On the Links between Probabilistic Graphical Models and Submodular Optimisation

Senanayak Sesh Kumar Karri

THÈSE DE DOCTORAT

de l'Université de recherche Paris Sciences Lettres
PSL Research University

Préparée à l'École normale supérieure

On the Links between Probabilistic Graphical Models and Submodular Optimisation

Liens entre modèles graphiques probabilistes et optimisation sous-modulaire

École doctorale n°386

ÉCOLE DOCTORALE DE SCIENCES MATHÉMATIQUES DE PARIS CENTRE

Spécialité INFORMATIQUE

COMPOSITION DU JURY :

M Andreas Krause
ETH Zurich, Rapporteur

M Nikos Komodakis
ENPC Paris, Rapporteur

M Francis Bach
Inria Paris, Directeur de thèse

M Josef Sivic
ENS Paris, Membre du Jury

M Antonin Chambolle
CMAP EP Paris, Membre du Jury

M Guillaume Obozinski
ENPC Paris, Membre du Jury

Soutenue par Senanayak Sesh Kumar KARRI
le 27.09.2016

Dirigée par **Francis BACH**

ENS ÉCOLE NORMALE SUPÉRIEURE | PSL RESEARCH UNIVERSITY PARIS

What is the purpose of life?
Proof and Conjecture, ...

Paul Erdős

Dedicated to my family

Abstract

A probabilistic graphical model encodes conditional independences among random variables, which is related to factorisable distributions. Moreover, the entropy of a probability distribution on a set of discrete random variables is always bounded by the entropy of its factorisable counterpart. This is due to the submodularity of entropy on the set of discrete random variables. Submodular functions are also generalisation of matroid rank function; therefore, linear functions may be optimised on the associated polytopes exactly using a greedy algorithm.

In this manuscript, we exploit these links between the structures of graphical models and submodular functions: we use greedy algorithms to optimise linear functions on the polytopes related to graphic and hypergraphic matroids for learning the structures of graphical models, while we use inference algorithms on graphs to optimise submodular functions.

The first main contribution of the thesis aims at approximating a probabilistic distribution with a factorisable tractable distribution under the maximum likelihood framework. Since the tractability of exact inference is exponential in the treewidth of the decomposable graph, our goal is to learn bounded treewidth decomposable graphs, which is known to be NP-hard. We pose this as a combinatorial optimisation problem and provide convex relaxations based on graphic and hypergraphic matroids. This leads to an approximate solution with good empirical performance.

In the second main contribution, we use the fact that the entropy of a probability distribution is always bounded by the entropy of its factorisable counterpart mainly as a consequence of submodularity. This property of entropy is generalised to all submodular functions and bounds based on graphical models are proposed. We refer to them as *graph-based bounds*. An algorithm is developped to maximise submodular functions, which is NP-hard, by maximising the graph-based bound using variational inference algorithms on graphs.

The third main contribution of the thesis deals with minimising submodular functions that can be written as sum of "simple" submodular functions. It is broadly subdivided into two parts. The first part deals with reviewing algorithms that minimise sum of "simple" submodular functions using minimisation oracles of the "simple" functions. Here, we specifically deal with cut functions in large scale problems and the minimisation oracles of the "simple" functions are graph inference algorithms. The second part proposes algorithms to minimise sum of general submodular functions using the structure of the polytopes related to individual "simple" submodular functions.

Keywords : Probabilistic graphical models, maximum likelihood trees, discrete optimisation, submodular optimisation, total variation, convex optimisation.

Résumé

Un modèle graphique probabiliste représente les relations d'indépendances conditionnelles parmi des variables aléatoires, qui sont liées à la factorisation de la densité. De plus, l'entropie d'une distribution sur un ensemble de variables aléatoires discrètes est toujours bornée par l'entropie de la distribution factorisée correspondante. Cette propriété est due à la sous-modularité de l'entropie. Par ailleurs, les fonctions sous-modulaires sont une généralisation des fonctions de rang des matroides ; ainsi, les fonctions linéaires sur les polytopes associés peuvent être minimisées exactement par un algorithme glouton.

Dans ce manuscrit, nous exploitons ces liens entre les structures des modèles graphiques et les fonctions sous-modulaires. Nous utilisons des algorithmes gloutons pour optimiser des fonctions linéaires sur des polytopes liés aux matroides graphiques et hypergraphiques pour apprendre la structure de modèles graphiques, tandis que nous utilisons des algorithmes d'inférence sur les graphes pour optimiser des fonctions sous-modulaires.

La première contribution de cette thèse consiste à approcher par maximum de vraisemblance une distribution de probabilité par une distribution factorisable et de complexité algorithmique contrôlée. Comme cette complexité est exponentielle dans la largeur arborescente du graphe, notre but est d'apprendre un graphe décomposable avec une largeur arborescente bornée, ce qui est connu pour être NP-difficile. Nous posons ce problème comme un problème d'optimisation combinatoire et nous proposons une relaxation convexe basée sur les matroides graphiques et hypergraphiques. Ceci donne lieu à une solution approchée avec une bonne performance pratique.

Pour la seconde contribution principale, nous utilisons le fait que l'entropie d'une distribution est toujours bornée par l'entropie de sa distribution factorisée associée, comme conséquence principale de la sous-modularité, permettant une généralisation à toutes les fonctions sous-modulaires de bornes basées sur les concepts de modèles graphiques. Un algorithme est développé pour maximiser les fonctions sous-modulaires, un autre problème NP-difficile, en maximisant ces bornes en utilisant des algorithmes d'inférence vibrationnels sur les graphes.

La troisième contribution principale de la thèse cherche à minimiser les fonctions sous-modulaires qui peuvent s'écrire comme la somme de fonctions sous-modulaires "simples", et est divisée en deux parties : la première propose une rappel sur les algorithmes permettant de minimiser de telles fonctions en utilisant des oracles de minimisation précis pour les fonctions simples. Nous nous focalisons principalement sur les fonctions de coupes dans les graphes où ces oracles correspondent à des al-

gorithmes d'inférence dans les modèles graphiques. La deuxième partie propose des algorithmes ces sommes de fonctions simples en utilisant la structure des polytopes associés.

Mots-clés : Modèles graphiques probabilistes, arbres de maximum de vraisemblance, optimisation discrète, optimisation sous-modulaire, variation totale, optimisation convexe.

Acknowledgements

I am eternally thankful to my thesis advisor Prof. Francis Bach for giving me an opportunity to work in the area of my interest. His passion for research and problem solving has always inspired me. His knowledge and vast experience has always helped me overcome the stumbling blocks during various phases of my doctoral studies. A simple discussion with him ensured that I am not stuck or demotivated, which I believe is very important for any doctoral student. Overall, Francis has been an excellent advisor and mentor. I could not have asked for a better person as a guide for my PhD. On a personal front too, he has been very accommodating on several occasions, very helpful when required and a great support. Overall Francis has been a great friend and an excellent advisor. This thesis was impossible without his support.

I would like to thank Andreas Krause and Nikos Komodakis for accepting to review the manuscript. I am grateful to Andreas Krause, Nikos Komodakis, Antonin Chambolle, Josef Sivic and Guillaume Obozinski for accepting to be part of the jury committee. It is a great honor to have such esteemed researchers in the jury.

I would like to extend my gratitude towards my collaborators Stefanie Jegelka, Alvaro Barbero and Suvrit Sra, working with whom I gained a lot of experience both theoretical and pratical.

I would like to thank Manoj Kumar Rajagopal for being my first guide to Paris. I would like to thank Gonzague Pichelin and Virginie Guarnerio for being the friends you are and creating a home away from home during my first years of stay in Paris. I would also like to thank Maison de l'Inde for providing me accommodation in Paris during my doctoral studies and ensuring a vibrant environment. I would like to thank everyone I came across in Maison de l'Inde for being a part of my life in Paris, importantly Emmanuel Stephen Victor. I am very grateful to Mr. Chris for organizing the hikes, which helped me explore the nature around Paris. Ofcourse, these explorations would not be the same without the fantastic and wonderful company of Anthony Gauci, Anja Fricke and Dominique Haleva. Nino Shervashidze and Anja Fricke have been a great company in exploring various cuisines in Paris as food exploration has been a major part of my PhD life.

A special thanks to Josef Sivic and Ivan Laptev who were kind enough to recruit me for the research engineer roles in their lab. I am very grateful to them for this. I should mention that the company of Marina Vinyes, Christophe Dupuy, Igor Colin, Maxime Oquab and Nicolas Flammarion during my masters at ENS Cachan a memorable experience. I would like to specially mention Piotr Bojanowski, Augustin Lefevre, Aymeric Dieuleveut and Gauthier Gidel for being amazing people

who I shared my office with during my PhD. Thanks for putting up with all my weirdness. Thanks to Remi Lajugie, Guillaume Seguin, Amande Ine, Damien Garreau, Christophe Dupuy and Francis Bach for always being around when I had troubles with the french language on numerous occasions. I would like to thank Simon Lacoste-Julien for presenting my ICML-2013 paper, which I could not attend. I would like to thank other members of WILLOW and SIERRA that I came across during my PhD like Vincent Roulet, Damien Scieur, Anastasia Podosinnikova, Jean-Baptiste Alayrac, Loïc Landrieu, Minsu Cho, Vadim Kantarov, Julia Peyre, Guilhem Cheron, Gul Varol, Anton Osokin, Mathieu Aubry, Armand Joulin, Vincent Delaitre, Edouard Grave, Olivier Duchenne, Mark Schmidt and others. I have thoroughly enjoyed all the discussions both technical and non-technical ones. I apologise to those who I have not named. I hope you understand that it is not on purpose but merely due to my bad memory.

I would like to thank Kiran Varanasi and Visesh Chari for the intellectually simulating philosophical discussions. I would like to thank OSK Chaitanya and Madhuri Vaidya, who were always ready with valuable advises when I was lost and confused. Thank you Sudhir for introducing me to "good" coffee. It has been an invaluable partner in this journey.

My parents: Rama Krishna and Krishna Veni have been amazing pillars of support all my life and they certainly played an important part in this achievement too. My brother Venu has always been one of the understanding and accommodating person without which this journey was not possible. My wife Sravani has played the most important role in this journey. Infact, she has walked every step with me since we got married 2 years ago through all the ups and downs. She has sacrificed several things to ensure, I reach this milestone. I would like to thank my family for the unending support in spite of who I am. I dedicated this thesis to them.

Contents

Contributions and thesis outline

Chapter 1: This chapter introduces probabilistic graphical models, submodular functions, matroids and convex optimisation, which are the main topics related to the manuscript. We recap the basics and some classical results, which we use as part of our work.

Chapter 2: This chapter considers the problem of learning undirected graphs with tractable inference under the maximum likelihood framework. This is equivalent to estimating a bounded treewidth decomposable graph that has a probabilistic distribution with least Kullback-Leibler divergence from the empirical distribution, which is known to be NP-hard. We pose this as a combinatorial optimisation problem and propose convex relaxations based on optimising linear functions on graphic and hypergraphic matroids that lead to an approximate solution. We show that our algorithm recovers the original graph structure for some graphs that have chain or star structured junction trees. We also compare the performance of our algorithm with state-of-art methods on standard datasets.

Chapter 3: This chapter considers the problem of maximising submodular functions. Mutual information, cover functions, etc., are interesting submodular functions, which model diversity and exhibit diminishing returns property. Although NP-hard, these class of problems admit constant factor approximation algorithms. In this work, directed acyclic graph (DAG) based bounds to submodular functions are proposed, which are further extended to decomposable graphs. It is shown that submodular functions can be maximised by maximising the bounds based on decomposable graphs using variational inference on underlying graph polytope. We compare the performance of our algorithm to solve max-cut on two toy graphs of different topologies with some standard greedy algorithms.

Chapter 4: This chapter considers the problem of minimising submodular functions, which can be decomposed into "simple" submodular functions. It specifically deals with minimising submodular energies on grid structures, often used in computer vision applications. These are cut functions, which are a subclass of submodular functions. The Lovász extension of a cut function is its total variation. The total variation oracles of the "simple" functions, i.e., lines in our case, are used to minimise the cut on grids. Proximal splitting methods like Douglas-Rachford, alternating projection,

block coordinate descent and FISTA are used to solve this problem. The performance of these algorithms is compared with the state-of-art combinatorial algorithms on some standard maxflow datasets with 2D and 3D grid structures. Our insights from this work has motivated us to work in the direction of the next chapter.

Chapter 5: This chapter extends the work of previous chapter to general submodular functions, which can be written as sum of "simple" submodular functions with no assumption on the type of submodular function. The goal is to be able to use submodular function minimisation (SFM) oracles of "simple" submodular functions to the minimise their sums. In general, SFM oracles are much less complex when compared to the total variation oracles that are used in the previous chapter. Initially, an active-set method is proposed to solve the total variation problem from the SFM oracles with warmstart, which would enable us to solve the total variation oracles of the "simple" functions quickly and enhance the performance of methods from the previous chapter to minimise the sum of the submodular functions. This is followed by use of local search techniques on active-sets only using SFM oracles and minimising the sum of submodular functions. Eventually, the algorithm is compared favorably with existing algorithms on standard datasets.

Chapter 6: This chapter concludes the thesis by highlighting our contributions and suggesting possible future research directions.

We list the publications related to this manuscript:

(a) Chapter 2 is based on the article: Convex Relaxations for Learning Bounded Treewidth Decomposable Graphs, K. S. Sesh Kumar, F. Bach, In proceedings of International Conference on Machine Learning, 2013 [Sesh Kumar and Bach, 2013b].

(b) Chapter 3 is based on the article: Maximizing Submodular Functions using Probabilistic Graphical Models, K. S. Sesh Kumar, F. Bach, In Workshop on Discrete and Combinatorial Problems in Machine Learning (DISCML), NIPS 2013: Theory and applications [Sesh Kumar and Bach, 2013a].

(c) Chapter 4 is based on our preprint: Convex Optimization for Parallel Energy Minimization, K. S. Sesh Kumar, A. Barbero, S. Jegelka, S. Sra and F. Bach [Sesh Kumar et al., 2015].

(d) Chapter 5 is based on our preprint: Active-set Methods for Submodular Optimization, K. S. Sesh Kumar, F. Bach [Sesh Kumar and Bach, 2015] and a journal: Active-set Methods for Submodular Minimisation Problems, K. S. Sesh Kumar, F. Bach under submission to International Journal of Computer Vision.

Chapter 1

Introduction

In this chapter, we introduce the four main topics related to the manuscript: *probabilistic graphical models*, *submodular functions*, *matroids* and *convex optimisation*. We also state some of the basic results from literature that we use. Probabilistic graphical models is more relevant to the Chapter 2 and Chapter 3 of the manuscript. Submodular functions are related to Chapter 3, Chapter 4 and Chapter 5 as they deal with submodular optimisation. We use concepts of greedy algorithms on polytopes related to matroids in Chapter 2. And convex optimisation has its presence in all the chapters of the manuscript.

1.1 Probabilistic graphical models

Probabilistic distributions on a set of *discrete* random variables represent a confidence measure with which the random variables take a particular value. The corresponding probability mass function is often a table with all possible outcomes and their confidences respectively. Such representations of general probabilistic distributions without any assumptions of interdependence between random variables is exponentially large in the number of variables. This leads to intractable inference tasks with complexity exponential in the number of variables.

This complexity may be considerably reduced by using knowledge such as conditional independence between random variables, which factorises the probability distribution into conditional probability distributions or potential functions. These are also represented by individual tables but are compact when compared to the huge table of general probability distributions.

Graphical models provide a versatile set of tools to encode the local interactions between interdependent random variables leading to well-defined probability distributions [Murphy, 2012, Studeny, 2010, Koller and Friedman, 2009]. There are two basic classes of graphical models. They are:

1. Directed acyclic graphs, also known as Bayesian networks [Pearl, 2000],
2. Undirected graphs, also known as Markov random fields [Lauritzen, 1996].

Directed acyclic graphs factorise probability distributions into conditional probabilities while undirected graphs factorise them into product of potential functions.

In general, each of these graphical models encode different types of conditional independences among random variables [Studeny, 2010]. However, there are a subclass of conditional independences, which can be encoded by both directed acyclic graphs and undirected graphical models. Such undirected graphical models have a characteristic called *decomposability* and are referred to as decomposable graphs.[1] See Figure 1-1.

Note that the factorisation of probability distributions based on conditional independences also extends to continuous random variables. However, the representation in this case is in the form of parameters that represent the corresponding probability density function. As we mostly deal with discrete random variables in the manuscript, we use them to explain the concepts.

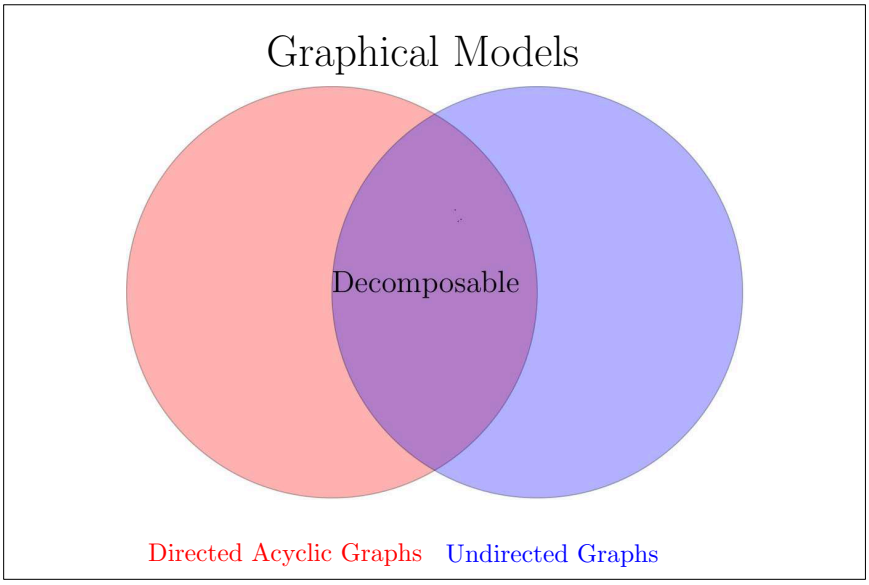


Figure 1-1 – Conditional independences of which some can be represented by directed acyclic graphs or undirected graphs. Decomposable graphs are undirected graphs, which can represent the conditional independences that can also be represented by directed acyclic graphs.

1.1.1 Graphical models and factorisation

In this section, we review the concepts of factorisability of probability distributions for a given graphical model. We also introduce notation, which we consistently use through out the manuscript.

Notations. We assume that a graphical model G is defined on a set of vertices $V = \{1, 2, \ldots, n\}$. Note that G is used to represent both directed acyclic graphs and undirected graphs. The nature of the graph is either mentioned explicitly or can be

1. Also called chordal or triangulated graphs.

understood from context. We consider n random variables $X_1, \ldots, X_n$ (referred to as X), associated with each vertex indexed by V.

We represent any general joint probability distribution on X as $p(x)$, where x is an instance in the domain of X, denoted by $\mathcal{X}$. Let $p_G(x)$ denote the projection of $p(x)$ onto a class of probability distributions that satisfy the conditional independence properties encoded by the graph G. We refer to p_G also as *graph-represented probability distribution* or *factorisable distribution* in this manuscript. As commonly done in graphical models literature, we overload the notation. We represent the marginal distribution of the set of random variables C, which is a subset of V by $p(x_C)$ instead of $p_C(x_C)$ for brevity. We also represent the conditional distribution of $x_1|x_2$, also read as "x_1 given x_2" by $p(x_1|x_2)$. The *Shannon entropy* of a probability distribution $p(x)$ on a set of random variables associated C is called $H(C)$ and is defined as [Cover and Thomas, 2006]:

$$H(C) = -\sum_{x_C \in \mathcal{X}_C} p(x_C) \log p(x_C).$$

Similarly, we use $H_G(C)$ to represent the Shannon entropy calculated based on the graph-represented probability distribution $p_G(x)$.

Directed acyclic graphs

Given a directed acyclic graph G defined on the set of vertices V, let $\pi_i(G)$ denote the parents of the node i in the graph G. The probability distribution p_G based on the directed acyclic graph G thus factorises as

$$p_G(x) \stackrel{\text{def}}{=} \prod_{i=1}^{n} p(x_i|x_{\pi_i(G)}) \tag{1.1}$$

Therefore, the entropy based on the factorisable probability distribution $p_G(x)$ also decomposes as follows

$$\begin{aligned} H_G(V) &= -\mathbb{E}_{p_G(x)} \log p_G(x) \\ &= -\mathbb{E}_{p_G(x)} \log \prod_{i=1}^{n} p(x_i|x_{\pi_i(G)}) = -\mathbb{E}_{p_G(x)} \log \prod_{i=1}^{n} \frac{p(x_i, x_{\pi_i(G)})}{p(x_{\pi_i(G)})} \\ &= -\sum_{i=1}^{n} \left\{ \mathbb{E}_{p_G(x)} \log p(x_i, x_{\pi_i(G)}) - \mathbb{E}_{p_G(x)} \log p(x_{\pi_i(G)}) \right\} \\ &= \textstyle\sum_{i=1}^{n} \left\{ H(i \cup \pi_i(G)) - H(\pi_i(G)) \right\}. \end{aligned} \tag{1.2}$$

Given a directed acyclic graph in Figure 1-2-(a), the probability distribution based on the graph G factorises as

$$p_G(x) = p(x_1)p(x_2|x_1)p(x_3|x_1)p(x_4|x_2, x_3).$$

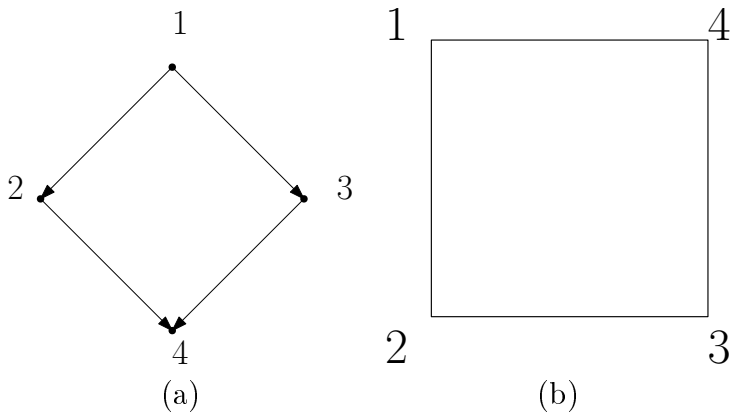


Figure 1-2 – (a) Directed acyclic graph and (b) Undirected graph on a a set of random variables associated to $V = \{1, 2, 3, 4\}$.

Undirected graphs

Let G now represent an undirected graph defined on the set of vertices V and let $\mathcal{C}(G)$ denote the *maximal cliques* of G. A clique is a subset of vertices that are completely connected in G and maximal cliques are cliques that lose their clique property if any adjacent vertex is added to the clique. The probability distribution p_G based on G now factorises as

$$p_G(x) \stackrel{\text{def}}{=} \frac{1}{Z} \prod_{C \in \mathcal{C}(G)} \psi_{X_C}(x_C), \tag{1.3}$$

where $\psi_{X_C}(x_C)$ is a *potential function* on the possible realisations x_C of the random variables X_C and Z is the normalization factor given by $Z = \sum_x \prod_{C \in \mathcal{C}(G)} \psi_{X_C}(x_C)$, also known as the *partition function*. The potential functions are assumed to be non-negative, real-valued functions, but are otherwise arbitrary and need not necessarily be probability distributions. Given an undirected graph in Figure 1-2-(b), the joint probability distribution factorises as

$$p_G(x) = \frac{1}{Z} \psi_{X_1,X_2}(x_1, x_2) \psi_{X_2,X_3}(x_2, x_3) \psi_{X_3,X_4}(x_3, x_4) \psi_{X_1,X_4}(x_1, x_4).$$

Decomposable graphs

An undirected graph G defined on the set V is *decomposable* when the set V can be partitioned into non-empty subsets (A, B, S) such that

1. S is the minimal set that separates the sets A and B, i.e., all paths from each node of the set A to each node of set B passes through at least one of the nodes that belong to S,
2. S is complete, i.e., all nodes of S are connected to each other, and
3. this decomposition induces subgraphs on the sets $A \cup S$ and $B \cup S$ that are *decomposable*.

For instance, any tree is a decomposable graph where any node of the graph is a minimal separator. When we remove a node from the graph, we partition the tree into two disjoint subgraphs, which are trees. Infact, all decomposable graphs have tree like characteristics that we will see now.

Every decomposable graph G can be represented by a *junction tree* [Wainwright and Jordan, 2008], i.e., a clique tree whose vertices are the maximal cliques $\mathcal{C}(G)$ and the unique path between any two maximal cliques A and B of the clique tree always contains $A \cap B$. This is called the *running intersection property*.

Let $\mathcal{T}(G)$ denote the edges of the junction tree over the set of cliques $\mathcal{C}(G)$. The probability distribution p_G of the random variables based on the decomposable graph G factorises [Lauritzen, 1996] as

$$p_G(x) \stackrel{\text{def}}{=} \frac{\prod_{C \in \mathcal{C}(G)} p(x_C)}{\prod_{(C,D) \in \mathcal{T}(G)} p(x_{C \cap D})} \ . \tag{1.4}$$

Similar to directed acyclic graphs, the entropy of the probability distribution $p_G(x)$ decomposes as follows:

$$\begin{aligned} H_G(V) &= -\mathbb{E}_{p_G(x)} \log p_G(x) \\ &= -\mathbb{E}_{p_G(x)} \log \frac{\prod_{C \in \mathcal{C}(G)} p(x_C)}{\prod_{(C,D) \in \mathcal{T}(G)} p(x_{C \cap D})} \\ &= -\sum_{C \in \mathcal{C}(G)} \mathbb{E}_{p_G(x)} \log p(x_C) - \sum_{(C,D) \in \mathcal{T}(G)} \mathbb{E}_{p_G(x)} \log p(x_{C \cap D}) \\ &= \textstyle\sum_{C \in \mathcal{C}(G)} H(C) - \sum_{(C,D) \in \mathcal{T}(G)} H(C \cap D). \end{aligned} \tag{1.5}$$

The *treewidth* of G is the maximal size of the cliques in G, minus one. For more details, see [Bishop et al., 2006, Wainwright and Jordan, 2008, Koller and Friedman, 2009, Lauritzen, 1996].

Let us consider a directed acyclic graph which encodes the same conditional independences as the decomposable graph with treewidth k. Therefore, the maximum neighbors for any node in the decomposable graph is k, which implies that the number of parents in the directed acyclic graph for any node $i \in V$ is at most k. Thus, the conditional probability distributions $p(x_i|x_{\pi_i(G)})$ is a table of length that is atmost exponential in k as the entries of the table are all possible values $(x_i, x_{\pi_i(G)})$ can take, which is given by their domain $(\mathcal{X}_i, \mathcal{X}_{\pi_i(G)})$. Depending on the inference task we devise an elimination of nodes from the graph, which can be done using perfect elimination ordering by removing "nodes without children" for a directed acyclic graph. This is equivalent to performing a summation ("sum-product message passing" algorithm for inference task of marginalisation) or finding a maximum ("max-product message passing" algorithm for inference task of estimating the mode) among the rows of the tables. This process only reduces the number of the entries in the table. Therefore, the maximum size of the table we deal with is at most exponential in k, which is the treewidth of the decomposable graph or equivalently the number of parents in a directed acylic graph. Thus, performing inference on a decomposable graph is always

exponential in treewidth of the graph.

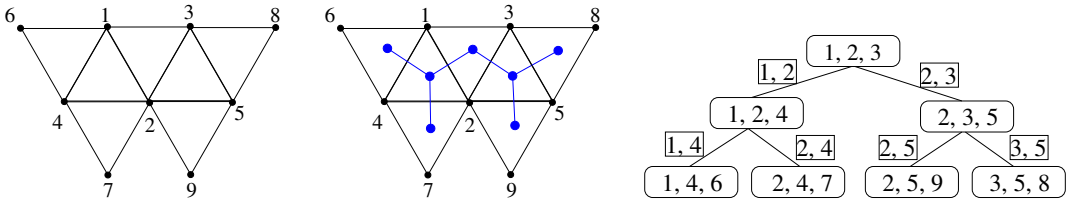


Figure 1-3 – (a) A decomposable graph on the set of vertices $V = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ having treewidth 2.(b) A junction tree embedded on the decomposable graph representing the maximal cliques by blue dots and the separator sets by blue lines. (c) The corresponding junction tree representation of the decomposable graph with ovals representing the maximal cliques and the rectangles representing the corresponding separator set.

Given a decomposable graph G in Figure 1-3, the treewidth of the graph is 2. The corresponding joint distribution factorises as

$$p_G(x) = \frac{p(x_1, x_2, x_3)p(x_1, x_2, x_4)p(x_2, x_3, x_5)p(x_1, x_4, x_6)p(x_2, x_4, x_7)p(x_2, x_4, x_9)p(x_3, x_5, x_8)}{p(x_1, x_2)p(x_2, x_3)p(x_1, x_4)p(x_2, x_4)p(x_2, x_5)p(x_3, x_5)},$$

Note that a separator set may occur more than once. For instance, if we consider a decomposable graph with maximal cliques: $\{\{1, 2, 3\}, \{1, 2, 4\}, \{1, 2, 5\}\}$ then the corresponding probability distribution factorises as

$$p_G(x) = \frac{p(x_1, x_2, x_3)p(x_1, x_2, x_4)p(x_1, x_2, x_5)}{p(x_1, x_2)^2}.$$

Remarks.

Any general joint probability distribution can be represented by a complete directed acyclic graph or a completely connected undirected graph, i.e., when G is a complete graph then $p_G(x) = p(x)$. Note that both directed acyclic graphs and undirected graphs represent different classes of conditional independences among the random variables. See [Koller and Friedman, 2009, Studeny, 2010] for in depth study of the differences in the conditional independences for both these models. For instance, there is no undirected graphical model that can represent the conditional independences encoded by the directed acyclic graph in Figure 1-2-(a). Similarly, there is no directed acyclic graph that can represent the conditional independences encoded by the undirected graph in Figure 1-2-(b). The Shannon entropy of any joint probability distribution is a submodular function. In Chapter 3, we use the property that entropy of a factorisable distribution represented by a graph over a set of random variables always bounds the entropy of any general distribution and generalise it to general submodular functions.

1.2 Submodular set functions

Submodular functions have been an important class of set functions in various areas of research such as economics, operations research and game theory. Most recently it has found applications in computer vision [Kolmogorov, 2006, Komodakis et al., 2011] and machine learning [Krause and Guestrin, 2005, Lin and Bilmes, 2011] as well. They are a class of discrete functions that can be minimised exactly in polynomial time. Maximising general submodular functions is NP-hard but admit constant factor approximation algorithms. In this section, we introduce the notations and basics related to submodular functions. We review the main properties of submodular functions. For a more complete treatment see [Bach, 2013, Fujishige, 2005].

Notations. Let us consider a set function F defined on a any subset of the groundset V with n elements , i.e., $V = \{1, \ldots, n\}$ and $F : 2^V \to \mathbb{R}$. Let us consider a unit hypercube in n dimensional space, $\{0, 1\}^n$. Each vertex of the hypercube is in bijection with a subset of V and represents its indicator vector as shown in Figure 1-4-(a). Therefore, the domain of F can also be represented by the vertices of the hypercube, i.e., $F : \{0, 1\}^n \to \mathbb{R}$.

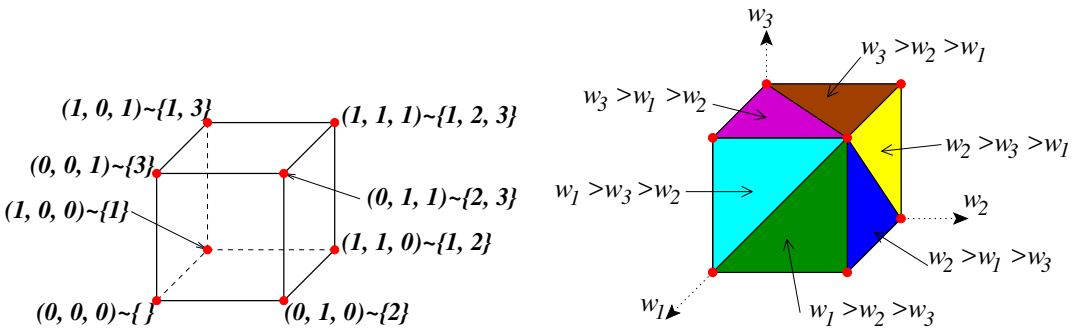


Figure 1-4 – (a) The bijection between vertices of the hypercube and subsets of $V = \{1, 2, 3\}$. (b) Division of the hypercube into simplices based on the ordering of the components of $w \in \{0, 1\}^3$. Figures are by courtesy of Bach [2013].

Lovász extension of a set function. Let f be the Lovász extension of F, which is defined on the complete hypercube, $[0, 1]^n$. The hypercube can be divided into $n!$ simplices based on the order of the components of any vector w in the hypercube, i.e, $w \in [0, 1]^n$ as shown in Figure 1-4-(b). Note that each vertex of each simplex coincides with a vertex of the hypercube, which belongs to the domain of F. The value of the Lovász extension at w, i.e. $f(w)$ can be estimated by interpolating the values of the set function F at the vertices of selected simplex as follows:

Definition 1. *Given any set function F and $w \in [0, 1]^n$ such that $w_{j_1} \geqslant \cdots \geqslant w_{j_n}$,*

we define its Lovász extension $f : [0,1]^n \to \mathbb{R}$ *as*

$$f(w) = \sum_{k=1}^{n-1} (w_{j_k} - w_{j_{k+1}}) F(\{j_1, \ldots, j_k\}) + w_{j_n} F(\{j_1, \ldots, j_n\})$$

Note that $(j_1, \ldots, j_n)$ gives the order of the components of w that determines the simplex. Extreme points of the simplex are indicator vectors of $\{j_1\}$, $\{j_1, j_2\}, \ldots,$ $\{j_1, \ldots, j_n\}$ in Figure 1-4-(b). At the vertices of the hypercube, both the set function and its Lovász extension have the same values, i.e. $f(1_A) = F(1_A), \forall A \subseteq V$, where 1_A represents the indicator vector of the set A. The Lovász extensions of any set functions is piecewise affine and positively homogeneous. Therefore, this can also be extended from the hypercube $[0,1]^n$ to a real subspace $\mathbb{R}^n$. See [Bach, 2013, Murota, 2003].

Submodular function. We now define submodular functions.

Definition 2. *A set function* F *is submodular if* $\forall A, B \subseteq V$

$$F(A) + F(B) \geq F(A \cup B) + F(A \cap B).$$

An equivalent definition, which uses the *diminishing returns property* is given by:

Definition 3. *A set function* F *is submodular if* $\forall S \subseteq T \subseteq V$ *and* $k \notin T$

$$F(S \cup \{k\}) - F(S) \geq F(T \cup \{k\}) - F(T).$$

Both theses definitions are equivalent by choosing $A = S \cup \{k\}$ and $B = T$ and rearranging the terms. In this manuscript, we consider only *normalized* submodular functions, i.e., $F(\varnothing) = 0$. The subclass of submodular functions that satisfy equality in the definitions are called modular functions. They are of the form $u \in \mathbb{R}^n$ and $u(A) = u^\top 1_A$. Note that even $-u$ is a modular function and hence submodular.

Submodular functions are intimately linked to convexity through their Lovász extensions. This is established by the following propositions due to Lovász [1982].

Proposition 1. *A set function,* F *is submodular if and only if its Lovász extension,* f *is convex.*

Proposition 2. *Let* F *be a submodular function and* f *its Lovász extension; then*

$$\min_{A \subseteq V} F(A) = \min_{w \in \{0,1\}^n} f(w) = \min_{w \in [0,1]^n} f(w). \tag{1.6}$$

Convex duality. The Lovász extension $f(w)$ is convex and absolutely homogeneous, that is, for any $w \in \mathbb{R}^n$ and $\lambda \in \mathbb{R}$, $f(\lambda w) = |\lambda| f(w)$. For all such functions, there exists a centrally symmetric convex body $K \subset \mathbb{R}^n$ such that for all $w \in \mathbb{R}^n$ [Rockafellar, 1997, §13],

$$f(w) = \max_{s \in K} s^\top w.$$

Note that when $f(w)$ happens to be equal to zero only for $w = 0$, then f is a norm and the set K is simply the unit ball of the dual norm.

Since f is piecewise affine, the set K is a *polytope* (i.e., the convex hull of finitely many points). The set K may be described precisely for general submodular functions and is the *base polytope* of the submodular function [Fujishige, 2005, Bach, 2013].

Submodular and base polyhedra. Let us now introduce polyhedra associated to submodular functions that further help us understand the links between submodularity and convexity.

