLD. Therefore the optimal must lie between the two, resulting in the error shown above. This bound however assumes that other parts of the network have not changed since the messages were last received.

Since the parallel JT approach is based upon information locally available at a particular node, if a particular clique is unable to gather all clique potentials that contain a particular set of states then those states cannot be sparsified. Thus, the communication cost associated with the parallel variant of CS is linked to how many cliques contain the states of interest. For example, in Figure 4.6 cliques $\mathcal{C}_1$, $\mathcal{C}_2$ and $\mathcal{C}_3$ are required to perform the clique split.

4.2.6 Parallel Sparsification Computational Cost

The computational cost to perform parallel clique splitting for a single clique is $\mathcal{O}(k^{6.5})$ where k is the maximum clique size. This is the same cost as performing the full CS split. However, the advantage becomes apparent when considering slitting multiple cliques that have disjoint Markov blankets. Using the joint approach to multiple clique splitting, the cost is $\mathcal{O}((tk)^{6.5})$, where t is the number of cliques to be split. For the sequential approach, the cost is reduced to $\mathcal{O}(tk^{6.5})$. However for the parallel approach the computational cost remains at $\mathcal{O}(k^{6.5})$, which is independent of t.

4.3 Comparison Between JT and Information Based Sparsification

There are several advantages for the JT with respect to the information based approach. By expressing the problem in cliques, JTs provide logical structures to build decentralised algorithms upon. This clique structure has the following advantages: (i) The JT clique size is a locally computable indicator of current solve complexity; (ii) JT message passing allows for marginals to be readily available at every clique; and (iii) the implicit chordal structure allows strong guarantees on feasibility of the problem. Applying the positive-definite decomposition also allows for parallel clique splitting while maintaining conservatism but trading off KLD accuracy.

The most significant disadvantage for the JT with respect to the information based approach is that the elimination ordering is predetermined. Therefore, an information matrix representation can always achieve as good or better an elimination ordering and the implicit chordal structure may obscure some extra sparsity that exists in the problem.

4.4 Linear SLAM Simulation

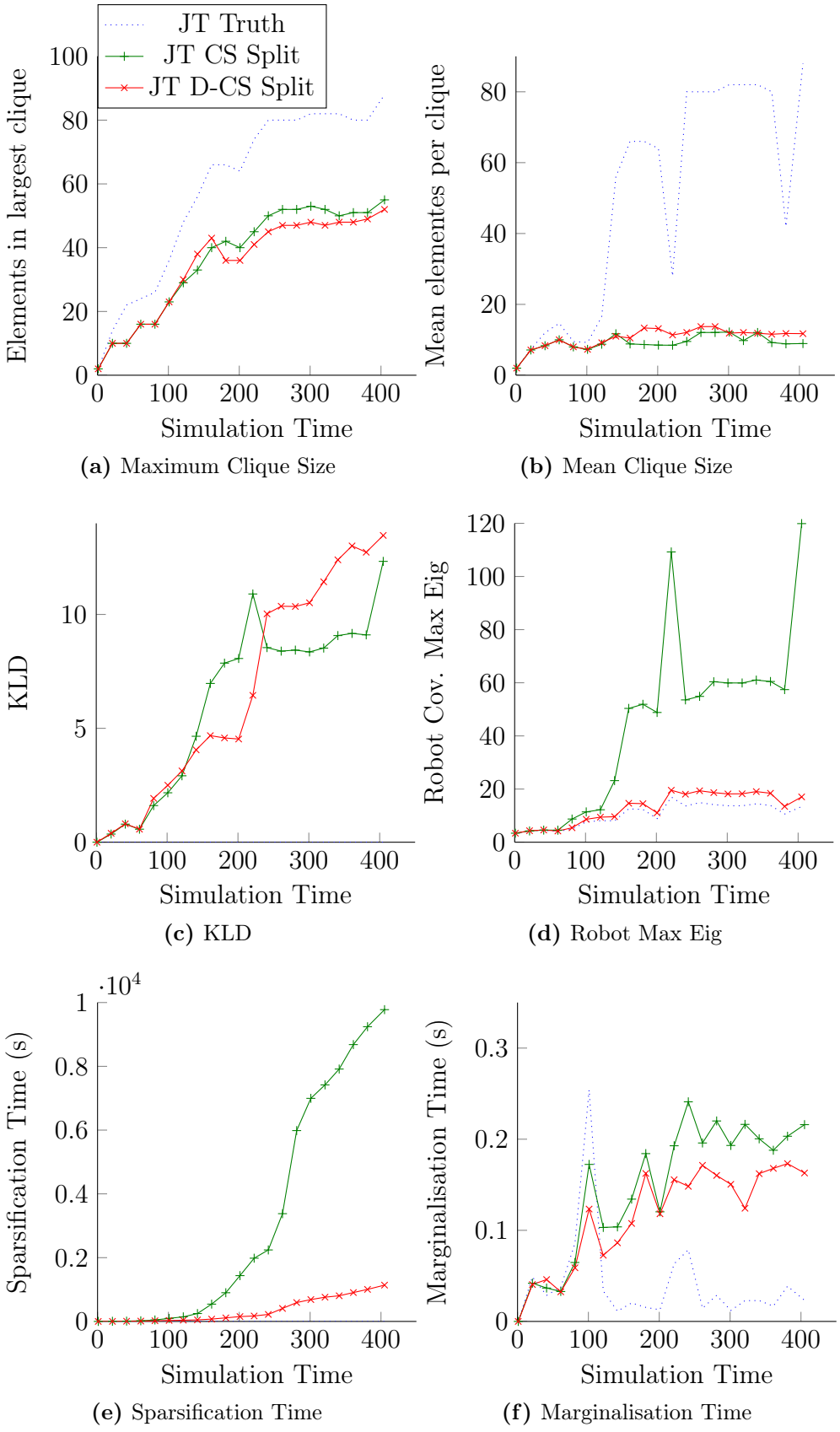
The synthetic Simultaneous Localisation and Mapping (SLAM) data as used in Section 3.3.1 is also used in the following SLAM example. A smoothing approach to the SLAM problem is used, but is implemented using the JT framework described in Section 2.3. Errors induced through sparsification will be carried by each approach throughout the simulation.

The CS JT algorithm (labeled JT CS Split), as well as the parallel variant of CS (labeled JT D-CS Split) are compared. During the simulation both approaches maintained conservative approximations with respect to the true state JT True. Due to the different numerical values in each filter, different cliques were best suited for sparsification but the overall sparsity was maintained approximately equal for both

filters. Figure 4.7(a) shows the number of elements in the maximum clique, while Figure 4.7(b) shows the mean number of elements per clique. Observe that both plots show the new techniques significantly improving upon the sparsity of the true solution. Figure 4.7(c) shows the KLD of both approaches. Numerical difficulties resulted in the JT D-CS Split approach maintaining a lower KLD initially, but this was eventually corrected and the JT-CS Split achieves a better KLD at the end. JT CS Split may alter all elements in the Markov blanket during its optimisation, while JT D-CS Split can only change the values that are in cliques that are being split, allowing it to achieve a lower KLD. Figure 4.7(d) shows the maximum eigenvalue of the robot covariance. Unexpectedly, the JT D-CS Split technique achieves a better KLD than the JT-CS Split. This is because JT-CS Split may have more aggressively sparsified the robot states than the JT D-CS split. Note that although the maximum eigenvalue is higher, the overall KLD is lower after time 250. This demonstrates that the uncertainty in the entire state space must be considered when comparing filters.

Figure 4.7(e) shows the time taken to perform sparsification. This time includes selecting the clique for sparsification and reforming a valid JT. Figure 4.7(f) shows the time taken to perform marginalisation. Note that the True JT approach appears to outperform the approximate approaches as the clique size increases. This is because large cliques can take advantage of optimized linear algebra routines in MATLAB, while the JT approaches, which involve message passing have a much lower performance when implemented in MATLAB.

This experiment shows that the JT based variants of CS are capable of maintaining sparse conservative approximations of the true distribution.



(a) Maximum Clique Size **(b)** Mean Clique Size

(c) KLD **(d)** Robot Max Eig

(e) Sparsification Time **(f)** Marginalisation Time

Figure 4.7 – Filtering SLAM example. All techniques are conservative with respect to the true value. Observe that both the maximum and mean clique size is lower than the true system. Note that the junction tree implementation was written in MATLAB, and that this resulted in very slow solving once the number cliques in the problem became large. Note that due to the reduced computational requirement of D-JT this is not as apparent.

Chapter 5

Conclusion and Future Work

Long term estimation can become intractable due to marginalisation of past states unless approximate techniques are used to maintain the sparse structure of the estimate. It is important for approximate techniques to be conservative approximations so that any risks associated with uncertainty in the state variables are not understated. This thesis has described three approaches that minimise the KLD while maintaining conservatism.

For problems that use the information form as their parameterisation, Chapter 3 details a new method for enforcing sparsity in the information matrix of a Gaussian distribution. The complexity of this method is a function of the maximum cardinal value of the graph, which can be quite small for sparse problems. This method was compared on a simple simulated SLAM data set and was capable of providing a better KLD than a competing technique, while guaranteeing conservatism.

For problems that require distributed estimation, Chapter 4 describes the application of conservative sparsification to Junction Trees. This formulation allows the marginals to be available at every clique, and therefore removes the need to consider the cost of forming the marginal when applying conservative sparsification to JT clique splitting.

Finally the Junction Tree clique splitting technique is extended to allow asynchronous parallel sparsification on Junction Trees. This technique, which is an approximation

to the conservative sparsification problem, enables sparsification in large decentralised estimation systems. A key feature of this technique is that it can be performed asynchronously and in parallel. Simulations show that the distributed approach has a worse KLD then the optimal CS based approach, but still maintains a conservative estimate.

5.1 Future Work

This thesis has not addressed the problem of choosing which edge (or edges) to remove. This problem is difficult as it is related to elimination orderings, for which the minimimum fill reducing elimination ordering is an NP Hard problem. Nonetheless, heuristics for choosing edges based on their weight or information content are commonly used. Section 2.5.2 describes some of the complications with choosing edges for removal. It would be interesting to incorporate the concept of expected fill-in from sparsifying a single link. For example, consider Figure 5.1, here two edges may have the same weight, yet removing the red edge is clearly better as it reduces the fill-in.

Perhaps a simpler yet related problem is choosing how to split cliques. This problem has a predetermined elimination ordering and so fill-in is easier to compute.

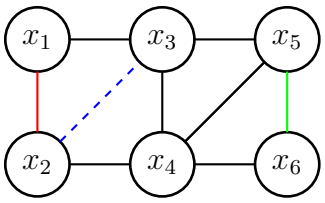


Figure 5.1 – Illustration of the fill-in inducing properties of edges. The dashed blue edge is fill-in induced by the elimination ordering $(x_1, x_2, x_3, x_4, x_5, x_6)$. Observe that removing the red edge will also remove the blue dashed fill-in, while removing the green edge will not result in less fill-in since x_4, x_5 is not a fill-in edge.

A direction arising immediately from this work is the concept of a *sparse marginalise* operation. This would apply conservative sparsification to the graph immediately after marginalisation, perhaps maintaining a spanning tree over the remaining states

(in a method similar to [37]). This is a good avenue as it avoids the problem of re-linearisation, which cannot be addressed by CS.

Applying these techniques to non-simulated data set is another exciting avenue. However care needs to be taken to ensure that observations involved in sparsification are not re-linearised, as this can destroy the induced sparsity. One possible approach is to sparsify an observation after re-linearisation and then fusing it into the system.

This thesis has been unable to prove convergence of iterative sparsification with disjoint Markov Blankets. This is considered for future work. A complicating factor is that the sequence of clique potentials are not necessarily decreasing over the Loewner ordering.

Bibliography

[1] Massoud A. and Wollenberg B. Toward a smart grid: power delivery for the 21st century. *Power and Energy Magazine, IEEE*, 3(5):34–41, 2005.

[2] P.R. Amestoy, T.A. Davis, and I.S. Duff. An approximate minimum degree ordering algorithm. *SIAM Journal on Matrix Analysis and Applications*, 17(4): 886–905, 1996.

[3] Education Australian Government Department of Resources and Tourism. Smart grid,smart city. February 2013.

[4] R. Bachmayer and N. E. Leonard. Vehicle networks for gradient descent in a sampled environment. In *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*, volume 1, pages 112–117. IEEE, 2002.

[5] T. Bailey and H. Durrant-Whyte. Simultaneous localization and mapping : Part ii. *Robotics & Automation Magazine, IEEE*, 13(3):108–117, 2006.

[6] T. Bailey, M. Bryson, H. Mu, J. Vial, L. McCalman, and H. Durrant-Whyte. Decentralised cooperative localisation for heterogeneous teams of mobile robots. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2859–2865. IEEE, 2011.

[7] O. Banerjee, L. El Ghaoui, and A. 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.

[8] C.M. Bishop et al. *Pattern recognition and machine learning*, volume 4. springer New York, 2006.

[9] A. Björk. Numerical methods for least squares problems. *SIAM*, 1996.

[10] S.P. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge Univ Pr, 2004. ISBN 0521833787.

[11] M. Bryson and S. Sukkarieh. Bearing-only slam for an airborne vehicle. In *Australasian Conf. Robot. Autom.(ACRA 2005), Sydney, Australia*, 2005.

[12] C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *Information Theory, IEEE Transactions on*, 14(3):462–467, 1968.

[13] J. Dahl, L. Vandenberghe, and V. Roychowdhury. Covariance selection for nonchordal graphs via chordal embedding. *Optimization Methods & Software*, 23(4):501–520, 2008.

[14] T.A. Davis. *Direct methods for sparse linear systems*, volume 2. Society for Industrial Mathematics, 2006.

[15] Timothy A Davis. The university of florida sparse matrix collection. In *NA digest*. Citeseer, 1994.

[16] A. P Dempster. Covariance selection. *Biometrics*, pages 157–175, 1972.

[17] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part i. *Robotics & Automation Magazine, IEEE*, 13(2):99–110, 2006.

[18] R. Eustice, H. Singh, J. Leonard, M. Walter, and R. Ballard. Visually navigating the rms titanic with slam information filters. In *Proc. Robotics: Science and Systems*. Cambridge, MA, 2005.

[19] R. Eustice, M. Walter, and J. Leonard. Sparse extended information filters: Insights into sparsification. In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 3281–3288. IEEE, 2005. ISBN 0780389123.

[20] R.M. Eustice, H. Singh, and J.J. Leonard. Exactly sparse delayed-state filters for view-based SLAM. *IEEE Transactions on Robotics*, 22(6):1100–1114, 2006. ISSN 1552-3098.

[21] U. Frese. *An O (log n) algorithm for simultaneous localization and mapping of mobile robots in indoor environments*. PhD thesis, PhD thesis, University of Erlangen-Nürnberg, 2004.

[22] U. Frese. Treemap: An o (log n) algorithm for indoor simultaneous localization and mapping. *Autonomous Robots*, 21(2):103–122, 2006.

[23] J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

[24] G. Guennebaud and B. Jacob. Eigen v3. http://eigen.tuxfamily.org. 2010.

[25] E. T Jaynes. *Probability theory: the logic of science*. Cambridge university press, 2003.

[26] A.H. Jazwinki. *Stochastic processes and filtering theory*. Academic Pr, 1970. ISBN 0123815509.

[27] E. Jones, T. Oliphant, and P. Peterson. SciPy: Open source scientific tools for Python. 2001–.

[28] M.I. Jordan and C. Bishop. Introduction to graphical models. 2004.

[29] S.J. Julier and J.K. Uhlmann. General decentralized data fusion with covariance intersection (ci). 2001.

[30] S.J. Julier and J.K. Uhlmann. A non-divergent estimation algorithm in the presence of unknown correlations. In *American Control Conference, 1997. Proceedings of the 1997*, volume 4, pages 2369–2373. IEEE, 2002. ISBN 0780338324.