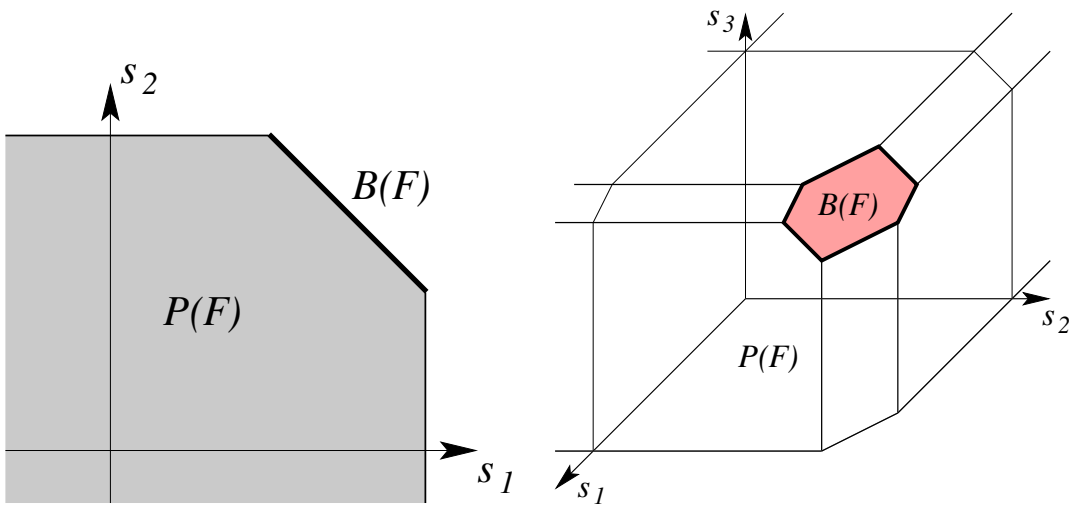


Figure 1-5 – Submodular polyhedron, $P(F)$ and Base polyhedron, $B(F)$ for (a) $V = \{1, 2\}$ and (b) $V = \{1, 2, 3\}$. Figures are by courtesy of Bach [2013].

Definition 4. *Given a submodular function F, the corresponding submodular polyhedron, $P(F)$ and the base polyhedron, $B(F)$ are defined as*

$$\begin{aligned} P(F) &= \big\{ s \in \mathbb{R}^p, \ \forall A \subseteq V, s(A) \leqslant F(A) \big\}, \\ B(F) &= \big\{ s \in \mathbb{R}^p, \ \forall A \subset V, s(A) \leqslant F(A), s(V) = F(V) \big\}. \end{aligned}$$

Maximising linear functions on these polyhedra may be done using a greedy algorithm. We formally state this from Bach [2013] but the result is due to Edmonds [2003].

Proposition 3. *Let F be a normalized submodular function, i.e., $F(\varnothing) = 0$. Let $w \in \mathbb{R}^n$, with components ordered in decreasing order, i.e. $w_{j_1} \geqslant \cdots \geqslant w_{j_n}$ and define $s_{j_k} = F(\{j_1, \ldots, j_k\}) - F(\{j_1, \ldots, j_{k-1}\})$. Then $s \in B(F)$ and*

1. *if $w \in \mathbb{R}^n_+$, s is a maximizer of $\max_{s \in P(F)} w^\top s$ and $\max_{s \in P(F)} w^\top s = f(w)$.*
2. *s is a maximizer of $\max_{s \in B(F)} w^\top s$ and $\max_{s \in B(F)} w^\top s = f(w)$.*

It can be seen that the Lovász extension $f(w)$, is the support function of the base polytope, $B(F)$. We use these results extensively in this manuscript.

$$f(w) = \max_{s \in B(F)} w^\top s \tag{1.7}$$

Operations that preserve submodularity. The set of submodular functions is a cone. Therefore, submodularity is preserved under addition and multiplication with positive scalars. There are other operations such as contraction, restriction and extension, which preserve submodularity. For more details please refer to [Bach, 2013, Fujishige, 2005] and references therein.

1.2.1 Maximising submodular functions

Let us consider a normalized submodular function, $F : 2^V \to \mathbb{R}$ with $F(\varnothing) = 0$. We consider the problem of maximising submodular functions of the form

$$\max_{A \subset V} F(A), \tag{1.8}$$

which we consider in Chapter 3 and only use definition of submodular set functions.

1.2.2 Submodular minimisation

Any general submodular function can be decomposed into a normalised submodular functions, i.e., $F : 2^V \to \mathbb{R}$ with $F(\varnothing) = 0$ and a modular function, $u \in \mathbb{R}^n$. Therefore, any general submodular function can be denoted by $F - u : 2^V \to \mathbb{R}$. In the context of submodular minimisation, as we shall see, we deal with projections onto a base polytope. Note that $B(F-u)$ has the exact same structure as $B(F)$, geometrically, but translated by u. For instance, projecting 0 onto $B(F-u)$ is equivalent to projecting u onto $B(F)$. Therefore, using $F - u$ to represent general submodular functions does not deal with operations such as translations of the base polytopes and gives us better understanding of the algorithms.

In Chapter 4 and Chapter 5, we consider solving a general submodular minimisation of the form:

$$\min_{A \subset V} F(A) - u(A). \tag{1.9}$$

Using the Lovász extension the corresponding continuous optimisation problem is

$$\min_{w \in [0,1]^n} f(w) - u^\top w, \tag{1.10}$$

and Eq. (1.9), Eq. (1.10) have the same optimal solution. The dual optimisation problem can be derived as

$$\min_{A \subseteq V} F(A) - u(A) \quad = \quad \min_{w \in \{0,1\}^n} f(w) - u^\top w$$

$$
\begin{aligned}
&= \min_{w\in[0,1]^n} f(w) - u^\top w \\
&= \min_{w\in[0,1]^n} \max_{s\in B(F)} s^\top w - u^\top w \text{ using Eq. (1.7)} \\
&= \max_{s\in B(F)} \min_{w\in[0,1]^n} s^\top w - u^\top w \\
&= \max_{s\in B(F)} \sum_{i=1}^n \min\{s_i - u_i, 0\}. \qquad (1.11)
\end{aligned}
$$

This dual problem allows to obtain certificates of optimality for the primal-dual pairs $w \in [0,1]^n$ and $s \in B(F)$ using the quantity,

$$\text{gap}(w,s) := f(w) - u^\top w - \min_{i=1}^n \{s_i - u_i, 0\},$$

which is always non-negative. It is equal to zero only at optimal and the corresponding (w,s) form the optimal primal-dual pairs. Note that the convex optimisation problems in Eq. (1.10) and Eq. (1.11) are convex but non-smooth.

Smooth minimisation problem. Here, we consider the optimisation problem in Eq. (1.10) and get rid of the box constraints, i.e., $[0,1]^n$ and add a quadratic penalty to the cost function, i.e.,

$$\min_{w\in\mathbb{R}^n} f(w) - u^\top w + \frac{1}{2}\|w\|_2^2. \qquad (1.12)$$

As a consequence of the representation of f as a support function leads to the following primal/dual pair [Bach, 2013, Sec. 8]:

$$
\begin{aligned}
&\min_{w\in\mathbb{R}^n} f(w) - u^\top w + \tfrac{1}{2}\|w\|_2^2 \\
= &\min_{w\in\mathbb{R}^n} \max_{s\in B(F)} s^\top w - u^\top w + \tfrac{1}{2}\|w\|_2^2 \text{ using Eq. (1.7)}, \\
= &\max_{s\in B(F)} \min_{w\in\mathbb{R}^n} s^\top w - u^\top w + \tfrac{1}{2}\|w\|_2^2, \\
= &\max_{s\in B(F)} -\tfrac{1}{2}\|s-u\|_2^2, \qquad (1.13)
\end{aligned}
$$

with $w = u - s$ at optimal. The primal in Eq. (1.12) is strongly convex and hence the dual in Eq. (1.13) is smooth.

The non-smooth optimisation problem in Eq. (1.10) and the smooth optimisation problem in Eq. (1.12) are tightly connected. Indeed, given the unique solution w of the Eq. (1.12), then we obtain a solution of $\min_{A\subseteq V} F(A) - u(A)$ by thresholding w at 0, i.e., by taking $A = \{i \in V, w_i \geqslant 0\}$ [Fujishige, 1980, Chambolle and Darbon, 2009]. Solving the dual problem in Eq. (1.13) is of clear interest in approximate Bayesian inference in log-supermodular models [Djolonga and Krause, 2014, 2015].

Conversely, one may solve the smooth problem in Eq. (1.12) by assuming oracles that solve the non-smooth problems of the form Eq. (1.10). The original divide-and-conquer algorithm may involve $O(n)$ oracle calls [Groenevelt, 1991]. The extended

algorithm of [Jegelka et al., 2013] can reach a precision ε in $O(\log \frac{1}{\varepsilon})$ but can only get the exact solution after $O(n)$ oracle calls. We provide another extended iterative algorithm to optimise the smooth problem assuming oracles of non-smooth problems in Chapter 5.

In the next section, we consider a specific class of submodular functions known as cut functions. We show that all the above optimisation problems are related in the context of cut functions to provide intuition. Note that the links between different optimisation problems holds for all submodular functions.

1.2.3 Cut functions and total variation denoising.

In Chapter 4, we focus on a subclass of submodular minimisation problems also referred to as energy minimization problems with pairwise potentials,

$$E(w) = -\sum\nolimits_{i=1}^{n} u_i w_i + \sum\nolimits_{i,j=1}^{n} \psi_{ij}(w_i, w_j), \tag{1.14}$$

where the variables w_i take values in a set of discrete labels. For simplicity, we here focus on binary labels, $w_i \in \{0, 1\}$. We assume the pairwise potentials to be submodular, i.e., $\psi_{ij}(0,1) + \psi_{ij}(1,0) \geq \psi_{ij}(0,0) + \psi_{ij}(1,1)$. (One may extend to non-submodular potentials via roof duality [Rother et al., 2007]). It is well known that all such submodular energy functions may be written as graph cut functions with nonnegative "edge weights" a_{ij}, up to a constant [Picard and Ratliff, 1975]:

$$E(w) = -\sum\nolimits_{i=1}^{n} u_i w_i + \sum\nolimits_{i,j=1}^{n} a_{ij}|w_i - w_j| + \text{const.} \tag{1.15}$$

This function consists of two parts: (1) a sum of unary potentials $-\sum_{i=1}^{n} u_i w_i = -u^\top w$; and (2) the sum of pairwise potentials, which is equivalent to a weighted graph cut between the set of indices i in $\{1, \ldots, n\}$ for which $w_i = 1$, and its complement. For $w \in \mathbb{R}^n$, this sum is the *total variation function*

$$f(w) \stackrel{\text{def}}{=} \sum_{i,j=1}^{n} a_{ij}|w_i - w_j|.$$

Note that this is a case of anisotropic weighted total variation. Since the weights a_{ij} are non-negative, the function f is convex. We refer to the graph cut problem as the *discrete problem*:

$$\min_{w \in \{0,1\}^n} f(w) - u^\top w, \tag{D}$$

which is precisely the submodular minimisation problem in Eq. (1.9).

We obtain a relaxation to the combinatorial problem in Eq. (D) by replacing $\{0, 1\}^n$ by its convex hull $[0, 1]^n$:

$$\min_{w \in [0,1]^n} f(w) - u^\top w. \tag{C}$$

We refer to Eq. (C) as the *continuous problem*. This relaxation is *exact*: since the continuous convex problem in (C) is a minimization problem over a larger set than the discrete problem, its minimal value has to be lower than (D). However, as a consequence of properties of the total variation and its relation to submodular graph cut functions (see, e.g., [Bach, 2013, Sec. 3.3] or [Hochbaum, 2001, Chambolle and Darbon, 2009] for a proof dedicated to cut functions), the two optimal values are equal and a solution to (D) may be obtained from a solution $x \in [0,1]^n$ of (C) by looking at all "level sets" of w, that is by rounding the values of w to zero or one by thresholding at a given level in $[0,1]$ (there are at most n possible thresholds, which can be obtained by first sorting the components of w).

The corresponding dual optimisation problem is given by Eq. (1.11). It is given by

$$\max_{s \in K} \sum_{i=1}^{n} \min\{s_i - u_i, 0\}.$$

While the cut problem is now reformulated as a convex optimization problem, it is still hard to minimize because neither the primal nor the corresponding dual are smooth, and thus iterative methods are typically slow (see detailed comparisons by Jegelka et al. [2013]). We now reformulate the problem so that the dual problem becomes smooth and potentially easier to optimize.

Equivalence to total-variation denoising

Following [Fujishige, 2005, Chambolle and Darbon, 2009, Bach, 2013, Nagano et al., 2011, Jegelka et al., 2013], we consider the *total variation denoising problem*:

$$\min_{w \in \mathbb{R}^n} f(w) + \frac{1}{2}\|u - w\|^2. \tag{TV}$$

By expanding $\frac{1}{2}\|u - w\|^2$ into $\frac{1}{2}\|u\|^2 - w^\top u + \frac{1}{2}\|u\|^2$, we see that going from the continuous problem (C) to (TV) means replacing the constraint $w \in [0,1]^n$ by the penalty $\frac{1}{2}\|w\|^2 = \frac{1}{2}\sum_{i=1}^n w_i^2$. This has a number of important consequences:

1. It makes the optimization problem strongly convex and thus the dual problem will be smooth.
2. A solution to (D) and hence (C) may be obtained by *thresholding* the unique solution w of (TV) at zero, that is, by defining $\hat{w}_i = 1$ if $w_i > 0$ and $\hat{w}_i = 0$ otherwise. This is usually not true for arbitrary convex functions f (even absolutely homogeneous) and is a direct consequence of submodularity.

Importantly, we *need not* solve the TV problem (TV) exactly to obtain a solution to (C), we only need to know which of the components are positive (resp. negative).

Let us recall the dual of the TV problem from Eq. (1.13)

$$\max_{s \in K} -\tfrac{1}{2}\|s - u\|_2^2.$$

The primal and dual solutions w and s have a simple correspondence $s = u - w$,

as can be seen from the dual derivation. We have now obtained a dual problem which may be simply interpreted as the *orthogonal projection* of the vector w onto the polytope K. The total variation relaxation (TV) can be solved by solving a series of discrete energy minimization problems (D). However, if we have a (TV) solved by solving the equivalent problem of projection onto K, then thresholding the solution at 0 solves the discrete problem (D). Therefore, any fast subroutine for any of these problems implies a fast subroutines for the other two.

While the reformulation is intuitive, orthogonal projections onto the polytope K are not fast to compute in general. Many special cases, however, are fast, including graphs that have fast cut subroutines (which may be combinatorial, e.g., max-flow or message passing). In Chapter 4, we solve this projection problem for cut functions using several optimisation algorithms and compare their performance with state-of-art discrete algorithms. In Chapter 5, we propose an iterative algorithm to solve the projection problem for general submodular functions. Solving the discrete optimisation problem (D) by solving (TV) allows us to use tools from convex optimisation, which naturally lead to algorithms that are easy to parallelise.

1.3 Matroids

In this section, we introduce matroids and related concepts that we use in Chapter 2. They were introduced by Whitney [1935] to characterise an abstract concept of independent sets of a ground set. We will define this formally now.

Definition 5. *Let E be a finite set. If a family $\mathcal{I}$ of subsets of E satisfies:*

1. $\varnothing \in \mathcal{I}$,
2. *"hereditary property"* : $I_1 \subseteq I_2 \in \mathcal{I} \implies I_1 \in \mathcal{I}$,
3. *"exchange property"* : $I_1, I_2 \in \mathcal{I}, |I_1| < |I_2| \implies \exists e \in I_2 \setminus I_1 : I_1 \cup \{e\} \in \mathcal{I}$,

then the pair $(E, \mathcal{I})$ is a matroid.

Each $I \in \mathcal{I}$ is called an *independent set* of the matroid $(E, \mathcal{I})$ and $\mathcal{I}$ is the *family of independent sets* of the matroid $(E, \mathcal{I})$.

Graphic matroid. We use graphic matroid as a running example to explain the concepts in this section. A graphic matroid is given by a pair $(E, \mathcal{I})$, where E is the set of edges of a graph $G = (V, E)$ and $\mathcal{I}$ is the family of subsets of the edges in E that do not form a loop. See Figure 1-6 for an example. It may be quickly verified that $\mathcal{I}$ satisfies all the conditions in Definition 5.

Any independent set which is maximal in family of subsets $\mathcal{I}$ with respect to set inclusion, i.e., it is not included in any other independent set is called a *base*. In the example of graphic matroid, all the *spanning trees*, i.e. the middle row in Figure 1-6-(b) form the bases as they are not included in any other independent set.

The submodular polyhedron of a graphic matroid is convex hull of indicator functions of the family of independent sets $\mathcal{I}$ and its base polyhedron is the convex hull of indicator function of all possible spanning trees. Using Proposition 3, optimising

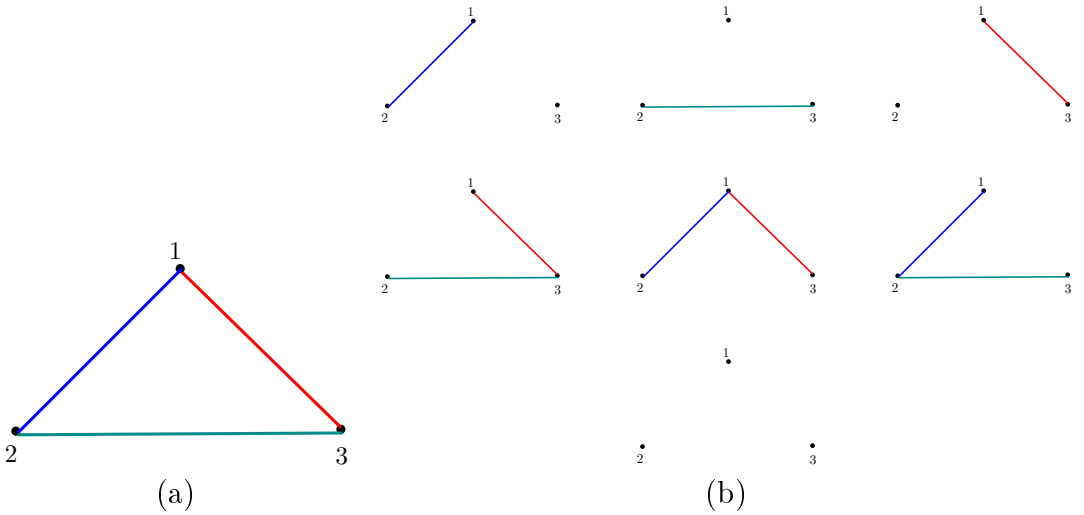


Figure 1-6 – Graphic Matroid is given by (a) the ground set is the set of edges E of the graph $G(V, E)$, i.e., $E = \{(1, 2), (2, 3), (1, 3)\}$ and (b) the family of independent sets $\mathcal{I}$, i.e., the subsets of the edges E that do not form a *loop*.

linear functions on these polyhedra may be done using a greedy algorithm [Edmonds, 2003]. For a graphic matroid, Kruskal [1956] proposed a non-polyhedral version of the greedy algorithm, which we use in Section 2.4.

1.4 Convex optimisation

In this section, we introduce basics of convex optimisation that we use in this manuscript.

Convex set. A subset K of $\mathbb{R}^n$ is said to be *convex* if:

$$\forall x, y \in K, \forall t \in [0, 1], (1 - t)x + ty \in K. \tag{1.16}$$

Convex cone. A subset K of $\mathbb{R}^n$ is said to be a *convex cone* if:

$$\forall x, y \in K, \alpha, \beta > 0, \alpha x + \beta y \in K.$$

Convex functions. Let us consider a function f defined on a convex set $K \subseteq \mathbb{R}^n$, i.e. the function is defined on elements of the convex set K. If we plot the function in $\mathbb{R}^{n+1}$ with $(x, f(x))$ for all x in the set K and $(x, +\infty)$ for all x that are not in K, then we get a curve. All the space that lies above this curve, is called its *epigraph*. Intuitively, the function f is convex if a line joining any two points on this curve always lies over the curve. Formally, this can be written as

$$\forall x, y \in K, \forall \alpha \in [0, 1], f(\alpha x + (1 - \alpha)y) \leq \alpha f(x) + (1 - \alpha) f(y).$$

The function is said to be *strictly* convex if

$$\forall x, y \in K, \forall \alpha \in [0,1], f(\alpha x + (1-\alpha)y) < \alpha f(x) + (1-\alpha) f(y).$$

In other words, a function f is convex if and only if its epigraph is a convex set. Note also that a function f is said to be *concave* if $-f$ is convex. In the process of plotting the function, it is important to note that we extended the function from a convex set K to a real subspace $\mathbb{R}^n$. Therefore, we can assume f is defined everywhere in $\mathbb{R}^n$. This is called an *extended function*. We will assume a extended function of f from now on to explain other concepts of convex functions.

Subgradients and gradients. Let $g \in \mathbb{R}^n$ be a vector. It is called the *subgradient* to the convex function f at x, if

$$\forall y \in \mathbb{R}^n, f(y) - f(x) \geq g^\top (y - x)$$

All the subgradients of f at x are denoted by $\partial f(x)$. A convex function always admits a non-empty subgradient set [Rockafellar, 1997].

If f is differentiable in the $\mathbb{R}^n$, then it is called a *smooth function.*. We denote the *gradient* of f at x by $\nabla f(x)$, which is composed of partial derivatives $\frac{\partial f(x)}{\partial x_i}$ of f at x along the i-th vector of the canonical basis. When f is differentiable the partial derivative and the gradient are identical.

Minimum of convex functions. One of the basic facts of convex functions is that the local minimum is the global minimum [Rockafellar, 1997, Boyd and Vandenberghe, 2004]. And if the function f obtains at minimum at x then $0 \in \partial f(x)$.

Fenchel conjugate. For a function f, its Fenchel conjugate f^* [Fenchel, 1949] is given by:

$$f^*(p) = \sup_{x \in \mathbb{R}^n} p^\top x - f(x) \tag{1.17}$$

We also refer to this as *conjugate* of f. The biconjugate, i.e., conjugate of the conjugate of a continuous convex function gives the original function, i.e., $f^{**}(x) = f(x)$. Intuitively, the conjugate of a convex function encodes its epigraph, which is a convex set using its supporting hyperplanes. Let us consider the following functions related to a closed convex set K.

1. $f(x) = \max_{y \in K} x^\top y$, then f is called the *support function* of K.
2. $\iota(x) = \begin{cases} 0 & x \in K, \\ \infty & \text{otherwise,} \end{cases}$ then ι is called the *indicator function* of K.

f and ι are conjugate to each other, i.e., $f^* = \iota$ and $\iota^* = f$. We use this in several places through Chapter 4 and Chapter 5.

Convex optimisation problems. A problem is said to be a convex, if the goals is to minimise a convex function f on a convex set K of the form

$$\min_{x \in K} f(x) \tag{1.18}$$

If f is an affine function of the form $a^\top x + b$ and K is a polytope, then the problem in Eq. (1.18) is a linear program(LP). If f is a quadratic function then it is a quadratic program(QP). Maximising a concave function is equivalent to minimising a convex function and is also referred to as a convex optimisation problem.

Orthogonal projection onto a convex set. The operation of orthogonal projection of a point $x \in \mathbb{R}^n$ onto a convex set is denoted by Π_K, which is defined as

$$\Pi_K(x) = \arg\min_{y \in K} \frac{1}{2}\|x - y\|_2^2. \tag{1.19}$$

However, the complexity of the orthogonal projection depends on the complexity of the convex set.

1.4.1 Projected gradient method.

In this section we briefly describe the *projected gradient method* to optimise problems of the form Eq. (1.18) with a stopping criteria of maximum number of iterations T.

Algorithm 1 Generic projected gradient method to optimise $\min_{x \in K} f(x)$

Input: $x_1 \in K$, stepsize:$\alpha(t)$
for $t = 1$ **to** T **do**
  $x_{t+1} = \Pi_K(x_t - \alpha(t)\nabla f(x_t))$
end for

When f is non-differentiable, then we use subgradients instead of gradients. This is called *projected subgradient method*, a variant of which we use in Section 2.5.

1.4.2 Best approximation problems.

In this manuscript, we come across a class of optimisation problems called *best approximation problems* [Deutsch, 2001]. We briefly introduce the problems in their standard form. Given two polytopes K_1 and K_2, the best approximation problem is defined as

$$\min_{\substack{x \in K_1 \\ y \in K_2}} \frac{1}{2}\|x - y\|_2^2. \tag{1.20}$$

Note that the orthogonal projection in Eq. (1.19) is a variant of the best approximation problems. In this manuscript, we encounter best approximation problems

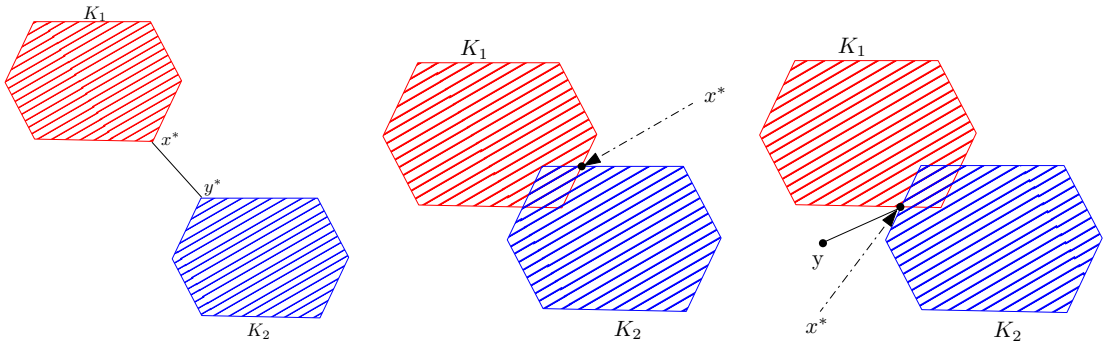


Figure 1-7 – (a) Best approximation problem in Eq. (1.21) when $K_1 \cap K_2 = \varnothing$. (b) Convex feasibility problem in Eq. (1.21). (c) Dykstra problem in Eq. (1.22).

where K_1 an K_2 are non-intersecting, i.e., $K_1 \cap K_2 = \varnothing$ in Chapter 4. See Figure 1-7-(a).

However, when K_1 and K_2 are intersecting polytopes, then $x = y$. This problem is referred to as the *convex feasibility problem* in literature [Bauschke and Borwein, 1996], which has the following canonical form,

$$\text{Find } x \text{ such that } x \in K_1 \cap K_2, \tag{1.21}$$

which is also shown in Figure 1-7-(b).

A variant of the convex feasibility problem is finding the nearest point to a point, $y \in \mathbb{R}^n$ in the intersection of convex polytopes. This is given by,

$$\min_{x \in K_1 \cap K_2} \frac{1}{2}\|x - y\|^2. \tag{1.22}$$

To be best of our knowledge, there is no specific name for Eq. (1.22) in literature. Therefore, we refer to this as the *Dykstra problem* in this manuscript to distinguish it from the other problems in Eq. (1.21) and Eq. (1.20). This is because these class of problems are often solved using Dykstra's alternating projection algorithms [Bauschke and Borwein, 1994]. See Figure 1-7-(c).

We now briefly describe the methods used to solve Eq. (1.20) in both cases of intersecting and non-intersecting convex polytopes.

Alternating projections (AP). The alternating projection algorithm [Bauschke and Borwein, 1996] was proposed to solve the convex feasibility problem, i.e., to obtain a feasible point in the intersection of the two polytopes. It is also used to solve the best approximation problem in the case of non-intersecting polytopes. Let z_0 be an arbitrary starting point. The update rule for z_t for each iteration t using the algorithm is given by

$$z_t = \Pi_{K_1}\Pi_{K_2}(z_{t-1}), \tag{1.23}$$

until $z_t = z_{t-1}$. In the case of non-intersecting polytopes, the pair $(z_t, \Pi_{K_2}(z_t))$ gives the points on the polytopes K_1 and K_2 respectively. We use variations of this in Chapter 4, which we describe there. This is equivalent to performing block coordinate descent in the dual.

Averaged Alternating reflections (AAR). The averaged alternating reflection algorithm [Bauschke and Luke, 2004], which is also known as Douglas-Rachford splitting is used to solve convex feasibility problem and the best approximation problem. We now introduce a reflection operator for the polytope K as R_K, i.e., $R_K = 2\Pi_K - I$, where I is an identity operator. Therefore, reflection of t on a polytope K is given by $R_K(t) = 2\Pi_K(t) - t$. Let z_0 be an arbitrary starting point. The update rule for z_t for each iteration t using the algorithm is given by

$$z_t = \frac{1}{2}(I + R_2 R_1)(z_{t-1}). \tag{1.24}$$

In case of intersecting polytopes, z_t converges to a point in the intersection. However, in the non-intersecting case, z_t is a diverging sequence. However, the projection of z_t onto the polytopes, i.e., $(\Pi_{K_1}(z_t), \Pi_{K_2}(z_t))$ gives the nearest points on the polytopes.

Dykstra's alternating projection. This alternating projection algorithm is mainly used to solve convex feasibility problems of the form Eq. (1.22) for a given $y \in \mathbb{R}^n$. However, this could also be used to solve the best approximation problem in Eq. (1.20) and convex feasibility problem in Eq. (1.21) by letting y to be an arbitrary point. It uses an auxiliary sequence (p_t, q_t). The updates are as follows after initializing $p_0 = q_0 = 0$ and $b_0 = y$.

$$\begin{aligned} a_t &= \Pi_{K_1}(b_{t-1} + p_{t-1}) \\ p_t &= b_{t-1} + p_{t-1} - a_t \\ b_t &= \Pi_{K_2}(a_t + q_{t-1}) \\ q_t &= a_t + q_{t-1} - b_t \end{aligned} \tag{1.25}$$

The sequence (a_t, b_t) converge to a point in the intersecting case and they converge to the nearest points on K_1 and K_2 in the non-intersecting case. This algorithm has a primal descent interpretation, i.e, as coordinate descent of a well formulated primal problem [Gaffke and Mathar, 1989]. One of the important variants of this is the accelerated Dykstra's alternating projection [Chambolle and Pock, 2015]. We derive these updates for specific problem in Chapter 5.

We will revisit these algorithms, sometimes with variations in Chapter 4 and Chapter 5 of this manuscript.

Chapter 2

Learning Bounded Treewidth Decomposable Graphs

Abstract

We consider the problem of learning the structure of undirected graphical models with bounded treewidth, within the maximum likelihood framework. This is an NP-hard problem and most approaches consider local search techniques. In this chapter, we pose it as a combinatorial optimization problem, which is then relaxed to a convex optimization problem that involves optimising linear functions over the graphic and hypergraphic matroids. A supergradient method is used to solve the dual problem, with a run-time complexity of $O(k^3 n^{k+2} \log n)$ for each iteration, where n is the number of variables and k is a bound on the treewidth. We compare our approach to state-of-the-art methods on synthetic datasets and classical benchmarks, showing the gains of the novel convex approach.

This chapter is based on our work "Convex Relaxations for Learning Bounded Treewidth Decomposable Graphs", K. S. Sesh Kumar and F.Bach, published in proceedings of International Conference on Machine Learning (ICML), 2013.

2.1 Goal

Goal. In this chapter, we consider the problem of learning the structure of undirected graphical models with bounded treewidth, under the maximum likelihood framework.

In many domains such as computer vision, natural language processing or bioinformatics, the structure of the graph follows naturally from the constraints of the problem at hand. In other situations, it might be desirable to estimate this structure from a set of observations. It allows

1. a statistical fit of rich probability distributions that can be considered for further use, and
2. discovery of structural relationship between different variables.

In the former case, distributions with tractable inference are often desirable, i.e., inference with run-time complexity that does not scale exponentially in the number of variables in the model. The simplest constraint to ensure tractability is to impose tree-structured graphs [Chow and Liu, 1968]. However, these distributions are not rich enough, and following earlier work [Malvestuto, 1991, Bach and Jordan, 2002, Narasimhan and Bilmes, 2004, Chechetka and Guestrin, 2007, Gogate et al., 2010, Szántai and Kovács, 2011], we consider models with bounded *treewidth*, not simply by one (i.e., trees), but by a small constant k as inference on decomposable graphs is $O(nr^k)$, where r is the number of values each random variable can take.

Motivation. Beyond the possibility of fitting tractable distributions (for which probabilistic inference has linear complexity in the number of variables), learning bounded-treewidth graphical models is a key to design approximate inference algorithms for graphs with higher treewidth. Indeed, as shown by [Saul and Jordan, 1995, Wainwright and Jordan, 2008, Kolmogorov and Schoenemann, 2012], approximating general distributions by tractable distributions is a common tool in variational inference. However, in practice, the complexity of variational distributions is often limited to trees (i.e., $k = 1$), since these are the only ones with exact polynomial-time structure learning algorithms. The convex relaxation designed in this work enables us to augment the applicability of variational inference, by allowing a finer trade-off between run-time complexity and approximation quality.

Related work. Learning the structure of a directed or undirected graphical model is often posed as an optimisation problem with model selection [Höfling and Tibshirani, 2009, Schmidt et al., 2007, Ravikumar et al., 2010, Koller and Friedman, 2009]. In this chapter, we consider learning the structure of a bounded treewidth graph, which naturally acts as a regulariser as there is bound in the number of parameters. Two types of algorithms have emerged using this criteria, based on the two equivalent definitions of graphical models: (a) by testing conditional independence relationships [Spirtes et al., 2001] or (b) by maximizing the log-likelihood of the data using the factorised form of the distribution [Friedman and Koller, 2003].

In the specific context of learning bounded-treewidth graphical models, the latter approach has been shown to be NP-hard [Srebro, 2002] and led to various approximate algorithms based on local search techniques [Malvestuto, 1991, Deshpande et al., 2001, Karger and Srebro, 2001, Bach and Jordan, 2002, Shahaf et al., 2009, Szántai and Kovács, 2011] while the former approach led to algorithms based on independence tests [Narasimhan and Bilmes, 2004, Chechetka and Guestrin, 2007, Gogate et al., 2010], which have recovery guarantees when the data-generating distribution has low treewidth. Malvestuto [1991] proposed a greedy heuristic of hyperedge selection with least incremental entropy. Deshpande et al. [2001] proposed a simple edge selection technique that maintains decomposability of the graph while minimizing the KL-divergence to the original distribution. Karger and Srebro [2001] proposed the first convex optimisation approach to learn the maximum weighted k-windmill, a subclass of the decomposable graph. Bach and Jordan [2002] gave an approach which

iteratively refines the hyperedge selection based on KL-divergence using iterative scaling. Shahaf et al. [2009] proposed another convex optimisation approach with Bethe approximation of the likelihood using graph-cuts. Szántai and Kovács [2011] proposed a hyperedge selection criteria based on high mutual information within a hyperedge. Narasimhan and Bilmes [2004] performs independence tests by solving submodular optimisation problems and derives a decomposable graph using dynamic programming. Chechetka and Guestrin [2007] used the weaker notion of conditional mutual information instead of conditional independence to learn approximate junction trees. Finally, Gogate et al. [2010] uses low mutual information criteria to recursively split the state space to smaller subsets until no further splits are possible.

Organisation. This chapter is organised as follows:

— We pose the problem of learning bounded-treewidth decomposable graphical models from data as a combinatorial optimisation problem in Section 2.2, which is relaxed to a convex optimisation problem that involves the graphic and hypergraphic matroids, as shown in Section 2.4.

— We show in Section 2.5 how a supergradient ascent method may be used to solve the dual optimisation problem, using greedy algorithms as inner loops on the two matroids. Each iteration has a run-time complexity of $O(k^3 n^{k+2} \log n)$, where n is the number of variables. We also show how to round the obtained fractional solution.

— We compare our approach to state-of-the-art methods on synthetic datasets and classical benchmarks in Section 2.6, showing the gains of the novel convex approach.

2.2 Maximum likelihood decomposable graphs

Given N observations $x^1, \ldots, x^N$ of X, we denote the corresponding empirical distribution of X by $\hat{p}(x) = \frac{1}{N} \sum_{i=1}^{N} \delta(x = x^i)$. Given the structure of a decomposable graph G, the maximum likelihood distribution that factorises in G may be obtained by combining the marginal empirical distributions on all maximum cliques and their separators as defined earlier in Section 1.1.1.

Let $\hat{p}(x)$ denote the empirical distribution and $\hat{p}_G(x)$ denotes the projected distribution on a decomposable graph G. Estimating the maximum likelihood decomposable graph which best approximates $\hat{p}$ is equivalent to finding the graph, G, which minimises the KL-divergence between the target distribution and the projected distribution, $\hat{p}_G$, defined by $D(\hat{p}||\hat{p}_G) =$.

$$
\begin{aligned}
&= \sum_{x \in \mathcal{X}} \hat{p}(x) \log \frac{\hat{p}(x)}{\hat{p}_G(x)} \\
&\propto \sum_{x \in \mathcal{X}} -\hat{p}(x) \log \hat{p}_G(x) \text{ as } \hat{p}(x) \text{ is independent of } G
\end{aligned}
$$

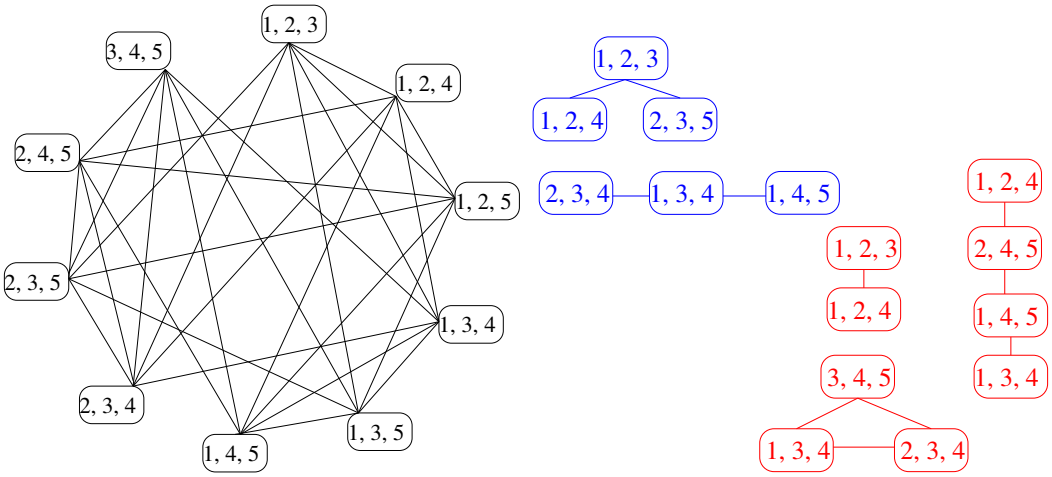


Figure 2-1 – Space of cliques $\mathcal{D}$ denoted by ovals and the space of feasible edges $\mathcal{E}$ denoted by lines for $V = \{1, 2, 3, 4, 5\}$ and treewidth 2(in Black). Clique and edge selections in blue represent decomposable graphs while those in red denote graphs that are not decomposable (best seen in color).

$$
\begin{aligned}
&= \sum_{x \in \mathcal{X}} -\hat{p}(x) \log \frac{\prod_{C \in \mathcal{C}(G)} \hat{p}(x_C)}{\prod_{(C,D) \in \mathcal{T}(G)} \hat{p}(x_{C \cap D})} \quad \text{from Eq. (1.4)} \\
&= \sum_{x \in \mathcal{X}} \left(-\hat{p}(x) \log \prod_{C \in \mathcal{C}(G)} \hat{p}(x_C) \right) - \sum_{x \in X} \left(-\hat{p}(x) \log \prod_{(C,D) \in \mathcal{T}(G)} \hat{p}(x_{C \cap D}) \right) \\
&= \sum_{C \in \mathcal{C}(G)} \sum_{x \in \mathcal{X}} -\hat{p}(x) \log \hat{p}(x_C) - \sum_{(C,D) \in \mathcal{T}(G)} \sum_{x \in \mathcal{X}} -\hat{p}(x) \log \hat{p}(x_{C \cap D}) \\
&= \sum_{C \in \mathcal{C}(G)} \sum_{x_C \in \mathcal{X}_C} -\hat{p}(x_C) \log \hat{p}(x_C) - \sum_{(C,D) \in \mathcal{T}(G)} \sum_{x_{C \cap D} \in \mathcal{X}_{C \cap D}} -\hat{p}(x_{C \cap D}) \log \hat{p}(x_{C \cap D}) \\
&= \sum_{C \in \mathcal{C}(G)} \hat{H}(C) - \sum_{(C,D) \in \mathcal{T}(G)} \hat{H}(C \cap D) \ , \qquad (2.1)
\end{aligned}
$$

where $\hat{H}(S)$ is the empirical entropy of the random variables indexed by the set $S \subseteq V$, defined by $\hat{H}(S) = \sum_{x_S} \{-\hat{p}(x_S) \log \hat{p}(x_S)\}$, and where the sum is taken over all possible values of x_S.

2.3 Combinatorial optimisation problem

We now consider the problem of learning a decomposable graph of treewidth less than k. We assume that we are given all entropies $\hat{H}(S)$ for subsets S of V of cardinality less than $k + 1$.

Since we do not add any model selection term, without loss of generality [Szántai and Kovács, 2012], we restrict the search space to the space of *maximal junction trees*, i.e., junction trees with $n - k$ maximal cliques of size $k + 1$ and $n - k - 1$ separator

sets of size k between two cliques of size $k+1$. Our natural search spaces are thus characterised by $\mathcal{D}$, the set of all subsets of size $k+1$ of V, of cardinality $\binom{n}{k+1}$, and $\mathcal{E}$, the set of all potential edges in a junction tree, i.e., $\mathcal{E} = \{(C,D) \in \mathcal{D} \times \mathcal{D}, C \cap D \neq \varnothing, |C \cap D| = k\}$. The cardinality of $\mathcal{E}$ is $\binom{n}{k+2}.\binom{k+2}{2}$ (number of subsets of size $k+2$ times the number of possibility of excluding two elements to obtain a separator).

A decomposable graph will be represented by a clique selection function $\tau : \mathcal{D} \to \{0,1\}$ and an edge selection function $\rho : \mathcal{E} \to \{0,1\}$ so that $\tau(C) = 1$ if C is a maximal clique of the graph and $\rho(C,D) = 1$ if (C,D) is an edge in the junction tree. Both ρ and τ will be referred to as incidence *functions* or incidence *vectors*, when seen as elements of $\{0,1\}^{\mathcal{D}}$ and $\{0,1\}^{\mathcal{E}}$.

Thus, minimizing the problem defined in Eq. (2.1) is equivalent to minimizing,

$$\mathcal{P}(\tau,\rho) = \sum_{C \in \mathcal{D}} \hat{H}(C)\tau(C) - \sum_{(C,D) \in \mathcal{E}} \hat{H}(C \cap D)\rho(C,D), \tag{2.2}$$

with the constraint that (τ, ρ) forms a decomposable graph.

At this time, we have merely reparameterised the problem with the clique and edge selection functions. We now consider a set of necessary and sufficient conditions for the pair to form a decomposable graph. Some are convex in (τ, ρ), while some are not. The latter ones will be relaxed in Section 2.4. From now on, we denote by $1_{i \in C}$ the indicator function for $i \in C$ (i.e., it is equal to 1 if $i \in C$ and zero otherwise).

— *Covering* V: Each vertex in V must be covered by at least one of the selected cliques,

$$\forall i \in V, \ \sum_{C \in \mathcal{D}} 1_{i \in C}\tau(C) \geq 1. \tag{2.3}$$

— *Number of edges*: Exactly $n-k-1$ edges from $\mathcal{E}$ must be selected,

$$\sum_{(C,D) \in \mathcal{E}} \rho(C,D) = n-k-1. \tag{2.4}$$

— *Number of cliques*: Exactly $n-k$ cliques from $\mathcal{D}$ must be selected,

$$\sum_{C \in \mathcal{D}} \tau(C) = n-k. \tag{2.5}$$

— *Running intersection property*: Every vertex, $i \in V$ must induce a tree, i.e., the number of selected edges containing the vertex, i, must be equal to the number of selected cliques containing the vertex, i , minus one.

$$\forall i \in V, \ \sum_{(C,D) \in \mathcal{E}} 1_{i \in (C \cap D)}\rho(C,D) - \sum_{C \in \mathcal{D}} 1_{i \in C}\tau(C) + 1 = 0. \tag{2.6}$$

— *Edges between selected cliques*: An edge in $\mathcal{E}$ is selected by ρ only if the cliques it is incident on is selected by τ.

$$\forall C \in \mathcal{D}, \ \tau(C) = \max_{D \in \mathcal{D}, \ (C,D) \in \mathcal{E}} \rho(C,D). \tag{2.7}$$

— *Acyclicity of* ρ: ρ selects edges in $\mathcal{E}$ such that they do not have loops, e.g., the blue lines in Figure 1-3-(b) cannot form loops,

$$\rho \text{ represents a subforest of the graph } (\mathcal{D}, \mathcal{E}). \tag{2.8}$$

— *Acyclicity of* τ: τ selects the hyperedges of V in $\mathcal{D}$ such that they are acyclic, i.e.,

$$\tau \text{ represents an acyclic hypergraph of } (V, \mathcal{D}). \tag{2.9}$$

The above constraints encode the classical definition of junction trees. Thus our combinatorial problem is exactly equivalent to minimizing $P(\tau, \rho)$ defined in Eq. (2.2), subject to the constraints in Eq. (2.3), Eq. (2.4), Eq. (2.5), Eq. (2.6), Eq. (2.7), Eq. (2.8) and Eq. (2.9). Note that the constraint Eq. (2.9) that τ represents an acyclic hypergraph is implied by the other constraints.

Figure 2-1 shows clique and edge selections in blue which satisfy all these constraints and hence represent a decomposable graph. The clique and edge selections in red violates at least one of these constraints.

2.4 Convex relaxation

We now provide a convex relaxation of the combinatorial problem defined in Section 2.3. The covering constraint in Eq. (2.3), the number of edges and the number of cliques constraints in Eq. (2.4) and Eq. (2.5) respectively, and the running intersection property in Eq. (2.6) are already convex in (τ, ρ).

The constraint in Eq. (2.7) that $\forall C \in \mathcal{D},\ \tau(C) = \max_{D \in \mathcal{D},\ (C,D) \in \mathcal{E}} \rho(C, D)$ may be relaxed into:

— *Edge constraint*: selection of edges only if the both the incident cliques are selected, i.e.,

$$\forall C \in \mathcal{D},\ \forall (C, D) \in \mathcal{E},\ \rho(C, D) \leq \tau(C). \tag{2.10}$$

— *Clique constraint*: selection of a clique if at least an edge incident on it is selected, i.e.,

$$\forall C \in \mathcal{D},\ \tau(C) \leq \sum_{(C,D) \in \mathcal{E}} \rho(C, D). \tag{2.11}$$

We now consider the two acyclicity constraints in Eq. (2.8) and Eq. (2.9).

2.4.1 Forest polytope

Given the graph $(\mathcal{D}, \mathcal{E})$, the *forest polytope* is the convex hull of all incidence vectors ρ of subforests of $(\mathcal{D}, \mathcal{E})$. Thus, it is exactly the convex hull of all $\rho : \mathcal{E} \to \{0, 1\}$ such that ρ satisfies the constraint in Eq. (2.8). We may thus relax it into:

— *Tree constraint*:

$$\rho \in \text{forest polytope of } (\mathcal{D}, \mathcal{E}). \tag{2.12}$$

While the new constraint in Eq. (2.12) forms a convex constraint, it is crucial that it may be dealt with empirically in polynomial time. This is made possible by

the fact that one may maximise any linear function over that polytope. Indeed, for a weight function $w : \mathcal{E} \times \mathcal{E} \to \mathbb{R}$, maximizing $\sum_{(C,D)\in\mathcal{E}} w(C,D)\rho(C,D)$ is exactly a maximum weight spanning forest problem, and its solution may along with algorithms be obtained by Kruskal's algorithm, i.e., (a) order all (potentially negative) weights $w(C,D)$ and (b) greedily select edges (C,D), i.e., set $\rho(C,D) = 1$, with higher weights first, as long as they form a forest and as long as the weights are positive. When we add the restriction that the number of edges is fixed (in our case $n-k-1$), then the algorithm is stopped when exactly the desired number of edges is selected (whether the corresponding weights are positive or not). See, e.g., [Schrijver, 2003].

The polytope defined above may also be defined as the independence polytope of the graphic matroid, which is the traditional reason why the greedy algorithm is exact [Schrijver, 2003]. In the next section, we show how this can be extended to hypergraphs.

2.4.2 Hypergraphic matroid

Given the set of potential cliques $\mathcal{D}$ over V, we consider functions $\tau : \mathcal{D} \to \{0,1\}$ that are equal to one when a clique is selected, and zero otherwise. Ideally, we would like to treat the acyclicity of the associated hypergraph in a similar way than for regular graphs. However, the set of acyclic subgraphs of the hypergraph defined from $\mathcal{D}$ does not form a matroid, and thus the polytope defined as the convex hull of all incidence vectors/functions of acyclic hypergraphs may be defined, but the greedy algorithm is not applicable. In order to define what is referred to as the *hypergraphic matroid*, one needs to relax the notion of acyclicity.

We now follow [Lorea, 1975, Frank et al., 2003, Fukunaga, 2010] and define a different notion of acyclicity for hypergraphs. An hypergraph $(V,\mathcal{F})$ is an *hyperforest* if and only if for all $A \subset V$, the number of hyperedges in $\mathcal{F}$ contained in A is less than $|A|-1$. A non-trivially equivalent definition is that we can select two elements in each hyperedge so that the graph with vertex set V and with edge set composed of these pairs is a forest.

Given an hypergraph with hyperedge set $\mathcal{D}$, the set of sub-hypergraphs which are hyperforests forms a matroid. This implies that given a weight function on $\mathcal{D}$, one may find the maximum weight hyperforest with a greedy algorithm that ranks all hyperedges and select them as long as they do not violate acyclicity (with the notion of acyclicity just defined and for which we exhibit a test below).

Checking acyclicity of an hypergraph $(V,\mathcal{F})$ (which is needed for the greedy algorithm above) may be done by minimizing with respect to $A \subset V$

$$|A| - \sum_{G\in\mathcal{F}} 1_{G\subset A}.$$

The hypergraph is an hyperforest if and only if the minimum is greater or equal to one. The minimisation of the above problem may be cast a min-cut/max-flow problem as follows [Fukunaga, 2010]:

— single source, single sink, one node per hyperedge in $\mathcal{F}$, one node per vertex

in V,

— the source points towards each hyperedge with unit capacity,

— each hyperedge points towards the vertices it contains, with infinite capacity,

— each vertex points towards the sink, with unit capacity.

The runtime complexity of this test is $O(n^2)$. The hypergraph obtained from the maximal cliques of a decomposable graph can easily be seen to be an hyperforest. But the converse is not true. We can now naturally define the hyperforest polytope as the convex hull of all incidence vectors of hyperforests. Thus the constraint in Eq. (2.9) may be relaxed into:

— *Hyperforest constraint*:

$$\tau \in \text{hyperforest polytope of } (V, \mathcal{D}). \tag{2.13}$$

In case of trees, where $k = 1$, the hyperforest polytope becomes the forest polytope.

2.4.3 Relaxed optimisation problem

We can now formulate our combinatorial problem from the constraints in Eq. (2.3), Eq. (2.4), Eq. (2.5), Eq. (2.6), Eq. (2.10), Eq. (2.11), Eq. (2.12) and Eq. (2.13) as follows

$$\min \mathcal{P}(\tau, \rho) \text{ subject to } \begin{cases} \tau \in \{0,1\}^{\mathcal{D}}, \\ \rho \in \{0,1\}^{\mathcal{E}}, \\ \forall i \in V, \sum_{C \in \mathcal{D}} 1_{i \in C} \tau(C) \geq 1, \\ \sum_{(C,D) \in \mathcal{E}} \rho(C,D) = n - k - 1, \\ \sum_{C \in \mathcal{D}} \tau(C) = n - k, \\ \forall i \in V, \sum_{(C,D) \in \mathcal{E}} 1_{i \in (C \cap D)} \rho(C,D) - \sum_{C \in \mathcal{D}} 1_{i \in C} \tau(C) + 1 = 0, \\ \forall C \in \mathcal{D}, \forall (C,D) \in \mathcal{E}, \rho(C,D) \leq \tau(C), \\ \forall C \in \mathcal{D}, \tau(C) \leq \sum_{(C,D) \in \mathcal{E}} \rho(C,D), \\ \rho \in \text{forest polytope of } (\mathcal{D}, \mathcal{E}), \\ \tau \in \text{hyperforest polytope of } (V, \mathcal{D}). \end{cases} \tag{2.14}$$

All constraints except the integrality constraints are convex. Let *τ-relaxed primal* be the partially relaxed primal optimisation problem formed by relaxing only the integral constraint on τ in Eq. (2.14), i.e., replacing $\tau \in \{0,1\}^{\mathcal{D}}$ by $\tau \in [0,1]^{\mathcal{D}}$. Note that this is not a convex problem due to the remaining integral constraint on ρ, but it remains equivalent to the original problem as the following proposition shows.

Proposition 4. *The combinatorial problem in Eq. (2.14) and the τ-relaxed primal problem are equivalent.*

Proof. Let us assume (τ^*, ρ^*) be a feasible solution for the relaxed primal with $0 < \tau^*(C) < 1$ for some $C \in \mathcal{D}$. The edge constraint in Eq. (2.10) ensures that there are no incident edges on C selected by ρ^* (as ρ^* is integral). This violates the clique constraint in Eq. (2.11). Therefore, the feasible solutions of relaxed primal are integral. Hence the optimal solutions of the primal and the relaxed primal are identical. □

The *convex relaxation* for the primal optimisation problem formed by relaxing the integral constraint on both τ and ρ can now be defined as

$$\min \mathcal{P}(\tau,\rho) \text{ subject to } \begin{cases} \tau \in [0,1]^{\mathcal{D}}, \\ \rho \in [0,1]^{\mathcal{E}}, \\ \forall i \in V, \sum_{C\in\mathcal{D}} 1_{i\in C}\tau(C) \geq 1, \\ \sum_{(C,D)\in\mathcal{E}} \rho(C,D) = n-k-1, \\ \sum_{C\in\mathcal{D}} \tau(C) = n-k, \\ \forall i \in V, \sum_{(C,D)\in\mathcal{E}} 1_{i\in(C\cap D)}\rho(C,D) - \sum_{C\in\mathcal{D}} 1_{i\in C}\tau(C) + 1 = 0, \\ \forall C \in \mathcal{D}, \forall (C,D) \in \mathcal{E}, \rho(C,D) \leq \tau(C), \\ \forall C \in \mathcal{D}, \tau(C) \leq \sum_{(C,D)\in\mathcal{E}} \rho(C,D), \\ \rho \in \text{forest polytope of } (\mathcal{D},\mathcal{E}), \\ \tau \in \text{hyperforest polytope of } (V,\mathcal{D}). \end{cases} \tag{2.15}$$

2.5 Solving the dual problem

We now show how the convex problem may be minimised in polynomial time. Among the constraints of our convex problem in Eq. (2.14), some are simple linear constraints, some are complex constraints depending on the forest and hyperforest polytopes defined in Section 2.4. We will define a dual optimisation problem by introducing the least possible number of Lagrange multipliers (a.k.a. dual variables) [Bertsekas, 1999] so that the dual function (and a supergradient) may be computed and maximised efficiently. We introduce the following dual variables:

— Set cover constraints in Eq. (2.3): $\gamma \in \mathbb{R}_+^V$.
— Running intersection property in Eq. (2.6): $\mu \in \mathbb{R}^V$.
— Edge constraints in Eq. (2.10): $\lambda \in \mathbb{R}_+^{2\mathcal{E}}$.
— Clique constraints in Eq. (2.11): $\eta \in \mathbb{R}_+^{\mathcal{D}}$.

Therefore, the dual variables are $(\gamma,\mu,\lambda,\eta)$. Let $\mathcal{L}(\tau,\rho,\gamma,\mu,\lambda,\eta)$ be the Lagrangian relating the primal and dual variables. It is derived from the primal cost function defined in Eq. (2.2) along with the covering constraint, running intersection property, the edge and the clique constraints defined in Eq. (2.3), Eq. (2.6), Eq. (2.10) and Eq. (2.11) respectively. The Lagrangian can be computed from the dual variables $(\gamma,\mu,\lambda,\eta)$ as follows:

$$\begin{aligned} & \mathcal{L}(\tau,\rho,\gamma,\mu,\lambda,\eta) \\ = & \sum_{C\in\mathcal{D}} \hat{H}(C)\tau(C) - \sum_{(C,D)\in\mathcal{E}} \hat{H}(C\cap D)\rho(C,D) \\ + & \sum_{C\in\mathcal{D}} \sum_{(C,D)\in\mathcal{E}} \lambda_{CD}\Big(\rho(C,D) - \tau(C)\Big) + \sum_{C\in\mathcal{D}} \eta_C\Big(\tau(C) - \sum_{(C,D)\in\mathcal{E}} \rho(C,D)\Big) \\ + & \sum_{i\in V} \mu_i\Big(\sum_{(C,D)\in\mathcal{E}} 1_{i\in(C\cap D)}\rho(C,D) - \sum_{C\in\mathcal{D}} 1_{i\in C}\tau(C) + 1\Big) \end{aligned}$$

$$
\begin{aligned}
&+ \sum_{i\in V}\gamma_i\Big(1-\sum_{C\in\mathcal{D}}1_{i\in C}\tau(C)\Big)\\
&= \sum_{C\in\mathcal{D}}\Big(\hat{H}(C)-\sum_{i\in C}(\mu_i+\gamma_i)-\sum_{(C,D)\in\mathcal{E}}\lambda_{CD}+\eta_C\Big)\tau(C)\\
&- \sum_{(C,D)\in\mathcal{E}}\Big(\hat{H}(C\cap D)-\sum_{i\in(C\cap D)}\mu_i-\lambda_{CD}-\lambda_{DC}+\eta_C+\eta_D\Big)\rho(C,D)\\
&+ \sum_{i\in V}(\mu_i+\gamma_i), \qquad (2.16)
\end{aligned}
$$

with the following *dual constraints* on the Lagrange multipliers

$$
\begin{aligned}
&\forall i\in V, && \gamma_i\geq 0,\\
&\forall C\in\mathcal{D}, \quad \forall (C,D)\in\mathcal{E}, && \lambda_{CD}\geq 0,\\
&\forall C\in\mathcal{D}, && \eta_C\geq 0. \qquad (2.17)
\end{aligned}
$$

We can now derive a dual optimisation problem with $\mathcal{Q}(\gamma,\mu,\lambda,\eta)$ represent the dual cost function, which can be derived from the Lagrangian in Eq. (2.16). We use the the number of edges constraint, the number of cliques constraint, tree constraint and hyperforest constraint given by Eq. (2.4), Eq. (2.5), Eq. (2.12) and Eq. (2.13) respectively in deriving the dual cost function, $\mathcal{Q}(\gamma,\mu,\lambda,\eta)$:

$$
\begin{aligned}
&= \inf_{\substack{\tau(C)\in[0,1]^{\mathcal{D}}\\ \sum_{C\in\mathcal{D}}\tau(C)=n-k\\ \tau\in\text{ hyperforest polytope of }(V,\mathcal{D})}}\Big(\hat{H}(C)-\sum_{i\in C}(\mu_i+\gamma_i)-\sum_{(C,D)\in\mathcal{E}}\lambda_{CD}+\eta_C\Big)\tau(C)\\
&- \sup_{\substack{\rho\in[0,1]^{\mathcal{E}}\\ \sum_{(C,D)\in\mathcal{E}}\rho(C,D)=n-k-1\\ \rho\in\text{forest polytope of }(\mathcal{D},\mathcal{E})}}\sum_{(C,D)\in\mathcal{E}}\Big(\hat{H}(C\cap D)-\sum_{i\in(C\cap D)}\mu_i-\lambda_{CD}-\lambda_{DC}+\eta_C+\eta_D\Big)\rho(C,D)\\
&+ \sum_{i\in V}(\mu_i+\gamma_i). \qquad (2.18)
\end{aligned}
$$

It is decomposed in three parts defined in Eq. (2.20), Eq. (2.21) and Eq. (2.22) respectively :

$$
\mathcal{Q}(\gamma,\mu,\lambda,\eta)=q_1(\gamma,\mu,\lambda,\eta)+q_2(\gamma,\mu,\lambda,\eta)+q_3(\gamma,\mu,\lambda,\eta), \qquad (2.19)
$$

where

$$
q_1(\gamma,\mu,\lambda,\eta)= \inf_{\substack{\tau(C)\in[0,1]^{\mathcal{D}}\\ \sum_{C\in\mathcal{D}}\tau(C)=n-k\\ \tau\in\text{ hyperforest polytope}\\ \text{of }(V,\mathcal{D})}}\sum_{C\in\mathcal{D}}\Big(\hat{H}(C)-\sum_{i\in C}(\mu_i+\gamma_i)-\sum_{(C,D)\in\mathcal{E}}\lambda_{CD}+\eta_C\Big)\tau(C). \qquad (2.20)
$$

$$q_2(\gamma,\mu,\lambda,\eta) = \quad - \sup_{\substack{\rho\in[0,1]^{\mathcal{E}} \\ \sum_{(C,D)\in\mathcal{E}}\rho(C,D)=n-k-1 \\ \rho\in \text{ forest polytope} \\ \text{of } (\mathcal{D},\mathcal{E})}} \sum_{(C,D)\in\mathcal{E}} \left(\hat{H}(C\cap D) - \sum_{i\in(C\cap D)} \mu_i - \lambda_{CD} - \lambda_{DC} + \eta_C + \eta_D\right)\rho(C,D). \tag{2.21}$$

$$q_3(\gamma,\mu,\lambda,\eta) = \quad \sum_{i\in V}(\mu_i+\gamma_i). \tag{2.22}$$

Therefore, the dual optimisation problem using the dual cost function defined in Eq. (2.18) and the dual constraints defined in Eq. (2.17) is given by

$$\max \mathcal{Q}(\gamma,\mu,\lambda,\eta) \text{ subject to } \begin{cases} \forall i\in V, \gamma_i \geq 0, \\ \forall C\in\mathcal{D}, \forall (C,D)\in\mathcal{E}, \lambda_{CD}\geq 0, \\ \forall C\in\mathcal{D}, \eta_C \geq 0. \end{cases} \tag{2.23}$$

The dual functions $q_1(\gamma,\mu,\lambda,\eta)$ and $q_2(\gamma,\mu,\lambda,\eta)$ may be computed using the greedy algorithms defined in Section 2.4.1 and Section 2.4.2; q_1 can be evaluated in $O(r\log(r))$, where r is the cardinality of the space of cliques, $\mathcal{D}$, i.e., $\binom{n}{k+1}$ and q_2 can be evaluated in $O(m\log(m))$, where m is the cardinality of feasible edges, $\mathcal{E}$, i.e., $\binom{n}{k+2}.\binom{k+2}{2}$. This complexity is due to sorting the edges and hyperedges based on their weights. This leads to an overall complexity of $O(k^3n^{k+2}\log n)$ per iteration of the projected supergradient method which we now present.

Algorithm 2 Projected Supergradient

Input: clique and edge entropies H, step-size constant a and number of iterations T
Output: sequence of clique and edge selections over iterations (τ^t,ρ^t)
Initialise $\gamma^0=0$, $\mu^0=0$, $\lambda^0=0$, $\eta^0=0$
for $t=0$ **to** T **do**
  solve Eq. (2.20) and evaluate $q_1(\gamma^t,\mu^t,\lambda^t,\eta^t)$ to obtain τ^t
  solve Eq. (2.21) and evaluate $q_2(\gamma^t,\mu^t,\lambda^t,\eta^t)$ to obtain ρ^t
  update dual variables, $(\gamma^{t+1},\mu^{t+1},\lambda^{t+1},\eta^{t+1})$ using supergradients and stepsize: $\alpha_t=\frac{a}{\sqrt{t}}$
  $\gamma_i^{t+1} = \left[\gamma_i^t + \alpha_t\left(1-\sum_{C\in\mathcal{D}} 1_{i\in C}\tau^t(C)\right)\right]^+$
  $\mu_i^{t+1} = \mu_i^t + \alpha_t\left(\sum_{(C,D)\in\mathcal{E}} 1_{i\in(C\cap D)}\rho^t(C,D) - \sum_{C\in\mathcal{D}} 1_{i\in C}\tau^t(C) + 1\right)$
  $\lambda_{CD}^{t+1} = \left[\lambda_{CD}^t + \alpha_t\left(\tau^t(C)-\rho^t(C,D)\right)\right]^+$
  $\eta_C^{t+1} = \left[\eta_C^t + \alpha_t\left(\sum_{(C,D)\in\mathcal{E}}\rho^t(C,D) - \tau^t(C)\right)\right]^+$
end for

Projected supergradient ascent. The dual optimisation problem defined by maximizing $Q(\gamma,\mu,\lambda,\eta)$ can be solved using the projected supergradient method. In

each iteration t of the algorithm, the dual cost function, $Q(\gamma^t, \mu^t, \lambda^t, \eta^t)$, is evaluated through estimation of q_1 and q_2 by solving Eq. (2.20) and Eq. (2.21) respectively. In the process of solving these equations, the corresponding primal variables (τ^t, ρ^t) are also estimated and allows the computations of the supergradients of Q (i.e., opposites of subgradients of $-Q$) [Bertsekas, 1999]. As shown in Algorithm 2, a step is made toward the direction of the supergradient and projection onto the positive orthant is performed for dual variables that are constrained to be nonnegative. With step sizes α_t proportional to $1/\sqrt{t}$, this algorithm is known to converge to a dual optimal solution at rate $1/\sqrt{t}$. Moreover, the average of all visited primal variables, i.e., after t steps, $(\hat{\tau}_t, \hat{\rho}_t) = \frac{1}{t}\sum_{u=0}^{t}(\tau^u, \rho^u)$ is known to be approximately primal-feasible (i.e., it satisfies all the linear constraints that were dualised up to a small constant that is also going to zero at rate $1/\sqrt{t}$) [Nedic and Ozdaglar, 2009].

Proposition 5. *If $k = 1$, the convex relaxation in Eq. (2.15) is equivalent to Eq. (2.14).*

Proof. If $k = 1$, all the cliques in the clique space contain only 2 vertices, i.e., $\forall C \in \mathcal{D}, |C| = 2$ and the number of elements in the feasible edges is only 1, i.e., $\forall (C, D) \in \mathcal{E}, |C \cap D| = 1$.

Solving the convex relaxation defined in Eq. (2.15) is equivalent to solving the dual defined in Eq. (2.23). On solving the dual variables, the optimal dual solution is given by

$$\begin{aligned} &\forall i \in V, \mu_i = \hat{H}(\{i\}), \\ &\forall i \in V, \gamma_i = 0, \\ &\forall C \in \mathcal{D}, \forall (C, D) \in \mathcal{E}, \lambda_{CD} = 0, \\ &\forall C \in \mathcal{D}, \eta_C = 0, \end{aligned} \tag{2.24}$$

where $\hat{H}(\{i\}) = -\hat{p}_i(x_i)\log(\hat{p}_i(x_i))$.

The optimal solution to the dual problem is given by

$$\begin{aligned} \mathcal{Q}^*(\gamma, \mu, \lambda, \eta) &= \inf_{\substack{\tau(C)\in[0,1]^{\mathcal{D}} \\ \sum_{C\in\mathcal{D}}\tau(C)=n-k \\ \tau\in \text{ hyperforest of } (V,\mathcal{D})}} \sum_{C\in\mathcal{D}} \left(\hat{H}(C) - \sum_{i\in C}\hat{H}(\{i\})\right)\tau(C) + \sum_{i\in V}\hat{H}(\{i\}) \\ &= \inf_{\substack{\tau(C)\in[0,1]^{\mathcal{D}} \\ \sum_{C\in\mathcal{D}}\tau(C)=n-k \\ \tau\in \text{ hyperforest of } (V,\mathcal{D})}} -I(C).\tau(C) + \sum_{i\in V}\hat{H}(\{i\}), \end{aligned} \tag{2.25}$$

where $\forall C \in \mathcal{D}, I(C) = \sum_{i\in C}\hat{H}(\{i\}) - \hat{H}(C)$, which defines the mutual information of the elements in the clique, i.e., an edge if $k = 1$. The constraints in Eq. (2.25) define a spanning tree polytope [Schrijver, 2003] and the optimal solution is a maximal information spanning tree, which is given by Chow-Liu trees [Chow and Liu, 1968]. They also form the optimal solution to the non-convex primal optimisation defined in Eq. (2.14). □

Approximate Greedy Primal Solution. We describe an algorithm to project from the average of a sequence of fractional primary infeasible solutions, estimated

Algorithm 3 Approximate Greedy Primal Solution

Input: primal infeasible sequence τ^t for Algorithm 2, treewidth k, number of Vertices n, set of cliques $\mathcal{D}$ and integer m such that $0 < m \leq T$
Output: approximate discrete primal feasible solution τ_m after m iterations of Algorithm 2
Initialise Adjacency Matrix $Adj = zeros(n, n)$, $\hat{\tau}_m = \frac{1}{m}\sum_{t=0}^{m}\tau^t$ and $\tau_m = zeros(length(\hat{\tau}_m))$
$order$ = Sorted indices in the descending order $\hat{\tau}_m$
repeat
  Initialise $decomposable = false$, $treewidth = 0$, $numConnectedComponents = 0$, $i = 1$
  update $TestAdj$ = AddClique(Adj, $\mathcal{D}(order(i))$)
  update $[decomposable, treewidth]$ = checkGraphDecomposability($TestAdj$)
  if $decomposable$ = true **and** $treewidth \leq k$ **then**
    update $Adj = TestAdj$
    update $\tau_m(Order(i)) = 1$
  end if
  $[numConnectedComponents]$ = getNumberConnectedComponents($TestAdj$)
  update $i = i + 1$
until $decomposable = true$, $treewidth = k$, $numConnectedComponents = 1$, $i =$ length($order$)

during the iterations of projective supergradient, to an integral primary feasible solution. “AddClique" adds all the edges of a clique to the adjacency matrix. “checkGraphDecomposability" checks if the maximal cardinality search is a perfect elimination ordering. For decomposable graphs the maximal cardinality search yields a perfect elimination ordering [Golumbic, 2004]. We refer to this as *decomposability test*. “getNumberConnectedComponents" gives the number of connected components in the graph using breadth-first search. Note that the projection only uses the average clique selection function, $\hat{\tau}_m$, to obtain the primary feasible solutions, τ_m. The corresponding edge selection, ρ_m, can be estimated from clique selection, τ_m, by selecting the edges between consecutive cliques of the perfect sequence of selected cliques [Lauritzen, 1996]. The time complexity of the projection algorithm is $O(n^{k+2})$. This is due to decomposability test with run time complexity $O(n^{k+1})$, that is performed on adding $O(n)$ cliques.

2.6 Experiments and results

In this section, we show the performance of the proposed algorithm on synthetic datasets and classical benchmarks.

Decomposable covariance matrices. In order to easily generate controllable distributions with entropies which are easy to compute, we use several decomposable

graphs and we consider a Gaussian vector [1] with covariance matrix Σ, generated as follows:

— sample a matrix Z of dimensions $n \times d'$ with entries uniform in $[0, 1]$ and consider the matrix

$$\Sigma' = \frac{d}{d'} ZZ^\top + (1 - \frac{d}{d'})I, \tag{2.26}$$

where Z is a random matrix of dimensions $n \times d'$, I is the n-dimensional identity matrix and d is a parameter to determine the correlations between the nodes of the graph, which takes values in $\{0, d'\}$. In our experiments, we choose d' to be 128. We have tight correlations between the nodes with higher values of d.

— normalise Σ' to unit diagonal,

— The normalised random positive definite covariance matrix, Σ', is projected onto a decomposable graph G as follows:

$$(\Sigma)^{-1} = \sum_{C \in \mathcal{C}(G)} [(\Sigma'_C)^{-1}]_n - \sum_{(C,D) \in \mathcal{T}(G)} [(\Sigma'_{C \cap D})^{-1}]_n, \tag{2.27}$$

where the operator $[(\Sigma'_X)^{-1}]_n$ gives an $n \times n$ matrix whose columns and rows representing the set $X \subseteq V$ are filled by $(\Sigma'_X)^{-1}$ and the rest of the elements of the matrix are filled with *zeroes*. The matrix, Σ, thus generated represents the covariance matrix of a multivariate Gaussian on a decomposable graph, G.

The projection ensures the following relationship between the random positive definite matrix, Σ' and the projected covariance matrix Σ:

$$\begin{aligned} \Sigma(i,j) &= \Sigma'(i,j) \text{ if } A(i,j) = 1 \text{ or } i = j, \\ \Sigma^{-1}(i,j) &= 0 \text{ if } A(i,j) = 0. \end{aligned} \tag{2.28}$$

where A is the adjacency matrix of the decomposable graph G onto which Σ' was projected.

The entropy of a multivariate Gaussian with a covariance matrix, Σ, is given by $\frac{1}{2}\log(2\pi e)^n|\Sigma|$, where $|\Sigma|$ denotes the determinant of the covariance matrix. However, for Gaussian distribution that is factored in $G \in \mathcal{G}$:

$$|\Sigma| = \frac{\prod_{C \in \mathcal{C}(G)} |\Sigma_C|}{\prod_{((C,D) \in \mathcal{T}(G)} |\Sigma_{C \cap D}|}, \tag{2.29}$$

where Σ_X is the sub-matrix of the covariance matrix whose rows and columns belong to the set $X \subseteq V$. Therefore, for any multivariate decomposable Gaussian graphical model, G:

$$\hat{H}(G) = \frac{1}{2}\log((2\pi e)^n|\Sigma|)$$

1. Note that Gaussian vector is a continuous random variable but entropy decomposition in Chapter 1 also holds for differential entropies

$$= \frac{1}{2}\Big(\sum_{C\in\mathcal{C}(G)} \log((2\pi e)^n|\Sigma_C|) - \sum_{(C,D)\in\mathcal{T}(G)} \log((2\pi e)^n|\Sigma_{C\cap D}|)\Big)$$

$$= \sum_{C\in\mathcal{C}(G)} \hat{H}(C) - \sum_{(C,D)\in\mathcal{T}(G)} \hat{H}(C\cap D). \tag{2.30}$$

Note that the entropy of any graph, G, is independent of the mean of the normal distribution, hence we consider only the covariance matrix.