[31] S.J. Julier and J.K. Uhlmann. Using covariance intersection for SLAM. *Robotics and Autonomous Systems*, 55(1):3–20, 2007. ISSN 0921-8890.

[32] Yoshifumi K and Yasutaka A. Triton: New ocean and atmosphere observing buoy network for monitoring enso. *Umi no Kenkyu*, 10(2):157–172, 2001.

[33] M. Kaess, V. Ila, R. Roberts, and F. Dellaert. The bayes tree: An algorithmic foundation for probabilistic robot mapping. *Algorithmic Foundations of Robotics IX*, pages 157–173, 2011.

[34] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert. isam2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3281–3288. IEEE, 2011.

[35] U. Kjærulff. Approximation of bayesian networks through edge removals. Technical Report IR-93-2007, Institute for Electronic Systems, Department of Mathematics and Computer Science, Aalborg University, 1993.

[36] U. Kjaerulff. Reduction of computational complexity in bayesian networks through removal of weak dependencies. Technical report, Aalborg University, February 1994.

[37] H. Kretzschmar and C. Stachniss. Information-theoretic compression of pose graphs for laser-based slam. *The International Journal of Robotics Research*, 31 (11):1219–1230, 2012.

[38] M. Laurent. Matrix completion problems. *The Encyclopedia of Optimization*, 3: 221–229, 2001.

[39] D. JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[40] I. Mahon. *Vision-based Navigation for Autonomous Underwater Vehicles*. PhD thesis, University of Sydney, 2008.

[41] I. Mahon, S. Williams, O. Pizarro, and M. Johnson-Roberson. Efficient view-based slam using visual loop closures. *Robotics, IEEE Transactions on*, 24 (5):1002–1014, 2008.

[42] A. Makarenko, A. Brooks, T. Kaupp, H. Durrant-Whyte, and F. Dellaert. Decentralised data fusion: A graphical model approach. In *Information Fusion, 2009. FUSION'09. 12th International Conference on*, pages 545–554. IEEE, 2009.

[43] MATLAB. *version 7.10.0 (R2010a)*. The MathWorks Inc., Natick, Massachusetts, 2010.

[44] H. Mu, T. Bailey, P. Thompson, and H. Durrant-Whyte. Decentralised solutions to the cooperative multi-platform navigation problem. *Aerospace and Electronic Systems, IEEE Transactions on*, 47(2):1433–1449, 2011.

[45] B. K. Natarajan. Sparse approximate solutions to linear systems. *SIAM journal on computing*, 24(2):227–234, 1995.

[46] H Niessner and K Reichert. On computing the inverse of a sparse matrix. *International journal for numerical methods in engineering*, 19(10):1513–1526, 1983.

[47] Andress Nuchter, Hartmut Surmann, Kai Lingemann, Joachim Hertzberg, and Sebastian Thrun. 6d slam with an application in autonomous mine mapping. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 2, pages 1998–2003. IEEE, 2004.

[48] M.A. Paskin. Thin junction tree filters for simultaneous localization and mapping. *Computer*, 2002.

[49] M.A. Paskin. *Exploiting locality in probabilistic inference*. PhD thesis, Citeseer, 2004.

[50] D. J. Rose, R. E. Tarjan, and G. S. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM Journal on computing*, 5(2):266–283, 1976.

[51] G.A.F. Seber. *A matrix handbook for statisticians*, volume 746. Wiley-Interscience, 2007.

[52] G. Strang. Linear algebra and its applications academic. *New York*, 19802, 1976.

[53] Kazuhiro Takahashi. Formation of a sparse bus impedance matrix and its application to short circuit study. In *IEEE PICA Conf., Minneapolis, 1973*, 1973.

[54] S. Thrun, Y. Liu, D. Koller, A.Y. Ng, Z. Ghahramani, and H. Durrant-Whyte. Simultaneous localization and mapping with sparse extended information filters. *The International Journal of Robotics Research*, 23(7-8):693, 2004. ISSN 0278-3649.

[55] GP Timms, JW McCulloch, P McCarthy, B Howell, PA De Souza, MD Dunbabin, and K Hartmann. The tasmanian marine analysis network (tasman). In *OCEANS 2009-EUROPE*, pages 1–6. IEEE, 2009.

[56] K.C. Toh, M.J. Todd, and R.H. Tütüncü. Sdpt3—a matlab software package for semidefinite programming, version 1.3. *Optimization Methods and Software*, 11(1-4):545–581, 1999.

[57] R.H. Tütüncü, K.C. Toh, and M.J. Todd. Solving semidefinite-quadratic-linear programs using sdpt3. *Mathematical programming*, 95(2):189–217, 2003.

[58] J.K. Uhlmann. Covariance consistency methods for fault-tolerant distributed data fusion. *Information Fusion*, 4(3):201–215, 2003. ISSN 1566-2535.

[59] J. Vial, H. Durrant-Whyte, and T. Bailey. Conservative sparsification for efficient and consistent approximate estimation. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 886–893. IEEE, 2011.

[60] M. Walter, F. Hover, and J Leonard. Slam for ship hull inspection using exactly sparse extended information filters. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1463–1470. IEEE, 2008.

[61] M.R. Walter, R.M. Eustice, and J.J. Leonard. Exactly sparse extended information filters for feature-based SLAM. *The International Journal of Robotics Research*, 26(4):335, 2007. ISSN 0278-3649.

[62] C. Wang, D. Sun, and K.C. Toh. Solving log-determinant optimization problems by a newton-cg primal proximal point algorithm. *SIAM Journal on Optimization*, 20(6):2994–3013, 2010.

[63] H. Wang, C. Chen, A. Ali, S. Asgari, R. E. Hudson, K. Yao, D. Estrin, and C. Taylor. Acoustic sensor networks for woodpecker localization. In *Optics & Photonics 2005*, pages 591009–591009. International Society for Optics and Photonics, 2005.

[64] SP Wu, L. Vandenberghe, and S. Boyd. MAXDET: software for determinant maximization problems. *Information Systems Laboratory, Stanford University*, 1996.

[65] M. Yannakakis. Computing the minimum fill-in is np-complete. *SIAM Journal on Algebraic Discrete Methods*, 2(1):77–79, 1981.

Appendix A

Proofs

A.1 Useful Identities

Positive Definite Matrices

Suppose that $\mathbf{A}$ and $\mathbf{B}$ are symmetric real matrices and $\mathbf{R}$ is non-singular, then (Eq. 10.46 [51]):

$$\mathbf{A} \succeq \mathbf{B} \iff \mathbf{R}^T\mathbf{A}\mathbf{R} \succeq \mathbf{R}^T\mathbf{B}\mathbf{R} \tag{A.1}$$

also note the useful identity (Eq. 10.58(a) of [51]):
let $\mathbf{A} \succ \mathbf{0}$ and $\mathbf{B} \succeq \mathbf{0}$ then

$$\mathbf{A} + \mathbf{B} \succ \mathbf{0}. \tag{A.2}$$

and the corollary (Eq. 10.58(c) of [51])

$$\text{if } \mathbf{A} - \mathbf{B} \succ \mathbf{0} \Rightarrow \det(\mathbf{A} - \mathbf{B}) < \det(\mathbf{A}). \tag{A.3}$$

Another useful identity (Eq. 10.55(a) of [51]). Let $\mathbf{V}$ have dimensions $n \times n$ and $\mathbf{V} \succ \mathbf{0}$. Also let $\mathbf{X}$ have dimensions $n \times p$ with $\text{rank}(\mathbf{X}) = p$ then

$$\mathbf{V} \succeq \mathbf{X}(\mathbf{X}^T\mathbf{V}^{-1}\mathbf{X})^{-1}\mathbf{X}^T \tag{A.4}$$

Partitioned Matrices

Suppose $\mathbf{A}$ is partitioned as:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix}, \tag{A.5}$$

then (Eq. 14.17(b) from [51]):

$$\det(\mathbf{A}) = \det(\mathbf{A}_{22}) \det(\mathbf{A}_{11} - \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{12}^T), \tag{A.6}$$

and using the shorthand introduced in Section 2.2.3,

$$\det(\mathbf{A}) = \det(\mathbf{A}_{22}) \det(\mathbf{A}/\mathbf{A}^{*2}), \tag{A.7}$$

Note the useful identities: From [51] p290 14.5 (b)

$$\mathbf{A} \succ \mathbf{0} \iff \mathbf{A}/\mathbf{A}^{*2} \succ \mathbf{0} \tag{A.8}$$

Shorthand

If $\mathbf{A}$ is symmetric, i.e. $\mathbf{A}_{12}^T = \mathbf{A}_{21}$, then (Eq. 14.26(a) from [51])

$$\mathbf{A} \succ \mathbf{0} \tag{A.9}$$

if and only if

$$\mathbf{A}_{11} \succ \mathbf{0},$$

and

$$\mathbf{A}/\mathbf{A}^{*1} \succ \mathbf{0}. \tag{A.10}$$

Matrix Inversion Lemma

The 2x2 matrix inversion lemma (Eq, 14.11(1) [51]):

$$\mathbf{A}^{-1} = \begin{bmatrix} (\mathbf{A}/\ \mathbf{A}^{*2})^{-1} & -\mathbf{A}_{11}^{-1}\mathbf{A}_{12}(\mathbf{A}_{22}^{-1} - \mathbf{A}_{21}\mathbf{A}_{11}^{-}1\mathbf{A}_{21}^{T}) \\ (-\mathbf{A}_{11}^{-1}\mathbf{A}_{12}(\mathbf{A}_{22}^{-1} - \mathbf{A}_{21}\mathbf{A}_{11}^{-}1\mathbf{A}_{21}^{T}))^{T} & \mathbf{A}_{22}^{-1} + \mathbf{A}_{22}^{-1}\mathbf{A}_{12}^{T}(\mathbf{A}/\ \mathbf{A}^{*2})^{-1}\mathbf{A}_{12}\mathbf{A}_{22}^{-1} \end{bmatrix} \tag{A.11}$$

A.2 Alternative Proof Approaches

This appendix demonstrates some alternative approaches for the proofs in Chapter 3. These proofs appeared in [59], however an error was later found. The error was in assuming that a $\mathcal{G}(\mathbf{x})$ marginals could be found that is positive definite when the marginal were subtracted. If the graph of $\mathcal{G}(\mathbf{x})$ was not locally chordal, then this cannot be guaranteed.

This appendix proves two key theorems that are vital for the problem reduction described in Section 3.2.1. We define the following notation; $(\mathbf{A})_{ij}$ accesses the ith element and jth row of the matrix expression $\mathbf{A}$ Let the matrix $\mathbb{E}_{nm}$ be known as the indicator matrix, defined as:

$$(\mathbb{E}_{nm})_{ij} = \begin{cases} \sqrt{2}, & \text{if } (i \neq j) \text{ and} \\ & ((n,m) = (i,j) \text{ or } (n,m) = (j,i)) \\ 1, & \text{if } (i = j)\text{and}(n = i)\text{and}(m = j) \\ 0, & \text{otherwise}\,. \end{cases}$$

We will also define a mapping from $\mathbb{R}^n$ to $\mathbb{S}^{m\times m}$ (where $\mathbb{S}^{m\times m}$ is the set of real

symmetric matrices):

$$F(\mathbf{x}) = \mathbf{F}_0 + \sum_{i=1}^{n} x_i \mathbb{E}_{qp}, \quad (q,p) \in \mathcal{Z} \tag{A.12}$$

where, $\mathcal{Z}$ is a set of indices in a $m \times m$ matrix space, $\mathbb{E}_{qp}, \mathbf{F}_0 \in \mathbb{S}^{m \times m}$ and $(\mathbf{F}_0)_{ij} = 0$ for all $(i,j) \in \mathcal{Z}$.

We now define the CS problem:

$$\underset{\mathbf{Y}_{app}}{\operatorname{argmin}} \operatorname{tr}(\mathbf{Y}_{app}\mathbf{Y}_{tr}^{-1}) - \operatorname{logdet}(\mathbf{Y}_{app})$$

subject to:

$$\begin{aligned} \mathbf{Y}_{tr} - \mathbf{Y}_{app} &\succeq \mathbf{0} \\ \mathbf{Y}_{app_{ij}} &= \mathbf{0} \quad \forall \{(i,j) \in \mathcal{I} \cup \mathcal{J}\}, \end{aligned}$$

where, $\mathbf{Y}_{tr}, \mathbf{Y}_{app} \in \mathbb{S}_{+}^{m \times m}$, $\mathcal{I}$ contains the set of indices that are to be sparsified, while $\mathcal{J}$ contains the set of indices that satisfy $(\mathbf{Y}_{tr})_{ij} = 0$. Here $\mathbb{S}_{+}^{m \times m}$ is the cone of $m \times m$ positive definite matrices.

We will also define another set of indices $\mathcal{K}$, which contains the indicies of all the elements that are to have a nonzero value in the final solution $\mathcal{K} \notin \mathcal{I} \cup \mathcal{J}$.

We can remove the equality constraints by performing a parameterisation using Equation A.12. Let,

$$\begin{aligned} \mathbf{Y}_{app} &= F(\mathbf{x}) \\ &= \mathbf{F}_0 + \sum_{i=1}^{n} x_i \mathbb{E}_{qp}, \{(q,p) \in \mathcal{K}\} \end{aligned}$$

and, $\mathbf{F}_0 = \mathbf{0}$.

We can now define the CS problem using the new parameterisation:

$$\underset{\mathbf{x}}{\operatorname{argmin}} \operatorname{tr}(F(\mathbf{x})\mathbf{Y}_{tr}^{-1}) - \operatorname{logdet}(F(\mathbf{x}))$$

subject to:

$$\mathbf{Y}_{tr} - F(\mathbf{x}) \succeq \mathbf{0}.$$

The Lagrange dual function for this problem is:

$$L(\mathbf{x}, \boldsymbol{\lambda}) = \operatorname{tr}(F(\mathbf{x})\mathbf{Y}_{tr}^{-1}) - \operatorname{logdet}(F(\mathbf{x})) \tag{A.13}$$

$$+ \operatorname{tr}(\boldsymbol{\lambda}(F(\mathbf{x}) - \mathbf{Y}_{tr})). \tag{A.14}$$

Using the Lagrange function we can form the first KKT[1] equation,

$$\frac{d}{d\mathbf{x}}\left(\operatorname{tr}(F(\mathbf{x})\mathbf{Y}_{tr}^{-1}) - \operatorname{logdet}(F(\mathbf{x}))\right) \tag{A.15}$$

$$+\frac{d}{d\boldsymbol{\lambda}}(\operatorname{tr}(\boldsymbol{\lambda}(F(\mathbf{x}) - \mathbf{Y}_{tr}))) = \mathbf{F}_{tr}^{-1} - \mathbf{F}_{sp}^{-1} + \mathbf{F}_{\lambda}. \tag{A.16}$$

where,

$$\mathbf{F}_{tr}^{-1} = \begin{cases} (\mathbf{Y}_{tr}^{-1})_{ij} & \forall\{(i,j) \in \mathcal{K}\} \\ \mathbf{0}, & \text{otherwise} \end{cases} \tag{A.17}$$

$$\mathbf{F}_{sp}^{-1} = \begin{cases} (\mathbf{F}^{-1})_{ij} & \forall\{(i,j) \in \mathcal{K}\} \\ \mathbf{0}, & \text{otherwise} \end{cases} \tag{A.18}$$

$$\mathbf{F}_{\lambda} = \begin{cases} (\boldsymbol{\lambda})_{ij} & \forall\{(i,j) \in \mathcal{K}\} \\ \mathbf{0}, & \text{otherwise} \end{cases} \tag{A.19}$$

It is sufficient for the optimal solution to this problem to satisfy the following equations (known as the KKT conditions):

$$\mathbf{F}_{tr}^{-1} - \mathbf{F}_{sp}^{-1} + \mathbf{F}_{\lambda} = \mathbf{0} \tag{A.20}$$

$$\mathbf{Y}_{tr} - \mathbf{F} \succeq \mathbf{0} \tag{A.21}$$

[1]The KKT or Karush-Kuhn-Tucker equations are a set of equations defining neccesary and sufficient conditions for the solution of a convex problem. See [10]

$$\boldsymbol{\lambda} \succeq \mathbf{0} \tag{A.22}$$

$$\mathrm{tr}(\boldsymbol{\lambda}(\mathbf{F} - \mathbf{Y}_{tr})) = \mathbf{0}. \tag{A.23}$$

The KKT conditions only hold under the assumption of strong duality. Slater's constraint states that a convex problem has strong duality if its inequality constraints hold strictly. For this problem, Slater's constraint holds if there exists a positive number ϵ, that satisfies the following (where $\mathbf{I}$ is the identity matrix):

$$\mathbf{Y}_{tr} - \epsilon\mathbf{I} \succ \mathbf{0}. \tag{A.24}$$

Since this is true for any matrix as long as $\mathbf{Y}_{tr} \succ \mathbf{0}$, the assumption of strong duality is valid for this problem.

Defining a Graph Cut: Consider a new operation, the *CS Graph Cut*, which can operate on any CS problem. This operation splits states in the problem into two disjoint sets of arbitrary size. The only constraint on a CS Graph Cut is that the full set of sparsification indices $\mathcal{I}$ must be included in only one of the sets. We can express this using matrix partitions as below:

$$\mathbf{Y}_{tr} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12}^T \\ \mathbf{A}_{12} & \mathbf{A}_{22} \end{bmatrix}, \quad F(\mathbf{x}) = \begin{bmatrix} \mathbf{A}_{11}^* & \mathbf{A}_{12}^{*\,T} \\ \mathbf{A}_{12}^* & \mathbf{A}_{22}^* \end{bmatrix},$$

where $\mathbf{A}_{11}$ contains the set of indices $\mathcal{I}$. We will also define another set of indices $\mathcal{L}$, which is the set of indices that satisfies,

$$(\mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{12}^T)_{ij} \neq 0.$$

We will define two sets of indices related to the cut. Let $\mathcal{J}_{11}$ be the subset of $\mathcal{J}$ which is contained in the cut $\mathbf{A}_{11}$, and let $\mathcal{K}_{11}$ be the subset of indices contained in the cut $\mathbf{A}_{11}$. It is important to recognise that the CS Graph Cut is not unique, there may be many valid cuts, all that is required is that the split is disjoint and indices $\mathcal{I}$ are entirely included in one of the sets.

Finally we will define the mapping from $\mathbb{R}^t$ to $\mathbb{S}^{u\times u}$:

$$G(\mathbf{x}_{11}) = \mathcal{G}_0 + \sum_{p=1}^{t} x_p \mathbb{E}_{i,j}, \forall\{(i,j) \in \mathcal{K}_{11}\} \tag{A.25}$$

where, $\mathbb{E}$ is a $u \times u$ indicator matrix, u is the size of the graph cut, and t is the number of free parameters in the cut, $\mathbf{x}_{11} \in \mathbb{R}^t$.

Theorem 1: [2] For any CS problem, choose a valid CS graph cut for which the following problem is feasible.
Let

$$\tilde{\mathbf{Y}}_{tr} = \mathbf{A}_{11} - \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{12}^T \tag{A.26}$$

$$(\mathcal{G}_0)_{ij} = \begin{cases} (-\mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{12}^T)_{ij}, & \forall\{(i,j) \in \mathcal{L}\} \\ 0, & \text{otherwise} \end{cases} \tag{A.27}$$

Now solve the problem (known as the reduced problem):

$$\underset{\mathbf{x}_{11}}{\operatorname{argmin}} \operatorname{tr}(G(\mathbf{x}_{11})\tilde{\mathbf{Y}}_{tr}^{-1}) - \operatorname{logdet}(G(\mathbf{x}_{11}))$$

subject to:

$$\tilde{\mathbf{Y}}_{tr} - \mathbf{G}(\tilde{\mathbf{x}}_{11}) \succeq \mathbf{0}$$

Let $\tilde{\mathbf{x}}_{11}$ and $\tilde{\boldsymbol{\lambda}}_{11}$ be the optimal primal and dual solutions to the reduced problem. We will then consider the optimal solution to the full problem to be:

$$F^*(\tilde{\mathbf{x}}) = \begin{bmatrix} \mathbf{G}(\tilde{\mathbf{x}}_{11}) + \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{12}^T & \mathbf{A}_{12} \\ \mathbf{A}_{12}^T & \mathbf{A}_{22} \end{bmatrix} \tag{A.28}$$

[2] Note that Theorem 1 is similar to the result in [35, Sect. 6.2] as both show that the KLD can be computed locally. Our theorem differs, as we show that the consistency constraint can also be optimised locally, and we do not require our final solution to be a Junction Tree (which requires a matrix ordering to be chosen and can cause fill in).

$$\boldsymbol{\lambda}^* = F^*(\tilde{\mathbf{x}})^{-1} - \mathbf{Y}_{tr}^{-1} \tag{A.29}$$

Proof Outline: This proof will show that the KKT conditions hold for the proposed solutions.

Proof. If $\tilde{\mathbf{x}}_{11}$ and $\tilde{\boldsymbol{\lambda}}_{11}$ are the optimal solutions to the reduced problem, then the following KKT equations for that system must hold:

$$\frac{d}{d\mathbf{x}} \begin{pmatrix} \operatorname{tr}(\mathbf{G}(\tilde{\mathbf{x}}_{11})\mathbf{Y}_{tr}^{-1}) \\ -\operatorname{logdet}(\mathbf{G}(\tilde{\mathbf{x}}_{11})) \\ +\operatorname{tr}(\tilde{\boldsymbol{\lambda}}_{11}(\mathbf{G}(\tilde{\mathbf{x}}_{11}) - \mathbf{Y}_{tr})) \end{pmatrix} = \mathcal{G}_{tr}^{-1} - \mathcal{G}_{sp}^{-1} + \mathcal{G}_{\lambda} \qquad = \mathbf{0}. \tag{A.30}$$

$$\begin{aligned} \frac{d}{d\mathbf{x}}\left(\tilde{\boldsymbol{\lambda}}_{11}\right) &= \mathbf{F}_{\lambda} \\ &= -\mathbf{F}_{tr}^{-1} + \mathbf{F}_{sp}^{-1} \end{aligned} \tag{A.31}$$

$$\mathcal{G}_{tr}^{-1} = \begin{cases} (\tilde{\mathbf{Y}}_{tr}^{-1})_{ij} & \forall\{(i,j) \in \mathcal{K}_{11}\} \\ 0, & \text{otherwise} \end{cases} \tag{A.32}$$

$$\mathbf{G}_{sp}^{-1} = \begin{cases} (\mathbf{G}(\tilde{\mathbf{x}}_{11})^{-1})_{ij} & \forall\{(i,j) \in \mathcal{K}_{11}\} \\ 0, & \text{otherwise} \end{cases} \tag{A.33}$$

$$\mathcal{G}_{\lambda} = \begin{cases} (\tilde{\boldsymbol{\lambda}}_{11})_{ij} & \forall\{(i,j) \in \mathcal{K}_{11}\} \\ 0, & \text{otherwise}\,. \end{cases} \tag{A.34}$$

Where,

$$\mathcal{G}_{tr}^{-1} - \mathcal{G}_{sp}^{-1} + \mathcal{G}_{\lambda} = \mathbf{0} \tag{A.35}$$

$$\tilde{\boldsymbol{\lambda}}_{11} \succeq \mathbf{0} \tag{A.36}$$

$$\operatorname{tr}(\tilde{\boldsymbol{\lambda}}_{11}(\mathbf{G}(\tilde{\mathbf{x}}_{11}) - \tilde{\mathbf{Y}}_{tr})) = \mathbf{0} \tag{A.37}$$

$$\mathbf{Y}_{tr} - \mathbf{G}(\tilde{\mathbf{x}}_{11}) \succeq \mathbf{0}. \tag{A.38}$$

Also, a solution to the reduced problem must lie in its domain. That is, the following must be true,

$$\mathbf{G}(\tilde{\mathbf{x}}_{11}) \succ \mathbf{0}. \tag{A.39}$$

Show that $F^*(\tilde{\mathbf{x}}) \succ \mathbf{0}$*:*
Using partition in Eqn. A.28 and since $\mathbf{G}(\tilde{\mathbf{x}}_{11}) \succ \mathbf{0}$ and $\mathbf{A}_{22}^{-1} \succ \mathbf{0}$ then,

$$\mathbf{G}(\tilde{\mathbf{x}}_{11}) + \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{12}^T \succ \mathbf{0}. \tag{A.40}$$

now,

$$\mathbf{G}(\tilde{\mathbf{x}}_{11}) + \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{12}^T - \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{12}^T = \mathbf{G}(\tilde{\mathbf{x}}_{11}) \succ \mathbf{0}, \tag{A.41}$$

and using $\mathbf{A}_{22} \succ \mathbf{0}$ implies $F^*(\tilde{\mathbf{x}}) \succ \mathbf{0}$.

Show KKT Condition (Equation A.20):

$$\mathbf{F}_{tr}^{-1} - \mathbf{F}_{sp}^{-1} + \mathbf{F}_{\lambda} = \mathbf{F}_{tr}^{-1} - \mathbf{F}_{sp}^{-1} + (-\mathbf{F}_{tr}^{-1} + \mathbf{F}_{sp}^{-1}) \tag{A.42}$$

$$= \mathbf{0}, \tag{A.43}$$

where the above has used Eqn. A.29

Show KKT Condition (Equation A.21):

$$\mathbf{Y}_{tr} - F^*(\tilde{\mathbf{x}}) = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12}^T \\ \mathbf{A}_{12} & \mathbf{A}_{22} \end{bmatrix} - \begin{bmatrix} \mathbf{G}(\tilde{\mathbf{x}}_{11}) + \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{12}^T & \mathbf{A}_{12} \\ \mathbf{A}_{12}^T & \mathbf{A}_{22} \end{bmatrix} \tag{A.44}$$

$$= \begin{bmatrix} \mathbf{A}_{11} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} - \begin{bmatrix} \mathbf{G}(\tilde{\mathbf{x}}_{11}) + \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{12}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{A.45}$$

$$= \mathbf{A}_{11} - \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{12}^T - \mathbf{G}(\tilde{\mathbf{x}}_{11}) \tag{A.46}$$

$$= \tilde{\mathbf{Y}}_{tr} - \mathbf{G}(\tilde{\mathbf{x}}_{11}) \tag{A.47}$$

$$\succeq \mathbf{0}. \tag{A.48}$$

Show KKT Condition (Equation A.22):
Note that,

$$\mathbf{Y}_{tr} - F^*(\tilde{\mathbf{x}}) \succeq \mathbf{0} \iff F^*(\tilde{\mathbf{x}})^{-1} - \mathbf{Y}_{tr}^{-1} \succeq \mathbf{0}, \tag{A.49}$$

therefore,

$$\boldsymbol{\lambda}^* = F^*(\tilde{\mathbf{x}})^{-1} - \mathbf{Y}_{tr}^{-1} \tag{A.50}$$

$$\succeq \mathbf{0}. \tag{A.51}$$

Show KKT Condition (Equation A.23)

$$\operatorname{tr}\left(\boldsymbol{\lambda}^* \left(F^*(\tilde{\mathbf{x}}) - \mathbf{Y}_{tr}\right)\right) = \operatorname{tr}\left(\tilde{\boldsymbol{\lambda}}_{11}\left(\mathbf{G}(\tilde{\mathbf{x}}_{11}) + \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{12}^T - \mathbf{A}_{11}\right)\right) \tag{A.52}$$

$$= \operatorname{tr}\left(\tilde{\boldsymbol{\lambda}}_{11}(\mathbf{G}(\tilde{\mathbf{x}}_{11}) - \tilde{\mathbf{Y}_{tr}})\right) \tag{A.53}$$

$$= \mathbf{0} \tag{A.54}$$

Where $\boldsymbol{\lambda}_{11}$ is the top left block of $\boldsymbol{\lambda}$. By re-arranging Eqn. A.35 we have

$$\mathcal{G}_\lambda = \mathcal{G}_{sp}^{-1} - \mathcal{G}_{tr}^{-1}.$$

Decomposing $\boldsymbol{\lambda}_{11}$:

$$\boldsymbol{\lambda}_{11} = \begin{cases} (\mathcal{G}_\lambda)_{ij} & \forall\{(i,j) \in \mathcal{J}_{11} \cup \mathcal{K}_{11}\} \\ (\boldsymbol{\lambda}_0)_{ij} & \forall\{(i,j) \in \mathcal{L}\}, \end{cases}$$

where,

$$\boldsymbol{\lambda}_0 = \begin{cases} (\boldsymbol{\lambda}^*)_{ij} & \forall\{(i,j) \in \mathcal{L}\} \\ \mathbf{0} & \text{otherwise}\,. \end{cases} \tag{A.55}$$

Also, note that

$$(F^*(\tilde{\mathbf{x}}) - \mathbf{Y}_{tr})_{ij} = 0, \forall\{(i,j) \in \mathcal{L}\}. \tag{A.56}$$

Now, Eqn A.53 becomes:

$$\begin{aligned} \text{tr}(\boldsymbol{\lambda}^*(F^*(\tilde{\mathbf{x}})\mathbf{Y}_{tr}) &= \text{tr}((\boldsymbol{\lambda}_0 + \mathcal{G}_\lambda)(\mathbf{G}(\tilde{\mathbf{x}}_{11}) - \tilde{\mathbf{Y}}_{tr})) && \text{(A.57)} \\ &= \text{tr}\,(\boldsymbol{\lambda}_0(F^*(\tilde{\mathbf{x}})\mathbf{Y}_{tr})) + \text{tr}(\mathcal{G}_\lambda(\mathbf{G}(\tilde{\mathbf{x}}_{11}) - \tilde{\mathbf{Y}}_{tr})) && \text{(A.58)} \\ &= \text{tr}\left(\boldsymbol{\lambda}_0(\mathbf{G}(\tilde{\mathbf{x}}_{11}) - \tilde{\mathbf{Y}}_{tr})\right) && \text{(A.59)} \end{aligned}$$

where we have used Eqn: A.37. Finally using equations A.56, A.55 and A.27 we have,

$$\begin{aligned} (\mathbf{G}(\tilde{\mathbf{x}}_{11}) - \tilde{\mathbf{Y}}_{tr})_{ij} &= (\mathcal{G}_0 - \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{12}^T)_{ij} && \text{(A.60)} \\ &= (\mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{12}^T - \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{12}^T)_{ij} && \text{(A.61)} \\ &= 0 && \text{(A.62)} \end{aligned}$$

where $(i,j) \in \mathcal{L}$. Which implies:

$$\text{tr}\left(\boldsymbol{\lambda}_0(\mathbf{G}(\tilde{\mathbf{x}}_{11}) - \tilde{\mathbf{Y}}_{tr})\right) = \mathbf{0}, \tag{A.63}$$

therefore,

$$\text{tr}(\boldsymbol{\lambda}^*(F^*(\tilde{\mathbf{x}}) - \mathbf{Y}_{tr}) = \mathbf{0}. \tag{A.64}$$

Since the KKT equations are satisfied, the proposed solution is optimal. □

The final part of the proof used the fact that we made the matrix $\mathbf{G}(\tilde{\mathbf{x}}_{11})$ contain the marginals, and since it was only for fill in for which we had no expression derived from the reduced solution, this made the fill in entries correspond with zeros in the right expression. Thus allowing our result.