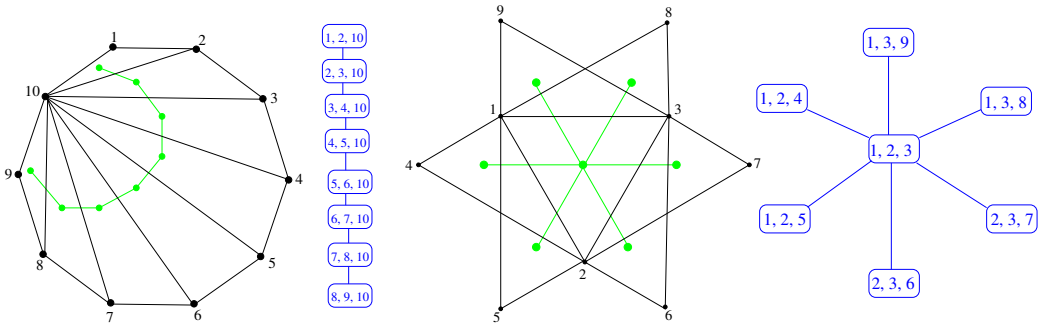


Figure 2-2 – Graph representing (a) chain junction tree, (b) star junction tree, with an embedded junction tree in green and its junction tree representation in blue.

We use the graph structures representing a *chain junction tree* as in Figure 2-2-(a) and a *star junction tree* as in Figure 2-2-(b) to analyse the performance of our algorithm for decomposable covariance matrices generated with different correlations.

Table 2.1 shows the performance of our algorithm on both chain and star graphs. Decomposable covariance matrices are generated as above with different values of the correlation parameter d (all averaged over ten different random covariance matrices). We show the suboptimality of the retrieved structure and the optimal structure by showing the different between the cost function in Eq. (2.2) of the retrieved structure and the optimal structure, i.e., the actual structure represented by the covariance matrices. Note that the suboptimality in the table is multiplied by 10^3 for brevity.

All represents the structure retrieved by solving the proposed convex relaxation and estimating the primal feasible solution using all the constraints proposed in this work. The column under **Without clique acyclicity** uses the same approach but does not use the constraint that the clique selection lies in a hyperforest. The **Greedy** algorithm learns a structure by sorting the mutual information and keeps adding the cliques with largest mutual information as long as decomposability is maintained. From the table it can be observed that the hyperforest constraints, i.e., the acyclicity on cliques constraint is key to obtain tighter relaxations. Empirically we have been able to get the optimal solution when the correlations are strong enough (i.e., large values of d) and outperforms the simple greedy algorithm.

Figure 2-3 represents the empirical integrality gaps of the retrieved structures with and without the acyclicity constraint on the cliques. It reinforces our observation that the acyclicity on cliques is important to achieve a tight relaxation.

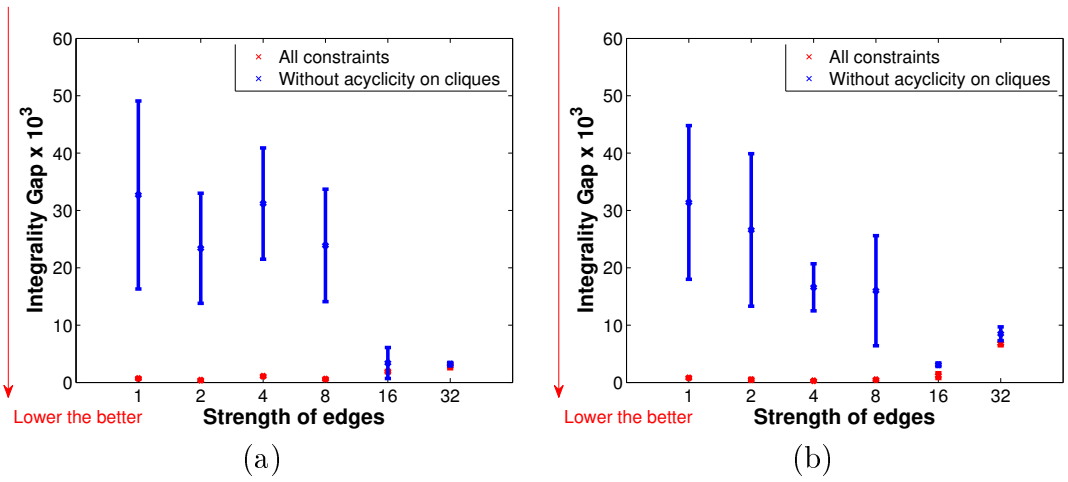


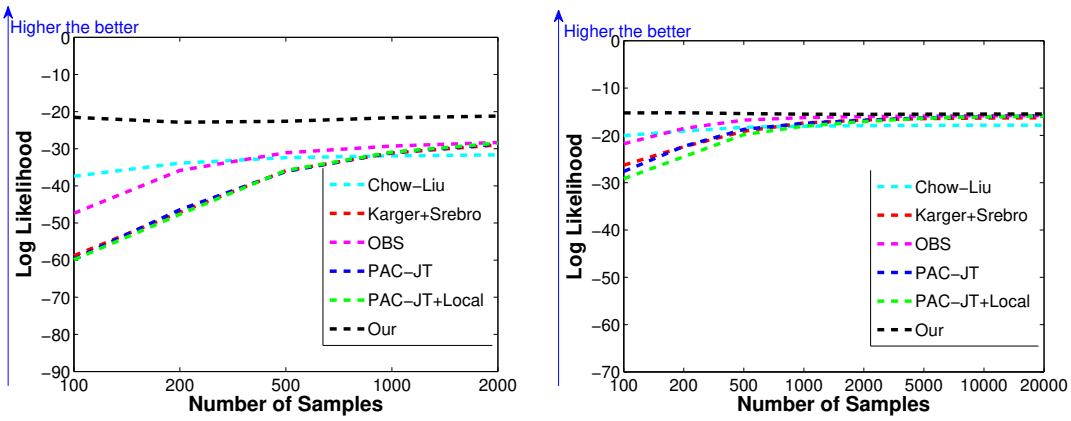
Figure 2-3 – Empirical integrality gaps of the learnt structures with and without the acyclicity constraint on cliques in the (a) chain junction tree, (b) star junction tree.

Performance Comparison. We compare the quality of the graph structures learned by the proposed algorithm with the ones produced by Ordering Based Search (OBS) [Teyssier and Koller, 2001], the combinatorial optimisation algorithm proposed by Karger and Srebro (Karger+Srebro) [Karger and Srebro, 2001], the Chow-Liu trees (Chow-Liu) [Chow and Liu, 1968] and different variations of PAC-learning based algorithms (PAC-JT, PAC-JT+local) [Chechetka and Guestrin, 2007]. We use a real-world dataset, TRAFFIC [Krause and Guestrin, 2005] and an artificial dataset, ALARM [Beinlich et al., 1989] to compare the performances of these algorithms.

This ALARM dataset was sampled from a known Bayesian network [Beinlich et al., 1989] of 37 nodes, which has a treewidth equal to 4. We learn an approximate decomposable graph of treewidth 3. The TRAFFIC dataset is the traffic flow information

d	All		Without clique acyclicity		Greedy	
	chain	star	chain	star	chain	star
1	0.2±0.1	0.2±0.1	0.4±0.1	0.5±0.1	0.2±0.1	0.9±0.1
2	0	0	0.3±0.2	0.4±0.1	0.5±0.2	0.4±0.3
4	0	0	0.3±0.1	0.2±0.1	1.9±0.3	1.7±0.2
8	0	0	0.2±0.1	0	7.9±0.3	6.9±0.3
16	0	0	0	0	25.6±1.2	26.3±1.5

Table 2.1 – Comparison of suboptimality for learnt structure vs true structure $\times 10^3$, where **All** represents the structures learnt by solving the proposed convex relaxation with all the constraints including the hyperforest constraint on cliques, **Without clique acyclicity** represents the structures learnt by solving the proposed convex relaxation with all constraints excluding hyperforest constraint on cliques and the structure learnt using an approximate **greedy** algorithm.



(a) (b)

Figure 2-4 – Log likelihood of the structures learnt using various algorithms on (a) TRAFFIC and (b) ALARM datasets with $k = 3$ except Chow-Liu ($k = 1$).

every 5 minutes for a month at 8000 locations in California [Krause and Guestrin, 2005]. The traffic flow information is collected at 32 locations in San Francisco Bay area and the values are discretised into 4 bins. We learn an approximate decomposable graph of treewidth 3 using our approach. Empirical entropies are computed from the generated samples of each data set and we infer the underlying structure from them using our algorithm. Figure 2-4(b) and Figure 2-4(c) show the log-likelihoods of structures learnt using various algorithms on Traffic and Alarm datasets respectively. Note that the performance is better with higher values as we compare log-likelihoods. These figures illustrate the gains of the convex approach over non-convex approaches.

2.7 Conclusion

In this chapter, we have provided a convex relaxation to the problem of finding the maximum likelihood decomposable graph with bounded treewidth. The convex relaxation relies on optimising linear functions on polytopes of graphic and hypergraphic matroids. It is a polynomial-time optimization algorithm, which empirically outperforms previously proposed algorithms.

Chapter 3

Maximising Submodular Functions using Probabilistic Graphical Models

Abstract

We consider the problem of maximizing submodular functions; while this problem is known to be NP-hard, several numerically efficient local search techniques with approximation guarantees are available. In this chapter, we propose a novel convex relaxation which is based on the relationship between submodular functions, entropies and probabilistic graphical models. In a graphical model, the entropy of the joint distribution decomposes as a sum of marginal entropies of subsets of variables; moreover, for any distribution, the entropy of the closest distribution factorizing in the graphical model provides an bound on the entropy. For directed graphical models, this last property turns out to be a direct consequence of the submodularity of the entropy function, and allows the generalization of graphical-model-based upper bounds to any submodular functions. These upper bounds may then be jointly maximized with respect to a set, while minimized with respect to the graph, leading to a convex variational inference scheme for maximizing submodular functions, based on outer approximations of the marginal polytope and maximum likelihood bounded treewidth structures. By considering graphs of increasing treewidths, we may then explore the trade-off between computational complexity and tightness of the relaxation. We also present extensions to constrained problems and maximizing the difference of submodular functions, which include all possible set functions.

This chapter is based on our work "Maximizing Submodular Functions using Probabilistic Graphical Models", K. S. Sesh Kumar, F. Bach, presented in workshop on Discrete and Combinatorial Problems in Machine Learning (DISCML), NIPS 2013.

3.1 Goal

Goal. In this chapter, our goal is to provide the first (to the best of our knowledge) generic convex relaxation of submodular function maximization, with a hierarchy of complexities related to known combinatorial hierarchies such as the Sherali-Adams

hierarchy [Sherali and Adams, 1990]. Our aim is to characterise the subclass of submodular functions that can be optimised exactly when compared to traditional approach of submodular maximisation algorithms, which aim at constant-factor approximation guarantees.

Beyond the graphical model tools that we are going to develop, having convex relaxations may be interesting for several reasons:

1. they may lead to better solutions,
2. they provide online bounds that may be used within branch-and-bound optimization and
3. they ease the use of such combinatorial optimization problems within structured prediction framework [Tsochantaridis et al., 2004].

Motivation. The relationship between submodularity and entropies has classically been useful in various probabilistic modeling tasks involving entropies, e.g., for proposing approximate algorithms for learning bounded treewidth graphical models [Narasimhan and Bilmes, 2004, Chechetka and Guestrin, 2007], for learning naive Bayes models [Krause and Guestrin, 2005] or for discriminative structure learning [Narasimhan and Bilmes, 2005]. In this chapter, we consider transfers in the opposite direction and will extend notions which are usually linked with entropies to all submodular functions. This will be achieved through *probabilistic graphical models*.

Discrete entropies are known to be non-decreasing submodular set functions– the submodularity being a consequence of the data-processing inequality [Cover and Thomas, 2006]. They are also known to be a strict subset of non-decreasing submodular set functions, i.e., when $n > 4$, there exist set functions which are non-decreasing and submodular but not entropies [Zhang and Yeung, 1998].

Related work. We deviate from the classical approach of proposing constant-factor guarentee algorithms. In our approach, we aim at characterising a subclass of submodular functions that can be maximised exactly. To the best of our knowledge, $M^\natural$-concave functions [Murota, 2003] is the only other work in literature in this direction.

Feige et al. [2011] proposed constant factor approximation algorithms for maximizing non-monotone submodular functions. They provide a randomized local search technique which optimizes a multi linear auxiliary function with some approximation guarantees. Buchbinder et al. [2012] proposed a ramdomized 1/2-approximation algorithm to maximize non-monotone submodular functions. They also use a randomized local search to remove or add an element for the existing set under consideration in each iteration of the algorithm. Maximising submodular functions over integer lattices has also gathered interest, recently. Gottschalk and Peis [2015] proposed an algorithm to maximise submodular functions on bounded integer lattice and Ward and Zivny [2014] proposed an algorithm to maximise k-submodular functions. However, these methods only consider unconstrained submodular maximization.

Recent works also consider maximization of non-monotone submodular functions [Vondrák et al., 2011] with packing-type constraints such as knapsack constraints, matroid constraints and their intersections with 0.309-approximation guarantee with respect to the best integer solution on the matroid polytope. They consider an extreme point of the polytope and provide a technique to replace an element of the extreme point fractionally using linear optimization. Iyer et al. [2013] proposed semi-differentials, discrete equivalent of gradients, to define linear bounds on submodular functions. The approximations thus obtained are optimized using CCCP-like [Yuille and Rangarajan, 2003] procedures.

Relationship to Chapter 2. In this chapter, we use the algorithm proposed in Chapter 2 to learn bounded treewidth decomposable graphs. We will use inner approximation of the polytope representing bounded treewidth decomposable graphs in contrast to outer approximation proposed in the previous chapter.

Notations. Throughout this chapter, we consider a normalised submodular function F defined on the set $V = \{1, 2, \ldots, n\}$ such that $F(\varnothing) = 0$.

Organisation. This chapter is organised as follows:

— For any directed acyclic graph G and a submodular function F, we define in Section 3.2 a bound $F_G(A)$ and study its properties (monotonicity, tightness). It is specialized to decomposable graphs in Section 3.3.

— In Section 3.4, we propose an algorithm to maximize submodular functions by maximizing the bound $F_G(A)$ with respect to A while minimizing with respect to the graph G, leading to a convex variational method based on outer approximation of the marginal polytope [Wainwright and Jordan, 2008] and inner approximation of the hypertree polytope. This is in contrast with our previous work in Chapter 2 where we propose a convex relaxation that gives outer approximation of the hypertree polytope.

— In Section 3.5, we propose extensions to constrained problems and maximizing the difference of submodular functions, which include all possible set functions.

— We illustrate our results on small-scale experiments in Section 3.6.

3.2 Directed graphical models

Let us consider a directed acyclic graph (DAG) G on the set of random variables associated with the groundset V. Let p denote a probability distribution on the random variables and let p_G denote the corresponding factorisable distribution with respect to G as defined in Eq. (1.1).

The KL-divergence between these probability distributions is always non-negative. From the definition of the KL-divergence,

$$D(p||p_G) = H(V) - H_G(V),$$

is always non-negative, where H_G is the entropy of the factorisable distribution p_G and is defined by $-\mathbb{E}_{p_G(x)} \log p_G(x)$, which decomposes as shown in Eq. (1.2) and H is the entropy of a general probability distribution p given by $-\mathbb{E}_{p(x)} \log p(x)$. Therefore entropy based on a graph based distribution always bound the entropy based on any general distribution. This bound is tight, i.e., KL-divergence is zero when p factorises in G.

Marginalisation. Given a DAG $G = (V, E)$, let G_A denote the induced subgraph of G by $A \subset V$ given by $G_A = (A, E \cap A \times A)$. Let p_{G_A} be the graph-represented probability distribution encoding the conditional independences of G_A on the random variables associated with A. It is important to note that probability distribution $p_{G_A}(x_A)$ is in general not equal to the marginal distribution $p_G(x_A)$. This is equal only if A is an *ancestral set* of G [Edwards, 2000], i.e., all parents of all elements of A are in A (in other words, we may recursively remove leaf nodes and preserves the factorization).

In the following, we denote by $H_{G_A}(A)$ the entropy of the projection $p_{G_A}(x_A)$ of $p(x_A)$ onto G_A. We have

$$H_{G_A}(A) = \sum_{i \in A} \big\{ H(A \cap (i \cup \pi_i(G)) - H(A \cap \pi_i(G)) \big\}. \tag{3.1}$$

Note that $H_{G_A}(A)$ is different from $H_G(A)$. From properties of entropies and graphical models, we have $H(A) \leqslant H_{G_A}(A)$ for any DAG G and set $A \subseteq V$ as G_A is also a DAG. We show in the next section that this property turns out to be a consequence of submodularity and thus applies to all submodular functions and not just entropies.

Structure learning. Although we will not use structure learning in this chapter, it is worth noting that several entropy-based approaches have been considered for finding the best possible graph (with some constraints) given a probability distribution. They are based on the decomposition of entropies and local search (see, e.g., [Chickering, 2002] and references therein).

3.2.1 Bounds on submodular functions

Given a submodular function $F : 2^V \to \mathbb{R}$ such that $F(\varnothing) = 0$, following Eq. (3.1), we define F_G as

$$F_G(A) \;=\; \sum_{i \in A} \Big\{ F\big(A \cap (\pi_i(G) \cup \{i\})\big) - F\big(A \cap \pi_i(G)\big) \Big\}. \tag{3.2}$$

When F is an entropy function, $F_G(A)$ is the entropy of the distribution closest to the distribution of X_A that factorises in G_A (which is not equal to the marginal entropy on A of the closest distribution that factorises in G). We now show that F_G bounds F and that the bound is tight for some subsets of V.

Proposition 6 (Upper bound)**.** *Let F be a submodular function and G a directed acyclic graph. The function F_G defined in Eq. (3.2) bounds F, i.e., for all $A \subseteq V$,*

$F(A) \leqslant F_G(A)$.

Proof. Without loss of generality, we assume that $\{1, \ldots, n\}$ is the topological ordering (i.e., $j \in \pi_i(G) \Rightarrow i > j$), without loss of generality. For all $A \subseteq V$, $F(A) =$

$$\begin{aligned}
& \sum_{i=1}^{n} F(A \cap \{1, \ldots, i\}) - F(A \cap \{1, \ldots, i-1\}) \text{ by telescoping the sums,} \\
\leqslant\ & \sum_{i \in V} F\big(A \cap (\pi_i(G) \cup \{i\})\big) - F\big(A \cap \pi_i(G)\big) \\
& \qquad \text{by submodularity, since } \pi_i(G) \subset \{1, \ldots, i-1\}, \\
=\ & F_G(A).
\end{aligned}$$

□

Proposition 7 (Tightness of the bound)**.** *For any element, $i \in V$, and any subset B of $\pi_i(G)$, i.e., $B \subseteq \pi_i(G)$, $F_G(B \cup \{i\}) - F_G(B) = F(B \cup \{i\}) - F(B)$.*

Proof. We have $F_G(B \cup \{i\}) - F_G(B) =$

$$\begin{aligned}
& \sum_{j \in B \cup \{i\}} \Big\{ F((B \cup \{i\}) \cap (\pi_j(G) \cup \{j\})) - F((B \cup \{i\}) \cap \pi_j(G)) \\
& \qquad\qquad - F(B \cap (\pi_j(G) \cup \{j\})) + F(B \cap \pi_j(G)) \Big\} \\
=\ & \sum_{j \in B \cup \{i\}} \Big\{ F\big[(B \cap \pi_j(G)) \cup (\{i\} \cap \pi_j(G)) \cup \{j\}\big] - F\big[(B \cap \pi_j(G)) \cup (\{i\} \cap \pi_j(G))\big] \\
& \qquad\qquad - F\big[(B \cap \pi_j(G)) \cup (B \cap \{j\})\big] + F(B \cap \pi_j(G)) \Big\} \\
=\ & \sum_{j \in B} \Big\{ F\big[(B \cap \pi_j(G)) \cup \varnothing \cup \{j\}\big] - F\big[(B \cap \pi_j(G)) \cup \varnothing\big] \\
& \qquad\qquad - F\big[(B \cap \pi_j(G)) \cup \{j\}\big] + F(B \cap \pi_j(G)) \Big\} \\
& + \Big\{ F\big[B \cup \varnothing \cup \{i\}\big] - F\big[B \cup \varnothing\big] - F\big[B \cup \varnothing\big] + F(B) \Big\} \\
=\ & F(B \cup \{i\}) - F(B),
\end{aligned}$$

where we have used acyclicity to ensure that for $j \in B$, $\{i\} \cap \pi_j(G) = \varnothing$. □

Note that a corollary of Proposition 7 is that the bound is tight on all singletons (by considering $B = \varnothing$). This implies that any modular properties of F are preserved and this notably implies that without loss of generality, we may consider only non-decreasing functions. This is because any general submodular function can be decomposed into a non-decreasing function and a modular function. See Bach [2013] for more details on ways to decompose. The bound also has other interesting monotonicity properties, which we now show.

Proposition 8 (Monotonicity of bounds - I)**.** *If G' is a subgraph of the DAG G, then $F_{G'} \geqslant F_G \geqslant F$, i.e., for all $A \subseteq V$, $F_{G'}(A) \geqslant F_G(A) \geqslant F(A)$.*

Proof. Let $G = (V, E)$ and $G' = (V, E')$. If G' is a subgraph of G, then $E' \subseteq E$ and hence for all the vertices, $i \in V$, $\pi_i(G') \subseteq \pi_i(G)$. Therefore, due to submodularity of F,

$$\begin{aligned} F_G(A) &= \sum_{i \in A} F(A \cap (\pi_i(G) \cup \{i\})) - F(A \cap \pi_i(G)) \\ &\leqslant \sum_{i \in A} F(A \cap (\pi_i(G') \cup \{i\})) - F(A \cap \pi_i(G')) \text{ by submodularity,} \\ &= F'_G(A). \end{aligned}$$

□

The following proposition shows that the difference between $F_G(V)$ and $F(V)$ (i.e., approximation for the full set) dominates the error for a specific class of subsets A, namely *ancestral sets*. These sets are also the sets A for which $p_A(x_A)$ factorised in G_A [Lauritzen, 1996].

Proposition 9 (Monotonicity of bounds - II)**.** *If $A \subset V$ is an ancestral set of the DAG G, then $0 \leqslant F_G(A) - F(A) \leqslant F_G(V) - F(V)$.*

Proof. Assuming, without loss of generality, that $\{1, \ldots, p\}$ is a topological ordering where $A = \{1, \ldots, k\}$, we have

$$\begin{aligned} F_G(V) - F(V) &= \sum_{i=1}^{p} \left\{ \left[F(\{1, \ldots, i\}) - F(\{1, \ldots, i-1\}) \right] - \left[F(\{i\} \cup \pi_i(G)) - F(\pi_i(G)) \right] \right\} \\ &\geqslant \sum_{i=1}^{k} \left\{ \left[F(\{1, \ldots, i\}) - F(\{1, \ldots, i-1\}) \right] - \left[F(\{i\} \cup \pi_i(G)) - F(\pi_i(G)) \right] \right\} \\ &= F_G(A) - F(A), \end{aligned}$$

since all terms are non-negative. □

Note that the bound in Proposition 9, does not hold if A is any subset of V. A simple counter-example may be obtained from the entropy of discrete distributions that factorize in the graphical model defined by G: in this case, $F_G(V) = F(V)$, but, for two leaf nodes $\{i, j\}$, $F_G(\{i, j\}) = F_G(\{i\}) + F_G(\{j\}) = F(\{i\}) + F(\{j\})$, which can only be equal to zero (i.e., between zero and $F_G(V) - F(V) = 0$), if the variables indexed by i and j are independent, which is not the case in general if the DAG has a single connected component.

Proposition 10 (Submodularity)**.** *If the DAG is a directed tree (at most one parent per node), then the bound $F_G(A)$ defines a submodular set function.*

Proof. For a directed tree the bound $F_G(A)$ is in fact a quadratic function of the indicator function 1_A, with quadratic terms equal to $F(\{i,j\}) - F(\{i\}) - F(\{j\})$ which are negative by submodularity of F. The function F_G is then a cut function and is submodular. □

Finally, when two DAGs are Markov equivalent, i.e., they entail the same conditional independeces among the random variables [Koller and Friedman, 2009], then the two bounds are equal:

Proposition 11 (Markov equivalence)**.** *If $G = (V, E)$ and $G' = (V, E')$ are two Markov equivalent graphs, then for all $A \subset V$, $F_G(A) = F_{G'}(A)$.*

Proof. Two Markov equivalent graphs may be obtained by reversing orders of edges that are not involved in a "v-structure" [Koller and Friedman, 2009]. The result is then straightforward. □

3.3 Decomposable graphs

Triangulated graphs. A graph $G = (V, E)$ is said to be triangulated if it contains no chordless cycles of length greater than 3 and all triangulated graphs are decomposable graphs [Koller and Friedman, 2009]. A vertex is *simplicial* if its neighbors in the graph form a clique. A graph is *recursively simplicial* if it contains a simplicial vertex $i \in V$ and when i is removed, the subgraph that remains is recursively simplicial. A graph is triangulated if and only if it is recursively simplicial [Lauritzen, 1996]. A perfect elimination ordering is the order in which simplicial vertices can be removed from the graph. The neighbors of the vertex $i \in V$ that are removed after the vertex i is eliminated is denoted by $\pi_i(G)$ [Golumbic, 2004]. This naturally defines a directed acyclic graph G.

Let us consider a decomposable graph G with maximal cliques $\mathcal{C}(G)$ and the edges in its junction tree represented by $\mathcal{T}(G)$.

3.3.1 Bounds on submodular functions

We now define the bound of the submodular function F by projection onto a decomposable graph $G = (V, E)$. Using recursive simpliciality, we define the projection function F_G, similar to that of Eq. (3.2) as:

$$F_G(A) = \sum_{i \in V} \Big\{ F(A \cap (\pi_i(G) \cup \{i\})) - F(A \cap \pi_i(G)) \Big\}, \tag{3.3}$$

where $\pi_i(G)$ denotes the neighbors of the simplicial vertex i during its elimination. We also define an equivalent bound with the junction tree representation; the projection function F_G, similar to Eq. (3.2), is then given by

$$F_G(A) = \sum_{C \in \mathcal{C}(G)} F(C \cap A) - \sum_{(C,D) \in \mathcal{T}(G)} F(C \cap D \cap A). \tag{3.4}$$

We can now show that the two definitions are equivalent and derive corollaries of Proposition 7, Proposition 8, Proposition 9, for decomposable graphs.

Proposition 12 (Bounds for decomposable graphs)**.** *Let F be a submodular function. Let G be a decomposable graph. The set function defined in Eq. (3.5) and Eq. (3.6) are equal and are bounds on the set function F. Moreover,*

(a) the bounds are tight on all cliques of the graph G,

(b) any decomposable subgraph of G will lead to a looser bound,

(c) if A is obtained by recursively removing simplicial vertices of the graph G, then we have $0 \leqslant F_G(A) - F(A) \leqslant F_G(V) - F(V)$.

Proof. We first recall the two definitions.

$$F_G(A) = \sum_{i \in V} \Big\{ F(A \cap (\pi_i(G) \cup \{i\})) - F(A \cap \pi_i(G)) \Big\}, \tag{3.5}$$

$$F_G(A) = \sum_{C \in \mathcal{C}(G)} F(C \cap A) - \sum_{(C,D) \in \mathcal{T}(G)} F(C \cap D \cap A). \tag{3.6}$$

Equivalence between Eq. (3.5) and Eq. (3.6) is a standard result in probabilistic graphical models [Lauritzen, 1996], which states that if $p(x)$ is a discrete distribution with strictly positive probability mass function that factorises in G, i.e.,

$$p(x) = \frac{\prod_{C \in \mathcal{C}(G)} p_C(x_C)}{\prod_{(C,D) \in \mathcal{T}(G)} p_{C \cap D}(x_{C \cap D})} = \prod_{i \in V} \frac{p(x_{\pi_i(G) \cup \{i\}})}{p(x_{\pi_i(G)})}. \tag{3.7}$$

To show tightness of bounds on all cliques, we can always choose an elimination ordering where a given maximal clique is eliminated first, and we then obtain the tightness as a consequence of Proposition 7.

In order to show the monotonicity, notice that if G' is a subgraph of G, then there is a sequence of decomposable graphs between G' and G so that a single edge is added between two graphs in the sequence [Giudici and Green, 1999]. We can then show that at every forward step, the bound has to increase.

Finally, if a set A is obtained by removing simplicial vertices of the graph, G the relationship between DAGs and decomposable graphs and Proposition 9 leads to the desired result. □

3.3.2 Decomposable graph structure learning

We have shown that a submodular function F, when projected onto a decomposable graph G, gives an bound F_G with interesting monotonic properties. In the next section, we will try to optimize the graph. Maximum likelihood structure learning happens to be equivalent to minimizing $F_G(V) - F(V)$ with respect to the graph. Typically, the set of decomposable graphs is restricted to have cliques of size $k + 1$, which leads to a *treewidth* bounded by k (the treewidth of a decomposable graph is exactly the maximal size of a clique minus one [Lauritzen, 1996]). These graphs are

usually considered because inference in these graphs may be performed in polynomial time, with a degree that grows linearly in k.

Some properties of maximum likelihood structures may be transferred to the general submodular case. For example, the best approximation is always given by *maximal junction trees* [Szántai and Kovács, 2012], i.e., decomposable graphs with maximal cliques of size $k+1$ and separators of size k. Therefore, as in the Chapter 2, we consider only the space of maximal junction trees with treewidth k. For these decomposable graphs, denoting $\mathcal{D}_k$ the set of subsets of V with cardinality less than $k+1$, we have

$$F_G(A) = \sum_{C \in \mathcal{D}_k} \nu_C F(C \cap A) \tag{3.8}$$

for a certain $\nu \in \mathbb{R}^{\mathcal{D}_k}$, with ν_C being zero for $|C| \leqslant k-1$. We denote by $\mathcal{J}_k \subset \mathbb{R}^{\mathcal{D}_k}$ the convex hull of all such vectors ν that correspond to a maximal decomposable graphical model with treewidth equal to k. We denote the subsets of V with cardinality $k+1$ as $\mathcal{D}_k^{max}$, which we use in Section 3.4.

Given $A \subset V$, the problem of learning the structure of the graph is to minimize $F_\nu(A)$ with respect to ν in the extreme points of $\mathcal{J}_k$, and since the objective is linear, this is equivalent to optimizing over the entire set $\mathcal{J}_k$. While the problem is NP-hard [Srebro, 2002], several algorithms have been designed, based on local search techniques [Szántai and Kovács, 2012], submodular function minimization [Narasimhan and Bilmes, 2004] or convex relaxations in Chapter 2.

We can build the vector ν from (τ, ρ) of Chapter 2. For all cliques B of size $k+1$, which represent the maximal cliques, $\nu(B) = \tau(B)$. For all cliques B of size k, which represent the separators, $\nu(B) = -\sum_{B=(C \cap D)} \rho(C, D)$.

Special case of trees. When $k = 1$, maximal decomposable graphs with treewidth equal to k are simple trees, and the problem of finding the best graph is equivalent to a maximum weight spanning tree problem [Chow and Liu, 1968], which can thus be found in polynomial time.

3.4 Variational submodular function maximization

We now show how the bounds described in Section 3.3 may be used for submodular function maximization. Given our graphical model framework, we follow the tree-reweighted framework of [Wainwright and Jordan, 2008]. Given a vertex ν of $\mathcal{J}_k$ (i.e., the incidence vector of a decomposable graph), from Proposition 12 and Eq. (3.8), we have the bound

$$\forall A \subset V,\ F(A) \leqslant \sum_{C \in \mathcal{D}_k} \nu_C F(C \cap A) = \sum_{C \in \mathcal{D}_k} F(C) \nu_C 1_{C \subset A}.$$

For best approximation we have, for all $A \subset V$,

$$F(A) \leqslant \min_{\nu \in \mathcal{J}_k} \sum_{C \in \mathcal{D}_k} F(C) \nu_C 1_{C \subset A}.$$

We may thus obtain a bound on $\max_{A\subset V} F(A)$ as

$$\max_{A\subset V} F(A) \;\leqslant\; \max_{A\subset V} \min_{\nu\in\mathcal{J}_k} \sum_{C\in\mathcal{D}_k} F(C)\nu_C 1_{C\subset A}.$$

Using weak duality, i.e., swapping min and max with an inequality, we obtain:

$$\max_{A\subset V} F(A) \;\leqslant\; \min_{\nu\in\mathcal{J}_k} \max_{A\subset V} \sum_{C\in\mathcal{D}_k} F(C)\nu_C 1_{C\subset A}.$$

We may equivalently parameterize $A \subset V$ as $x \in \{0,1\}^n$ through the bijection $A \mapsto 1_A$. This leads to the bound

$$\max_{A\subset V} F(A) \;\leqslant\; \min_{\nu\in\mathcal{J}_k} \max_{x\in\{0,1\}^n} \sum_{C\in\mathcal{D}_k} F(C)\nu_C \prod_{i\in C} x_i.$$

The maximization problem $\max_{x\in\{0,1\}^n} \sum_{C\in\mathcal{D}_k} \nu_C F(C) \prod_{i\in C} x_i$ is typically NP-hard (however, it is not NP-hard when ν is an extreme point of $\mathcal{J}_k$) [Wainwright and Jordan, 2004]. We may relax it by first introducing the set

$$\mathcal{M}_k = \Big\{ y \in \{0,1\}^{\mathcal{D}_k},\ \exists x \in \{0,1\}^n, y_C = \prod_{i\in C} x_i \Big\}.$$

The maximization problem may then be reformulated as $\max_{y\in\mathcal{M}_k} \sum_{C\in\mathcal{D}_k} \nu_C F(C) y_C$, and thus on the convex hull of $\mathcal{M}_k$. This convex hull is usually referred to as the *marginal polytope* [Wainwright et al., 2005, Wainwright and Jordan, 2008] and has exponentially many vertices and faces. A common outer relaxation is based on considering only the local consistencies between probabilities defined by y_C, $C \in \mathcal{D}$. This leads to [Wainwright and Jordan, 2004]

$$\mathcal{N}_k = \Big\{ y \in [0,1]^{\mathcal{D}_k}, \forall D \in \mathcal{D}_k^{max}, \forall C \subset D,\ \sum_{B:C\subseteq B\subseteq D} (-1)^{|B\setminus C|} y_B \geqslant 0 \Big\}.$$

We may now state the main proposition of this section:

Proposition 13. *Let F be a submodular function. Then*

$$\max_{A\subset V} F(A) \leqslant \min_{\nu\in\mathcal{J}_k} \max_{y\in\mathcal{N}_k} \sum_{C\in\mathcal{D}_k} F(C)\nu_C y_C = \max_{y\in\mathcal{N}_k} \min_{\nu\in\mathcal{J}_k} \sum_{C\in\mathcal{D}_k} F(C)\nu_C y_C.$$

If there exists a k-bounded treewidth decomposable graph G such that for all $A \subset V$, $F_G(A) = F(A)$, then the bound is tight.

Proof. If $y_C = \prod_{i\in C} x_i$ for all cliques in $\mathcal{D}_k$, then for all $D \in \mathcal{D}_k^{max}$ and $C \subset D$

$$0 \leq \prod_{i\in C} x_i \prod_{i\in D\setminus C} (1 - x_i) = \sum_{A\subset D\setminus C} (-1)^{|A|} \prod_{A\cup C} x_i,$$

which implies that $\mathcal{M}_k \subseteq \mathcal{N}_k$.