Both the dual and primal solutions are as hard as each other to solve for.

Theorem 2: If the Markov blanket is wholly included in one of the sets of a CS Graph cut, then the associated reduced problem will always have a feasible solution.

Proof Outline: This proof first describes the inequalities which can result in an infeasible solution. It then describes a choice of the reduction which will always result in a feasible solution. A problem is known as infeasible if there are no solutions which satisfy the equality and inequality constraints.

Proof. We denote a solution to the reduced problem by, $\mathbf{G}(\tilde{\mathbf{x}}_{11}) = \tilde{\mathbf{A}}_{11}$. The inequality for the reduced problem corresponding to the consistency constraint is:

$$\mathbf{A}_{11} - \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{12}^{T} - \tilde{\mathbf{A}}_{11} \succeq \mathbf{0} \tag{A.65}$$

$$\mathbf{A}_{11} - \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{12}^{T} \succeq \tilde{\mathbf{A}}_{11} \succ \mathbf{0}. \tag{A.66}$$

The subtraction of the term $(\mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{12}^{T})$ reduces the upper bound on the solution $\tilde{\mathbf{A}}_{11}$, if $\mathcal{I} \cap \mathcal{L} \neq \emptyset$, then it is possible that the problem will have no solution (i.e. infeasible).

Exploiting Structure: Consider a matrix $\mathbf{Y}_{tr}$ with the following structure:

$$\mathbf{Y}_{tr} = \begin{bmatrix} \mathbf{B}_{11} & \mathbf{B}_{12} & \mathbf{0} \\ \mathbf{B}_{12}^{T} & \mathbf{B}_{22} & \mathbf{B}_{23} \\ \mathbf{0} & \mathbf{B}_{23}^{T} & \mathbf{B}_{33} \end{bmatrix}. \tag{A.67}$$

If we choose to reduce the problem so that $\tilde{\mathbf{A}}_{11}$ encompasses $\mathbf{B}_{11}$ and $\mathbf{B}_{22}$ (the Markov blanket of the equality constraints contained in $\mathbf{B}_{11}$), then the above inequality becomes:

$$\begin{bmatrix} \mathbf{B}_{11} & \mathbf{B}_{12} \\ \mathbf{B}_{12}^T & \mathbf{B}_{22} - \mathbf{B}_{23}\mathbf{B}_{33}^{-1}\mathbf{B}_{23}^T \end{bmatrix} - \begin{bmatrix} \tilde{\mathbf{B}}_{11} & \tilde{\mathbf{B}}_{12} \\ \tilde{\mathbf{B}}_{12}^T & \tilde{\mathbf{B}}_{22} \end{bmatrix} \succeq \mathbf{0} \tag{A.68}$$

If this structure exists for a particular reduction choice, then this problem will always have a feasible solution as long as $\mathcal{I}$ is contained in $\mathbf{B}_{11}$, and $\mathcal{L}$ is contained in $\mathbf{B}_{22}$. A feasible solution is demonstrated below (where ϵ satisfies the $\mathbf{B}_{11} \succ \epsilon\mathbf{I}$)

$$\tilde{\mathbf{A}}_{11} = \begin{bmatrix} \epsilon\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_{23}\mathbf{B}_{33}^{-1}\mathbf{B}_{23}^T \end{bmatrix}. \tag{A.69}$$

This satisfies the equality and inequality constraints, which is demonstrated below:

$$\begin{bmatrix} \mathbf{B}_{11} & \mathbf{B}_{12} \\ \mathbf{B}_{12}^T & \mathbf{B}_{22} - \mathbf{B}_{23}\mathbf{B}_{33}^{-1}\mathbf{B}_{23}^T \end{bmatrix} - \begin{bmatrix} \epsilon\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_{23}\mathbf{B}_{33}^{-1}\mathbf{B}_{23}^T \end{bmatrix} \succeq \mathbf{0} \tag{A.70}$$

$$\begin{bmatrix} \epsilon\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_{23}\mathbf{B}_{33}^{-1}\mathbf{B}_{23}^T \end{bmatrix} \succ \mathbf{0}. \tag{A.71}$$

Note that we cannot choose $\tilde{\mathbf{B}}_{22} = \epsilon\mathbf{I}$ as the equality constraints associated with $\mathcal{L}$ require off diagonal terms to be non-zero.

Therefore, the reduced problem will always have a feasible solution if the Markov blanket is included in the reduced problem.

□

Appendix B

Details of Previous Work in Sparse Approximations

B.1 Approximate Sparsification for Data Fusion

This section describes two important approaches for sparsification in data fusion, Covariance Intersection (CI) and Approximately Marginalised - Sparse Extended Information Filter (AM-SEIF). CI is a commonly used approach for fusion of datums whose correlation is unknown, and guarantees conservatism in its result. AM-SEIF differs from CI in that it was developed to solve the SLAM problem, however its solution is sufficiently general to be applicable to general estimation problems.

B.1.1 Covariance Intersection and Related Approaches

A common problem in data fusion is maintaining records of cross-correlation terms between states. Standard approaches that use the Kalman filter [26] require knowledge of the cross correlation terms to produce estimates that are consistent.

The CI family of approaches use a simple geometric property to guarantee consistency and conservatism. There are three key members of the CI family, CI, Covariance Union (CU) and split Covariance Intersection (split-CI).

CI is a method for fusing two Gaussian distributions when the cross-correlation between the two is unknown. Let the two distributions have means $\bar{\mathbf{a}}$, $\bar{\mathbf{b}}$ and covariances $\mathbf{A}$, $\mathbf{B}$ for the distributions $\mathcal{N}_A$ and $\mathcal{N}_B$, respectively. Let the resultant distribution have mean $\bar{\mathbf{c}}$ and covariance $\mathbf{C}$.

The fundamental equation for covariance intersection is then [30]:

$$\mathbf{C}^{-1} = w\mathbf{A}^{-1} + (1-w)\mathbf{B}^{-1}, \tag{B.1}$$

$$\mathbf{C}^{-1}\bar{\mathbf{c}} = w\mathbf{A}^{-1}\bar{\mathbf{a}} + (1-w)\mathbf{B}^{-1}\bar{\mathbf{b}}, \tag{B.2}$$

where $w \in [0, 1]$. The parameter w can be optimised to reduce some cost, typically the trace or determinant of the resulting system.

The CU approach attempts to solve the problem of conflicting observations [58]. That is, two observations that purport to be of the same thing but predict very different results. The algorithm assumes that one of the estimates must be true but cannot know which one. CU creates a resultant distribution that encompasses both distributions. The problem is formed as either a Maximum Determinant (MAXDET) problem or a Semi-Definite Program (SDP):

$$\begin{aligned} &\min_{\mathbf{U}\in\mathcal{S}^n, \mathbf{u}\in\mathbb{R}^n} F(\mathbf{U}) \\ &\quad \text{s.t.} \\ &\qquad \mathbf{U} \succeq \mathbf{A} + (\mathbf{u}-\mathbf{a})(\mathbf{u}-\mathbf{a})^T \\ &\qquad \mathbf{U} \succeq \mathbf{B} + (\mathbf{u}-\mathbf{b})(\mathbf{u}-\mathbf{b})^T \end{aligned} \tag{B.3}$$

where F is either the negative log determinant (for MAXDET) or trace (for SDPs).

Split-CI is an extension to Covariance Intersection that has also been used in SLAM to increase the accuracy of results [31]. Split-CI allows the incorporation of observations that are known or assumed to be uncorrelated. That is, the data fusion is split into independent and dependent parts [29]. Let $\mathbf{A} = \frac{1}{w}\mathbf{A}_1 + \mathbf{A}_2$, and $\mathbf{B} = \frac{1}{1-w}\mathbf{B}_1 + \mathbf{B}_2$, then split-CI is:

$$\mathbf{C} = \left[\mathbf{A}^{-1} + \mathbf{B}^{-1}\right]^{-1} = \left[\left(\frac{1}{w}\mathbf{A}_1 + \mathbf{A}_2\right)^{-1} + \left(\frac{1}{1-w}\mathbf{B}_1 + \mathbf{B}_2\right)^{-1}\right]^{-1} \quad \text{(B.4)}$$

$$\bar{\mathbf{c}} = \left[\mathbf{A}^{-1}\bar{\mathbf{a}} + \mathbf{B}^{-1}\bar{\mathbf{b}}\right]^{-1} = \mathbf{C}\left[\left(\frac{1}{w}\mathbf{A}_1 + \mathbf{A}_2\right)^{-1}\bar{\mathbf{a}} + \left(\frac{1}{1-w}\mathbf{B}_1 + \mathbf{B}_2\right)^{-1}\bar{\mathbf{b}}\right]. \quad \text{(B.5)}$$

B.1.1.1 Limitations of Covariance Intersection Approaches

CI is very useful for guaranteeing conservative data fusion when two pieces of data need to be fused. However, it is not possible to introduce an arbitrary sparsity pattern into an existing estimate using CI based approaches. To illustrate this, consider the following information matrix ($\mathbf{Y}$) and its desired sparsity ($\tilde{\mathbf{Y}}$) :

$$\mathbf{Y} = \begin{bmatrix} a & b & d \\ b & c & e \\ d & e & f \end{bmatrix} \rightarrow \tilde{\mathbf{Y}} = \begin{bmatrix} a & b & 0 \\ b & c & e \\ 0 & e & f \end{bmatrix} \quad \text{(B.6)}$$

CI cannot in general be used to transform $\mathbf{Y}$ to $\tilde{\mathbf{Y}}$. However, CI could be used to perform this sparsification if all the updates that included non-zero information about d were known.

Although CI is a conservative approach to sparsification, its assumption that the correlations between states are unknown is not used in this thesis. Thus comparisons with CI are flawed as CI will always perform worse than other approaches because it makes no use of the known correlations.

B.1.2 Approximately Marginalised Sparse Filtering

Approximately Marginalised - Sparse Extended Information Filters (AM-SEIFs) [54] is a technique for removing conditional dependencies in a way that provides a close approximation to the original distribution. It was designed for use in SLAM scenarios to disconnect robot states from the set of stationary states (landmarks). Notably,

this approach allows for constant time filtering prediction steps, although it produces inconsistent estimates [19].

AM-SEIF induces sparsity through combining marginal distributions to form a new approximate distribution with the desired sparsity. This requires the system to be divided into four sets. The robot state will be denoted $\mathbf{x}$, while the states that are to be disconnected from $\mathbf{x}$ are denoted as $\mathbf{y}^0$. Let $\mathbf{y}^+$ indicate the states that are to remain connected to $\mathbf{x}$, and $\mathbf{y}^-$ the states that are not connected to $\mathbf{x}$, that is $\mathbf{x} \perp\!\!\!\perp \mathbf{y}^-$.

Now consider the system:

$$p(\mathbf{x}, \mathbf{y}^+, \mathbf{y}^-, \mathbf{y}^0) = p(\mathbf{y}^-|\mathbf{y}^+, \mathbf{y}^0)p(\mathbf{x}|\mathbf{y}^+, \mathbf{y}^0)p(\mathbf{y}^+, \mathbf{y}^0) \tag{B.7}$$

To induce sparsity or conditional independance between $\mathbf{x}$ and $\mathbf{y}^0$, the new factorisation should be of the form:

$$p(\mathbf{x}, \mathbf{y}^+, \mathbf{y}^-, \mathbf{y}^0) = p(\mathbf{x}|\mathbf{y}^+, \mathbf{y}^-)p(\mathbf{y}^0|\mathbf{y}^+, \mathbf{y}^-)p(\mathbf{y}^+, \mathbf{y}^-) \tag{B.8}$$

This can also be expressed as:

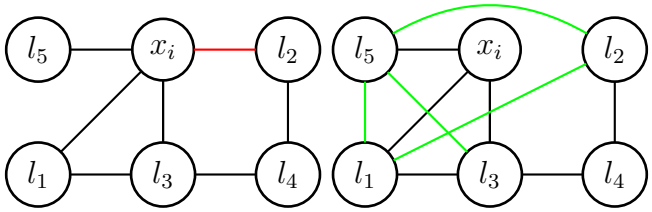
$$p(\mathbf{x}, \mathbf{y}^+, \mathbf{y}^-, \mathbf{y}^0) = \frac{p(\mathbf{x}, \mathbf{y}^+|\mathbf{y}^-)}{p(\mathbf{y}^+|\mathbf{y}^-)}p(\mathbf{y}^+, \mathbf{y}^0, \mathbf{y}^-) \tag{B.9}$$

Each of the probability distributions in equation B.9 can be formed by appropriate marginalisation of the original distribution. Thus, the approximation requires forming the marginals and combining them as in equation B.9. Let the information matrix for the distribution $p(\mathbf{x}, \mathbf{y}^+|\mathbf{y}^-)$, $p(\mathbf{y}^+|\mathbf{y}^-)$ and $p(\mathbf{y}^+, \mathbf{y}^-, \mathbf{y}^0)$ be $\mathbf{M}_1$, $\mathbf{M}_2$ and $\mathbf{M}_3$ respectively.

These matricies are combined to form the approximate system.

$$\mathbf{Y} = \mathbf{M}_1 - \mathbf{M}_2 + \mathbf{M}_3. \tag{B.10}$$

Although AM-SEIF is tailored to a SLAM filtering scenario, the fundamental equations are not specific to the SLAM problem, and can be used for more general Gaussian estimation problems where sparsity is desired.



(a) Original graph **(b)** Sparsified graph

Figure B.1 – Sparsifying link between x_i and l_2 using AM-SEIF approach (shown in red), this link was induced through marginalisation of a prior state. Green links are induced by AM-SEIF. The landmark sets are: $\mathbf{Y}^0$ which contains landmark l_3; $\mathbf{Y}^-$ which contains l_2 and set Y^+ which contains the active landmarks l_1, l_4 and l_5. Fill-in has occurred between the sparsified set $\mathbf{Y}^-$ and the active set $\mathbf{Y}^+$ but not the deactive set $\mathbf{Y}^0$. After this operation, the robot state has 3 rather than 4 edges, and is in this sense more sparse, although the overall graph sparsity has been increased.

B.1.2.1 Link removal heuristic

AM-SEIFs link removal heuristic is used when the number of links connected to the robot pose reaches a predetermined threshold. AM-SEIF then removes links to ensure that the number of connections to the robot pose is reduced below the threshold. Weaker links are chosen for deletion first. This approach tends to disconnect landmarks that are further away from the robot [54]. Link strength is determined by its normalised weight. The weight for a link a_{ij} is:

$$w_{ij} = \frac{a_{ij}^2}{a_{ii}a_{jj}}. \tag{B.11}$$

w_{ij} will range from [0,1], since a property of positive definite matrices is that an off-diagonal must be less than or equal to its diagonal [51, Eq 10.34 (a)].

Example B.1.1. *This example will demonstrate the AM-SEIF sparsification process on a matrix. Consider the system below:*

$$\mathbf{Y} = \begin{bmatrix} \mathbf{A} & \mathbf{B}^T & \mathbf{C}^T & \mathbf{0} \\ \mathbf{B} & \mathbf{D} & \mathbf{E}^T & \mathbf{F}^T \\ \mathbf{C} & \mathbf{E} & \mathbf{G} & \mathbf{H}^T \\ \mathbf{0} & \mathbf{F} & \mathbf{H} & \mathbf{I} \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} \tag{B.12}$$

It is desired to remove the dependence between y_1 and y_2, which is equivalent to setting the matrix $\mathbf{B} = \mathbf{0}$. To apply AM-SEIF to $\mathbf{y}$ form information matrices of the probabilities in equation B.9.

$$p(\mathbf{x}, \mathbf{y}^+|\mathbf{y}^-) = \mathbf{M}_1 = \begin{bmatrix} \mathbf{A} & \mathbf{0} & \mathbf{C}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{C} & \mathbf{0} & \mathbf{G} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} - \begin{bmatrix} \mathbf{B}\mathbf{D}^{-1}\mathbf{B}^T & \mathbf{0} & \mathbf{B}\mathbf{D}^{-1}\mathbf{E}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{E}\mathbf{D}^{-1}\mathbf{B}^T & \mathbf{0} & \mathbf{E}\mathbf{D}^{-1}\mathbf{E}^T & \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{B.13}$$

$$p(\mathbf{y}^+|\mathbf{y}^-) = \mathbf{M}_2 = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{G} - \mathbf{L} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{B.14}$$

Where $\mathbf{L} = \begin{bmatrix} \mathbf{C} & \mathbf{E} \end{bmatrix} \begin{bmatrix} \mathbf{A} & \mathbf{B}^T \\ \mathbf{B} & \mathbf{D} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{C}^T \\ \mathbf{E}^T \end{bmatrix}$,

$$p(\mathbf{y}^+, \mathbf{y}^0, \mathbf{y}^-) = \mathbf{M}_3 = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{D} - \mathbf{B}\mathbf{A}^{-1}\mathbf{B}^T & \mathbf{E}^T - \mathbf{B}\mathbf{A}^{-1}\mathbf{C}^T & \mathbf{F}^T \\ \mathbf{0} & \mathbf{E} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B}^T & \mathbf{G} - \mathbf{C}\mathbf{A}^{-1}\mathbf{C}^T & \mathbf{H}^T \\ \mathbf{0} & \mathbf{F} & \mathbf{H} & \mathbf{I} \end{bmatrix}, \tag{B.15}$$

So combining these as in equation B.9 $\tilde{\mathbf{y}} = \mathbf{M}_1 - \mathbf{M}_2 + \mathbf{M}_3$ *:*

$$\tilde{\mathbf{y}} = \begin{bmatrix} \mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{B}^T & \mathbf{0} & \mathbf{C}^T - \mathbf{B}\mathbf{D}^{-1}\mathbf{E}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{D} - \mathbf{B}\mathbf{A}^{-1}\mathbf{B}^T & \mathbf{E}^T - \mathbf{B}\mathbf{A}^{-1}\mathbf{C}^T & \mathbf{F}^T \\ \mathbf{C} - \mathbf{E}\mathbf{D}^{-1}\mathbf{B}^T & \mathbf{E} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B}^T & \mathbf{G} - \mathbf{C}\mathbf{A}^{-1}\mathbf{C}^T - \mathbf{E}\mathbf{D}^{-1}\mathbf{E}^T + \mathbf{L} & \mathbf{H}^T \\ \mathbf{0} & \mathbf{F} & \mathbf{H} & \mathbf{I} \end{bmatrix}, \tag{B.16}$$

Clearly AM-SEIF has induced the desired sparsity, however the presence of many Schur complement complement fill terms will result in significant fill-in.

The AM-SEIF approach is not a conservative approach to sparsification. The following proposition was first proved in [19] using covariance matrices. What follows is a similar approach, instead focusing on information matrices.

Proposition 1. *The AM-SEIF approach to sparsification is not conservative.*

Proof. Consider the conservatism constraint using the matrix decomposition from example B.1.1.

$$\mathbf{Y} - \tilde{\mathbf{Y}} = \begin{bmatrix} \mathbf{B}\mathbf{D}^{-1}\mathbf{B}^T & \mathbf{B} & \mathbf{B}\mathbf{D}^{-1}\mathbf{E}^T & \mathbf{0} \\ \mathbf{B} & \mathbf{B}\mathbf{A}^{-1}\mathbf{B}^T & \mathbf{B}\mathbf{A}^{-1}\mathbf{C}^T & \mathbf{0} \\ \mathbf{E}\mathbf{D}^{-1}\mathbf{B}^T & \mathbf{C}\mathbf{A}^{-1}\mathbf{B}^T & \mathbf{C}\mathbf{A}^{-1}\mathbf{C}^T + \mathbf{E}\mathbf{D}^{-1}\mathbf{E}^T - \mathbf{L} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}. \tag{B.17}$$

Examine the diagonal term of the third row:

$$\mathbf{C}\mathbf{A}^{-1}\mathbf{C}^T + \mathbf{E}\mathbf{D}^{-1}\mathbf{E}^T - \begin{bmatrix} \mathbf{C} & \mathbf{E} \end{bmatrix} \begin{bmatrix} \mathbf{A} & \mathbf{B}^T \\ \mathbf{B} & \mathbf{D} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{C}^T \\ \mathbf{E}^T \end{bmatrix}, \tag{B.18}$$

This can be re-written as:

$$\begin{bmatrix} \mathbf{C} & \mathbf{E} \end{bmatrix} \left(\begin{bmatrix} \mathbf{A}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}^{-1} \end{bmatrix} - \begin{bmatrix} \mathbf{A} & \mathbf{B}^T \\ \mathbf{B} & \mathbf{D} \end{bmatrix}^{-1} \right) \begin{bmatrix} \mathbf{C}^T \\ \mathbf{E}^T \end{bmatrix}, \tag{B.19}$$

In order to be positive definite, all the diagonals must be positive and all diagonal sub-blocks must be positive definite. The following shows that one of the diagonal sub-blocks is not positive definite.

Consider the top left sub-block of the inner term, and using the Woodman-Morrisey matrix inversion lemma [51, Eq. 15.3(c)]:

$$\mathbf{A}^{-1} - \left(\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{B}^T\right)^{-1} = \mathbf{A}^{-1} - \left(\mathbf{A}^{-1} + \mathbf{K}\right) \tag{B.20}$$

$$= -\mathbf{K} \tag{B.21}$$

where $\mathbf{K} = \mathbf{A}^{-1}\mathbf{B}(\mathbf{D} - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{B}^T\mathbf{A}^{-1}$. Therefore if $\mathbf{K}$ is positive definite, then equation B.18 will not be positive definite. All that remains to be shown is that $\mathbf{K}$ is positive definite. Assume that $\mathbf{Y} \succ \mathbf{0}$, this implies $\mathbf{D} - \mathbf{B}\mathbf{A}^{-1}\mathbf{B}^T \succ \mathbf{0}$, using [51, Eq. 10.46 (a)] then implies $\mathbf{K} \succeq \mathbf{0}$, which implies that $-\mathbf{K} \preceq \mathbf{0}$. Therefore

$$\mathbf{Y} - \tilde{\mathbf{Y}} \not\succeq \mathbf{0}. \tag{B.22}$$

□

The fill-in properties of AM-SEIF are described by the graph in Figure B.1. It can be seen that AM-SEIF will induce fill-in between the disconnected states and the currently connected states. It will also induce internal fill-in in the 'robot' states, as

well as internal AM-SEIF fill-in for the connected landmarks. However, AM-SEIF will not induce any fill-in between any state not connected to the robot pose.

AM-SEIFs property of inducing internal fill-in in the connected states is suitable for a filtering system, where marginalisation can often induce this fill-in anyway. However, if the state being separated is not going to be marginalised, then this extra fill-in becomes an undesirable property of the approach, this is demonstrated in Figure B.1.

B.1.3 Approximate Sparsification for SLAM

The motivating example for these approaches will be the SLAM problem.

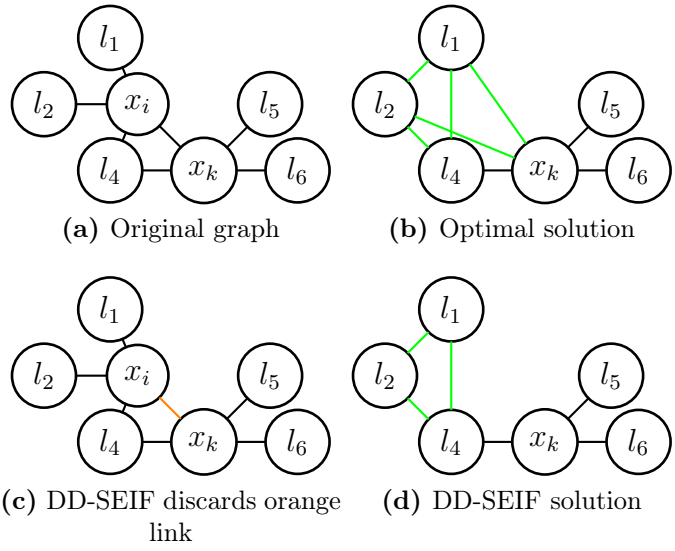
B.1.4 Short overview of SLAM

The SLAM problem involves the concurrent localisation of a robot in an unknown environment. Typically states are split into robot states ($\mathbf{x}$), and landmarks. Many different approaches to SLAM exist, this thesis will consider only linear SLAM with known data association. That is, the robot movement model and observation models are linear and correspondences between observations and landmarks is known.

B.1.4.1 DD-SEIF

The Data Discarding Sparse Extended Information Filters (DD-SEIF) approach ([61]) uses the insight that by selectively choosing to discard odometry data, as used by the vehicle's prediction step, the sparsity of the graph can be improved. The algorithm also requires the robot to re-observe a landmark in order to reinitialise the robot pose.

In the case of the filtering SLAM problem (Section 2.4.1), DD-SEIF chooses to discard odometry whenever the number of links to the robot state rises above a predefined threshold. This motion observation is then discarded, the previous robot pose is marginalised into the landmarks, and the robot pose is reinitialised through a landmark re-observation. An example of this process is given in Figure B.2.



(a) Original graph **(b)** Optimal solution

(c) DD-SEIF discards orange link **(d)** DD-SEIF solution

Figure B.2 – The optimal approach is shown in (a) and (b), while the DD-SEIF approach is shown in (c) and (d). In this example the robot predicts its state forward and then observes two new landmarks (l_5 and l_6) and a previously surveyed landmark, l_4. The graph in (b) shows the network after the state x_i has been marginalised and has formed a large clique. In (c) the DD-SEIF algorithm chooses to discard the motion update (represented by an orange link). In (d) marginalising x_i landmarks l_1 and l_2 are not connected to state x_k. DD-SEIF has induced sparsity by ignoring a motion observation.

B.1.4.2 Laser Based Pose Graph Sparsification

A recent approach to sparsification is presented by H. Kretzschmar et. al [37]. In this work a map is constructed using robot poses and laser scans. Each scan is associated with a particular robot pose, and each pose is correlated to other poses using laser based features seen in the environment. The key insight in this work is that many laser scans may overlap, and therefore not all laser scans are necessary to create a representative map of the environment.

The technique begins by finding a pose whose mutual information with all other poses is large. This pose is then marginalised out of the graph, inducing fill-in between its neighbours. A Chow-Liu [12] maximum spanning tree is then constructed over the markov blanket of the marginalised node. The tree edges are calculated by computing

each weights mutual information, using the following formula c.f. [37]:

$$I(x_i; x_j) = \frac{1}{2}\left(\text{logdet}(\Sigma_{ii}) - \text{logdet}(\Sigma_{ii} - \Sigma_{ij}\Sigma_{jj}^{-1}\Sigma_{ji})\right), \tag{B.23}$$

where Σ_{ij} is the covariance matrix between the states i and j.

This approach is not a conservative estimate as the Chow-Liu spanning tree does not guarantee conservatism, rather only connectivity [12].

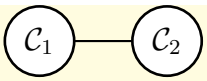
B.2 Previous Work in Sparsification for Junction Trees

B.2.1 Thin Junction Trees (TJT)

Junction Trees have been used in approximate estimation [49], and applied to the SLAM problem in [48]. Although the system is represented as a JT, the approach to sparsification is similar to AM-SEIF in that it involves a combination of marginals to form a new approximate distribution.

This approach forms a JT over the estimation space and then identifies large cliques, which are then 'thinned' to increase the efficiency of the underlying JT message passing algorithm. The thinning operation is performed by marginalising a state from a particular clique, but not from the entire network, example B.2.1 demonstrates this using the Shafer-Shenoy message passing algorithm. Thin Junction Tree (TJT) uses the Hugin message passing scheme, Example B.2.1 is repeated using Hugin message passing in Example B.2.2.

Example B.2.1. *This example demonstrates the clique contraction technique. Consider the following junction tree ($\mathcal{C}_1 = \{1, 2, 3\}, \mathcal{C}_2 = \{2, 3, 4\}$, assume message passing has already occurred):*



where the potentials are:

$$\mathbf{Y}_{\Psi_1} = \begin{bmatrix} \mathbf{A} & \mathbf{B}^T & \mathbf{C}^T \\ \mathbf{B} & \mathbf{D} & \mathbf{E}^T \\ \mathbf{C} & \mathbf{E} & \mathbf{G} \end{bmatrix}, \quad \mathbf{y}_{\Psi_1} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}, \tag{B.24}$$

$$\mathbf{Y}_{\Psi_2} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{F}^T \\ \mathbf{0} & \mathbf{0} & \mathbf{H}^T \\ \mathbf{F} & \mathbf{H} & \mathbf{I} \end{bmatrix}, \quad \mathbf{y}_{\Psi_2} = \begin{bmatrix} 0 \\ 0 \\ y_4 \end{bmatrix}. \tag{B.25}$$

It is desired to thin the tree by removing state 3 from $\mathcal{C}_2$. The first step is to form the conditional distribution for all the states that are neighbours of state 3 in Ψ_2.

$$\mathbf{Y}_c = \begin{bmatrix} \mathbf{D} & \mathbf{E}^T & \mathbf{F}^T \\ \mathbf{E} & \mathbf{G} & \mathbf{H}^T \\ \mathbf{F} & \mathbf{H} & \mathbf{I} \end{bmatrix}, \quad \mathbf{y}_c = \begin{bmatrix} y_2 \\ y_3 \\ y_4 \end{bmatrix}, \tag{B.26}$$

The marginals $\mathbf{Y}_c$/ $\mathbf{Y}_c^{*3}$ and $\mathbf{y}_c$/ $\mathbf{y}_c^*$ are calculated, and the change in $\mathbf{Y}_c$ and $\mathbf{y}_c$ results in the following system (however any change in information related to a state that is in a separator of $\mathcal{C}_2$ is nullified):

$$\mathbf{Y}_{\Delta_c} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & -\mathbf{E}\mathbf{G}^{-1}\mathbf{H}^T \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ -\mathbf{H}\mathbf{G}^{-1}\mathbf{E}^T & \mathbf{0} & -\mathbf{H}\mathbf{G}^{-1}\mathbf{H}^T \end{bmatrix}, \tag{B.27}$$

$$\mathbf{y}_{\Delta_c} = \begin{bmatrix} 0 \\ 0 \\ -\mathbf{H}\mathbf{G}^{-1}y_3 \end{bmatrix} \tag{B.28}$$

The information is then redistributed such that:

$$\mathbf{Y}_{\Psi_1} = \begin{bmatrix} \mathbf{A} & \mathbf{B}^T & \mathbf{C}^T \\ \mathbf{B} & \mathbf{D} & \mathbf{E}^T \\ \mathbf{C} & \mathbf{E} & \mathbf{G} \end{bmatrix}, \quad \mathbf{y}_{\Psi_1} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}, \tag{B.29}$$

$$\mathbf{Y}^{app}_{\Psi_2} = \begin{bmatrix} \mathbf{0} & \mathbf{F}^T - \mathbf{E}\mathbf{G}^{-1}\mathbf{H}^T \\ \mathbf{F} - \mathbf{H}\mathbf{G}^{-1}\mathbf{E}^T & \mathbf{I} - \mathbf{H}\mathbf{G}^{-1}\mathbf{H}^T \end{bmatrix}, \quad \mathbf{y}^{app}_{\Psi_2} = \begin{bmatrix} y_2 \\ y_4 - \mathbf{H}\mathbf{G}^{-1}y_3 \end{bmatrix} \tag{B.30}$$

Here it can be seen that $\mathcal{C}_2$ no longer has reference to state 3, effectively removing any link between states 3 and 4.