In the context of probabilistic graphical models, this is equivalent to defining pseudo-marginals y_C on the cliques and ensuring that the pseudo-marginals satisfy the local constraints of the marginal polytope. These are the k^{th} order relaxations. The outer relaxation consists of all the extreme points of the marginal polytope as extreme points. However, it also consists of other additional extreme points with fractional elements. In the case of decomposable graph models, which are also known as hypertrees, these relaxations are shown to be tight and yield the same optimal solution [Wainwright and Jordan, 2004, 2008]. Therefore,

$$\max_{A \subset V} F(A) \leq \min_{\nu \in \mathcal{J}_k} \max_{y \in \mathcal{M}} \mathcal{P}(\nu, y) \leq \min_{\nu \in \mathcal{J}_k} \max_{y \in \mathcal{N}} \mathcal{P}(\nu, y) \tag{3.9}$$

To prove the tightness, let us assume that there exists a decomposable graph, G denoted by a vertex $\nu_G \in \mathcal{J}_k$ such that $F(A) = F_G(A)$. Therefore,

$$\begin{aligned}
\max_{A \subset V} F(A) &= \max_{A \subset V} F_G(A) \\
&= \max_{y \in \mathcal{M}} \mathcal{P}(\nu_G, y) \text{ by definition of the marginal polytope} \\
&= \max_{y \in \mathcal{N}} \mathcal{P}(\nu_G, y) \text{ as } G \text{ is a decomposable graph} \\
&\geq \max_{y \in \mathcal{N}} \min_{\nu \in \mathcal{J}_k} \mathcal{P}(\nu, y) \qquad (3.10)
\end{aligned}$$

Eq. (3.9) and Eq. (3.10) show that they are tight. □

The last proposition shows that a convex-concave saddle point problem may be considered to provide an bound for $\max_{A \subset V} F(A)$ and that it is tight for certain submodular functions. Note that the tightness result is still valid if we restrict $\mathcal{J}_k$ to a subclass of graphical models that includes the graph G. The proof of the previous proposition is a consequence of the exactness of the relaxation of inference in graphical models, based on outer relaxations of the marginal polytope and its relationship to the Sherali-Adams hierarchy [Wainwright and Jordan, 2004]. By increasing the treewidth k, we can get tighter relaxations for growing sets of submodular functions, thus replacing set functions which are low-order polynomials of the indicator vectors by submodular functions. Note that these two sets are not included in one another (see also differences of submodular functions in Section 3.5).

Rounding. Given optimal vectors y and ν, following [Sontag et al., 2011], a set may be obtained by thresholding the values of $y_{\{k\}}$ for all singletons.

3.4.1 Optimization algorithm

In this section, we propose an algorithm to optimize the variational bound for maximizing submodular functions in Proposition 13. We denote by $\mathcal{P}(\nu, y) = \sum_{C \in \mathcal{D}_k} F(C) \nu_C y_C$

the bilinear cost function, and the goal is to perform the following optimization

$$\min_{\nu \in \mathcal{J}_k} \max_{y \in \mathcal{N}_k} \mathcal{P}(\nu, y), \tag{3.11}$$

where the two domains are polytopes. We are going to use a simplicial method [Bertsekas and Yu, 2011], which operates as follows.

We denote by $\mathcal{R}(\nu)$ the convex function $\max_{y \in \mathcal{N}_k} \mathcal{P}(\nu, y)$. Our problem is to minimize $\mathcal{R}(\nu)$ on $\mathcal{J}_k$. Given a set of extreme points $\nu_1, \dots, \nu_t$ of $\mathcal{J}_k$, we will minimize $\mathcal{R}(\nu)$ not on $\mathcal{J}_k$, but only on the convex hull $\mathcal{J}_k^t$ of all points $\nu_1, \dots, \nu_t$, thus obtaining a point $\bar{\nu}_t$ and the corresponding optimal vector y_t at $\bar{\nu}_t$. This point $\bar{\nu}_t$ is optimal if and only if $\min_{\nu \in \mathcal{J}_k} \mathcal{P}(\nu, y_t) = \mathcal{P}(\bar{\nu}_t, y_t)$. If the equality above is met, we have the optimal solution; if not, then any minimizer ν_{t+1} of $\min_{\nu \in \mathcal{J}_k} \mathcal{P}(\nu, y_t)$ may be added to the list of extreme points and the algorithm iterates.

This algorithm converges in finitely many iterations [Bertsekas and Yu, 2011] for polytopes. However, the number of iterations is not known a priori (much like the simplex method). Given the algorithm described above, there are still two algorithmic pieces that are missing: obtaining $\min_{\nu \in \mathcal{J}_k^t} \max_{y \in \mathcal{N}_k} \mathcal{P}(\nu, y)$, i.e., the optimization problem on the convex hull, and computing $\min_{\nu \in \mathcal{J}_k} \mathcal{P}(\nu, y_t)$, i.e., finding the next graph to add.

Optimization on the convex hull. Since $\mathcal{N}_k$ is defined by polynomially many linear inequalities, we may introduce Lagrange multipliers z_{CD} for each of the constraints $\sum_{B : C \subseteq B \subseteq D} (-1)^{|B \setminus C|} y_B \geqslant 0$, for $D \in \mathcal{D}_k^{\max}$ and each subset C of D. This leads to

$$\begin{aligned} \max_{y \in \mathcal{N}_k} \mathcal{P}(\nu, y) &= \min_{z \geqslant 0} \max_{y \in [0,1]^{\mathcal{D}_k}} \left\{ \sum_{C \in \mathcal{D}_k} F(C) \nu_C y_C + \sum_{(C,D)} z_{CD} \Big(\sum_{B : C \subseteq B \subseteq D} (-1)^{|B \setminus C|} y_B \Big) \right\} \\ &\overset{\text{def}}{=} \min_{z \geqslant 0} \mathcal{Q}(\nu, z), \end{aligned}$$

with $\mathcal{Q}(\nu, z)$ a function which may be computed in closed form (as the maximum of an affine function with respect to $y \in [0,1]^{\mathcal{D}_k}$), and which is jointly convex in (ν, z). Our optimization problem is then equivalent to

$$\min_{\eta \geqslant 0, \eta^\top 1 = 0} \min_{z \geqslant 0} \mathcal{Q}\Big(\sum_{i=1}^t \eta_i \nu_i, z \Big),$$

which can be solved by projected subgradient descent techniques, that can obtain both approximate primal variables (η, z), but also dual variables y [Nedic and Ozdaglar, 2009].

Finding optimal graphs. When $k = 1$, maximizing linear functions over $\mathcal{J}_k$ is a maximum-weight spanning tree problem. However, as mentioned in Section 3.3.2, it is NP-hard as soon as $k > 1$. There are two ways of dealing with the impossibility of maximizing linear functions: (a) using a reduced convex hull by generating a large number of random graphs–a strategy often used in variational inference in graphical models, or (b) approximate minimization [Narasimhan and Bilmes, 2004] and the

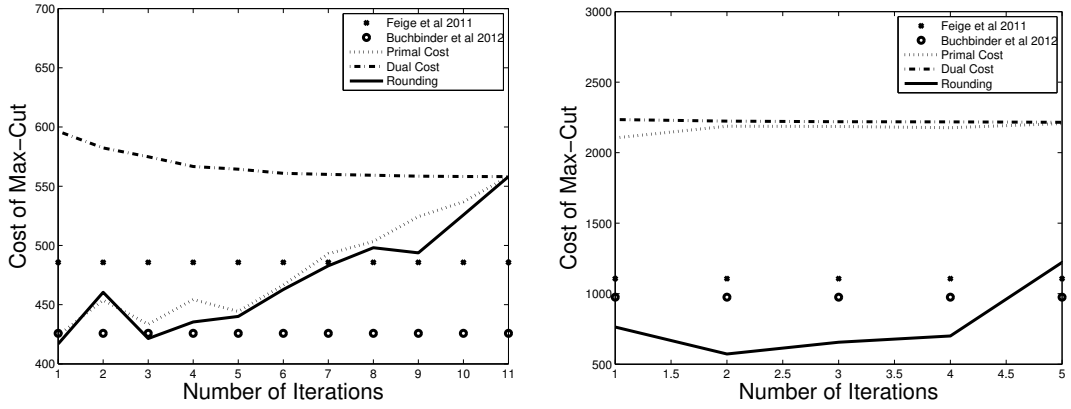


Figure 3-1 – Performance on max-cut for (a) 2D-grid and (b) a random graph; the primal cost is $\min_{\nu \in \mathcal{J}_k} \mathcal{P}(\nu, y_t)$ and the dual cost is $\min_{\nu \in \mathcal{J}_k^t} \mathcal{R}(\nu)$ in our algorithm.

algorithm proposed in Chapter 2. In this situation, the algorithm still provides an bound on the submodular maximization problems, but the algorithm may stop too early.

3.5 Extensions

In this section, we provide possible extensions of this framework to optimise difference of submodular functions and constrained submodular maximisation. **Difference of submodular functions.** As shown by [Narasimhan and Bilmes, 2005], any set function may be written as the difference of two submodular functions F and H. In order to maximize $F(A) - H(A)$, we can use the variational formulation $H(A) = \max_{s \in B(H)} s^\top 1_A$, where $B(H) = \{y \in \mathbb{R}^n, \ \forall A \subset V, \ y^\top 1_A \leqslant H(A), \ y^\top 1_V = H(V)\}$ (see, e.g., [Bach, 2013, Fujishige, 2005]). We then have, for all $A \subset V$, $\nu \in \mathcal{J}_k$ and $s \in B(H)$, $F(A) - H(A) \leqslant F_\nu(A) - s^\top 1_A$. This leads to the convex relaxation:

$$\max_{A \subset V} F(A) \leqslant \max_{y \in \mathcal{N}_k} \min_{\nu \in \mathcal{J}_k, s \in B(H)} \sum_{C \in \mathcal{D}_k} F(C) \nu_C y_C - \sum_{k \in V} s_k y_{\{k\}}.$$

Constrained problems. One common practical benefit of having convex relaxations is their flexibility: it is easy to add constraints on the problem. In our variational framework, any constraints that can be expressed as convex constraints on $y \in \mathcal{M}_k$ may be added. For instance, it includes the cardinality constraint.

3.6 Experiments

In this section, we show the results of our algorithm to solve max-cut on graphs with different configurations: trees, 2D-grid and random graphs. In all our exper-

iments we restrict ourselves to $k = 1$, i.e., simple spanning trees. Given a set of weights in an undirected graph, $d : V \times V \rightarrow \mathbb{R}_+$, a cut is defined as $F(A) = d(A, V \setminus A) = \sum_{i \in A, j \in V \setminus A} d(i, j)$. The function F is known to be a non-monotone submodular function. To illustrate our algorithm, we generated synthetic graphs of different configurations with $|V| = 100$ nodes and random positive edge weights. In the case of a tree-based cut functions, the algorithm converges to an optimal solution in the first iteration. In the case of 2D grid (10×10), the algorithm converges to an optimal solution as shown Figure 3-1-(a). We also show the performance of other constant factor approximation algorithm proposed by Buchbinder et al. [Buchbinder et al., 2012] and Feige et al. [Feige et al., 2011] on this configuration. For generating random graphs, we considered $|V| = 100$ nodes with random edge incident on each vertex with probability 0.9. It can be observed in Figure 3-1-(b) that our algorithm solves a convex optimization problem but with a larger integrality gap. This gap could be reduced by using higher treewidth graphs, i.e., $k > 1$ instead of trees.

3.7 Conclusion

In this chapter, we have developed a novel approximation framework for submodular functions, which enables us to provide convex relaxations of submodular function maximization and related problems. While we have considered only trees in our experiments, it is of clear interest to consider higher treewidths and explore empirically the trade-offs between computational complexity and tightness of our relaxations.

Chapter 4

Convex Relaxations for Parallel Energy Minimisation

Abstract

Energy minimization has been an intensely studied core problem in computer vision. With growing image sizes (2D and 3D), it is now highly desirable to run energy minimization algorithms in parallel. But many existing algorithms, in particular, some efficient combinatorial algorithms, are difficult to parallelize. By exploiting results from convex and submodular theory, we reformulate the quadratic energy minimization problem as a total variation denoising problem, which, when viewed geometrically, enables the use of projection and reflection based convex methods. The resulting min-cut algorithm is conceptually very simple, and solves a sequence of TV denoising problems. We perform an extensive empirical evaluation comparing state-of-the-art combinatorial algorithms and convex optimization techniques. On small problems the iterative convex methods match the combinatorial max-flow algorithms, while on larger problems they offer other flexibility and important gains: (a) their memory footprint is small; (b) their straightforward parallelizability fits multi-core platforms; (c) they can easily be warm-started; and (d) they quickly reach approximately good solutions, thereby enabling faster "inexact" solutions. A key consequence of our approach based on submodularity and convexity is that it is allows to combine *any arbitrary combinatorial or convex methods as subroutines*, which allows one to obtain hybrid combinatorial and convex optimization algorithms that benefit from the strengths of both.

This chapter is based on our preprint "Convex Optimization for Parallel Energy Minimization", K. S. Sesh Kumar, A. Barbero, S. Jegelka, S. Sra and F. Bach [Sesh Kumar et al., 2015].

4.1 Goal

Goal. In this chapter, we explore and compare methods for energy minimisation problems typical to computer vision setting that allow combining convex and com-

binatorial optimisation, and thereby offer a way to parallelise recent successful combinatorial methods [Boykov and Kolmogorov, 2004, Chandran and Hochbaum, 2009, Goldberg et al., 2011]. Our algorithms can run on large datasets while using only limited memory, and are flexible enough to be ported to different hardware architectures.

Motivation. Energy minimisation has become a key element in many low-level to mid-level tasks in computer vision, such as segmentation or stereo correspondence (see [Blake et al., 2011] for a survey). Many minimisation problems in computer vision can be solved using graph cuts that have efficient codes available [Boykov et al., 2001]. These can be used to solve submodular quadratic penalties and can also be extended to higher-order and non-submodular potentials too. However, when applying widely used graph-cut code to huge problems in 3D or video, running time and memory usage becomes problematic. Ideally, we would wish to have algorithmic flexibility to decompose the problem into arbitrary subproblems that can be solved in parallel and adapt to new architectures such as GPU clusters. These latter needs can be met through convex optimisation.

Energy minimisation problems reformulated as min-cut problems as shown in Section 1.2.3 and its relaxed form of total variation minimisation lead to tight relaxations. This enabled Komodakis et al. [2011], Komodakis et al. [2008] and Savchynskyy et al. [2011] to propose iterative convex optimisation methods for solving such problems. However, the lack of smoothness (in the primal and dual problems) poses some difficulties in solving the problem efficiently.

Hence, they can be reformulated into total variation denoising outlined by [Hochbaum, 2001, Jegelka et al., 2013]. The gist of this approach is to replace the non-smooth relaxation by the total variation (TV) denoising problem

$$\min_{w\in\mathbb{R}^n} f(w) + \frac{1}{2}\|u - w\|^2, \tag{TV}$$

from which one may obtain an optimal solution of the energy minimisation problem by thresholding. The benefit of this formulation is its smooth dual problem, which has a natural geometric interpretation: it reduces to computing the distance between two convex sets. Moreover, via the equivalence between energy minimisation and projections we obtain fast projection subroutines, which in turn enable the use of classical, popular projection methods [Jegelka et al., 2013].

Organisation. This chapter is organised as follows.

— We assume that the total variation f can be decomposed into sum of r total variation functions, which can be solved in parallel. We derive the optimisation problem to optimising the original total variation problem in Section 4.2.

— We show how algorithms proposed to solve the best approximation problem introduced in Chapter 1 can be used to solve the total variation minimisation in the decomposable setting in Section 4.3.

— In Section 4.4 and Section 4.5, we give implementation details and a thorough

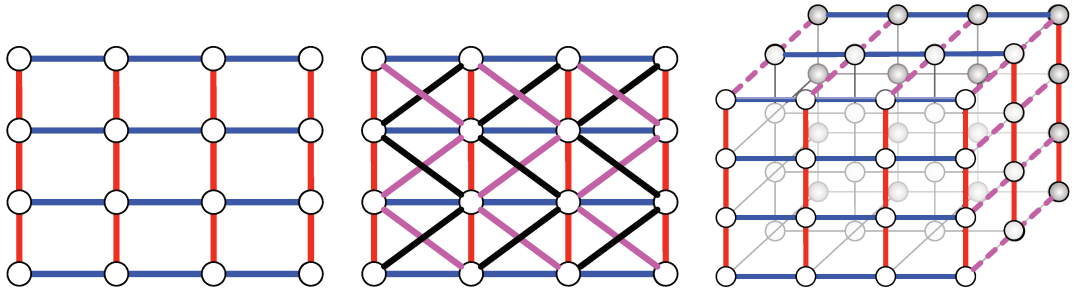


Figure 4-1 – Grid graph structures and decompositions, indicated by different colors. (a) 2D, 4-connected. (b) 2D, 8-connected. (c) 3D tensor, 6-connected.

analysis of our experiments when compared to some standard combinatorial algorithms.

4.2 Decomposition of graphs

We assume that the total variation f (and equivalently the energy function) may be decomposed into a sum $\sum_{j=1}^{r} f_j(w)$ of r total variation functions f_j. Each function f_j can be defined on 2D grids, trees or chains. The only criterion for the decomposition is to efficiently perform orthogonal projections onto the polytope K_j corresponding to f_j. In fact, f_j does not need to have full support.

While other decompositions are possible [Kolmogorov, 2006, Sontag and Jaakkola, 2009], in this chapter, we focus on chains to illustrate our ideas; we note that 1D-TV (on chains) can be solved very efficiently–see Section 4.2.2. Note that the same idea works for decompositions into trees [Kolmogorov et al., 2015], stars, small "cubes" and 2D sheets.

4.2.1 2D and 3D grids

The most frequently used 2D and 3D topologies in computer vision decompose straightforwardly in several ways [Strandmark and Kahl, 2010, Komodakis et al., 2011, Kolmogorov, 2012, Savchynskyy et al., 2011, Kolmogorov et al., 2015]. In our experiments, we split the 2D 4-connected grid into vertical and horizontal lines (Figure 4-1-(a)), the 2D 8-connected grid additionally into zig-zagged paths (Figure 4-1-(b)), and 3D 6-connected grid into vertical, horizontal and depth lines (Figure 4-1-(c)). Diagonal lines are also possible 2D 8-connected grids but less efficient with respect to memory access.

Even though the set of 1D lines of equal color in the figures may be considered one f_j, the lines are independent and may be addressed in parallel. Further parallelisation (across colors) results from our splitting formulation. Finally, any fast algorithm for solving a 2D or 3D TV (or graph cut) problem may be used as a subroutine when

setting f_j to be an entire 2D grid sheet or 3D tensor.

4.2.2 Efficient 1D TV

As chains are the building blocks of our decomposition, performance of the overall method is heavily influenced by the speed at which 1D-TV can be solved. Being a classic regulariser for image denoising, literature on solvers for different variations of 1D and 2D TV abounds, though only recently fast direct methods for chains have been proposed. A notable example is the dynamic programming method of Johnson [Johnson, 2013], which guarantees linear complexity. Another outstanding method is that of Condat [Condat, 2012]; it is based on a thorough analysis of the KKT conditions and manages to achieve faster running times in practice, despite a pathological quadratic cost worst-case. These TV solvers, however, only apply to chains with constant weights.

To permit varying weights we use a recent method of [Barbero and Sra, 2014] that obtains Condat's method through a taut-string viewpoint [Grasmair, 2007], in a way that allows weights along the chain. Experiments [Barbero and Sra, 2014] indicate that this method shares the same performance as the original procedure, therefore rapidly solving TV chains in linear time in practice.

We also point out that the choice of the 1D-TV solver is independent of the overlying topologies and optimisers. This allows us to localise complexity to highly tuned TV chain solvers for the architecture under use (multicore, GPUs, etc.), thus providing overall modularity and adaption to the underlying hardware. In this chapter, we use a general implementation for CPUs.

Message passing. Alternatively to the method that we have used, we may also use message passing techniques, which could be more efficient on certain architectures. These are directly adapted to solve the min-cut problem on a chain or a tree, not the total variation problem. However, it is known that by a sequence of at most n min-cut problems, one may obtain the exact TV solution [Hochbaum and Hong, 1995].

4.2.3 Decomposed dual problems

By their form (total variation or Lovász extensions of submodular functions), the functions f_i may be represented as a maximum of linear functions, that is, $f_j(w) = \max_{s_j \in K_j} \; s_j^\top w$, for K_j a certain polytope, $j \in \{1, \dots, r\}$. This form as well as the decomposability of the total variation may be used to obtain a decomposed dual problem for the continuous Problem (C) that is described in Chapter 1. The dual splits in the same way as the primal, and admits parallel optimisation algorithms [Strandmark and Kahl, 2010, Komodakis et al., 2011, Kolmogorov, 2012, Savchynskyy et al., 2011]. It however has a non-smooth objective function that makes optimisation harder.

We hence next describe two dual problems for the (TV) problem [Jegelka et al., 2013] that extends the dual formulation in Eq. (1.13) to sums of submodular functions.

First dual problem. We use a standard reformulation for dual decomposition: we introduce a variable $\mathbf{w} = (w_1, \dots, w_r) \in \mathbb{R}^n \times \cdots \times \mathbb{R}^n$ composed of r copies of the input variable w, with the constraint that $w_i = w$ for each $i \in \{1, \dots, r\}$ (see, e.g., [Boyd et al., 2011]). We then add Lagrange multipliers $\lambda_i \in \mathbb{R}^n$ for each of these constraints. Writing f_j as a maximum of linear functions introduces a dual variable vector $s_j \in K_j$. We collect those variables in a vector $\mathbf{s} = (s_1, \dots, s_r) \in \mathbf{K} = K_1 \times \cdots \times K_r$. Overall, we obtain the following traditional formulation for minimisation of sum of functions [Boyd et al., 2011].

$$
\begin{aligned}
&\min_{w \in \mathbb{R}^n} \ \sum_{j=1}^{r} f_j(w) + \frac{1}{2}\|u - w\|^2 \\
&= \max_{\boldsymbol{\lambda} \in \mathbb{R}^{n \times r},\ \mathbf{s} \in \mathbf{K}} \ \min_{w \in \mathbb{R}^n,\ \mathbf{w} \in \mathbb{R}^{n \times r}} \sum_{j=1}^{r} w_j^\top s_j + \frac{1}{2}\|u\|^2 - w^\top u + \frac{1}{2r}\sum_{j=1}^{r}\|w_j\|^2 + \sum_{j=1}^{r}\lambda_j^\top (w - w_j) \\
&= \max_{\boldsymbol{\lambda} \in \mathbf{L},\ \mathbf{s} \in \mathbf{K}} \frac{1}{2}\|u\|_2^2 - \frac{r}{2}\sum_{j=1}^{r}\|s_j - \lambda_j\|^2 \\
&= \max_{\boldsymbol{\lambda} \in \mathbf{L},\ \mathbf{s} \in \mathbf{K}} \frac{1}{2}\|u\|_2^2 - \frac{r}{2}\|\mathbf{s} - \boldsymbol{\lambda}\|^2, \qquad (4.1)
\end{aligned}
$$

where $\mathbf{L}$ denotes the set of $\boldsymbol{\lambda} \in \mathbb{R}^{n \times r}$ such that $\sum_{j=1}^{r} \lambda_j = u$. We are thus faced with the problem of finding the closest point between two convex sets, which we explore in Sections 4.3.2 and 4.3.3.

Note that if the function f_j only depends on a subset of variables of w, then we may restrict the corresponding variable λ_j to be zero on the complement of that subset in order to have faster convergence for the iterative methods presented below.

Second dual problem. Given $\mathbf{s} \in \mathbf{K}$ in Eq. (4.1), the optimal $\boldsymbol{\lambda} \in \mathbf{L}$ may be obtained in closed form:

$$\lambda_j = \frac{u}{r} + s_j - \frac{1}{r}\sum_{k=1}^{r} s_k. \qquad (4.2)$$

This leads to another dual problem, where the variables $\boldsymbol{\lambda} \in \mathbf{L}$ are maximised out:

$$\max_{\mathbf{s} \in \mathbf{K}} \ \frac{1}{2}\|u\|_2^2 - \frac{1}{2}\Big\|\sum\nolimits_{j=1}^{r} w_j - u\Big\|^2. \qquad (4.3)$$

The problem above has *separable constraints* $s_i \in K_j$, $j \in \{1, \dots, r\}$ and a smooth objective function. We will discuss optimisation procedures in Section 4.3.1.

Special case $r = 2$. When the function f is split into two functions, then the problem in Eq. (4.3) is equivalent to finding the distance between the convex set K_1 and the set $\{u - w_2 \mid \ w_2 \in K_2\}$, for which methods presented in Sections 4.3.2 and 4.3.3 may be used.

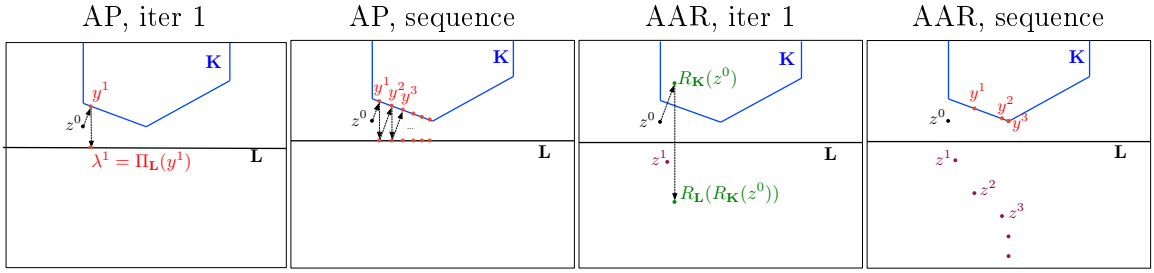


Figure 4-2 – Illustration of alternating projections (AP) and averaged alternating reflections (AAR) for Problem (4.1), the problem of finding the closest points between a polytope $\mathbf{K}$ and a subspace $\mathbf{L}$. The iterations start at the black point z^0. AP alternatingly projects onto $\mathbf{K}$ and then $\mathbf{L}$. AAR reflects at $\mathbf{K}$ and then at $\mathbf{L}$; the next iterate z^{t+1} (magenta) is the midpoint between z^t and $R_{\mathbf{L}}R_{\mathbf{K}}(z^t)$. For AAR, the sequence z^t diverges, but the projected shadow sequence of $y^t = \Pi_{\mathbf{K}}(z^t)$ converges (red). Here, AAR takes larger steps than AP and hence converges more quickly.

4.3 Optimisation for decomposable problems

Next, we describe optimisation procedures that exploit the decomposable structure of the dual problems Eq. (4.1) and Eq. (4.3), where in particular the Cartesian decomposition $\mathbf{K} = K_1 \times \cdots \times K_r$ of the constraint sets plays an important role. In particular, we exploit that the projection onto $\mathbf{K}$ consists of r independent projections onto the sets K_i. By the above derivations, each of those projections can be done quickly via a TV subroutine for each f_i.

4.3.1 Cyclic projections

The first method we consider for problem Eq. (4.3) is *block coordinate descent* (BCD), a classic method [Bertsekas, 1999] that has recently witnessed a huge resurgence of interest in large-scale optimisation [Richtárik and Takáč, 2012, Beck and Tetruashvili, 2013]. Since the cost function is a separable quadratic, BCD assumes a form that is more commonly known as *cyclic projections* (more precisely, this is so if we go through the constraint blocks in a cyclic order). Specifically, we update coordinate blocks $i = 1, \ldots, r$ as follows:

$$s_i \leftarrow \operatorname*{arg\,max}_{z \in K_i} \; -\frac{1}{2}\Big\|z - u + \sum_{j \neq i} s_i\Big\|^2 = \Pi_{K_i}\big(u - \sum_{j \neq i} s_i\big),$$

where Π_{K_i} denotes orthogonal projection onto set K_i. This projection is solved by solving a (fast) TV problem with f_i. Notice that the variable s_i is overwritten after the update, so that when updating s_{i+1}, the latest values $s_1, \ldots, s_i$ are used in the projection.

In addition to cyclic projections, one could solve the smooth dual Eq. (4.3) using a gradient-based method like FISTA [Beck and Teboulle, 2009]. Such a method is also easy to implement because the Lipschitz constant of the gradient is easily seen

to be r and the required projections decompose due to the structure of $\mathbf{K}$.

4.3.2 Alternating projections in product space

The method of cyclic projections offers a practical choice. However, it is inherently serial. To solve the problem in parallel, the first dual formulation Eq. (4.1) turns out to be more suited (note that this provides a second source of parallelisation, beyond the fact that each polytope K_j is itself a product of polytopes corresponding to individual lines).

The key idea is to exploit the "product space" $K_1 \times \cdots \times K_r$. Since w is constant, as previously mentioned, Eq. (4.1) is nothing but the problem of finding the closest point between two convex sets [Jegelka et al., 2013]. Applying BCD, except this time with just two coordinate blocks, we obtain the classic *alternating projections* (AP) (cast in a product space setting), which performs for $k = 0, 1, \ldots,$ the iteration:

$$\begin{aligned}\mathbf{s}^{k+1} &\leftarrow \operatorname*{arg\,max}_{\mathbf{s}\in\mathbf{K}} \; -\frac{r}{2}\|\mathbf{s}-\boldsymbol{\lambda}^k\|^2 = \Pi_{\mathbf{K}}(\boldsymbol{\lambda}^k),\\ \boldsymbol{\lambda}^{k+1} &\leftarrow \operatorname*{arg\,max}_{\boldsymbol{\lambda}\in\mathbf{L}} \; -\frac{r}{2}\|\boldsymbol{\lambda}-\mathbf{s}^{k+1}\| = \Pi_{\mathbf{L}}(\mathbf{s}^{k+1}).\end{aligned}$$

The key point here is that the projection $\Pi_{\mathbf{K}}$ decomposes

$$\Pi_{\mathbf{K}}(\boldsymbol{\lambda}) = (\Pi_{K_1}(\lambda_1), \ldots, \Pi_{K_r}(\lambda_r)),$$

so that each of the coordinate blocks may be computed in parallel (our implementation exploits this fact), while the projection $\Pi_{\mathbf{L}}$ is merely an averaging step detailed in Eq. (4.2).

4.3.3 Alternating reflections in product space

The recent work [Jegelka et al., 2013] provided strong experimental evidence that for projection problems of the form Eq. (4.1), AP is often outperformed by a more refined method of [Bauschke and Luke, 2004], namely, *averaged alternating reflections (AAR)*. Here, instead of alternating between the projection operations $\Pi_{\mathbf{K}}$ and $\Pi_{\mathbf{L}}$, one uses *reflection operators*

$$R_{\mathbf{K}} := 2\Pi_{\mathbf{K}} - I, \quad R_{\mathbf{L}} := 2\Pi_{\mathbf{L}} - I, \tag{4.4}$$

while averaging them to ensure firm nonexpansivity, a property that greatly simplifies convergence analysis [Bauschke and Luke, 2004]. To apply the AAR method, one first introduces the auxiliary vector $\mathbf{z}$, which represents $\mathbf{s} - \boldsymbol{\lambda}$. Then, AAR takes the form

$$\mathbf{z}^{k+1} = \tfrac{1}{2}(R_{\mathbf{L}}R_{\mathbf{K}} + I)\mathbf{z}^k. \tag{4.5}$$

However since usually $\mathbf{K} \cap \mathbf{L} = \emptyset$, the sequence $(\mathbf{z}^k)$ generated by Eq. (4.5) diverges to infinity! The remarkable fact is that from this diverging sequence, we can extract a solution by maintaining a "shadow sequence" $\mathbf{s}^k \equiv \Pi_{\mathbf{K}}(\mathbf{z}^k)$. See Figure 4-2

for an illustration, and Theorem 3.13 in [Bauschke and Luke, 2004] for a proof of convergence.

4.3.4 Extensions

Above, we outlined flexible, parallelizable convex optimisation algorithms for energy minimisation with pairwise submodular potentials. These algorithms straightforwardly generalise from binary labels to the multi-label case, to submodular higher-order potentials, and to related problems. The reasons are two-fold: (1) the above algorithms solve a minimum cut problem, and any methodological machinery that builds on graph cuts as a subroutine will work with the above algorithms too; (2) the decomposition theory and tightness of the relaxations hold generically for submodular functions, not only graph cuts.

For multi-label energy minimisation, one may use move-making algorithms [Boykov et al., 2001] that reduce the multi-label problem to a series of binary submodular energy minimisation problems. The methods above solve those binary problems. For combinatorial algorithms, it has proved useful to reuse existing solutions and data structures [Kohli and Torr, 2005]. "Warm-starting" is possible for the convex case too: we simply use the y_i vectors of the previous problem to initialise the new problem. If the geometry of the polytopes K has not changed too much, this can save many iterations (see Figure 4-3(b)).

Second, the convex approach directly generalises to submodular potentials that involve more than two nodes at a time (following [Jegelka et al., 2013]); such potentials include [Chambolle and Darbon, 2009, Stobbe and Krause, 2010, Kohli et al., 2009a,b, Hein et al., 2013]. Many of those potentials correspond to sufficiently simple submodular functions, often with small support, such that the relaxation (the equivalent to total variation for graph cuts) can be solved fast. Moreover, the same methods may even generalise to be used with roof duality [Rother et al., 2007].

Finally, since the above methods also solve the parametric version of the discrete problem (by thresholding the solution of Eq. (TV) at different levels) as a byproduct, they are also applicable to the numerous applications of parametric graph cuts [Kolmogorov et al., 2007, Hochbaum, 2013].

4.4 Implementation details

The algorithms are inherently parallel by design as each projection/reflection onto a chain graph is independent of the other. Our implementation assumes decomposed functions and the decomposition depends on the problem at hand. In Section 4.2.1, we described some possible decompositions of grid-like graph structures on 2D and 3D graphs. However, this extends to many other decompositions. Empirically and theoretically [Nishihara et al., 2014], longer connected structures lead to faster convergence than decomposition e.g. into single edges.

4.4.1 Parallelisation

We use the efficient 1D-TV implementation of [Barbero and Sra, 2014] to solve the projection/reflection on chains in parallel. Our implementation is in C++ and uses OpenMP; it ensures that the memory access pattern across threads is streamlined, since bad memory access patterns can lead to considerable slowdowns. While the 1D-TV solver is not optimised for GPUs, as explained in Section 4.2.2, it can be replaced by message passing based subroutines, which are inherently parallel and also friendly to GPU architectures.

4.4.2 Memory footprint

In our implementation every decomposable function must maintain states of dual variables for each node in the graph. Thus, the memory requirement of our methods increases bilinearly in the number of decomposed functions and the number of nodes. Our experiments suggest that the projection-based algorithms require less memory than standard combinatorial algorithms—see Table 4.1. Unlike the 32 bit integers used in many other implementations, we use (64 bit) double precision numbers. Reducing those to 32 bit would reduce the memory requirements of the projection methods even further.

Algorithm	Memory(GB)
AAR	26.82
BK [Boykov and Kolmogorov, 2004]	42.83
IBFS [Goldberg et al., 2011]	44.16
HPF [Chandran and Hochbaum, 2009]	55.46

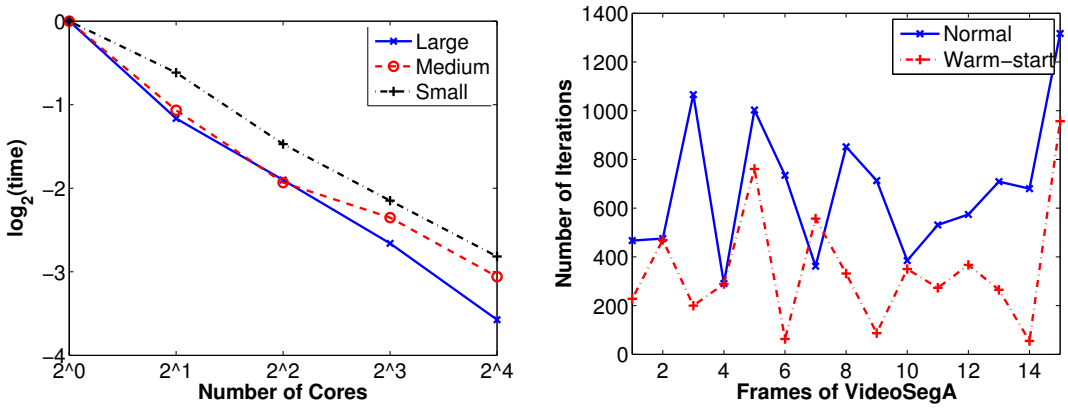
Table 4.1 – Memory footprint for *Abdomen* dataset ($512 \times 512 \times 551$)

4.4.3 Running time

With the TV subroutine we use, each projection/reflection step scales in the worst case quadratically in the length of the chain (the chain length is typically equal to $\sqrt{n}$ or $\sqrt[3]{n}$, where n is the total number of nodes in the 2D or 3D graphs), but is in practice usually linear [Barbero and Sra, 2014]. In fact, it did not scale quadratically for any of our data. Hence, empirically, the cost of each iteration grows bilinearly with the number f of functions f_i, and with the number of nodes in the graph.

4.5 Experiments

Our experiments study the performance of the projection algorithms on 2D and 3D-maxflow datasets [max], exploiting in particular the parallel nature of the algorithms. We compare the algorithms to standard, popular maxflow implementa-



(a) (b)

Figure 4-3 – Speedup of the AAR algorithm with parallelisation and warm starts. (a) Normalised scale of performance with increasing number of cores on Bunny datasets [max] of different resolutions. (b) Number of iterations taken by each from of a video with "normal" initialisation and "warm start" from the dual variables of the previous frame.

tions such as BK [Boykov and Kolmogorov, 2004], IBFS [Goldberg et al., 2011], and HPF [Chandran and Hochbaum, 2009]. For other algorithms [Strandmark and Kahl, 2010, Komodakis et al., 2011], we have not been able to find implementations that were easily portable to 3D datasets.