This new estimate is not necessarily representative of the original distribution. The following proposition proves that the TJT approach is not conservative.

Proposition 2. *The Thin Junction Tree is not a conservative method for sparsification.*

Proof. All that is required is to show that the conservatism constraint does not hold. Using the decomposition from example B.2.1. Let $\mathbf{Y} = \mathbf{Y}_{\Psi_1} + \mathbf{Y}_{\Psi_2}$ and $\tilde{\mathbf{Y}} = \mathbf{Y}_{\Psi_1} + \mathbf{Y}^{sp}_{\Psi_2}$

$$\mathbf{Y} - \tilde{\mathbf{Y}} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{E}\mathbf{G}^{-1}\mathbf{H}^T \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{H}^T \\ \mathbf{0} & \mathbf{H}\mathbf{G}^{-1}\mathbf{E}^T & \mathbf{H} & \mathbf{H}\mathbf{G}^{-1}\mathbf{H}^T \end{bmatrix} \not\succeq \mathbf{0} \tag{B.31}$$

Where the result is not positive semi-definite since a row with a zero diagonal has a non-zero off-diagonal [51, Eq. 10.34(c)]. □

The clique contraction technique described above can only be used in situations where the state to be removed is on the edge of a subtree. To allow arbitrary link removal [49] introduced the concept of clique cloning, and subsequent contraction of two states

from each clone to create a clique split, resulting in a removed link in an arbitrary position in the underlying graph.

B.2.1.1 Link removal heuristic

As the complexity of the JT algorithm is related to the clique width, large cliques become good candidates for variable contraction.

In the SLAM variant of TJT [48], states are removed from a clique based on the minimisation of its conditional mutual information with the separator. Consider the case where i contains one or more states from the target clique $\mathcal{C}$ to be removed. Let $\mathbf{Y}_c$ be the information matrix of the marginal formed by marginalising the target potential Ψ_{c2} down to the states i. Let $\mathbf{Y}_s$ be the information matrix of the marginal formed by finding the marginal over the messages across the seperator (in both directions) μ_s to the states i. The conditional mutual information is then c.f. [48, Prop. 6]:

$$I = \frac{1}{2}(\text{logdet}(\mathbf{Y}_c) - \text{logdet}(\mathbf{Y}_s)) \tag{B.32}$$

TJT then chooses the elements of i such that equation B.32 is minimised.

The matrix $\mathbf{Y}_c$ is composed by calculating the marginal around i, while the matrix $\mathbf{Y}_s$ is constructed by combining only the messages from the sub-tree associated with the states i and the clique potential, and then marginalising that result to the states i.

$$\mathbf{Y}_c = \sum_{k=\mathcal{C}/i} \mathbf{Y}_{\mu_k} + \mathbf{Y}_{\Psi_k} \tag{B.33}$$

$$\mathbf{Y}_s = \sum_{k=\mathcal{C}/i} \mathbf{Y}_{\mu_k} + \mathbf{Y}_{\Psi_{\mathcal{S}_{im}}}, \tag{B.34}$$

where $m = \mathcal{C}_i \notin \mathcal{S}_{ij}$ is the set of states not in the neighbouring subtree of i and k is all states not in i. Here $\Psi_{\mathcal{S}_{im}}$ is the conditional potential, conditioned on the states m.

B.2.1.2 Comparison to SEIF

Both the TJT and SEIF share a similar approach to sparsification. That is, they attempt to induce sparsity through the combination of marginals from the true distribution. TJT differs in that its sparsification does not suffer from the drawback of incidental non-zero's being introduced as junction tree cliques are assumed to already be triangulated. TJT also uses a significantly different approach to choosing links to remove.

B.2.1.3 TJT Sparsification using Hugin

The Hugin method of inference uses the concept of a seperator potential to distribute information. Each link maintains a potential ϕ_s, that is a marginal distribution over the states in the seperator. Also, each clique potential is also a marginal over the particular states of the interest. The marginal over two clique potentials is calculated using:

$$p(\mathbf{x}_1, \mathbf{x}_2) = \frac{\phi_1 \phi_2}{\phi_s} \tag{B.35}$$

If the distributions are parametrised using a Gaussian with the information form, then the information matrix for the above equation becomes:

$$\mathbf{Y}_{12} = \mathbf{Y}_{\phi_1} + \mathbf{Y}_{\phi_2} - \mathbf{Y}_{\phi_s} \tag{B.36}$$

Example B.2.2. *Consider the simple case below:*

$$\mathbf{Y} = \begin{bmatrix} \mathbf{A} & \mathbf{B}^T & \mathbf{C}^T & \mathbf{0} \\ \mathbf{B} & \mathbf{D} & \mathbf{E}^T & \mathbf{F}^T \\ \mathbf{C} & \mathbf{E} & \mathbf{G} & \mathbf{H}^T \\ \mathbf{0} & \mathbf{F} & \mathbf{H} & \mathbf{I} \end{bmatrix}, \tag{B.37}$$

which is split into the following using the Hugin junction tree approach [28]:

$$\mathbf{Y}_{\phi_{c1}} = \begin{bmatrix} \mathbf{A} & \mathbf{B}^T & \mathbf{C}^T & \mathbf{0} \\ \mathbf{B} & \mathbf{D} & \mathbf{E}^T & \mathbf{0} \\ \mathbf{C} & \mathbf{E} & \mathbf{G} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}, \mathbf{Y}_{\phi_s} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{D} & \mathbf{E}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{E} & \mathbf{G} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}, \tag{B.38}$$

$$\mathbf{Y}_{\phi_{c2}} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{D} & \mathbf{E}^T & \mathbf{F}^T \\ \mathbf{0} & \mathbf{E} & \mathbf{G} & \mathbf{H}^T \\ \mathbf{0} & \mathbf{F} & \mathbf{H} & \mathbf{I} \end{bmatrix}. \tag{B.39}$$

Let the states associated with $\mathbf{G}$ *be chosen for contraction from the clique* ϕ_{c2}*. This requires marginalising the states associated with* $\mathbf{G}$ *from* ϕ_2 *and* ϕ_s*. Resulting in the following:*

$$\mathbf{Y}_{\phi_{c1}} = \begin{bmatrix} \mathbf{A} & \mathbf{B}^T & \mathbf{C}^T & \mathbf{0} \\ \mathbf{B} & \mathbf{D} & \mathbf{E}^T & \mathbf{0} \\ \mathbf{C} & \mathbf{E} & \mathbf{G}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}, \mathbf{Y}_{\phi_s} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{D} - \mathbf{E}\mathbf{G}^{-1}\mathbf{E}^T & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}, \tag{B.40}$$

$$\mathbf{Y}_{\phi_{c2}} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{D} - \mathbf{E}\mathbf{G}^{-1}\mathbf{E}^T & \mathbf{0} & \mathbf{F}^T - \mathbf{E}\mathbf{G}^{-1}\mathbf{H}^T \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{F} - \mathbf{H}^T\mathbf{G}^{-1}\mathbf{E} & \mathbf{0} & \mathbf{I} - \mathbf{H}\mathbf{G}^{-1}\mathbf{H}^T \end{bmatrix}. \tag{B.41}$$

Since the size of the seperator and the clique have been reduced, this new system will require less computation and communication to solve. To demonstrate that this has infact induced sparsity, reform the information matrix using the formula:

$$\tilde{\mathbf{Y}} = \phi_{c1} - \phi_s + \phi_{c2}, \tag{B.42}$$

resulting in:

$$\tilde{\mathbf{Y}} = \begin{bmatrix} \mathbf{A} & \mathbf{B}^T & \mathbf{C}^T & \mathbf{0} \\ \mathbf{B} & \mathbf{D} & \mathbf{E}^T & \mathbf{F}^T - \mathbf{E}\mathbf{G}^{-1}\mathbf{H}^T \\ \mathbf{C} & \mathbf{E} & \mathbf{G} & \mathbf{0} \\ \mathbf{0} & \mathbf{F} - \mathbf{H}\mathbf{G}^{-1}\mathbf{E}^T & \mathbf{0} & \mathbf{I} - \mathbf{H}\mathbf{G}^{-1}\mathbf{H}^T \end{bmatrix} \tag{B.43}$$

Notice the induced zeros in the previous position of the matrix $\mathbf{H}$.

B.2.2 TreeMap

Treemap [22] is an architecture to perform SLAM. Although it is specifically a SLAM algorithm, its sparsification procedure can be applied to any information matrix. The procedure is now presented c.f. [21]:
Consider an information matrix (where a is a scalar, $\mathbf{b}$ and $\mathbf{c}$ are vectors and $\mathbf{A}$ is a matrix),

$$\mathbf{Y} = \begin{bmatrix} a & \mathbf{b}^T & \mathbf{c}^T \\ \mathbf{b} & \mathbf{A}_{22} & \mathbf{A}_{32}^T \\ \mathbf{c} & \mathbf{A}_{32} & \mathbf{A}_{33} \end{bmatrix}, \tag{B.44}$$

Then the matrix that sets the vector $\mathbf{b}$ from $\mathbf{Y}$ to zero is:

$$\mathbf{B} = \mathbf{x}\mathbf{x}^T, \quad \mathbf{x} = \mathbf{Y} \begin{bmatrix} \lambda \\ \delta\mathbf{A}_{22}^{-1}\mathbf{b} \\ \mathbf{0} \end{bmatrix}, \tag{B.45}$$

Using:

$$\alpha = \mathbf{b}^T \mathbf{A}_{22}^{-1} \mathbf{b}, \quad \beta = (a - \alpha)^{-1} \tag{B.46}$$

$$\lambda = \sqrt[4]{a(\mathbf{b}^T \mathbf{A}_{22} \mathbf{b})}, \quad \gamma = \beta(\lambda - \lambda^{-1}\alpha), \quad \delta = \beta(-\lambda + a\lambda^{-1}). \tag{B.47}$$

The sparsification can be performed using the equation:

$$\tilde{\mathbf{Y}} = \mathbf{Y} - \mathbf{B} \tag{B.48}$$

Lemma B.1. *The Treemap approach to sparsification is a conservative approach to sparsification, assuming that* $\mathbf{Y} - \mathbf{B} \succ \mathbf{0}$.

Proof. If $\mathbf{Y} - \mathbf{B} \succ \mathbf{0}$ then the Treemap approach will result in a well formed information matrix. Consider the consistency constraint:

$$\mathbf{Y} - \tilde{\mathbf{Y}} = \mathbf{Y} - (\mathbf{Y} - \mathbf{B}) \tag{B.49}$$

$$= \mathbf{B} \tag{B.50}$$

$$\succeq \mathbf{0} \tag{B.51}$$

where the final step has used $\mathbf{B} = \mathbf{x}\mathbf{x}^T \succeq \mathbf{0}$ ([51, Eq. 10.10]). □

Treemap is not guaranteed to produce a result that can be used for sparsification. This is because of a flaw in Theorem 3 from [21]. This theorem's proof optimizes the quantity $\mathbf{x}$, but makes no claim of its optimal value, while Lemma 17 from the same work requires that $\mathbf{x}^T \mathbf{A}^{-1} \mathbf{x} < 1$ for the matrix $\mathbf{A} - \mathbf{x}\mathbf{x}^T \succ \mathbf{0}$. Thus in the case that Treemap's value for $\mathbf{x}$ does not satisfy $\mathbf{x}^T \mathbf{A}^{-1} \mathbf{x} < 1$ then using the result from treemap will cause the system to become indefinite, that is, the value $\mathbf{Y} - \mathbf{B} \not\succeq \mathbf{0}$.

B.2.2.1 Flexibility of induced sparsity

This approach cannot delete an arbitrary sparsity pattern, rather it can only delete links connected to a single state. In [21] this method is extended to allow for multiple link removal steps, but these steps lose the closeness bounds guaranteed by the single state case. This method will also compound the numerical instabilities for large dimension matrices.

The subtraction of a rank one matrix is not in general a sparse operation, and in this case the $\mathbf{B}$ matrix will be dense in every other part except for the offdiagonal columns being sparsified. This technique is therefore not useful for information form sparsification where existing sparsity needs to be maintained. The technique can be used in scenarios where a dense matrix can still represent overall sparsity, such as a junction tree clique.

Algorithm B.1: Multi-row Treemap Sparsify

input : $\mathbf{Y}$: matrix to be sparsified
output: $\mathbf{Y}$: sparse approximation of $\mathbf{Y}$
$\mathbf{W} := \mathbf{Y}$
for *i=1 to number of rows of* $\mathbf{Y}$ **do**
  $\mathbf{B}$:= elimination matrix for row i of $\mathbf{W}_2$ from Eq. B.48
  $\mathbf{Y} := \mathbf{Y} - \mathbf{B}$
  $\mathbf{W} := \mathbf{W} - \mathbf{B}$
  ww^T := elimination matrix of row/column i of $\mathbf{W}$
  $\mathbf{W} := \mathbf{W} - ww^T$

B.2.2.2 Link removal heuristic

The heuristic used in Treemap is based upon the particular structure of the SLAM problem. Treemap avoids recording robot states at all, and only does so when the number of observed landmarks is insufficient to constrain the robots location. If Treemap does maintain a robot state, it immediately sparsifies odometry links between robot states. This is similar to DD-SEIF, except that the information contained in the link can be approximated rather than discarded.

B.2.3 CS-JT Comparison to Chordal Embedding

Dahl et al [13] describe a method for improving the efficiency of Newton's method for covariance selection through chordal embedding. This is similar to the approach taken in Section 4.2.3 as both use the concept of a junction tree (or clique tree) to decompose the graph. In this method however, the chordal sparsity pattern is exploited to enable fast computation of the gradient and Hessian for the associated convex optimisation problem. The approach in this thesis is different in the following ways:

- Conservative Sparsification adds a conservatism constraint,
- chordal embedding aims to solve the problem globally, whereas CS with JT's focuses on removing links in a small subset of the graph,
- chordal embedding uses the clique tree to speed up optimisation rather than using it as a basis for estimation, and,
- chordal embedding does not allow for approximate solutions like the parallel CSJT.

CS using junction trees attempts to solve a different problem, and maintains the junction tree form with conservatism constraints to enable data fusion after sparsification.

B.2.4 CS-JT Comparison to Treemap

The idea of working with a sum of positive definite matrices was posed with the SLAM solution Treemap [21], which is described in Section B.2.2. This work considers a BIB (Basic Information Block) as a data structure for the information matrices. BIB's play a similar role to clique potentials in a junction tree, but they only contain landmark states. Treemap is not a general framework, and requires a SLAM problem for its use. It requires the concept of robot and landmark states and its sparsification is only performed upon robot odometry measurements (see Section B.2.2 for a description of Treemap sparsification procedure).

B.3 Matrix Approximation

This section discusses alternate approaches to matrix approximation that also induce sparsity or reduce the complexity of a matrix while maintaining its dimensions. These approaches are considered in the context of large scale, distributed and incremental linear estimation.

B.3.1 Optimisation Based Approaches

The set of positive semi-definite matrices forms a convex cone and is therefore a convex set. This has enabled many different approaches to matrix approximation using convex analysis.