Table 4.2 shows running time and the number of iterations for the projection algorithms and others on a multicore machine[1]. The timings are recorded using the *gettimeofday* command. All the timings are for the optimisation phase only and exclude data I/O (which is common to all methods).

4.5.1 2D problems

As a 2D example, we use the tsukuba data [max], a multi-label task on 2D images corresponding to 4-neighborhood grids. To cope with multiple labels, we use alpha expansion [Boykov et al., 2001]. Notably, the decomposition for 2D 4-connected grids uses only two functions (vertical and horizontal), and therefore corresponds to the special dual for $r = 2$ in Section 4.2.3. For this formulation, AAR converges remarkably faster than other iterative algorithms, and even outperforms the combinatorial methods. The time comparisons at the bottom of Table 4.2 show that in general, the running times of our methods are comparable to standard combinatorial algorithms on images of size 384×288.

Warm starts. Figure 4-3(b) shows the number of iterations required for the algorithm to converge on each frame of size 480×360 of the VideoSegA [max] dataset.

1. 20 core, Intel(R) Xeon(R) CPU E5-2670 v2 @ 2.50GHz with 100Gigabytes of memory. We only use up to 16 cores of the machine to ensure accurate timings.

These are the consecutive frames of a video, which are 2D images with 8-neighborhood grids (see Figure 4-1). We use the dual variables at the convergence of the previous frame to warm-start the projection/reflection process. This makes the method converge to the optimal solution substantially quicker than with other initialisations.

4.5.2 3D problems

On the 3D data, the running times of the algorithms differs more widely. In particular, the number of iterations for the algorithms to converge appears to depend on two important characteristics: (i) number of nodes in the graph (dimensionality) (ii) the edge weights in the graph (weights of the TV term). The latter affects the size of the polytopes K, i.e., the diameter of the domain of the dual problem, a parameter that commonly influences the convergence of convex optimisation methods (see also [Nishihara et al., 2014]).

Effect of edge weights. Table 4.2 shows results for larger edge weights, as well as weights scaled by a factor of 0.1. The iterative methods become faster with smaller weights, while the combinatorial methods robustly perform well with large weights too. On many instances, AAR converges faster than the cyclic BCD, while on others BCD is faster.

Approximate solutions. Since we can obtain a feasible solution (discrete cut) from any iterate of the projection methods by thresholding the continuous vector, Table 4.2 also shows the time taken to obtain an approximate solution with limited error (10% and 2%, measured by Jaccard distance). The results suggest that, while a complete dual certificate of convergence for the discrete problem takes a bit longer, a reasonable approximate solution can be obtained fairly quickly.

Parallel speedup. Figure 4-3-(a) shows the speedup of AAR achievable with an increasing number of cores. This figure reports the running time on the Bunny dataset (3D) with different resolutions: *Large* ($401 \times 396 \times 312$), *Medium* ($202 \times 199 \times 157$), and *Small* ($102 \times 100 \times 79$). It is evident that more cores can improve the performance of the algorithm considerably. When using GPUs, it is important to consider their limited video memory, and hence the algorithms need to have a low memory footprint to perform well.

Memory. Apart from running time, we also investigate the memory footprint of the algorithms. Table 4.1 shows the memory footprint of all algorithms on the *Abdomen* data [max], which is $512 \times 512 \times 551$. AAR uses considerably less memory than the standard algorithms.

Dataset	Time in seconds								Iterations				
	AAR (< 10%)	AAR (< 2%)	AAR-JD (< 0.1)	AAR-JD (< 0.02)	AAR	BK	IBFS	HPF	AAR	AAR (0.1x)	AP	BCD	FISTA
(3D) bone-100	4.2	7.4	3.9	10.9	14.4	8.5	6.3	1.0	105	73	846	146	422
(3D) bone-10	4.5	7.4	4.7	9.6	18.5	5.6	3.4	0.9	134	25	1183	206	592
(3D) bone_x-100	0.08	0.08	0.08	0.08	3.90	2.42	1.02	1.08	45	34	42	9	25
(3D) bone_x-10	0.09	0.09	0.09	0.09	3.70	1.86	0.93	0.75	45	23	44	10	26
(3D) bone_xy-100	0.01	0.01	0.02	0.02	1.25	0.79	0.68	0.40	44	36	26	6	16
(3D) bone_xy-10	0.01	0.01	0.03	0.03	1.18	0.64	0.53	0.29	36	21	27	7	17
(3D) bone_xyz-100	0.39	0.54	0.51	0.51	0.91	0.47	0.32	0.16	57	43	185	36	98
(3D) bone_xyz-10	0.34	0.57	0.46	0.48	0.82	0.36	0.33	0.11	57	21	185	37	98
(3D) bone_xyz_x-100	0.42	0.43	0.42	0.43	0.53	0.14	0.09	0.06	53	48	618	88	256
(3D) bone_xyz_x-10	0.43	0.47	0.39	0.39	0.48	0.12	0.03	0.05	50	23	615	97	290
(3D) bone_xyz_xy-c100	0.24	0.24	0.21	0.21	0.26	0.05	0.03	0.03	29	28	117	25	63
(3D) bone_xyz_xy-c10	0.18	0.22	0.18	0.22	0.24	0.04	0.02	0.02	30	23	120	25	64
(3D) babyface-100	3.6	8.2	9.2	32	33.5	25.7	12.3	9.4	509	346	873	550	1360
(3D) babyface-10	6.2	9.4	7.5	13.9	35.2	14.3	7.9	7.6	543	223	793	420	1162
(3D) bunny-lrg	16.7	28.3	1.28	1.28	186.6	9.5	6.3	41.2	145	52	796	133	406
(3D) bunny-med	1.72	2.72	0.14	0.14	7.47	1.07	1.27	2.27	52	25	94	17	52
(3D) bunny-sml	0.12	0.19	0.11	0.11	0.38	0.11	0.17	0.23	35	16	111	18	58
(3D) liver-100	10.3	15.0	4.88	4.88	38.5	7.3	4.7	4.7	654	503	1682	1444	2873
(3D) liver-10	10.4	15.8	5.06	5.06	33.1	3.4	3.2	3.3	523	407	1586	1290	2754
(3D) abdomen_long	525	701	441	1445	1445	212	110	68	468	349	2532	939	1432
(3D) abdomen_short	578	772	468	1540	1593	119	60	29	485	231	2373	953	1428
(3D) adhead-100	9.5	27.2	0.2	8.5	42.2	10.1	8.1	13.6	208	176	453	104	148
(3D) adhead-10	9.1	25.4	0.2	4.5	42.3	6.3	8.4	10.5	208	105	395	111	176
(2D) BVZ-tsukuba	0.15	0.18	0.02	0.01	0.21	0.31	0.20	0.24	30	25	110	79	50

Table 4.2 – Performance comparison of AAR with BK [Boykov and Kolmogorov, 2004], IBFS [Goldberg et al., 2011], and HPF [Goldberg et al., 2011] on 3D datasets with 6 connectivity. AAR($< p\%$) denotes the time taken for the algorithm to find a cut whose difference to the optimal cut is $p\%$ of the difference between the cut in the first iteration and the optimal cut. AAR-JD($< p$) denotes time taken by the algorithm to reduce Jaccard Distance to p. AAR(0.1x) is the number of iterations taken by AAR after scaling the pairwise weights by 0.1.

4.6 Conclusion

We have proposed parallel iterative algorithms for binary energy minimisation problems. The algorithms rely on a fast projection subroutine. For binary submodular potentials (graph cuts), this subroutine is simply a total variation problem, which can be efficiently solved on sub-graphs with special structure. In other examples, these subroutines could be fast algorithms for solving cuts on arbitrary subgraphs, or for simpler submodular energies. Hence, while the experiments here concentrate on cuts and decompositions into line graphs, the same methods apply to decompositions into 2D sheets, 3D cubes or any other subgraphs, and to sums of simple higher-order potentials.

We observed that the iterative methods perform similarly to combinatorial methods on 2D grid graphs, and require less memory than other, popular implementations of maximum flow algorithms. The tradeoffs between convex and combinatorial methods illustrated here have some interesting implications, and suggest a wider study of integrating combinatorial and convex methods via different decompositions. For example, instead of TV oracles for line graphs, one may use discrete oracles such as BK [Boykov and Kolmogorov, 2004] for larger specialised subgraphs. A 3D tensor can then be easily decomposed into two components: grids and lines.

We propose an algorithm which enables us use to these discrete oracles instead of total variation oracles for solving the Problem (TV) in Chapter 5.

Chapter 5

Active-Set Methods for Submodular Minimisation Problems

Abstract

We consider submodular optimization problems such as submodular function minimization, referred to as SFM and quadratic problems regularized by the Lovász extension; for cut functions, this corresponds respectively to graph cuts and total variation (TV) denoising. Given a submodular function with an SFM oracle, we propose a new active-set algorithm for total variation denoising, which is more flexible than existing ones; the algorithm may be seen as a local descent algorithm over ordered partitions with explicit convergence guarantees. For functions that decompose into the sum of two functions F_1 and F_2 with efficient SFM oracles, we propose a new active-set algorithm for total variation denoising (and hence for SFM by thresholding the solution at zero). This algorithm also optimizes over ordered partitions and improves over existing ones based on TV or SFM oracles for F_1 and F_2.

This chapter is based on our preprint "Active-set Methods for Submodular Optimization", K. S. Sesh Kumar, F. Bach [Sesh Kumar and Bach, 2015] and another article "Active-set Methods for Submodular Minimisation Problems", K. S. Sesh Kumar, F. Bach under submission to International Journal on Computer Vision.

5.1 Goal

Goal. In this chapter, we consider a normalised submodular function F defined on $V = \{1, \dots, n\}$, i.e., $F : 2^V \to \mathbb{R}$ such that $F(\varnothing) = 0$ and a modular function u, i.e., $u \in \mathbb{R}^n$. We aim at minimizing with respect to $w \in \mathbb{R}^n$:

$$f(w) - u^\top w + \tfrac{1}{2}\|w\|_2^2, \tag{5.1}$$

where f is the Lovász extension of F. If F is a cut function in a weighted undirected graph, then f is its total variation, hence the denomination of *total variation denoising* problem, which we use in this chapter—since it is equivalent to minimizing $\frac{1}{2}\|u - w\|_2^2 + f(w)$.

We also consider the general submodular function minimisation (SFM) problem:

$$\min_{w\in[0,1]^n} f(w) - u^\top w = \min_{A\subseteq V} F(A) - u(A), \tag{5.2}$$

where we use the convention $u(A) = u^\top 1_A$, with $1_A \in \{0,1\}^n$ is the indicator vector of the set A.

In this chapter, we assume that F can be decomposed into "simple" submodular functions such that $F = \sum_{i=1}^r F_i$ and we have SFM oracles that solve,

$$\min_{A\subseteq V} F_i(A) - u_i(A), \tag{5.3}$$

where u_i is a modular function. Our goal is to propose iterative algorithms to solve these two problems given in Eq. (5.1) and Eq. (5.2) using SFM oracles of "simple" functions that solve Eq. (5.3). Note that our algorithms minimise general submodular functions as any submodular function can be decomposed into a normalised submodular function, F, i.e., $F(\varnothing) = 0$ and a modular function, u [Bach, 2013].

Motivation. Generic algorithms to optimise SFM in Eq. (5.2) or TV in Eq. (5.1) problems which only access F through function values (e.g., subgradient descent or min-norm-point algorithm) are too slow without any assumptions [Bach, 2013], as for signal processing applications, high precision is typically required (and often the exact solution).

For decomposable problems, i.e., when $F = F_1 + \cdots + F_r$, where each F_j is "simple". when algorithms use more powerful oracles than function evaluations, the running times improve. When only SFM oracles are used for each function F_j [Stobbe and Krause, 2010], they remain significantly slower than existing algorithms. However, when total variation oracles for each F_j are used, they become competitive as shown in Chapter 4 and [Jegelka et al., 2013]. Note that, in general, the exact total variation oracles are at most $O(n)$ times expensive than their SFM oracles. However, there are a subclass of submodular functions (cut functions and other submodular functions that can be written in form of cuts) whose total variation oracles are only $O(1)$ times expensive than the corresponding SFM oracles but are still more expensive than their total variation oracles. Therefore, the goal is to design fast optimisation strategies using efficient SFM oracles for each function F_j rather than their expensive TV oracles in Chapter 4 and [Jegelka et al., 2013] to solve the SFM and TV denoising problems of F given by Eq. (5.2) and Eq. (5.1) respectively. An algorithm was proposed by [Landrieu and Obozinski, 2016] to search over partition space for solving more generic problems of the form Eq. (5.1) with the unary terms $(-u^\top w)$ replaced by a convex differentiable function. Probabilistic graphical models provide a classical example of functions with efficient SFM oracles.

Organisation. This chapter is organised as follows.

— We exploit the polytope structure of these non-smooth optimisation problems, where each face is indexed by a partition of the underlying ground set $V =$

$\{1, \dots, n\}$. The main insight of this chapter is that once given a face of the main polytope associated with the submodular function and its tangent cone, orthogonal projections may be done in linear time by isotonic regressions. We will only need SFM oracles, i.e., the minimisation of $F(A) - s(A)$ with respect to $A \subseteq V$ for all possible $s \in \mathbb{R}^n$, to check optimality of this partition and/or generate a new partition.

— Given a submodular function F with an SFM oracle, we propose a new active-set algorithm for total variation denoising in Section 5.2, which is more efficient and flexible than existing ones (i.e., it allows warm restarts). This algorithm may be seen as a local descent algorithm over ordered partitions.

— Given a decomposition of $F = F_1 + F_2$, with available SFM oracles for each F_j, we propose an active-set algorithm for total variation denoising in Section 5.3 (and hence for SFM by thresholding the solution at zero). These algorithms optimises over ordered partitions (one per function F_j). Following [Jegelka et al., 2013], they are also naturally parallelizable. Given that only SFM oracles are needed, it is much more flexible, and allow more applications as shown in Section 5.4.

5.2 Ordered Partitions and Isotonic Regression

The main insight of this chapter is

(a) to consider the detailed face structure of the base polytope $B(F)$ and

(b) to notice that for the outer approximation of $B(F)$ based on the tangent cone to a certain face, the orthogonal projection problem (which is equivalent to constrained TV denoising) may be solved efficiently in $O(n)$ using a simple algorithm used to solve isotonic regression. This allows an explicit efficient local search over ordered partitions.

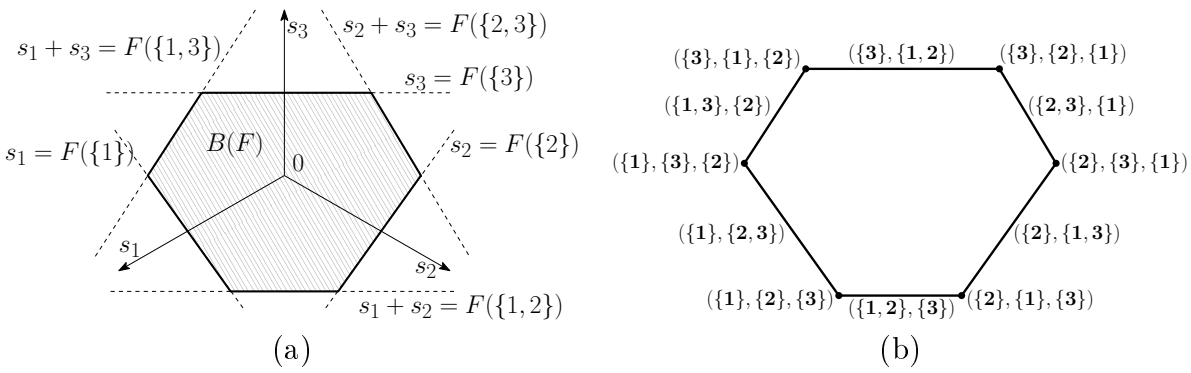


Figure 5-1 – Base polytope for $n = 3$. (a) Definition from its supporting hyperplanes $\{s(A) = F(A)\}$. (b) Each face (point or segment) of $B(F)$ is associated with an ordered partition.

5.2.1 Outer approximations of $B(F)$

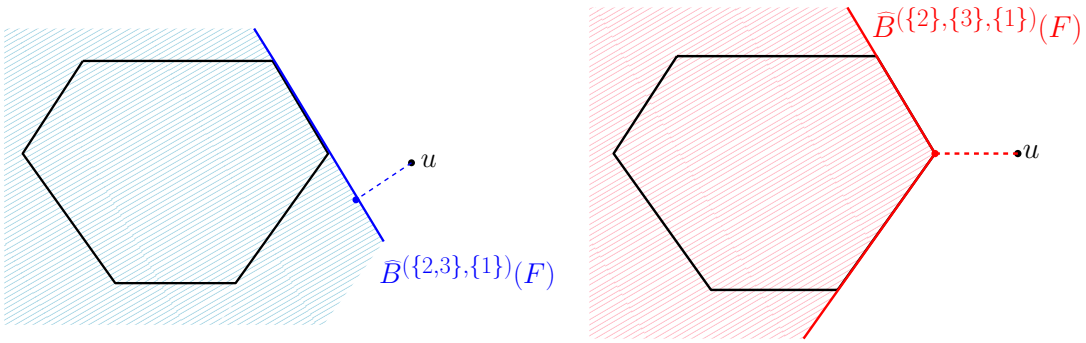
In this section, we use ordered partitions to define outer approximations of the base polytope and derive the support function of the outer approximation.

Supporting hyperplanes. The base polytope is defined as the intersection of half-spaces $\{s(A) \leqslant F(A)\}$, for $A \subseteq V$. Therefore, faces of $B(F)$ are indexed by subsets of the power set. As a consequence of submodularity [Bach, 2013, Fujishige, 2005], the faces of the base polytope $B(F)$ are characterised by "ordered partitions" $\mathcal{A} = (A_1, \dots, A_m)$ with $V = A_1 \cup \cdots \cup A_m$. Then, a face of $B(F)$ is such that $s(B_i) = F(B_i)$ for all $B_i = A_1 \cup \cdots \cup A_i$, $i = 1, \dots, m$. See the Figure 5-1-(b) for the enumeration of faces for $n = 3$ based on an enumeration of all ordered partitions. Such ordered partitions are associated to vectors $w = \sum_{i=1}^m v_i 1_{A_i}$ with $v_1 > \ldots > v_m$ with all solutions of $\max_{s \in B(F)} w^\top s$ being on the corresponding face.

From a face of $B(F)$ defined by the ordered partition $\mathcal{A}$, we may define its *tangent cone* $\widehat{B}^{\mathcal{A}}(F)$ at this face as the set

$$\widehat{B}^{\mathcal{A}}(F) = \Big\{ s \in \mathbb{R}^n, s(V) = F(V), \forall i \in \{1, \dots, m-1\}, s(B_i) \leqslant F(B_i) \Big\}. \tag{5.4}$$

These are outer approximations of $B(F)$, as illustrated in Figure 5-2 for two ordered partitions.



(a) (b)

Figure 5-2 – Projection algorithm for a single polytope: projecting on the outer approximation (a) $\widehat{B}^{(\{2,3\},\{1\})}(F)$, with a projected element which is not in $B(F)$ (blue), then on (b) $\widehat{B}^{(\{2\},\{3\},\{1\})}(F)$, with a projected element being the projection of s onto $B(F)$ (red).

Support function. We may compute the support function of $\widehat{B}^{\mathcal{A}}(F)$, which should be an upper bound on $f(w)$ since this set is an outer approximation of $B(F)$.

$$\sup_{s \in \widehat{B}^{\mathcal{A}}(F)} w^\top s \;=\; \sup_{s \in \mathbb{R}^n} \inf_{\lambda \in \mathbb{R}_+^{m-1} \times \mathbb{R}} w^\top s - \sum_{i=1}^m \lambda_i (s(B_i) - F(B_i))$$

$$
\begin{aligned}
& \text{(using Lagrangian duality)}, \\
= & \inf_{\lambda \in \mathbb{R}_+^{m-1} \times \mathbb{R}} \sup_{s \in \mathbb{R}^n} s^\top \Big(w - \sum_{i=1}^m (\lambda_i + \cdots + \lambda_m) 1_{A_i} \Big) \\
& + \sum_{i=1}^m (\lambda_i + \cdots + \lambda_m) \big[F(B_i) - F(B_{i-1}) \big], \\
= & \inf_{\lambda \in \mathbb{R}_+^{m-1} \times \mathbb{R}} \sum_{i=1}^m (\lambda_i + \cdots + \lambda_m) \big[F(B_i) - F(B_{i-1}) \big] \\
& \text{such that } w = \sum_{i=1}^m (\lambda_i + \cdots + \lambda_m) 1_{A_i}.
\end{aligned}
$$

Thus, by defining $v_i = \lambda_i + \cdots + \lambda_m$, which are decreasing, the support function is finite for w having ordered level sets corresponding to the ordered partition $\mathcal{A}$ (we then say that w is *compatible* with $\mathcal{A}$), i.e., $w = \sum_{i=1}^m v_i 1_{A_i}$; then it is then equal to the Lovász extension $f(w)$. Otherwise, when w is not compatible with $\mathcal{A}$, the support function is infinite.

Let us now denote $\mathcal{W}^{\mathcal{A}}$ as a set of all weight vectors w that are compatible with the ordered partition $\mathcal{A}$. This can be defined as

$$
\mathcal{W}^{\mathcal{A}} = \{ w \in \mathbb{R}^n \mid \exists v \in \mathbb{R}^m, w = \sum_{i=1}^m v_i 1_{A_i}, v_1 \geq \ldots \geq v_m \}.
$$

Therefore,

$$
\sup_{s \in \widehat{B}^{\mathcal{A}}(F)} w^\top s = \begin{cases} f(w) & \text{if } w \in \mathcal{W}^{\mathcal{A}}, \\ \infty & \text{otherwise.} \end{cases} \tag{5.5}
$$

5.2.2 Isotonic regression for restricted problems

Given an ordered partition $\mathcal{A} = (A_1, \ldots, A_m)$ of V, we consider the original TV problem restricted to w in $\mathcal{W}^{\mathcal{A}}$. Since on this constraint set $f(w) = \sum_{i=1}^m v_i \big[F(B_i) - F(B_{i-1}) \big]$ is a linear function, this is equivalent to

$$
\min_{v \in \mathbb{R}^m} \sum_{i=1}^m v_i \big[F(B_i) - F(B_{i-1}) - u(A_i) \big] + \tfrac{1}{2} \textstyle\sum_{i=1}^m |A_i| v_i^2 \text{ such that } v_1 \geqslant \cdots \geqslant v_m. \tag{5.6}
$$

This may be done by isotonic regression in complexity $O(m)$ by the weighted pool-adjacent-violator algorithm [Best and Chakravarti, 1990]. Typically the solution v will have some values which are equal to each other, which corresponds to merging some sets A_i. If these merges are made, we now obtain a *basic ordered partition*[1]

1. Given a submodular function F and an ordered partition $\mathcal{A}$, when the unique solution problem in Eq. (5.6) is such that $v_1 > \cdots > v_m$, we say that we $\mathcal{A}$ is a *basic ordered partition* for $F - u$. Given any ordered partition, isotonic regression allows to compute a coarser partition (obtained by partially merging some sets) which is basic.

such that our optimal w has *strictly decreasing* values. Primal stationarity leads to explicit values of v given by $v_i = u(A_i)/|A_i| - (F(B_i) - F(B_{i-1}))/|A_i|$, i.e., given $\mathcal{A}$, the exact solution of the TV problem may be obtained in closed form.

Dual interpretation. Eq. (5.6) is a constrained TV denoising problem that minimises the cost function in Eq. (5.1) but with the constraint that weights are compatible with the ordered partition $\mathcal{A}$, i.e. $\min_{w \in \mathcal{W}^{\mathcal{A}}} f(w) - u^\top w + \frac{1}{2}\|w\|_2^2$. The dual of the problem can be derived exactly the same way as shown in Eq. (1.13), using the definition of the support function defined by Eq. (5.5). The corresponding dual is given by $\max_{s \in \widehat{B}^{\mathcal{A}}(F)} -\frac{1}{2}\|s-u\|_2^2$, with the relationship $w = u - s$ at optimality. Thus, this corresponds to projecting u on the outer approximation of the base polytope, $\widehat{B}^{\mathcal{A}}(F)$, which only has m constraints instead of the $2^n - 1$ constraints defining $B(F)$. See an illustration in Figure 5-2.

5.2.3 Checking optimality of a basic ordered partition

Given a basic ordered partition $\mathcal{A}$, the associated $w \in \mathbb{R}^n$ is optimal for the TV problem in Eq. (5.1) if and only if $s = u - w \in B(F)$ due to optimality conditions in Eq. (1.13), which can be checked by minimizing the submodular function $F - s$. For a basic partition, a more efficient algorithm is available.

By repeated application of submodularity, we have for all sets $C \subseteq V$, if $C_i = C \cap A_i$:

$$
\begin{aligned}
F(C) - s(C) &= F(V \cap C) - \sum_{i=1}^m s(C_i) \text{ (as } s \text{ is a modular function)}, \\
&= F(B_m \cap C) - \sum_{i=1}^m s(C_i) + \sum_{i=1}^{m-1} F(B_i \cap C) - F(B_i \cap C) \text{ (as } B_m = V), \\
&= \sum_{i=1}^m F(B_i \cap C) - F(B_{i-1} \cap C) - s(C_i) \text{ (let } B_0 = \varnothing \text{ and as } F(\varnothing) = 0), \\
&= \sum_{i=1}^m F((B_{i-1} \cup A_i) \cap C) - F(B_{i-1} \cap C) - s(C_i) \text{ (since } B_i = B_{i-1} \cup A_i), \\
&= \sum_{i=1}^m F((B_{i-1} \cap C) \cup (A_i \cap C)) - F(B_{i-1} \cap C) - s(C_i), \\
&= \sum_{i=1}^m F((B_{i-1} \cap C) \cup C_i) - F(B_{i-1} \cap C) - s(C_i), \\
&\geqslant \sum_{i=1}^m \big[F(B_{i-1} \cup C_i) - F(B_{i-1}) - s(C_i)\big] \\
&\qquad \text{(as } (B_{i-1} \cap C) \subseteq B_{i-1} \text{ and due to submodularity of } F).
\end{aligned}
$$

Moreover, we have $s(A_i) = F(B_i) - F(B_{i-1})$, which implies $s(B_i) = F(B_i)$ for all $i \in \{1, \dots, m\}$, all subproblems $\min_{C_i \subseteq A_i} F(B_{i-1} \cup C_i) - F(B_{i-1}) - s(C_i)$ have non-positive

values. This implies that we may check optimality by solving these m subproblems: s is optimal if and only if all of them have zero values. This leads to smaller subproblems whose overall complexity is less than a single SFM oracle calls. Moreover, for cut functions, it may be solved by a single oracle call on a graph where some edges have been removed [Tarjan et al., 2006].

Given all sets C_i, we may then define a new ordered partition by splitting all A_i for which $F(B_{i-1} \cup C_i) - F(B_{i-1}) - s(C_i) < 0$. If no split is possible, the pair (w, s) is optimal for Eq. (5.1). Otherwise, this new strictly finer partition may not be basic, the value of the optimisation problem in Eq. (5.6) is strictly lower as shown in Section 5.2.5 (and leads to another basic ordered partition), which ensures finite convergence of the algorithm.

5.2.4 Active-set algorithm

This leads to the active-set algorithm below.

— **Input**: Submodular function F with SFM oracle, $u \in \mathbb{R}^n$, ordered partition $\mathcal{A}$.

— **Algorithm**: iterate until convergence

(a) Solve Eq. (5.6) by isotonic regression.

(b) Merge the sets with equal values of v_i to define a new ordered partition $\mathcal{A}$. Define $w = \sum_{i=1}^m v_i 1_{A_i}$ and $s = u - w$.

(c) Check optimality by solving $\min_{C_i \subseteq A_i} F(B_{i-1} \cup C_i) - F(B_{i-1}) - s(C_i)$ for $i \in \{1, \dots, m\}$.

(d) If s not optimal, for all C_i which are different from $\varnothing$ and A_i, add the new set $B_{i-1} \cup C_i$ in the ordered partition $\mathcal{A}$.

— **Output**: $w \in \mathbb{R}^n$ and $s \in B(F)$.

Relationship with divide-and-conquer algorithm. When starting from the trivial ordered partition $\mathcal{A} = (V)$, then we exactly obtain a parallel version of the divide-and-conquer algorithm [Groenevelt, 1991], that is, the isotonic regression problem in (a) is always solved without using the constraints of monotonicity, i.e., there are no merges in (b), and thus in (c), it is not necessary to re-solve the problems where nothing has changed. This shows that the number of iterations is then less than n. The key added benefits in our formulation is the possibility of warm-starting, which can be very useful for building paths of solutions with different weights on the total variation. This is also useful for decomposable functions where many TV oracles are needed with close-by input. See experiments in Section 5.4.

5.2.5 Proof of convergence

In order to prove convergence of the algorithm, we only need to show that if the optimality check fails in step (c), then step (d) introduces splits in the partition, which ensures that the isotonic regression in step (a) of the next iteration has a strictly lower value. Let us recall the isotonic regression problem solved in step (a):

$$\min_{v \in \mathbb{R}^m} \sum_{i=1}^m \left(v_i \big[F(B_i) - \quad F(B_{i-1}) - u(A_i) \big] + \tfrac{1}{2} |A_i| v_i^2 \right) \tag{5.7}$$

$$\text{such that } v_1 \geqslant \cdots \geqslant v_m. \tag{5.8}$$

Steps (a-b) ensure that the ordered partition $\mathcal{A}$ is a basic ordered partition warranting that the inequality constraints are strict, i.e., no two partitions have the same value v_i and the values v_i for each element of the partition $i = \{1, \ldots, m\}$ is given through

$$v_i|A_i| = u(A_i) - (F(B_i) - F(B_{i-1})), \tag{5.9}$$

which can be calculated in closed form.

The optimality check in step (c) decouples into checking the optimality in each subproblem as shown in Section 5.2.3. If the optimality test fails, then there is a subset of C_i of A_i for some of elements of the partition $\mathcal{A}$ such that $F(B_{i-1} \cup C_i) - F(B_{i-1}) - s(C_i) < 0$. We will show that the splits introduced by step (d) strictly reduces the function value of isotonic regression in Eq. (5.7), while maintaining the feasibility of the problem. The splits modify the cost function of the isotonic regression as follows

$$\begin{aligned}
\text{Eq. (5.7)} \;=\; & \sum_{i=1}^{m} \Big(v_i \big[F(B_{i-1} \cup C_i) - F(B_{i-1}) - u(C_i) \big] \\
& \quad + v_i \big[F(B_i) - F(B_{i-1} \cup C_i) - u(A_i \setminus C_i) \big] \\
& \quad + \tfrac{1}{2} v_i^2 |C_i| + \tfrac{1}{2} v_i^2 |A_i \setminus C_i| \Big).
\end{aligned} \tag{5.10}$$

Let us assume a positive t, which is small enough. The direction that the isotonic regression moves is $v_i + t$ for the partition corresponding to C_i and $v_i - t$ for the partition corresponding to $A_i \setminus C_i$ maintaining the feasibility of the isotonic regression problem, i.e., $v_1 \geqslant \ldots \geqslant v_i + t > v_i - t \geqslant \ldots \geqslant v_m$. The function value is given by

$$\begin{aligned}
& \sum_{i=1}^{m} \Big((v_i + t) \big[F(B_{i-1} \cup C_i) - F(B_{i-1}) - u(C_i) \big] \\
& \quad + (v_i - t) \big[F(B_i) - F(B_{i-1} \cup C_i) - u(A_i \setminus C_i) \big] \\
& \quad + \tfrac{1}{2} (v_i + t)^2 |C_i| + \tfrac{1}{2} (v_i - t)^2 |A_i \setminus C_i| \Big) \\
= \; & \sum_{i=1}^{m} \Big(\big(v_i \big[F(B_{i-1} \cup C_i) - F(B_{i-1}) - u(C_i) \big] \\
& \quad + v_i \big[F(B_i) - F(B_{i-1} \cup C_i) - u(A_i \setminus C_i) \big] \\
& \quad + \tfrac{1}{2} v_i^2 |C_i| + \tfrac{1}{2} v_i^2 |A_i \setminus C_i| \big) \\
& \quad + t \, \big(2F(B_{i-1} \cup C_i) - F(B_{i-1}) - F(B_i) \\
& \quad - u(C_i) + u(A_i \setminus C_i) + v_i |C_i| - v_i |A_i \setminus C_i| \big) \\
& \quad + \tfrac{1}{2} t^2 |A_i| \Big).
\end{aligned}$$

From this we can compute the directional derivative of the function, which is given by

$$\begin{aligned}
& 2F(B_{i-1}\cup C_i) - F(B_{i-1}) - F(B_i) \\
& \quad -u(C_i) + u(A_i\setminus C_i) + |C_i|v_i - |A_i\setminus C_i|v_i \\
= \;& 2F(B_{i-1}\cup C_i) - F(B_{i-1}) - F(B_i) \\
& \quad -2u(C_i) + u(A_i) + 2|C_i|v_i - |A_i|v_i \\
= \;& 2\big(F(B_{i-1}\cup C_i) - F(B_{i-1}) - u(C_i) + v_i|C_i|\big) \\
& \quad \text{(substituting Eq. (5.9))} \\
= \;& 2\big(F(B_{i-1}\cup C_i) - F(B_{i-1}) - s(C_i)\big) < 0 \\
& \quad \text{(as } s = u - w \text{ and Eq. (5.2.5)).}
\end{aligned}$$

This shows that the function strictly decreases with the splits introduced in step (d).

Certificates of optimality. The algorithm has dual-infeasible iterates s (they only belong to $B(F)$ at convergence). However, after step (c), we have that for all $C\subset V$, $F(C) - s(C) \geqslant -\varepsilon$. This implies that $s\in B(F + \varepsilon 1_{\mathrm{Card}\in(1,n)})$, i.e., $s\in B(F_\varepsilon)$ with $F_\varepsilon = F + \varepsilon 1_{\mathrm{Card}\in(1,n)}$. Since by construction $w = u - s$, we have:

$$\begin{aligned}
f_\varepsilon(w) - u^\top w + \tfrac{1}{2}\|w\|_2^2 + \tfrac{1}{2}\|s-u\|_2^2 &= \varepsilon\big|\max_{j\in V} w_j - \min_{j\in V} w_j\big| + f(w) - u^\top w + \|w\|^2 \\
&= \varepsilon\big|\max_{j\in V} w_j - \min_{j\in V} w_j\big| \\
&+ \sum_{i=1}^m v_i\big[F(B_i) - F(B_{i-1}) - u(A_i)\big] + \sum_{i=1}^m |A_i|v_i^2 \\
&= \varepsilon\big|\max_{j\in V} w_j - \min_{j\in V} w_j\big| \text{ (using Eq. (5.9))} \\
&= \varepsilon\,\mathrm{range}(w),
\end{aligned}$$

where $\mathrm{range}(w) = \max_{k\in V} w_k - \min_{k\in V} w_k$. This means that w is approximately optimal for $f(w) - u^\top w + \frac{1}{2}\|w\|_2^2$ with *certified gap* less than $\varepsilon\,\mathrm{range}(w) + \varepsilon\,\mathrm{range}(w^*)$.

Maximal range of an active-set solution. For any ordered partition $\mathcal{A}$, and the optimal value of w (which we know in closed form), we have $\mathrm{range}(w) \leqslant \mathrm{range}(u) + \max_{i\in V}\big\{F(\{i\}) + F(V\backslash\{i\}) - F(V)\big\}$. Indeed, for the u part of the expression, this is because values of w are averages of values of u; for the F part of the expression, we always have by submodularity:

$$\begin{aligned}
F(B_i) - F(B_{i-1}) &\leqslant \sum_{k\in A_i} F(\{k\}) \\
&\text{and} \\
F(B_i) - F(B_{i-1}) &\geqslant -\sum_{k\in A_i} F(V) - F(V\backslash\{k\}).
\end{aligned}$$

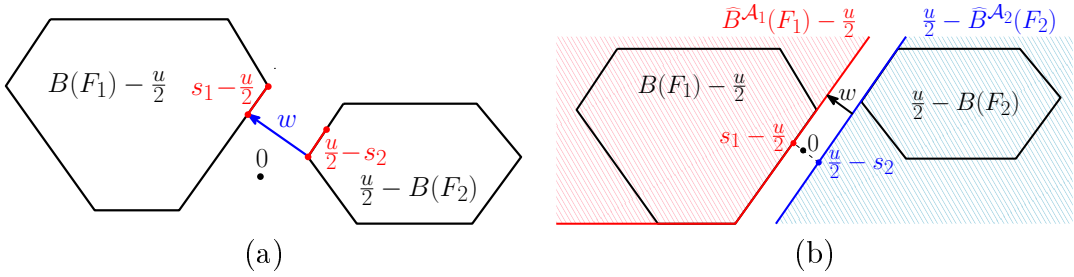


Figure 5-3 – Closest point between two polytopes. (a) Output of Dykstra's alternating projection algorithm for the TV problem, the pair (s_1, s_2) may not be unique while $w = s_1 + s_2 - u$ is. (b) Dykstra's alternating projection output for outer approximations.

Exact solution. If the submodular function only takes integer values and we have an approximate solution of the TV problem with gap $\varepsilon \leqslant \frac{1}{4n}$, then we have the optimal solution [Chakrabarty et al., 2014].

Relationship with traditional active-set algorithm. Given an ordered partition $\mathcal{A}$, an active-set method solves the unconstrained optimisation problem in Eq. (5.6) to obtain a value of v using the primary stationary conditions. The corresponding primal value $w = \sum_{i=1}^{m} v_i 1_{A_i}$ and dual value $s = u - w$ are optimal, if and only if,

$$\text{Primal feasibility : } \quad w \in \mathcal{W}^{\mathcal{A}}, \tag{5.11}$$

$$\text{Dual feasibility : } \quad s \in B(F). \tag{5.12}$$

If Eq. (5.11) is not satisfied, a move towards the optimal w is performed to ensure primal feasibility by performing line search, i.e., two consecutive sets A_i and A_{i+1} with increasing values of v, i.e., $v_i < v_{i+1}$ are merged and a potential w is computed until primal feasibility is met. Then dual feasibility is checked and potential splits are proposed.

In our approach, we consider a different strategy which is more direct and does many merges simultaneously by using *isotonic regression*. Our method explicitly moves from ordered partitions to ordered partitions and computes optimal vector w, which is feasible.

5.3 Decomposable Problems

Many interesting problems in signal processing and computer vision naturally involve submodular functions F that decompose into $F = F_1 + \cdots + F_r$, with r "simple" submodular functions. For example, a cut function in a 2D grid decomposes into a function F_1 composed of cuts along vertical lines and a function F_2 composed

of cuts along horizontal lines. For both of these functions, SFM oracles may be solved in $O(n)$ by message passing. For simplicity, we consider the case $r = 2$ functions, but following [Komodakis et al., 2011, Jegelka et al., 2013], our framework easily extends to $r > 2$.

5.3.1 Reformulation as the distance between two polytopes

Following our derivations in Chapter 1, we have the primal/dual problems :

$$\begin{aligned} & \min_{w \in \mathbb{R}^n} f_1(w) + f_2(w) - u^\top w + \tfrac{1}{2}\|w\|_2^2 \\ = & \min_{w \in \mathbb{R}^n} \max_{s_1 \in B(F_1),\ s_2 \in B(F_2)} w^\top (s_1 + s_2) - u^\top w + \tfrac{1}{2}\|w\|_2^2 \\ = & \max_{s_1 \in B(F_1),\ s_2 \in B(F_2)} \min_{w \in \mathbb{R}^n} (s_1 + s_2 - u)^\top w + \tfrac{1}{2}\|w\|_2^2 \\ = & \max_{s_1 \in B(F_1),\ s_2 \in B(F_2)} -\tfrac{1}{2}\|s_1 + s_2 - u\|_2^2, \end{aligned} \tag{5.13}$$

with $w = u - s_1 - s_2$ at optimality.

This is the projection of u on the sum of the base polytopes $B(F_1) + B(F_2) = B(F)$. Further, this may be interpreted as finding the distance between two polytopes $B(F_1) - u/2$ and $u/2 - B(F_2)$. Note that these two polytopes typically do not intersect (they will if and only if $w = 0$ is the optimal solution of the TV problem, which is an uninteresting situation).

Alternating projections (AP). The updates of each iteration of alternating projections described in Chapter 1 leads to the following updates for our problem.

$$z_t = \Pi_{u/2 - B(F_2)} \Pi_{B(F_1) - u/2}(z_{t-1}),$$

where z_0 is an arbitrary starting point. Thus each of these steps require TV oracle for F_1 and F_2 since projection onto the base polytope is equivalent to performing TV denoising as shown in Eq. (1.13).

Averaged alternating reflections (AAR). The updates of the each iteration of the averaged alternating reflections, also described in Chapter 1, which starts with an auxiliary sequence z_0 initialised to 0 vector, are given by

$$z_t = \tfrac{1}{2}(I + R_{u/2 - B(F_2)} R_{B(F_1) - u/2})(z_{t-1}).$$

In the feasible case, i.e., intersecting polytopes, the sequence z weakly converges to a point in the intersection of the polytopes on using both the algorithms. However, in our case, we have non intersecting polytopes which leads to a converging sequence of z with AP but a diverging sequence of z with AAR. However, when we project z by using the following projection operation, i.e, $t_{1,t} = \Pi_{B(F_1) - u/2}(z_t); t_{2,t} = \Pi_{u/2 - B(F_2)}(s_{1,t})$ the

sequences t_1 and t_2 converge to nearest points on the polytopes, $B(F_1) - u/2$ and $u/2 - B(F_2)$ [Bauschke and Luke, 2004].

Dykstra's alternating projection algorithm [Bauschke and Borwein, 1994] retrieves a convex feasible point closest to an arbitrary point, which we assume to be 0. It can also be used and has a form of primal descent interpretation, i.e., as coordinate descent for a well-formulated primal problem [Gaffke and Mathar, 1989]. Let us denote ι_K as the indicator function of a convex set K. In our case we consider finding the nearest points on the polytopes $B(F_1) - u/2$ and $u/2 - B(F_2)$ closest to 0, which can be formally written as:

$$\begin{aligned}
\min_{\substack{s \in B(F_1) - \frac{u}{2} \\ s \in \frac{u}{2} - B(F_2)}} \tfrac{1}{2}\|s\|_2^2 &= \min_{s \in \mathbb{R}^n} \frac{1}{2}\|s\|_2^2 + \iota_{B(F_1) - \frac{u}{2}}(s) + \iota_{\frac{u}{2} - B(F_2)}(s) \\
&= \min_{s \in \mathbb{R}^n} \frac{1}{2}\|s\|_2^2 + \iota_{B(F_1) - \frac{u}{2}}(s) + \iota_{B(F_2) - \frac{u}{2}}(-s) \\
&= \min_{s \in \mathbb{R}^n} \frac{1}{2}\|s\|_2^2 + \max_{w_1 \in \mathbb{R}^n} w_1^\top s - f_1(w_1) + \frac{w_1^\top u}{2} \\
&\qquad + \max_{w_2 \in \mathbb{R}^n} -w_2^\top s - f_2(w_2) + \frac{w_2^\top u}{2} \\
&= \max_{\substack{w_1 \in \mathbb{R}^n \\ w_2 \in \mathbb{R}^n}} -f_1(w_1) - f_2(w_2) + \frac{(w_1 + w_2)^\top u}{2} \\
&\qquad + \min_{s \in \mathbb{R}^n} \frac{1}{2}\|s\|_2^2 + (w_1 - w_2)^\top s \\
&= \max_{\substack{w_1 \in \mathbb{R}^n \\ w_2 \in \mathbb{R}^n}} -f_1(w_1) - f_2(w_2) + \frac{(w_1 + w_2)^\top u}{2} \\
&\qquad - \frac{1}{2}\|w_1 - w_2\|_2^2 \\
&= \min_{\substack{w_1 \in \mathbb{R}^n \\ w_2 \in \mathbb{R}^n}} f_1(w_1) + f_2(w_2) - \frac{(w_1 + w_2)^\top u}{2} \\
&\qquad + \frac{1}{2}\|w_1 - w_2\|_2^2,
\end{aligned}$$

where $s = w_2 - w_1$ at optimal. The block coordinate descent gives

$$\begin{aligned}
s_{1,t} &= \Pi_{B(F_1) - u/2}(w_{2,t-1}), \\
w_{1,t} &= w_{2,t-1} - s_{1,t}, \\
s_{2,t} &= \Pi_{B(F_2) - u/2}(w_{1,t}), \\
w_{2,t} &= w_{1,t} - s_{2,t},
\end{aligned}$$

with s_1 and s_2 converging to the nearest points on the polytopes closest to the origin.

We have implemented it, which behaves similar to alternating projections, but it still requires TV oracles for projection (see experiments in Section 5.4). There is however a key difference: while alternating projections and alternating reflections always converge to a pair of closest points, Dykstra's alternating projection algorithm converges to a *specific* pair of points, namely the pair closest to the initialisation of the algorithm [Bauschke and Borwein, 1994]; see an illustration in Figure 5-3-(a). This insight will be key in our algorithm to avoid cycling.

Assuming TV oracles are available for F_1 and F_2, we have shown in the Chapter 4 the use of alternating projection [Bauschke et al., 1997] and alternating reflection [Bauschke and Luke, 2004] algorithms. However, these algorithms are equivalent to block *dual* coordinate descent and cannot be be cast explicitly as descent algorithms for the primal TV problem. On the other hand, Dykstra's alternating projection is a descent algorithm on the primal, which enables local search over partitions. Complex TV oracles are often implemented by using SFM oracles recursively with the divide and conquer strategy on the individual functions. Using our algorithm in Section 5.2.4, they can be made more efficient using warms tarts. (see experiments in Section 5.4).

5.3.2 First attempt at an active-set method

Given our algorithm for a single function, it is natural to perform a local search over two partitions $\mathcal{A}_1$ and $\mathcal{A}_2$, one for each function F_1 and F_2, and consider in the primal formulation a weight vector w compatible with both $\mathcal{A}_1$ and $\mathcal{A}_2$; or, equivalently, in the dual formulation, two outer approximations $\widehat{B}^{\mathcal{A}_1}(F_1)$ and $\widehat{B}^{\mathcal{A}_2}(F_2)$. That is, given the ordered partitions $\mathcal{A}_1$ and $\mathcal{A}_2$, using a similar derivation as in Eq. (5.13), we obtain the primal/dual pairs of optimisation problems

$$\max_{\substack{s_1 \in \widehat{B}^{\mathcal{A}_1}(F_1) \\ s_2 \in \widehat{B}^{\mathcal{A}_2}(F_2)}} -\tfrac{1}{2}\|u - s_1 - s_2\|_2^2 = \min_{\substack{w \in \mathcal{W}^{\mathcal{A}_1} \\ w \in \mathcal{W}^{\mathcal{A}_2}}} f_1(w) + f_2(w) - u^\top w + \tfrac{1}{2}\|w\|_2^2,$$

with $w = u - s_1 - s_2$ at optimality.

Primal solution by isotonic regression. The primal solution w is unique by strong convexity. Moreover, it has to be compatible with both $\mathcal{A}_1$ and $\mathcal{A}_2$, which is equivalent to being compatible with the *coalesced* ordered partition $\mathcal{A} = \text{coalesce}(\mathcal{A}_1, \mathcal{A}_2)$ defined as the coarsest ordered partition compatible by both. As shown in Appendix 5.A, $\mathcal{A}$ may be found in time $O(\min(m_1, m_2)n)$.

Given $\mathcal{A}$, the primal solution w of the subproblem may be found by isotonic regression like in Section 5.2.2 in time $O(m)$ where m is the number of sets in $\mathcal{A}$. However, finding the optimal dual variables s_1 and s_2 turns out to be more problematic. We know that $s_1 + s_2 = u - w$ and that $s_1 + s_2 \in \widehat{B}^{\mathcal{A}}(F)$, but the split in (s_1, s_2) is unknown.

Obtaining dual solutions. Given an ordered partition $\mathcal{A}$, a unique well-defined pair (s_1, s_2) can be obtained by solving a set of linear systems. We propose such

an algorithm in Section 5.3.4 using the primal active-set method. However, due to high complexity of these methods, we could use any convex feasibility algorithm such as alternating projections [Bauschke et al., 1997] or alternating reflections [Bauschke and Luke, 2004]. However, the result would depend in non understood ways on the initialisation, and we have observed cycling of the active-set algorithm. Using Dykstra's alternating projection algorithm allows us to converge to a unique well-defined pair (s_1, s_2) that will lead to a provably non-cycling algorithm.

When running our algorithms starting from 0 on the polytopes $\widehat{B}^{\mathcal{A}_1}(F_1) - u/2$ and $u/2 - \widehat{B}^{\mathcal{A}_2}(F_2)$, if w is the unique distance vector between the two polytopes, then the iterates converge to the projection of 0 onto the convex sets of elements in the two polytopes that achieve the minimum distance [Bauschke and Borwein, 1994]. See Figure 5-3-(b) for an illustration. This algorithm is however slow to converge when the polytopes do not intersect (they will notice here for the most interesting situations when $w \neq 0$) and convergence is hard to monitor because primal iterates diverge [Bauschke and Borwein, 1994].

Translated intersecting polytopes. In our situation, we want to reach the solution *while knowing the vector* w (as mentioned earlier, it is obtained cheaply from isotonic regression). Indeed, from Lemma 2.2 and Theorem 3.8 from [Bauschke and Borwein, 1994], given this vector w, we may translate the two polytopes and now obtain a formulation where the two polytopes do intersect; that is we aim at projecting 0 on the (non-empty) intersection of $\widehat{B}^{\mathcal{A}_1}(F_1) - u/2 + w/2$ and $u/2 - w/2 - \widehat{B}^{\mathcal{A}_2}(F_2)$. See Figure 5-4. We also refer to this as *translated Dykstra problem*[2] in the rest of the chapter. This is equivalent to solving the following optimisation problem

$$
\begin{aligned}
\min_{\substack{s \in \widehat{B}^{\mathcal{A}_1}(F_1) - \frac{u-w}{2} \\ s \in \frac{u-w}{2} - \widehat{B}^{\mathcal{A}_2}(F_2)}} \tfrac{1}{2}\|s\|_2^2 &= \min_{s \in \mathbb{R}^n} \tfrac{1}{2}\|s\|_2^2 + \iota_{\widehat{B}^{\mathcal{A}_1}(F_1) - \frac{u-w}{2}}(s) + \iota_{\frac{u-w}{2} - \widehat{B}^{\mathcal{A}_2}(F_2)}(s), \\
&= \min_{s \in \mathbb{R}^n} \tfrac{1}{2}\|s\|_2^2 + \iota_{\widehat{B}^{\mathcal{A}_1}(F_1) - \frac{u-w}{2}}(s) + \iota_{\widehat{B}^{\mathcal{A}_2}(F_2) - \frac{u-w}{2}}(-s), \\
&= \min_{s \in \mathbb{R}^n} \Bigg(\tfrac{1}{2}\|s\|_2^2 + \max\nolimits_{w_1 \in \mathcal{W}^{\mathcal{A}_1}} w_1^\top s - f_1(w_1) + \tfrac{w_1^\top (u-w)}{2} \\
&\qquad + \max_{w_2 \in \mathcal{W}^{\mathcal{A}_2}} -w_2^\top s - f_2(w_2) + \frac{w_2^\top (u-w)}{2} \Bigg), \\
&= \max_{\substack{w_1 \in \mathcal{W}^{\mathcal{A}_1} \\ w_2 \in \mathcal{W}^{\mathcal{A}_2}}} \Bigg(- f_1(w_1) - f_2(w_2) + \frac{(w_1 + w_2)^\top (u-w)}{2} \\
&\qquad + \min_{s \in \mathbb{R}^n} \frac{1}{2}\|s\|_2^2 + (w_1 - w_2)^\top s \Bigg), \\
&= \max_{\substack{w_1 \in \mathcal{W}^{\mathcal{A}_1} \\ w_2 \in \mathcal{W}^{\mathcal{A}_2}}} \Bigg(- f_1(w_1) - f_2(w_2) + \frac{(w_1 + w_2)^\top (u-w)}{2}
\end{aligned}
$$

2. We refer to finding a Dykstra solution for translated intersecting polytopes as translated Dykstra problem

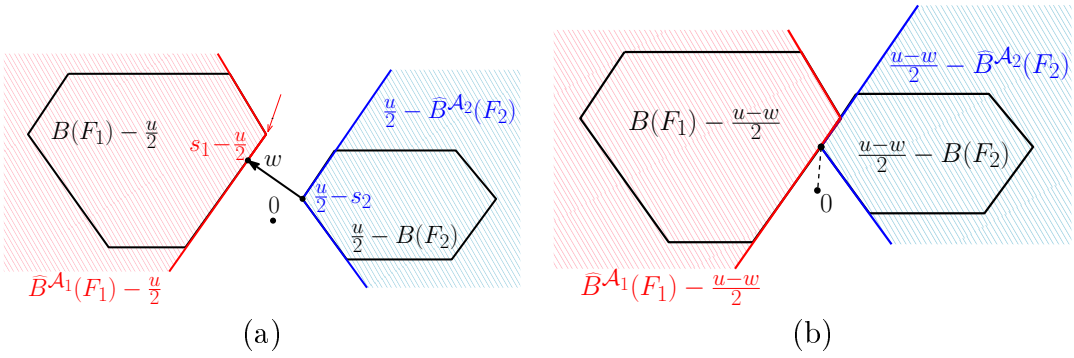


Figure 5-4 – Translated intersecting polytopes. (a) output of our algorithm before translation. (b) Translated formulation.

$$
\begin{aligned}
& & & -\frac{1}{2}\|w_1 - w_2\|_2^2\Big), \\
&= \min_{\substack{w_1 \in \mathcal{W}^{\mathcal{A}_1} \\ w_2 \in \mathcal{W}^{\mathcal{A}_2}}} & \Big(f_1(w_1) + f_2(w_2) - \frac{(w_1 + w_2)^\top (u - w)}{2} \\
& & & +\frac{1}{2}\|w_1 - w_2\|_2^2\Big), \qquad (5.14)
\end{aligned}
$$

with $s = w_2 - w_1$ at optimality.

In Section 5.3.5 we propose primal active-set algorithm and accelerated Dykstra's algorithm in Section 5.3.6 to solve the above optimisation problems. Assuming that we are able to solve this step efficiently, we now present our active-set algorithm for decomposable functions below.

5.3.3 Active-set algorithm for decomposable functions

— **Input**: Submodular function F_1 and F_2 with SFM oracles, $u \in \mathbb{R}^n$, ordered partitions $\mathcal{A}_1, \mathcal{A}_2$

— **Algorithm**: iterate until convergence (i.e., $\varepsilon_1 + \varepsilon_2$ small enough)

(a) Find $\mathcal{A} = \text{coalesce}(\mathcal{A}_1, \mathcal{A}_2)$ and run isotonic regression to minimise $f(w) - u^\top w + \frac{1}{2}\|w\|_2^2$ such that w is compatible with $\mathcal{A}$.

(b) Find the projection of 0 onto the intersection of $\widehat{B}^{\mathcal{A}_1}(F_1) - u/2 + w/2$ and $u/2 - w/2 - \widehat{B}^{\mathcal{A}_2}(F_2)$ using any of the algorithms described in Section 5.3.4.

(c) Merge the sets in $\mathcal{A}_j$ which are tight for s_j, $j \in \{1, 2\}$.

(d) Check optimality by solving $\min_{C_{j,i_j} \subseteq A_{j,i_j}} F_j(B_{j,i_j-1} \cup C_{j,i_j}) - F_j(B_{j,i+j-1}) - s_j(C_{j,i_j})$ for $i_j \in \{1, \dots, m_j\}$, Monitor ε_1 and ε_2 such that $F_j(C_j) - s_j(C_j) \geqslant -\varepsilon_j$, $j = 1, 2$.

(e) If both s_1 and s_2 not optimal, for all C_{j,i_j} which are different from $\varnothing$ and A_{j,i_j}, split partitions.

— **Output**: $w \in \mathbb{R}^n$ and $s_1 \in B(F_1)$, $s_2 \in B(F_2)$.

Given two ordered partitions $\mathcal{A}_1$ and $\mathcal{A}_2$, we obtain $s_1 \in \hat{B}^{\mathcal{A}_1}(F_1)$ and $s_2 \in \hat{B}^{\mathcal{A}_2}(F_2)$ as described in the following section. The solution $w = u - s_1 - s_2$ is optimal if and only if both $s_1 \in B(F_1)$ and $s_2 \in B(F_2)$. When checking the optimality described in Section 5.2.3, we split the partition. As shown in Appendix 5.B, either (a) $\|w\|_2^2$ strictly increases at each iteration, or (b) $\|w\|_2^2$ remains constant but $\|s_1 - s_2\|_2^2$ strictly increases. This implies that the algorithm is finitely convergent.

5.3.4 Optimizing the "translated Dykstra problem"

In this section, we describe algorithms to optimise the translated Dykstra problem in Eq. (5.14), i.e.,

$$\min_{\substack{w_1 \in \mathcal{W}^{\mathcal{A}_1} \\ w_2 \in \mathcal{W}^{\mathcal{A}_2}}} f_1(w_1) + f_2(w_2) - \frac{(w_1+w_2)^\top (u-w)}{2} + \frac{1}{2}\|w_1 - w_2\|^2. \tag{5.15}$$

The corresponding dual optimisation problem is given by

$$\min_{\substack{s \in \widehat{B}^{\mathcal{A}_1}(F_1) - \frac{u-w}{2} \\ s \in \frac{u-w}{2} - \widehat{B}^{\mathcal{A}_2}(F_2)}} \frac{1}{2}\|s\|_2^2 \tag{5.16}$$

with the optimality condition $s = w_2 - w_1$. Note that the only link to submodularity is that f_1 and f_2 are linear functions on $\mathcal{W}^{\mathcal{A}_1}$ and $\mathcal{W}^{\mathcal{A}_2}$, respectively. The rest of this section primarily deals with optimizing a quadratic program, which we show in the next section.

5.3.5 Primal active-set method

In this section, we find the projection of the origin onto the intersection of the translated base polytopes given by Eq. (5.14) using active-set methods by solving a set of linear equations. For this purpose, we derive the equivalent optimisation problems using equality constraints.

The ordered partition, $\mathcal{A}_j$ is given by $(A_{j,1}, \ldots, A_{j,m_j})$, where m_j is the number of elements in the ordered partitions. Let B_{j,i_j} be defined as $(A_{j,1} \cup \ldots \cup A_{j,i_j})$. Therefore,

$$f_j(w_j) = \sum_{i_j=1}^{m_j} v_{j,i_j} \Big(F_j(B_{j,i_j}) - F_j(B_{j,i_j-1}) \Big) \tag{5.17}$$

$$w_j = \sum_{i_j=1}^{m_j} v_{j,i_j} 1_{A_{j,i_j}} \tag{5.18}$$

$$\text{with the constraints, } v_{j,1} \geq \ldots \geq v_{j,m_j}. \tag{5.19}$$

On substituting Eq. (5.17), Eq. (5.18) and Eq. (5.19) in Eq. (5.14), we have and equivalent optimisation problem:

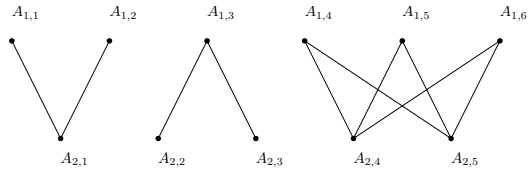


Figure 5-5 – Bipartite graph to estimate $Q(\mathcal{A}_1, \mathcal{A}_2)$ with $\mathcal{A}_1$ having $m_1 = 6$ components and $\mathcal{A}_2$ having $m_2 = 5$.

$$\begin{aligned}\min_{\substack{v_{1,1}\geq\ldots\geq v_{1,m_1}\\ v_{2,1}\geq\ldots\geq v_{2,m_2}}} \quad & \sum_{i_1=1}^{m_1} \left(F_1(B_{1,i_1}) - F_1(B_{1,i_1-1}) - \frac{u(A_{1,i_1}) - w(A_{1,i_1})}{2}\right) v_{1,i_1} + \\ & \sum_{i_2=1}^{m_2} \left(F_2(B_{2,i_2}) - F_2(B_{2,i_2-1}) - \frac{u(A_{2,i_2}) - w(A_{2,i_2})}{2}\right) v_{2,i_2} + \\ & \sum_{i_1=1}^{m_1} \frac{1}{2}|A_{1,i_1}|v_{1,i_1}^2 + \sum_{i_2=1}^{m_2} \frac{1}{2}|A_{2,i_2}|v_{2,i_2}^2 - \sum_{i_1=1}^{m_1}\sum_{i_2=1}^{m_2} v_{1,i_1}v_{2,i_2}1_{A_{1,i_1}}^\top 1_{A_{2,i_2}}.\end{aligned}$$

This can be written as a quadratic program in $x = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}$ with inequality constraints in the following form

$$\min_{\substack{x\in\mathbb{R}^{m_1+m_2}\\ D(\mathcal{A}_1,\mathcal{A}_2)x\geqslant 0}} \frac{1}{2}x^\top Q(\mathcal{A}_1, \mathcal{A}_2)x + c(\mathcal{A}_1, \mathcal{A}_2)^\top x, \tag{5.20}$$

where $D(\mathcal{A}_1, \mathcal{A}_2)$ is a sparse matrix of size $(m_1 + m_2 - 2) \times (m_1 + m_2)$, which is a block diagonal matrix containing the difference or first order derivative matrices of sizes $m_1 - 1 \times m_1$ and $m_2 - 1 \times m_2$ as the blocks and $c(\mathcal{A}_1, \mathcal{A}_2)$ is a linear vector that can be estimated using the functions evaluations of F_1 and F_2. Note that these evaluations need to be done only once.

Estimating $Q(\mathcal{A}_1, \mathcal{A}_2)$. Let us consider a bipartite graph, $G = (\mathcal{A}_1, \mathcal{A}_2, E)$, with $m_1 + m_2$ nodes representing the ordered partitions of $\mathcal{A}_1$ and $\mathcal{A}_2$ respectively. The weight of the edge between each element of ordered partitions of $\mathcal{A}_1$, represented by A_{1,i_1} and each element of ordered partitions of $\mathcal{A}_2$, represented by A_{2,i_2} is the number of elements of the ground set V that lie in both these partitions and can be written as $e(A_{1,i_1}, A_{2,i_2}) = 1_{A_{1,i_1}}^\top 1_{A_{2,i_2}}$ for all $e \in E$. The matrix $Q(\mathcal{A}_1, \mathcal{A}_2)$ represents the Laplacian matrix of the graph G. Figure 5-5 shows a sample bipartite graph with $m_1 = 6$ and $m_2 = 5$.

Optimizing the quadratic program in Eq. (5.20) by using active-set methods is equivalent to finding the face of the constraint set on which the optimal solution lies. For this purpose, we need to be able to solve the quadratic program in Eq. (5.20) with equality constraints.

Equality constraint QP. Let us now consider the following quadratic program with equality constraints

$$\min_{\substack{p\in\mathbb{R}^{m_1+m_2}\\ D'p=0}} \frac{1}{2}p^\top Q(\mathcal{A}_1,\mathcal{A}_2)p + \big(Q(\mathcal{A}_1,\mathcal{A}_2)x_k + c(\mathcal{A}_1,\mathcal{A}_2)\big)^\top p, \tag{5.21}$$

where D' is the subset of the constraints in $D(\mathcal{A}_1,\mathcal{A}_2)$, i.e. indices of its rows that are tight and x_k is a primal feasible point. The vector p gives the direction of strict descent of the cost function in Eq. (5.20) from feasible point x_k [Nocedal and Wright, 2006].

Without loss of generality, let us assume that the equality constraints are $v_{j,k_j} = v_{j,k_j+1}$ for any k_j in $[0, m_j)$. Let $\mathcal{A}'_j$ be the new ordered partition formed by merging A_{j,k_j} and A_{j,k_j+1} as $v_{j,k_j} = v_{j,k_j+1}$. Finding the optimal vector p using the quadratic program in Eq. (5.21) with respect to the ordered partition $\mathcal{A}'_j$ is equivalent to solving the following unconstrained quadratic problem,

$$\mathcal{Q}(\mathcal{A}'_1,\mathcal{A}'_2,x_t) = \min_{p'\in\mathbb{R}^{m'_1+m'_2}} \left(\frac{1}{2}p'^\top Q(\mathcal{A}'_1,\mathcal{A}'_2)p' + \big(Q(\mathcal{A}'_1,\mathcal{A}'_2)x_t + c(\mathcal{A}'_1,\mathcal{A}'_2)\big)^\top p'\right) \tag{5.22}$$

where m'_j is the number of elements of the ordered partition $\mathcal{A}'_j$. This can be estimated by solving a linear system using conjugate gradient descent. The complexity of each iteration of the conjugate gradient is given by $O((m'_1+m'_2)k)$ where k is the number of non-zero elements in the sparse matrix, $Q(\mathcal{A}'_1,\mathcal{A}'_2)$ [Vishnoi, 2013]. We can build p from p' by repeating the values for the elements of the partition that were merged.

Primal active-set algorithm. We now describe step (b) of the active-set algorithm to optimise decomposable functions using a primal active-set method. Note that we can warmstart this step by using w estimated from step (a) of the algorithm.

- **Input**: Laplacian matrix $Q(\mathcal{A}_1,\mathcal{A}_2)$ and vector $c(\mathcal{A}_1,\mathcal{A}_2)$, ordered partitions $\mathcal{A}_1,\mathcal{A}_2$ and vector w.
- **Algorithm**: Iterate on t until both primal and dual feasibility conditions in Eq. (5.15) and Eq. (5.16) respectively are satisfied.
- **Initialise**: x_0 using w, Working set WS_0 with tight sets of $\mathcal{A}_j$. Estimate $\mathcal{A}'_1,\mathcal{A}'_2$ from WS_0.
  (a) Solve $\mathcal{Q}(\mathcal{A}'_1,\mathcal{A}'_2,x_t)$ in Eq. (5.22) to find optimal p.
  (b) **Check Primal Feasibility**: If $p == 0$
    1. **Estimate Dual**: $\lambda = D(\mathcal{A}_1,\mathcal{A}_2)^{-\top}\big(Q(\mathcal{A}_1,\mathcal{A}_2)x_t + c(\mathcal{A}_1,\mathcal{A}_2)\big)$.
    2. **Check Dual Feasibility**:
       if $\lambda_i \geq 0$ for all in $i \in WS_t$
       — return Optimal $x^* = x_t$.
    3. else
       — **Most violated Constraint**:
       update $j = \arg\min_{j\in WS_t} \lambda_j$.

— **Update Working Set**:
update $WS_{t+1} = WS_t \setminus \{j\}$.
update $\mathcal{A}'_1$, $\mathcal{A}'_2$ from WS_{t+1}.
— update $x_{t+1} = x_t$
— Goto step (a).

4. endif

(c) else

1. **Line Search**: Find least α that retains feasibility of $x_{t+1} = x_t + \alpha p$ and find the blocking constraints B_t.
2. **Update Working Set**: $WS_{t+1} = WS_t \cup B_t$ and update $\mathcal{A}'_1$, $\mathcal{A}'_2$ from WS_{t+1}.
3. Goto step (a).

(d) endif

— **Output**: $x^* \in \mathbb{R}^{m_1+m_2}$.

We can estimate w_1 and w_2 from x^*, which will enable us to estimate s feasible in Eq. (5.16). Therefore we can estimate the dual variable $s_1 \in B^{\mathcal{A}_1}(F_1)$ and $s_2 \in B^{\mathcal{A}_2}(F_2)$ using s.

5.3.6 Accelerated Dykstra's algorithm

In this section, we find the projection of the origin onto the intersection of the translated base polytopes obtained by solving the optimisation problem in Eq. (5.14) given by

$$\min_{\substack{w_1 \in \mathcal{W}^{\mathcal{A}_1} \\ w_2 \in \mathcal{W}^{\mathcal{A}_2}}} f_1(w_1) + f_2(w_2) - \tfrac{(w_1+w_2)^\top (u-w)}{2} + \tfrac{1}{2}\|w_1 - w_2\|^2$$

using Dykstra's alternating projection. It can be solved using the following Dykstra's iterations.

$$\begin{aligned}
s_{1,t} &= \Pi_{\widehat{B}^{\mathcal{A}_1}(F_1)}(u/2 - w/2 + w_{2,t-1}), \\
w_{1,t} &= u/2 - w/2 + w_{2,t-1} - s_{1,t}, \\
s_{2,t} &= \Pi_{\widehat{B}^{\mathcal{A}_2}(F_2)}(u/2 - w/2 + w_{1,t}), \\
w_{2,t} &= u/2 - w/2 + w_{1,t} - s_{2,t},
\end{aligned}$$

with Π_C denoting the orthogonal projection onto the sets C, solved here by isotonic regression. Note that the value of the auxiliary variable w_2 can be warm-started. The algorithm converges linearly on two intersecting convex sets [Shusheng, 2000].

In our simulations, we have used the recent accelerated version of [Chambolle and Pock, 2015], which led to faster convergence. In order to monitor convergence, we compute the value of $\|u - w - s_{1,t} - s_{2,t}\|_1$ which is equal to zero at convergence. The optimisation problem can also be decoupled into smaller optimisation problems by using the knowledge of the face of the base polytopes on which s_1 and s_2 lie. See details in Appendix 5.D.

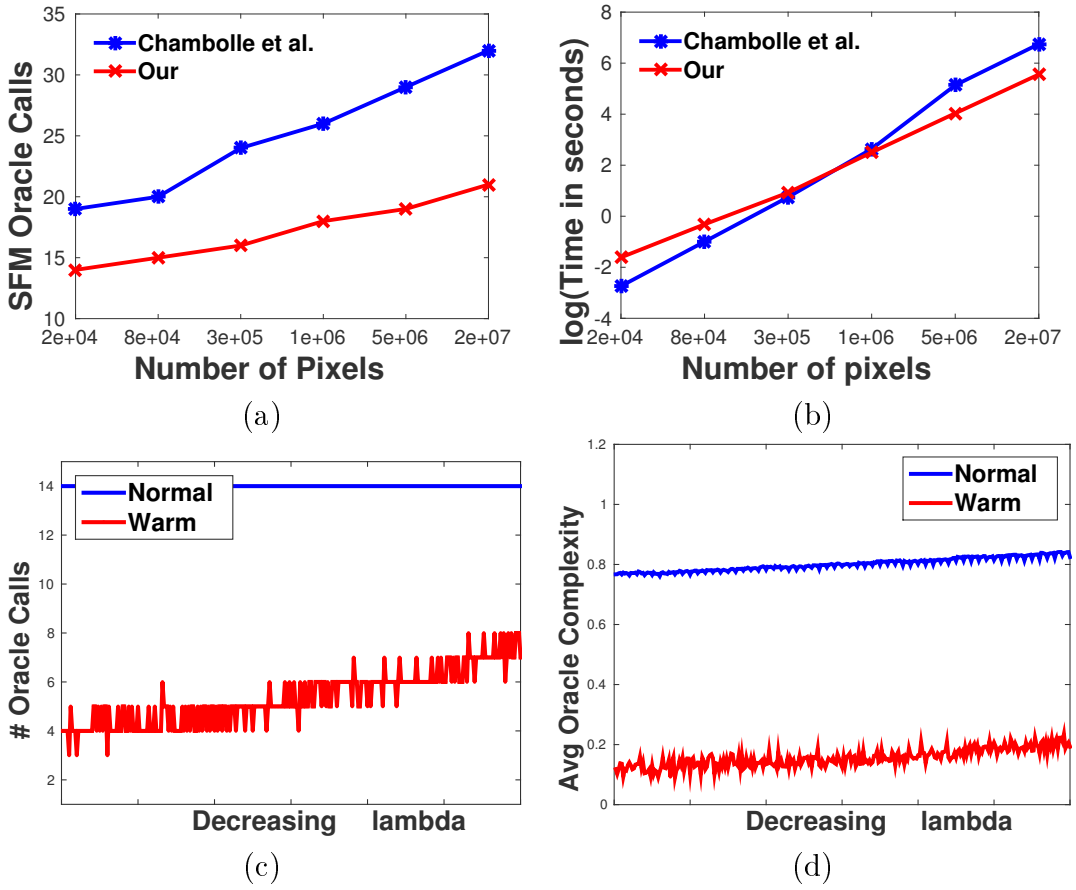


Figure 5-6 – (a) Number of SFM oracle calls for images of various sizes, (b) Time taken for images of various sizes, (c) Number of iterations with and without warm start, (d) Average complexity of the oracle with and without warm start.

5.3.7 Decoupled problem

In our context, the quadratic program Eq. (5.20) can be decoupled into smaller optimisation problems. Let us consider the bipartite graph $G = (\mathcal{A}_1, \mathcal{A}_2, E)$ of which Q is the Laplacian matrix. The number of connected components of the graph, G is equal to the number of levelsets of w.

Let m be the total number of connected components in G. These connected components define a partition on the ground set V and a total order on elements of the partition can be obtained using the levels sets of w. Let k denote the index of each bipartite subgraph of G represented by $G_k = (\mathcal{A}_{1,k}, \mathcal{A}_{2,k}, E_k)$, where $k = 1, 2, \ldots, m$. Let J_k denote the indices of the nodes of G_k in G.

$$x^*_{J_k} = \underset{\substack{x \in \mathbb{R}^{m_{1,k}+m_{2,k}} \\ D(\mathcal{A}_1,\mathcal{A}_2)_k x \succcurlyeq 0}}{\arg\min} \frac{1}{2} x^\top Q(\mathcal{A}_1, \mathcal{A}_2)_{J_k J_k} x + c(\mathcal{A}_1, \mathcal{A}_2)^\top_{J_k} x, \tag{5.23}$$

where $m_{j,k}$ is size of $\mathcal{A}_{j,k}$. Therefore, $m_{1,k} + m_{2,k}$ is the total number of nodes in the subgraph G_k. Note that this is exactly equivalent to decomposition of base polytope of F_j into base polytopes of submodular functions formed by contracting F_j on each individual components representing the connected component k. See Appendix 5.C for more details.