B.3.1.1 Matrix Completion and Covariance Selection

The MAXDET problem, as specified in [64] is shown below:

$$\begin{aligned} &\min_{\mathbf{x}} \mathbf{c}^T\mathbf{x} + \operatorname{logdet} G(\mathbf{x})^{-1} \\ &\quad s.t. \\ &\quad G(\mathbf{x}) \succ 0 \\ &\quad F(\mathbf{x}) \succeq 0 \end{aligned}$$

where $\mathbf{x}, \mathbf{c} \in \mathbb{R}^m$, and the functions $G(\mathbf{x}) : \mathbb{R}^m \to \mathbb{R}^{l\times l}$, $F(\mathbf{x}) : \mathbb{R}^m \to \mathbb{R}^{n\times n}$ are affine. That is, they are of the form:

$$\begin{aligned} G(x) &= \mathbf{G}_0 + x_1\mathbf{G}_1 + \ldots + x_m\mathbf{G}_m, \\ F(x) &= \mathbf{F}_0 + x_1\mathbf{F}_1 + \ldots + x_m\mathbf{F}_m, \end{aligned}$$

where $\mathbf{F}_i = \mathbf{F}_i^T$ and $\mathbf{G}_i = \mathbf{G}_i^T$. By appropriately choosing the $\mathbf{F}_i$ and $\mathbf{G}_i$ matrices this can represent many matrix inequalities. Example problems that can be represented are the matrix completion problem [38] (where some elements of a matrix are set

and a positive definite matrix is formed using the undefined other elements), or the covariance selection problem [16] (where particular offdiagonals in a covariance matrix are set to zero, while minimizing a cost such as the matrix determinant).

The worst case runtime complexity for this is $\mathcal{O}(\sqrt{n}(n^2 + l^2)m^2)$ [64], if $\mathbf{x}$ is the vectorised form of a dense matrix of dimension r, then it will have $m = \frac{r(r-1)}{2}$ elements. Assuming that the matrix inequalities are of similar size, then the runtime complexity becomes:

$$\begin{aligned}
\mathcal{O}(\sqrt{n}(n^2 + l^2)m^2) &= \mathcal{O}\left(\sqrt{r}(r^2 + r^2)(\frac{r(r-1)}{2})^2\right) \\
&= \mathcal{O}\left(\sqrt{r}(r^2 + r^2)r^4\right) \\
&= \mathcal{O}\left(r^{.5}r^2r^4\right) \\
&= \mathcal{O}\left(r^{6.5}\right)
\end{aligned}$$

However, if the matrix $\mathbf{x}$ is sparse, then assuming it has s nonzeros the complexity becomes:

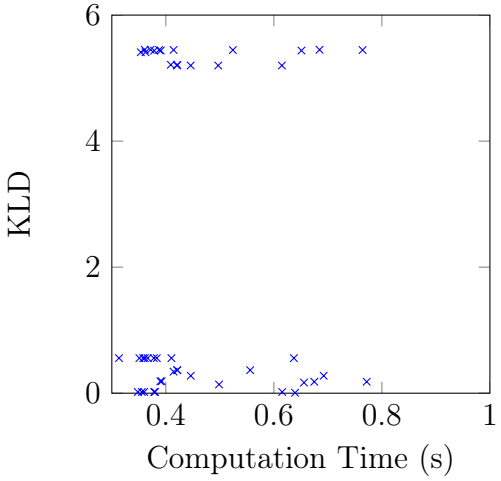
$$\begin{aligned}
\mathcal{O}\left(\sqrt{n}(n^2 + l^2)m^2\right) &= \mathcal{O}\left(\sqrt{r}(r^2 + r^2)\, s^2\right) \\
&= \mathcal{O}\left(r^{2.5}s^2\right)
\end{aligned}$$

Appendix C

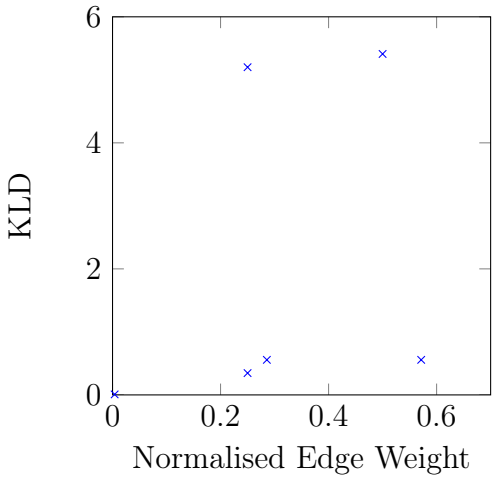
Results

C.1 Extra Benchmark Matrices

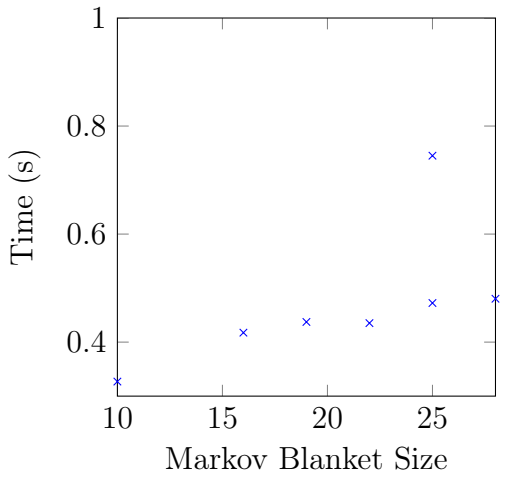
This section contains extra matrices that were used to benchmark the CS algorithm in Chapter 3. These figures show the algorithms performance on matrices which have different sparse structures. Each off diagonal in the matrix was sparsified independently and then sorted by their runtime, initial normalised edge weight and Markov blanket size. These experiments were performed on an i7 2.2Ghz processor with 16Gb of RAM using SDPT-3 [56] as the solver. Many of the Markov blanket vs computation time results show that there is no strong connection between run-time and Markov blanket size. This is due to the matrices being small in dimension, with small Markov blankets. It is expected that larger Markov blankets will demonstrate the cost increase. It is good to note however, that for Markov blankets less than 22 states, all the problems can be solved in less than 2 seconds.



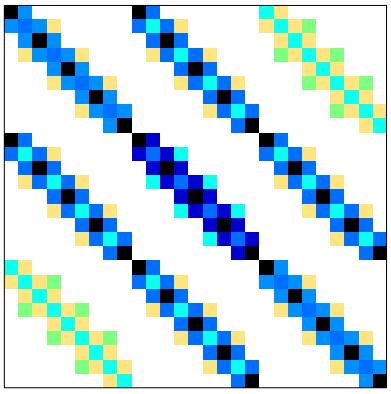
(a) KLD as Computation Time Increases



(b) KLD as a function of normalised edge weight

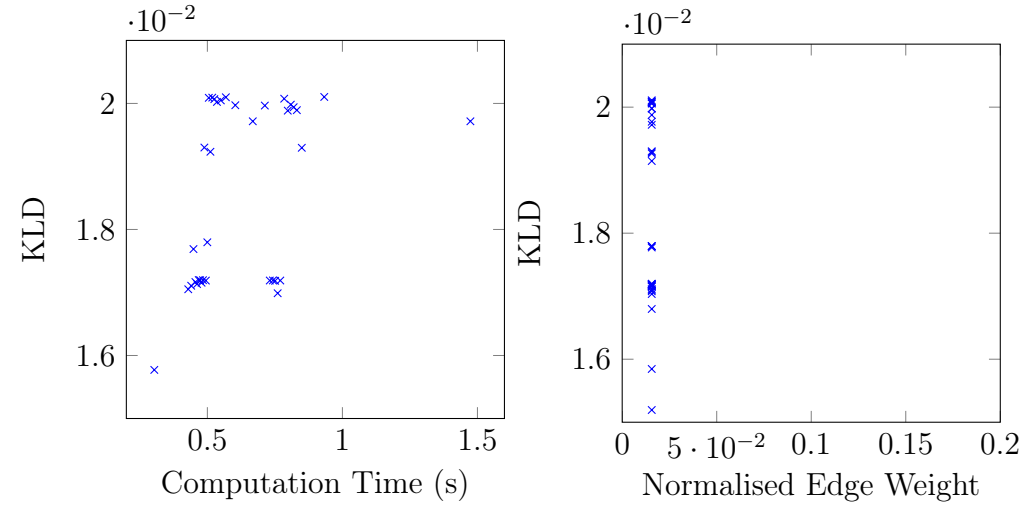


(c) Time required to solve as Markov Blanket size increases

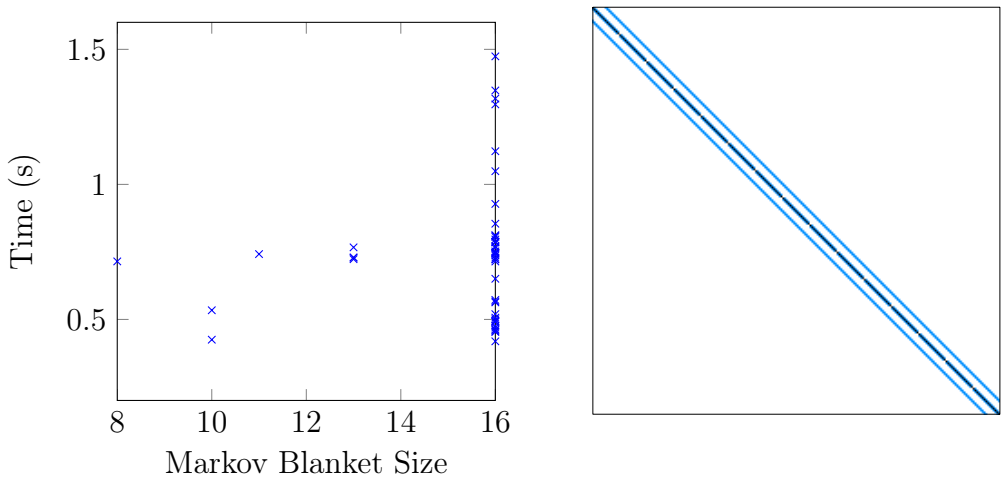


(d) CSPY Plot

Figure C.1 – FIDAP EX-5 Matrix sparsification result. This banded diagonal matrix shows very weak trend as in computation time as the Markov blanket increases. Likely because of the large sparsity in this problem. Observe that the KLD of the final solution appears unrelated to the computation time required or the initial edge weight.



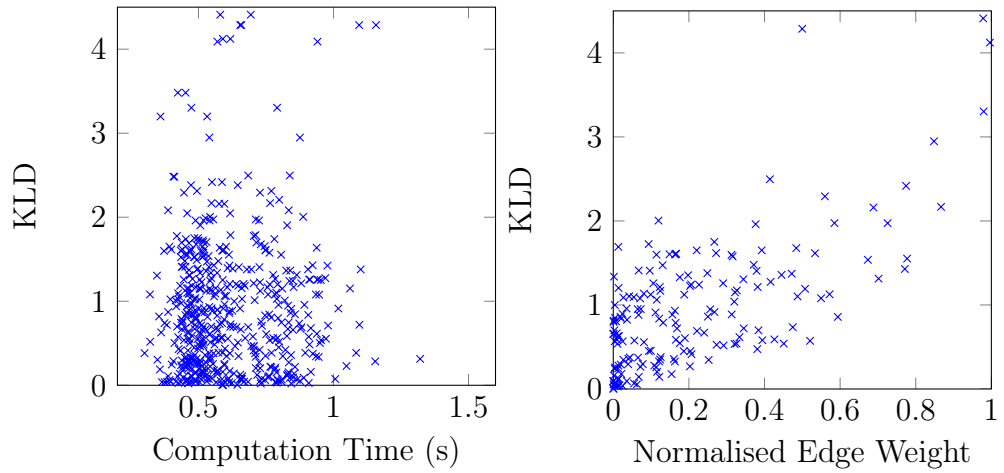
(a) KLD as Computation Time Increases **(b)** KLD as a function of normalised edge weight



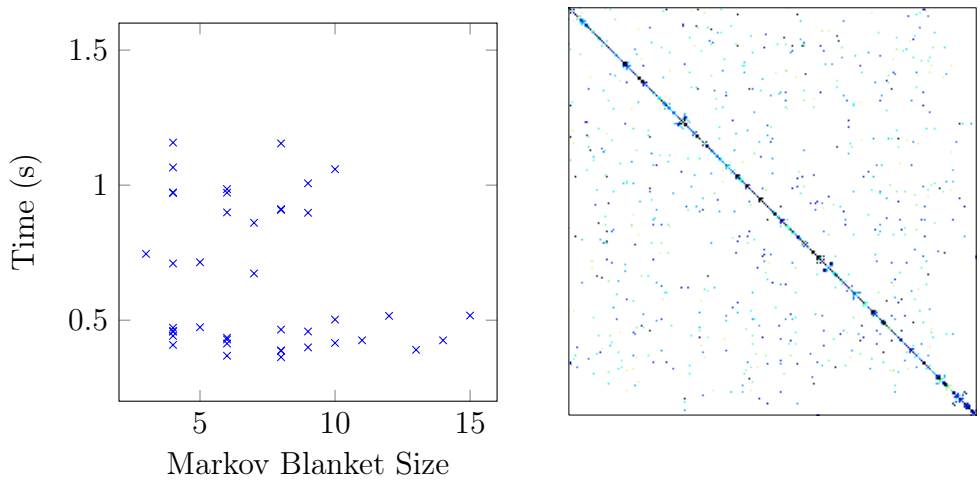
(c) Time required to solve as Markov Blanket size increases

(d) CSPY Plot

Figure C.2 – HB gr-30-30 Matrix sparsification result. This 30×30, singly banded diagonal matrix has off-diagonals with the same value. Despite this, there is a large variance in the KLD's as shown by (b). Also, there is a large variance in the run-times for a given Markov blanket size.



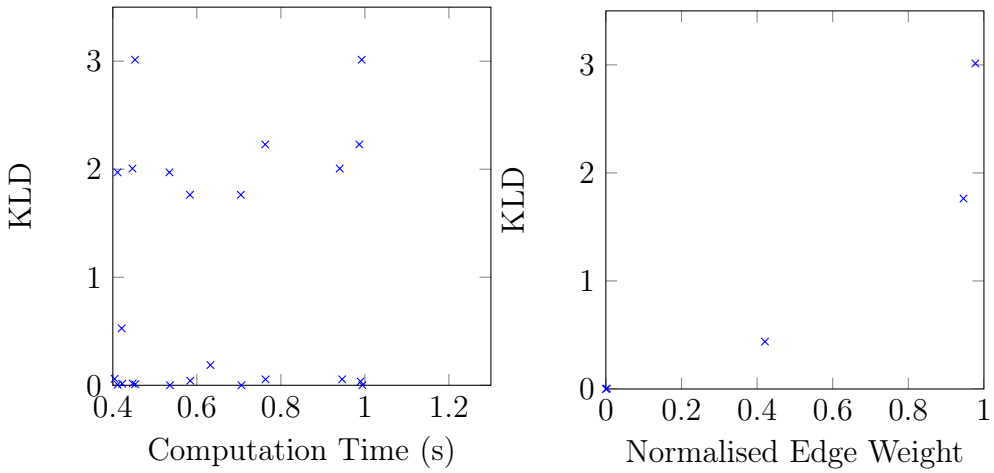
(a) KLD as Computation Time Increases **(b)** KLD as a function of normalised edge weight



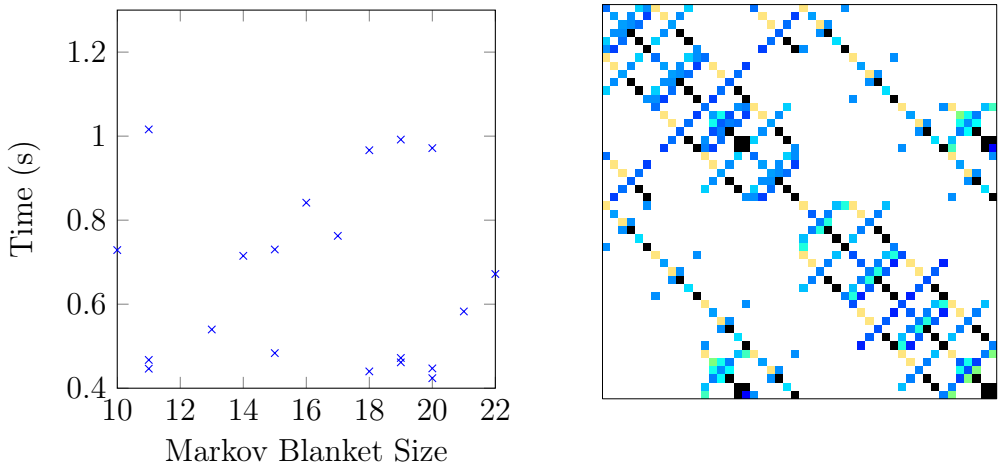
(c) Time required to solve as Markov Blanket size increases

(d) CSPY Plot

Figure C.3 – HB 494 bus Matrix sparsification result. This large sparse matrix shows a noisy upwards trend in KLD as the edge weight increases. Strangely the computation time appears to be unaffected by the size of the Markov blanket.



(a) KLD as Computation Time Increases **(b)** KLD as a function of normalised edge weight



(c) Time required to solve as Markov Blanket size increases

(d) CSPY plot

Figure C.4 – HB bcsstk01 Matrix sparsification result. This strongly structured sparse matrix shows an increase in KLD as the normalised edge weight increases. However the computation time as a function of the Markov blanket size remains flat.

C.2 Suite Sparse matrices, Chordal CS vs. CS comparisons

These matrices were tested to compare the Chordal CS and CS approaches described in Section 3.2.3. These results are presented to demonstrate the technique applied to a variety of different matrix structures. Links were selected if they had a Markov blanket which was not locally chordal. They were then ordered and sparsified independently. In most cases, the problems can be solved, however there are some cases where only Chordal CS is capable of finding a solution.

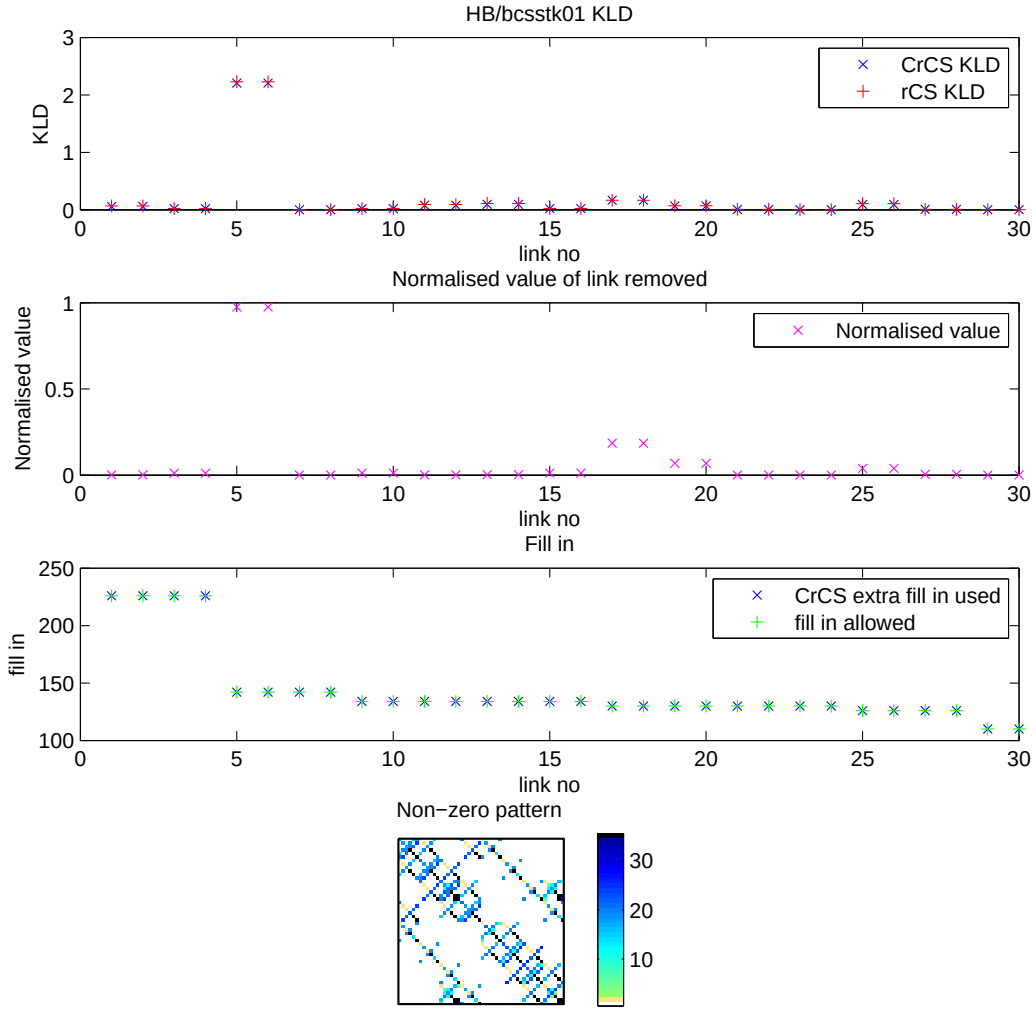


Figure C.5 – For this matrix every single non locally chordal link removal could be solved by both Chordal Conservative Sparsification (CCS) and CS. CCS always used all of the allowed edges. The difference in KLD is very small for both approaches, but CCS always uses all of the available fill-in.

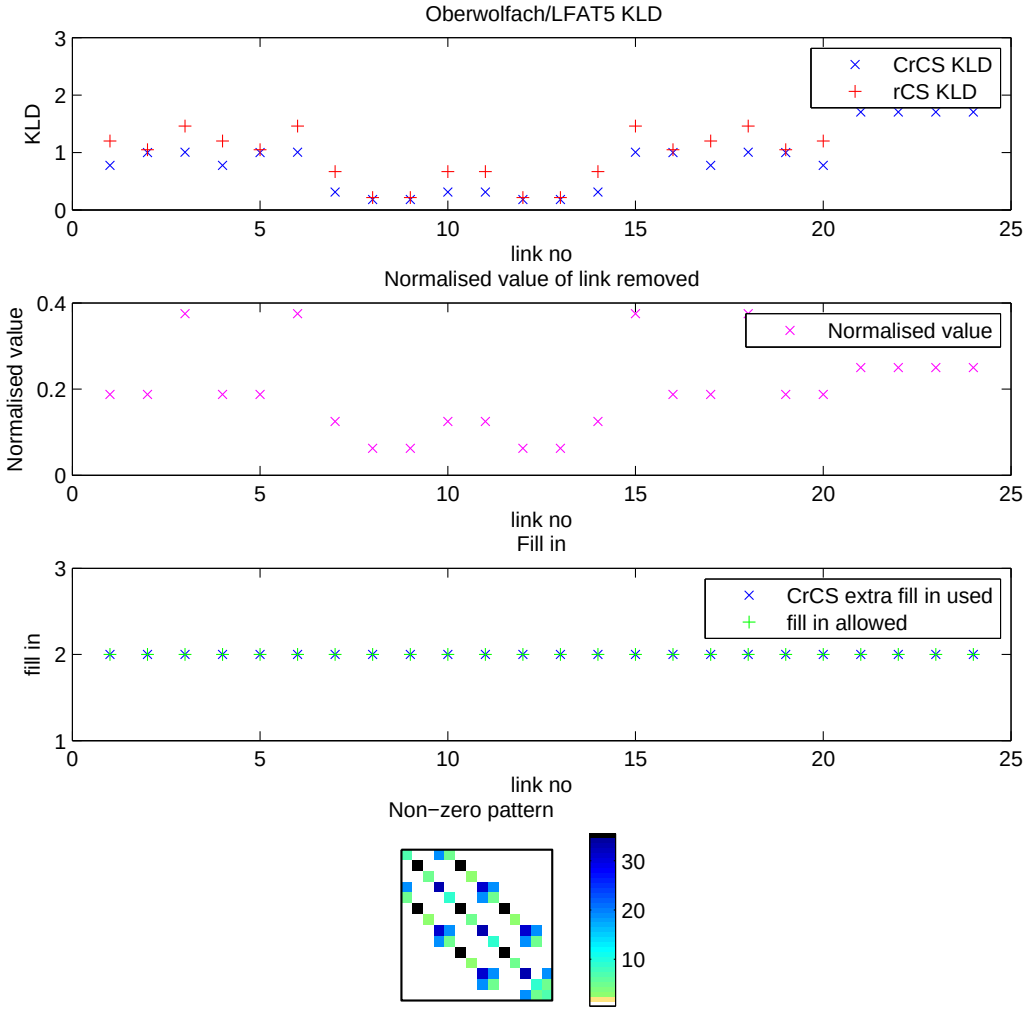


Figure C.6 – For this matrix every single non locally chordal link removal could be solved by both CCS and CS. CCS always used all of the allowed edges.

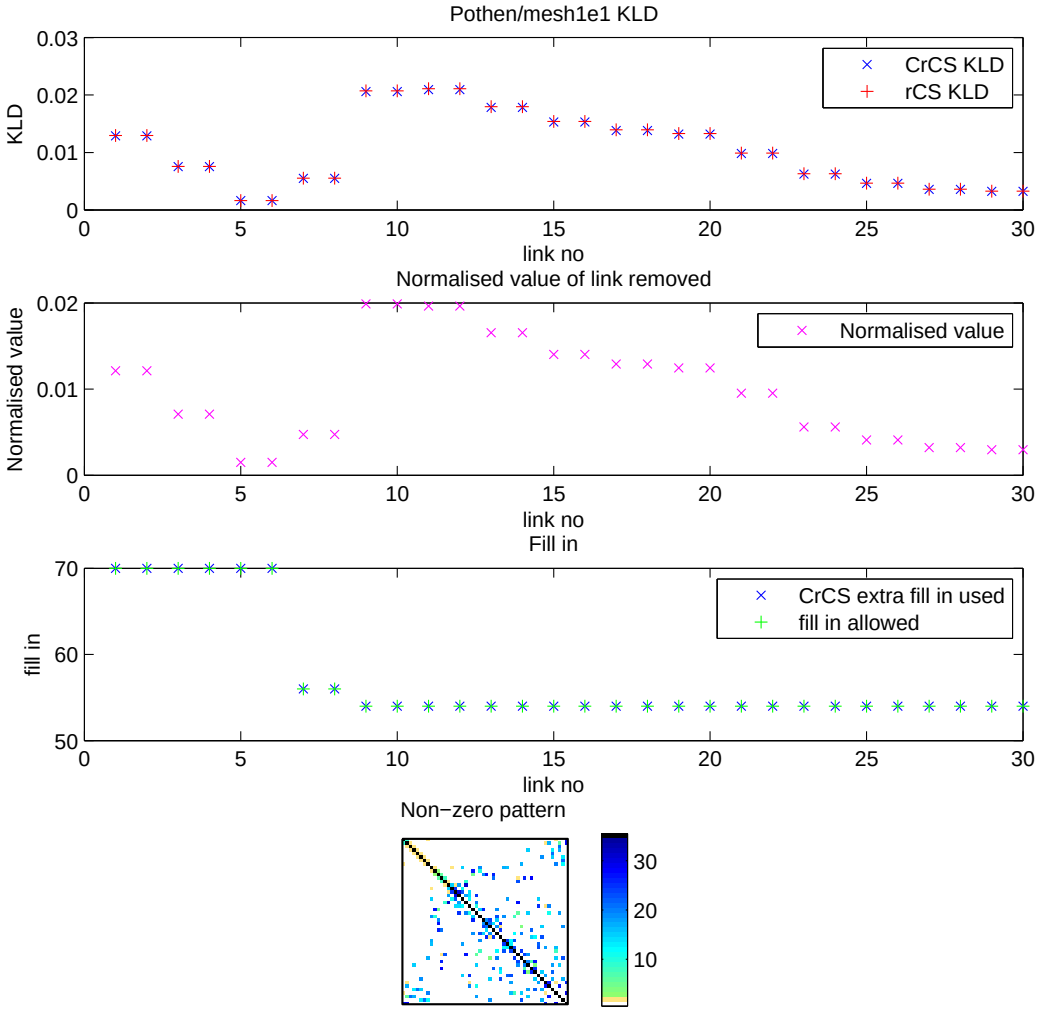


Figure C.7 – For this matrix every single non locally chordal link removal could be solved by both CCS and CS. CCS always used all of the allowed edges.

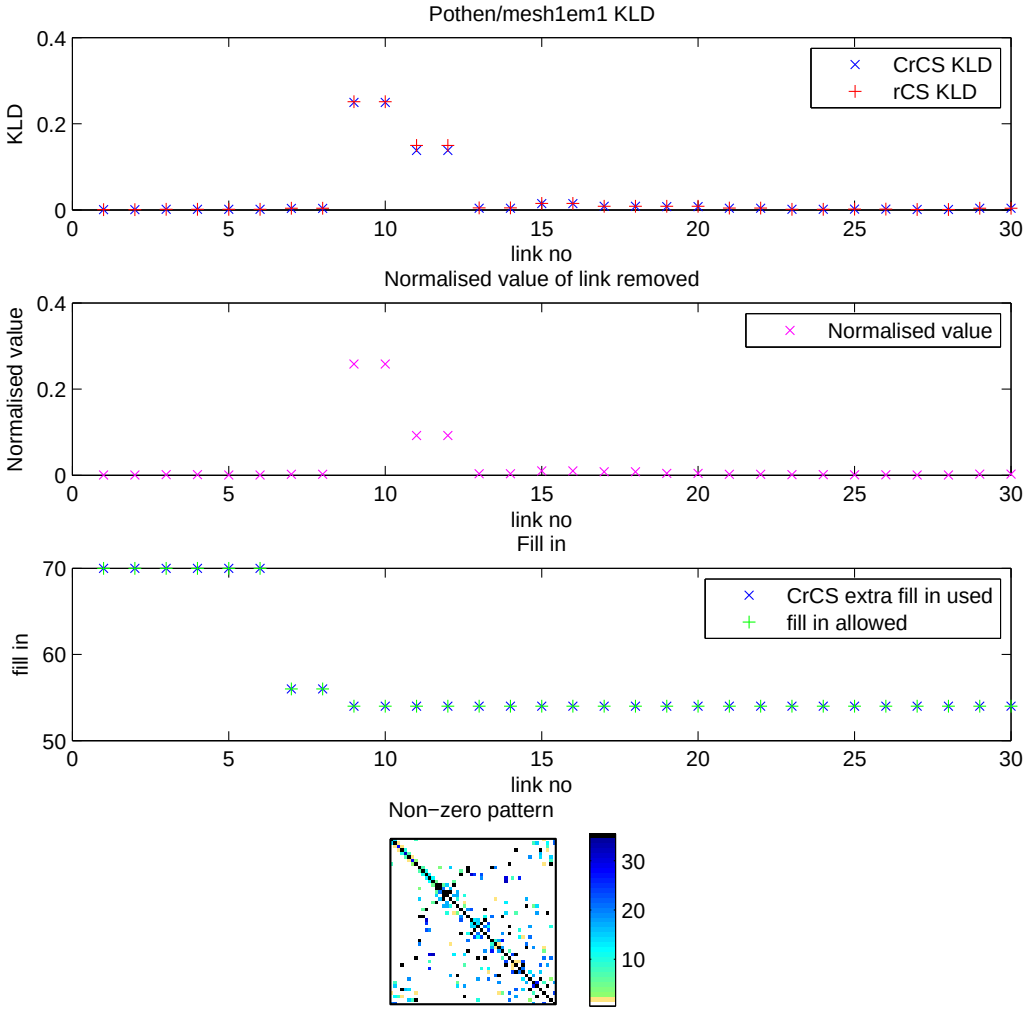


Figure C.8 – For this matrix every single non locally chordal link removal could be solved by both CCS and CS. CCS always used all of the allowed edges.