5.4 Experiments

In this section, we show the results of the algorithms proposed on various problems. We first consider the problem of solving total variation denoising for a non decomposable function using active-set methods in Section 5.4.1, specifically cut functions. In Section 5.4.2, we consider cut functions on a 3D grid decomposed into a function of 2D grid and a function of chains. We then consider 2D grid and a concave function on cardinality, which is not a cut function. We show these results using primal active-set method proposed in Section 5.3.5. The primal active-set method in Section 5.3.5 is compared with the accelerated Dykstra's algorithm proposed in Section 5.3.6 to optimise TV on a 2D grid, which is decomposed into 2 functions of 1D chains.

5.4.1 Non decomposable total variation denoising

Our experiments consider images, which are 2-dimensional grids with 4-neighborhood. The dataset comprises of 6 different images of varying sizes. We consider a large image of size 5616×3744 and recursively scale into a smaller image of half the width and half the height maintaining the aspect ratio. Therefore, the size of each image is four times smaller than size of the previous image. We restrict to anisotropic uniform-weighted total variation to compare with Chambolle et al. [Chambolle and Darbon, 2009] but our algorithms works as well with weighted total variation, which is standard in computer vision, and on any graph with SFM oracles. Therefore, the

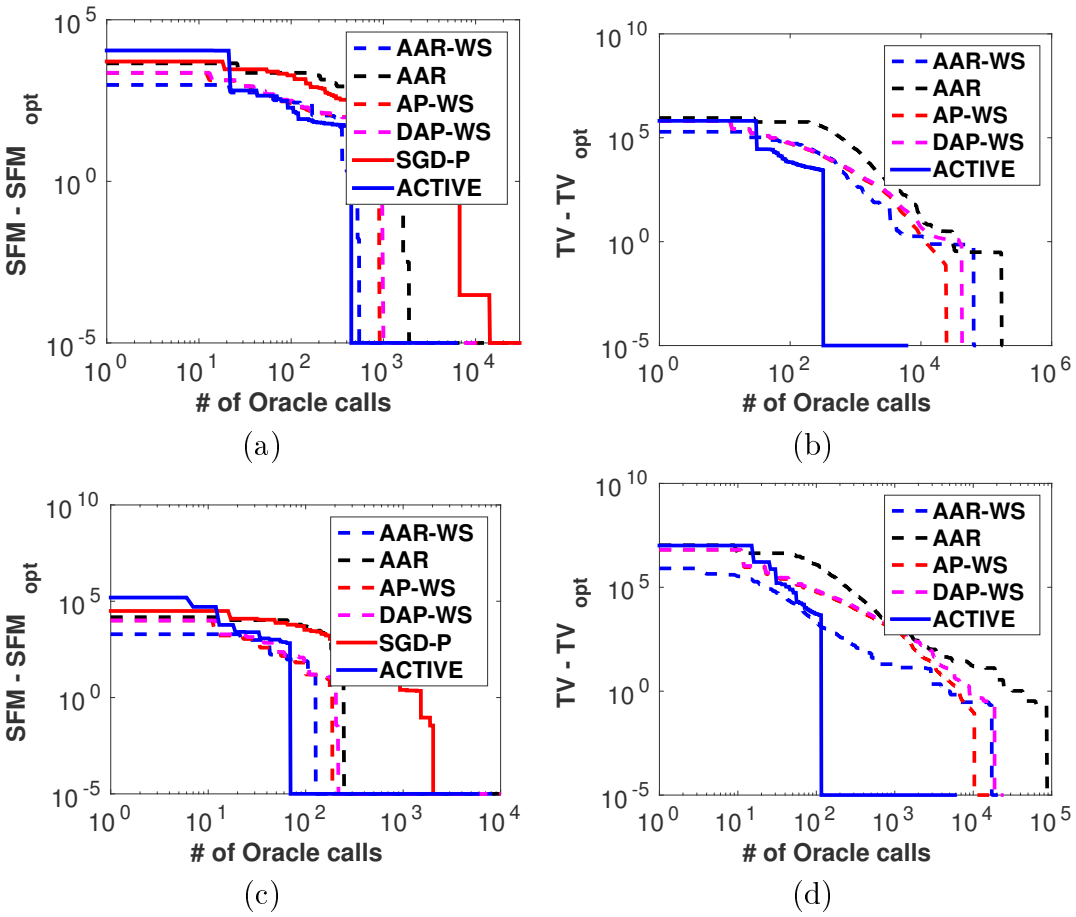


Figure 5-7 – (a) Number of 2D SFM calls to obtain 3D SFM, (b) Number of 2D SFM calls to obtain 3D TV, (c) Number of 2D SFM calls to obtain SFM of 2D + concave function, (d) Number of 2D SFM calls to obtain TV of 2D + concave function.

unweighted total variation is

$$f(w) \;=\; \lambda \sum_{i \sim j} |w_i \;-\; w_j|,$$

where λ is a regularizing constant for solving the total variation problem in Eq. (5.1). Note that we restrict to uniform weights only to be able to perform a fair comparison with their method [Chambolle and Darbon, 2009].

Maxflow [Boykov and Kolmogorov, 2004] is used as the SFM oracle for checking the optimality of the ordered partitions. Figure 5-6-(a) shows the number of SFM oracle calls required to solve the TV problem for images of various sizes. Note that in [Chambolle and Darbon, 2009] each SFM oracle call optimises smaller problems sequentially, while each SFM oracle call in our method optimises several independent smaller problems in parallel. Therefore, our method has lesser number of oracle calls than [Chambolle and Darbon, 2009]. However, oracle complexity of each call is higher for the our method when compared to [Chambolle and Darbon, 2009]. Figure 5-6-(b) shows the time required for each of the methods to solve the TV problem to convergence. We have an optimised code and only use the oracle as plugin which takes about 80-85 percent of the running time. This is primarily the reason our approach takes more time than [Chambolle and Darbon, 2009] inspite of having lesser oracle calls.

Figure 5-6-(c) also shows the ability to warm start by using the output of a related problem, i.e., when computing the solution for several values of λ (which is typical in practice). In this case, we use optimal ordered partitions of the problem with larger λ to warm start the problem with smaller λ. It can be observed that warm start of the algorithm requires lesser number of oracle calls to converge than using the initialisation with trivial ordered partition. Warm start also largely helps in reducing the burden on the SFM oracle. With warm starts the number of ordered partitions does not change much over iterations. Hence, it suffices to query only ordered partitions that have changed. To analyze this we define *oracle complexity* as the ratio of pixels in the elements of the partitions that need to be queried with the full set. Oracle complexity is averaged over iterations to understand the average burden on the oracle per iteration. With warm starts this reduces drastically, which can be observed in Figure 5-6-(d).

5.4.2 Decomposable total variation denoising and SFM

Cut functions. In the decomposable case, we consider a 3D-grid that decomposes into a function F_1 composed of 2D grids and a function F_2 composed of chains. For each of these functions, the corresponding SFM oracle is a maxflow-mincut [Boykov and Kolmogorov, 2004] and message passing algorithm on chains respectively. We consider averaged alternating reflection (AAR) [Bauschke and Luke, 2004] by solving each projection without warmstart and counting the number of 2D SFM oracle calls to solve the corresponding projection as our baseline. We compare our algorithm with other methods like alternating projection (AP-WS), averaged alternating

reflection (AAR-WS) [Bauschke and Luke, 2004], Dykstra's alternating projection (DAP-WS) [Bauschke and Borwein, 1994] and dual subgradient based method (SGD-P) [Komodakis et al., 2011] modified with Polyak's [Bertsekas, 1999] rule. All these algorithms use the same optimisation algorithm but use ordered partitions to warmstart the projection using ordered partitions as described in Section 5.2.4. The performance of the activeset algorithm proposed in Section 5.3.3 with inner loop solved using the primal active-set method proposed in Section 5.3.5 is represented by (ACTIVE).

In our experiments, we consider the 3D volumetric dataset of the Stanford bunny [max] of sizes $102 \times 100 \times 79$. F_1 denotes 102 2D frame of size 100×79 and F_2 represents the $100 \times 79 = 7900$ chains of length 102. Figure 5-7-(a) and (b) shows that our algorithm converges for solving TV quickly by using only SFM oracles and relatively less number of oracle calls. Note that we count 2D SFM oracle calls as they are more expensive than the SFM oracles on chains.

Concave functions on cardinality. In this experiment we consider SFM problem of sum of a 2D cut on a graph of size 5616×3744 and a super pixel based concave function on cardinality [Stobbe and Krause, 2010, Jegelka et al., 2013]. The unary potentials of each pixel is calculated using the Gaussian mixture model of the color features. The edge weight $a(i, j) = \exp(-\|y_i - y_j\|^2)$, where y_i denotes the RGB values of the pixel i. In order to evaluate the concave function, regions R_j are extracted via superpixels and, for each R_j, defining the function $F_2(S) = |S||R_j \setminus S|$. We use 200 and 500 regions. Figure 5-7-(c) and (d) shows that our algorithm converges for solving TV quickly by using only SFM oracles and relatively less number of oracle calls. Note that we count 2D SFM oracle calls.

5.5 Conclusion

We have presented an active-set method to solve submodular function minimisation (SFM) and total variation (TV) denoising problem. For decomposable problems, we have showed that we can solve both TV denoising and SFM problems by using cheaper SFM oracles of the individual functions, while other competitive methods use expensive TV oracles, which restricts the function decomposition, as for cut functions, they are efficient only for chains and trees. This provides us flexibility to decompose into functions for which we have efficient SFM oracles. Due to inherent parallelism, this approach can be very useful in solving large scale optimization problems. As future work, it could be interesting to improve the solutions for "translated Dykstra problem" in the decomposable case and extend this formulation to solve constrained submodular optimization.

Appendix

This section contains the appendices related to this chapter.

5.A Algorithms for coalescing partitions

The basic interpretation in coalescing two ordered partitions is as follows. Given an ordered partition $\mathcal{A}_1$ and $\mathcal{A}_2$ with m_1 and m_2 elements in the partitions respectively, we define for each $j = 1, 2$, $\forall i_j = (1, \ldots, m_j)$,

$$B_{j,i_j} = (A_{j,1} \cup \ldots \cup A_{j,i_j}).$$

The inequalities defining the outer approximation of the base polytopes are given by hyperplanes defined by $\forall i_j = (1, \ldots, m_j), s_j(B_{j,i_j}) \leq F_j(B_{j,i_j})$. The hyperplanes defined by common sets of both these partitions, defines the coalesced ordered partitions. The following algorithm performs coalescing between these partitions.

— **Input**: Ordered partitions $\mathcal{A}_1$ and $\mathcal{A}_2$.
— **Initialize**: $x = 1$, $y = 1$, $z = 1$ and $C = \varnothing$.
— **Algorithm**: Iterate until $x = m_1$ and $y = m_2$ with $m = z$
  (a) If $|B_{1,x}| > |B_{2,y}|$ then $y := y + 1$.
  (b) If $|B_{1,x}| < |B_{2,y}|$ then $x := x + 1$.
  (c) If $|B_{1,x}| == |B_{2,y}|$ then
    — If $B_{1,x} == B_{2,y}$ then
      — $A_z = (B_{1,x} \setminus C)$,
      — $C = B_{1,x}$, and
      — $z := z + 1$.
— **Output**: $m = z$, ordered partitions $\mathcal{A} = (A_1, \ldots, A_m)$.

5.B Optimality of algorithm for decomposable problems

In step (d) of the algorithms, when we split partitions, the value of the primal/dual pair of optimization algorithms

$$\max_{\substack{s_1 \in \widehat{B}^{\mathcal{A}_1}(F_1) \\ s_2 \in \widehat{B}^{\mathcal{A}_2}(F_2)}} -\tfrac{1}{2}\|u - s_1 - s_2\|_2^2,$$

$$= \min_{\substack{w\in\mathcal{W}^{\mathcal{A}_1}\\ w\in\mathcal{W}^{\mathcal{A}_2}}} f_1(w) + f_2(w) - u^\top w + \tfrac{1}{2}\|w\|_2^2,$$

cannot increase. This because, when splitting, the constraint set for the minimization problem only gets bigger. Since at optimality, we have $w = u - s_1 - s_2$, $\|w\|_2$ cannot decrease, which shows the first statement.

Now, if $\|w\|_2$ remains constant after an iteration, then it has to be the same (and not only have the same norm), because the optimal s_1 and s_2 can only move in the direction orthogonal to w.

In step (b) of the algorithm, we project 0 on the (non-empty) intersection of $\widehat{B}^{\mathcal{A}_1}(F_1) - u/2 + w/2$ and $u/2 - w/2 - \widehat{B}^{\mathcal{A}_2}(F_2)$. This corresponds to minimizing $\frac{1}{2}\|s_1 - u/2 + w/2\|^2$ such that $s_1 \in \widehat{B}^{\mathcal{A}_1}(F_1)$ and $s_2 = u - w - s_1 \in \widehat{B}^{\mathcal{A}_2}(F_2)$. This is equivalent to minimizing $\frac{1}{8}\|s_1 - s_2\|^2$. We have:

$$\begin{aligned}
\max_{\substack{s_1\in\widehat{B}^{\mathcal{A}_1}(F_1)\\ s_2\in\widehat{B}^{\mathcal{A}_2}(F_2)\\ s_1+s_2=u-w}} -\frac{1}{8}\|s_1 - s_2\|_2^2 &= \min_{\substack{w_1\in\mathcal{W}^{\mathcal{A}_1}\\ w_2\in\mathcal{W}^{\mathcal{A}_2}}} \max_{\substack{s_1\in\mathbb{R}^n\\ s_2\in\mathbb{R}^n\\ s_1+s_2=u-w}} \Big(-\frac{1}{8}\|s_1 - s_2\|_2^2 + f_1(w_1) + f_2(w_2) \\
&\qquad\qquad -w_1^\top s_1 - w_2^\top s_2\Big) \\
&= \min_{\substack{w_1\in\mathcal{W}^{\mathcal{A}_1}\\ w_2\in\mathcal{W}^{\mathcal{A}_2}}} \max_{s_2\in\mathbb{R}^n} \Big(-\frac{1}{8}\|u - w - 2s_2\|_2^2 + f_1(w_1) + f_2(w_2) \\
&\qquad\qquad -w_1^\top(u - w - s_2) - w_2^\top s_2\Big) \\
&= \min_{\substack{w_1\in\mathcal{W}^{\mathcal{A}_1}\\ w_2\in\mathcal{W}^{\mathcal{A}_2}}} \max_{s_2\in\mathbb{R}^n} \Big(-\frac{1}{8}\|u - w\|_2^2 - \frac{1}{2}\|s_2\|_2^2 + \frac{1}{2}s_2^\top(u - w) \\
&\qquad\qquad +f_1(w_1) + f_2(w_2) - w_1^\top(u - w - s_2) - w_2^\top s_2\Big) \\
&= \min_{\substack{w_1\in\mathcal{W}^{\mathcal{A}_1}\\ w_2\in\mathcal{W}^{\mathcal{A}_2}}} -w_1^\top(u - w) + f_1(w_1) + f_2(w_2) - \frac{1}{8}\|u - w\|_2^2 \\
&\qquad\qquad + \max_{s_2\in\mathbb{R}^n} -\frac{1}{2}\|s_2\|_2^2 + s_2^\top\big(\tfrac{u-w}{2} + w_1 - w_2\big) \\
&= \min_{\substack{w_1\in\mathcal{W}^{\mathcal{A}_1}\\ w_2\in\mathcal{W}^{\mathcal{A}_2}}} \Big(-w_1^\top(u - w) + f_1(w_1) + f_2(w_2) - \frac{1}{8}\|u - w\|_2^2 \\
&\qquad\qquad +\frac{1}{2}\|\tfrac{u-w}{2} + w_1 - w_2\|_2^2\Big) \\
&= \min_{\substack{w_1\in\mathcal{W}^{\mathcal{A}_1}\\ w_2\in\mathcal{W}^{\mathcal{A}_2}}} \Big(-w_1^\top(u - w) + f_1(w_1) + f_2(w_2) + \frac{1}{2}\|w_1 - w_2\|_2^2 \\
&\qquad\qquad +\frac{1}{2}(u - w)^\top(w_1 - w_2)\Big)
\end{aligned}$$

$$= \min_{\substack{w_1 \in \mathcal{W}^{\mathcal{A}_1} \\ w_2 \in \mathcal{W}^{\mathcal{A}_2}}} \Big(f_1(w_1) + f_2(w_2) - \frac{1}{2}(u-w)^\top (w_1 + w_2) + \frac{1}{2}\|w_1 - w_2\|_2^2 \Big),$$

Thus s_1 and s_2 are dual to certain vectors w_1 and w_2, which minimize a decoupled formulation in f_1 and f_2. To check optimality, like in the single function case, it decouples over the constant sets of w_1 and w_2, which is exactly what step (c) is performing.

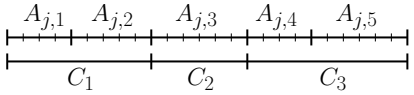
If the check is satisfied, it means that w_1 and w_2 are in fact optimal for the problem above without the restriction in compatibilities, which implies that they are the Dykstra solutions for the TV problem.

If the check is not satisfied, then the same reasoning as for the one function case, leads directions of descent for the new primal problem above. Hence it decreases; since its value is equal to $-\frac{1}{8}\|s_1 - s_2\|_2^2$, the value of $\|s_1 - s_2\|_2^2$ must increase, hence the second statement.

5.C Decoupled problems.

Given that we deal with polytopes, knowing w implies that we know the faces on which we have to looked for. It turns outs that for base polytopes, these faces are products of base polytopes for modified functions (a similar fact holds for their outer approximations).

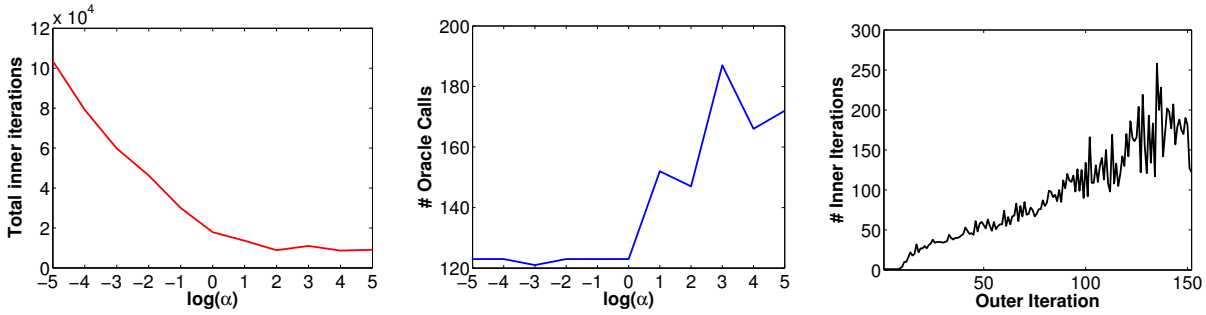
Given the ordered partition $\mathcal{A}'$ defined by the level sets of w (which have to be finer than $\mathcal{A}_1$ and $\mathcal{A}_2$), we know that we may restrict $\widehat{B}^{\mathcal{A}_j}(F_j)$ to elements s such that $s(B) = F(B)$ for all sup-level sets B of w (which have to be unions of contiguous elements of A_j); see an illustration below.



More precisely, if $C_1, \dots, C_{m'}$ are constant sets of w ordered with decreasing values. Then, we may search for s_j independently for each subvector $(s_j)_{C_k} \in \mathbb{R}^{C_k}$, $k \in \{1, \dots, m'\}$ and with the constraint that

$$(s_j)_{C_k} \in \hat{B}^{\mathcal{A}_j \cap C_k}\big[(F_j)_{C_k | C_1 \cup \cdots \cup C_{k-1}}\big],$$

where $\mathcal{A}_j \cap C_k$ is the ordered partition obtained from $\mathcal{A}_j$ once restricted onto C_k and the submodular function is the so-called contraction of F on C_k given $C_1 \cup \cdots \cup C_{k-1}$, defined as $S \mapsto F_j(S \cup C_1 \cup \cdots \cup C_{k-1}) - F(C_1 \cup \cdots \cup C_{k-1})$. Thus this corresponds to solving m different smaller subproblems.



(a) (b) (c)

Figure 5-8 – (a) Total number of inner iterations for varying α. (b) Total number of outer iterations for varying α. and (c) Number of inner iterations per each outer iteration for the $\alpha = 10^1$

5.D Choice of α

The Dykstra step, i.e., step (b) of the algorithm proposed in Section 5.3.6 is not finitely convergent. Therefore, it needs to be solved approximately. For this purpose, we introduce a parameter α to approximately solve the Dykstra step such that $\|s_1 + s_2 - u + w\|_1 \leq \alpha(\epsilon_1 + \epsilon_2)$. Let ϵ be defined as $\alpha(\epsilon_1 + \epsilon_2)$. This shows that the s_1 and s_2 are ϵ-accurate. Therefore, α must be chosen in such a way that we avoid cycling in our algorithm. However, another alternative is to warm start the dykstra step with w_1 and w_2 of the previous iteration. This ensures we dont go back to the same w_1 and w_2, which we have already encountered and avoid cycling. Figure 5-8 shows the performance of our algorithm for a simple problem of 100×100 2D-grid with 4-neighborhood and uniform weights on the edges with varying α. Figure 5-8-(a) shows the total number of inner iterations required to solve the TV problem. Figure 5-8-(b) gives the total number of SFM oracle calls required to solve the TV problem. In Figure 5-8-(c), we show the number of inner iterations in every outer iteration for the best α we have encountered.

Chapter 6

Conclusion and Future Work

6.1 Summary of the thesis

In this thesis, we focused on some problems specific to probabilistic graphical models and submodular optimisation. We leverage the links between them to propose solutions to some of the related problems.

In our first contribution, we consider the problem of learning bounded treewidth decomposable graphs under the maximum likelihood framework. Chow and Liu [1968] proposed an exact algorithm to learn maximum likelihood trees. Srebro [2002] proved that learning higher treewidth decomposable graphs is NP-hard. The first convex relaxation approach to learn bounded treewidth graphs was proposed by Karger and Srebro [2001]. We pose the problem as a combinatorial optimisation problem and propose a convex relaxation. The convex relaxation leads to optimising linear functions on graphic and hypergraphic matroids. Use of these matroids enabled us to use simple greedy algorithms to propose a subgradient based optimisation algorithm. We have shown that the algorithm is able to recover the structure in toy problems on some instances. On the rest, it is shown to have recovered an approximate structure. We also show that the algorithm empirically outperforms on some standard datasets when compared to state-of-art algorithms.

In our second main contribution, we propose a graph based bounds to submodular functions and use variational inference on graphs to maximise general submodular functions. Maximising submodular functions is an NP hard problem. However, greedy algorithms [Buchbinder et al., 2012, Feige et al., 2011] give constant factor approximations to maximise non-monotone submodular functions. However, all these algorithms use function value oracles to optimise them. There are functions like entropy and mutual information where the function value oracles have complexity exponential in cardinality of the set. Our work proposes an approach where the access to function value oracles are available only for sets up to cardinality $(k+1)$. This is the first such work in this direction. We use graphs with treewidth bounded by k to approximate the submodular functions by using the algorithm from our first contribution and use variational inference algorithms on bounded treewidth graphs to propose an algorithm to maximise submodular functions.

In our final contribution, we borrow our motivation from convex optimisation literature to minimise sum of submodular functions. In the first subpart of this contribution, we make an extensive analysis of existing algorithms based on proximal splitting methods to minimise sum of large scale submodular functions using TV oracle [Jegelka et al., 2013]. We use cut functions specifically in this context and compare them with state-of-art combinatorial algorithms. We observed that the proximal splitting based methods have very low memory footprint when compared to combinatorial algorithms, which use heavy data-structures. However there are two drawbacks to this approach. The first one is assumption of exact TV oracles of individual functions, which is not generally the case. And the second is lack of quick convergence in cases when we consider minimising submodular functions, which are decomposed into more than two functions. We address the former case in the next subpart of this contribution. We only assume SFM oracles instead of TV oracles of the individual submodular functions. We use ordered partitions to characterise the base polytope of the individual submodular function and use local search over ordered partitions to solve the corresponding TV problem of the individual submodular functions. This already improves the performance of the existing proximal splitting methods as ordered partitions provide the flexibility to warm start. We further extend this framework to decomposable case by using extending this framework to search over ordered partitions.

6.2 Perspectives

Our work has triggered a few questions, which are still open.

(a) Learning bounded treewidth decomposable graphs under maximum likelihood framework is known to be NP-hard. We provide a convex relaxation to learn bounded treewidth graphs. However, there is a lack of theoretical analysis. One possible direction is to study the duality gap, which provides a gap on the integrality gap of the problem of learning bounded treewidth decomposable graphs. This could possibility give an insight into approximability of the problem of learning bounded treewidth decomposable graph under the maximum likelihood framework. The main motivation is the following open problem.

(b) Maximising submodular functions admit constant factor approximation algorithms with function value oracles. Optimal bounds are available for approximability on various submodular maximisation problems and all of them assume function value oracle. It is an open problem to estimate the optimal bound on having access to restricted function value oracles, i.e, an oracle, which can return a function value only for sets of up to cardinality k. Note that in some problems, it is not possible to compute higher order entropies.

(c) In most of our work and literature [Komodakis et al., 2011, Kappes et al., 2013, Jegelka et al., 2013, Strandmark and Kahl, 2010], the decomposition of submodular functions into "simple" functions is by design based on availablity of oracles. There are two cases of decomposition of submodular functions,

where the algorithms we proposed can be of help. In the first case, the decomposition is based on decomposing the ground set, i.e., V into disjoint sets V_i, which ensures quicker convergence in the outerloop as the base polytopes become orthogonal in this case. Projection methods converge linearly in estimating nearest points on on the polytopes in this case [Nishihara et al., 2014]. Here, the "simple" functions are the same functions with smaller problem size V_i but the oracle complexity may still be high. In the second case, as we did in our thesis, the "simple" functions are characterised by simplicity of the oracles with the same problem size. This could lead to slower convergence of the outerloop. It is of clear interest to consider the whole regime of problems that lie in between these two extreme cases of decomposition and find an optimal balance.

(d) We used SFM and TV oracles of individual functions to minimise the sum of submodular functions. However, in our work we mainly concentrated on solving the binary problem or discrete optimisation problem related to sets. The same framework naturally extends to solve multi-label problems with total order on the labels, i.e., to solve the formulation of Ishikawa [2003]. We believe it is important to propose similar algorithms to optimise tree-submodular functions [Kolmogorov, 2010] or k-submodular functions [Gridchyn and Kolmogorov, 2013].

(e) A subclass of non-convex functions can be characterised by submodular functions [Bian et al., 2016, Bach, 2015] and algorithms have been proposed to optimise these class of functions. Our work could also contribute in this direction by assuming structure to the submodular function that characterises the non-convex function. This could lead to large scale distributed non-convex optimisation.

Bibliography

Maxflow dataset online. `http://vision.csd.uwo.ca/maxflow-data`.

F. R. Bach. Submodular functions: from discrete to continous domains. *ArXiv e-prints*, 2015.

F.R. Bach. *Learning with Submodular Functions: A Convex Optimization Perspective*, volume 6 of *Foundations and Trends in Machine Learning*. NOW, 2013.

F.R. Bach and M.I. Jordan. Thin junction trees. In *Advances in Neural Information Processing Systems*, 2002.

.Á Barbero and S. Sra. Modular proximal optimization for multidimensional total-variation regularization. *ArXiv e-prints*, 2014.

H. H. Bauschke and J. M. Borwein. Dykstra's alternating projection algorithm for two sets. *Journal of Approximation Theory*, 79(3):418–443, 1994.

H. H. Bauschke, J. M. Borwein, and A. S. Lewis. The method of cyclic projections for closed convex sets in Hilbert space. *Contemporary Mathematics*, 204:1–38, 1997.

Heinz H. Bauschke and Jonathan M. Borwein. On projection algorithms for solving convex feasibility problems. *SIAM Review*, 38(3), 1996.

P. L. Bauschke, H. H.and Combettes and D. Luke. Finding best approximation pairs relative to two closed convex sets in Hilbert spaces. *Journal of Approximation theory*, 127(2):178–192, 2004.

A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

A. Beck and L. Tetruashvili. On the convergence of block coordinate descent type methods. *SIAM Journal on Optimization*, 23(4):2037–2060, 2013.

I. A. Beinlich, H. J. Suermondt, R. M. Chavez, and G. F. Cooper. The ALARM Monitoring System: A Case Study with Two Probabilistic Inference Techniques for Belief Networks. In *Proceeding of European Conference on AI in Medicine*, 1989.

D. P. Bertsekas. *Nonlinear programming*. Athena Scientific, 1999.

D. P. Bertsekas and H. Yu. A unifying polyhedral approximation framework for convex optimization. *SIAM Journal on Optimization*, 21(1):333–360, 2011.

M. J. Best and N. Chakravarti. Active set algorithms for isotonic regression: a unifying framework. *Mathematical Programming*, 47(1):425–439, 1990.

Y. Bian, B. Mirzasoleiman, J.M. Buhmann, and A. Krause. Guaranteed non-convex optimization: Submodular maximization over continuous domains. *ArXiv e-prints*, 2016.

C.M. Bishop et al. *Pattern recognition and machine learning*. springer New York, 2006.

A. Blake, P. Kohli, and C. Rother, editors. *Markov Random Fields for Vision and Image Processing*. MIT Press, 2011.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers*, volume 3 of *Foundations and Trends in Machine Learning*. NOW, 2011.

Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis Machine Intelligence*, September 2004.

Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23 (11):1222–1239, 2001.

N. Buchbinder, M. Feldman, N. Naor, Joseph, and R. Schwartz. A tight linear time (1/2)-approximation for unconstrained submodular maximization. In *IEEE Symposium on Foundations of Computer Science*, 2012.

D. Chakrabarty, P. Jain, and P. Kothari. Provable submodular minimization using Wolfe's algorithm. In *Advances in Neural Information Processing Systems*, 2014.

A. Chambolle and J. Darbon. On total variation minimization and surface evolution using parametric maximum flows. *International Journal of Computer Vision*, 84 (3):288–307, 2009.

A. Chambolle and T. Pock. A remark on accelerated block coordinate descent for computing the proximity operators of a sum of convex functions. *HAL*, 2015.

B. G. Chandran and D. S. Hochbaum. A computational study of the pseudoflow and push-relabel algorithms for the maximum flow problem. *Operations Research*, 2009.

A. Chechetka and C. Guestrin. Efficient principled learning of thin junction trees. In *Advances in Neural information Processing Systems*, 2007.

D. M. Chickering. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507–554, November 2002.

C. I. Chow and C. N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14, 1968.

Laurent Condat. A direct algorithm for 1d total variation denoising. Technical report, GREYC laboratory, CNRS-ENSICAEN-Univ. of Caen, 2012.

T. M. Cover and J. A. Thomas. *Elements of information theory*. John Wiley & Sons, 2006.

A. Deshpande, M.N. Garofalakis, and M.I. Jordan. Efficient stepwise selection in decomposable models. In *Proceedings of Uncertainity in Artifial Intelligence*, 2001.

F. R. Deutsch. *Best approximation in Inner Product Spaces*. Springer Verlag, 2001.

J. Djolonga and A. Krause. From map to marginals: Variational inference in bayesian submodular models. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

J. Djolonga and A. Krause. Scalable variational inference in log-supermodular models. In *International Conference on Machine Learning*, 2015.

J. Edmonds. Submodular functions, matroids, and certain polyhedra. In *Combinatorial optimization - Eureka, you shrink!*, pages 11–26. Springer, 2003.

D. Edwards. *Introduction to Graphical Modelling*. Springer, 2000.

U. Feige, V. S. Mirrokni, and J. Vondrák. Maximizing non-monotone submodular functions. *SIAM Journal on Computing*, 2011.

W. Fenchel. On conjugate convex functions. *Canadian Journal of Mathematics*, 1: 73–77, 1949.

A. Frank, T. Király, and M. Kriesell. On decomposing a hypergraph into k connected sub-hypergraphs. *Discrete Applied Mathematics*, 131(2):373–383, 2003.

N. Friedman and D. Koller. Being Bayesian about network structure. a bayesian approach to structure discovery in bayesian networks. *Machine learning*, 50(1): 95–125, 2003.

S. Fujishige. Lexicographically optimal base of a polymatroid with respect to a weight vector. *Mathematics of Operations Research*, pages 186–196, 1980.

S. Fujishige. *Submodular Functions and Optimization*. Annals of Discrete Mathematics. Elsevier, 2005.

T. Fukunaga. Computing minimum multiway cuts in hypergraphs from hypertree packings. *Integer Programming in Combinatorial Optimization*, pages 15–28, 2010.

N. Gaffke and R. Mathar. A cyclic projection algorithm via duality. *Metrika*, 36(1): 29–54, 1989.

P. Giudici and P.J. Green. Decomposable graphical gaussian model determination. *Biometrika*, 86(4), 1999.

V. Gogate, W.A. Webb, and P. Domingos. Learning efficient Markov networks. In *Advances in Neural Information Processing Systems*, 2010.

A. V. Goldberg, S. Hed, H. Kaplan, R. E. Tarjan, and R. F. Werneck. Maximum flows by incremental breadth-first search. In *Proceedings of European Conference on Algorithms*, 2011.

M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. North Holland, 2004.

C. Gottschalk and B. Peis. Submodular function maximization on the bounded integer lattice. *ArXiv e-prints*, 2015.

M. Grasmair. The equivalence of the taut string algorithm and BV-regularization. *Journal of Mathematical Imaging and Vision*, 27(1):59–66, 2007.

I Gridchyn and V. Kolmogorov. Potts model, parametric maxflow and k-submodular functions. *ArXiv e-prints*, 2013.

H. Groenevelt. Two algorithms for maximizing a separable concave function over a polymatroid feasible region. *European Journal of Operational Research*, 54(2): 227–236, 1991.

M. Hein, S. Setzer, L. Jost, and S.S. Rangapuram. The total variation on hypergraphs - learning on hypergraphs revisited. In *Advances in Neural Information Processing Systems*, 2013.

D. S. Hochbaum. An efficient algorithm for image segmentation, markov random fields and related problems. *Journal of the ACM*, 48(2), 2001.

D. S. Hochbaum. A polynomial time algorithm for rayleigh ratio on discrete variables: Replacing spectral techniques for expander ratio, normalized cut, and cheeger constant. *Operations Research*, 61(1):184–198, 2013.

D.S. Hochbaum and S.-P. Hong. About strongly polynomial time algorithms for quadratic optimization over submodular constraints. *Mathematical Programming*, 69(1):269–309, 1995.

H. Höfling and R. Tibshirani. Estimation of sparse binary pairwise markov networks using pseudo-likelihoods. *Journal of Machine Learning Research*, 2009.

H. Ishikawa. Exact optimization for markov random fields with convex priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2003.

R. Iyer, S. Jegelka, and J. Bilmes. Fast semidifferential-based submodular function optimization. In *Proceedings of International Conference in Machine Learning*, pages 855–863, 2013.

S. Jegelka, F. R. Bach, and S. Sra. Reflection methods for user-friendly submodular optimization. In *Advances in Neural Information Processing Systems*, 2013.

N. A. Johnson. A dynamic programming algorithm for the fused Lasso and l_0-segmentation. *Journal Computational and Graphical Statistics*, 2013.

J. Kappes, B. Andres, C. Schnoerr, F. Hamprecht, S. Nowozin, D. Batra, J. Lellmann, N. Komodakis, S. Kim, B. Kausler, and C. Rother. A comparative study of modern inference techniques for discrete energy minimization problems. In *Proceedings of Computer Vision and Pattern Recognition*, 2013.

D. Karger and N. Srebro. Learning Markov networks: maximum bounded tree-width graphs. In *Proc. ACM-SIAM symposium on Discrete algorithms*, volume 10, 2001.

P. Kohli and P. Torr. Efficiently solving dynamic markov random fields using graph cuts. In *Proceedings of International Conference on Computer Vision*, 2005.

P. Kohli, M.P. Kumar, and P. Torr. P3 & beyond: Move making algorithms for solving higher order functions. *IEEE Transactions on Pattern Analysis and Machine Learning*, pages 1645—-1656, 2009a.

P. Kohli, L. Ladický, and P.H.S. Torr. Robust higher order potentials for enforcing label consistency. *International Journal of Computer Vision*, 82(3):302—-324, 2009b.

D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE Transaction on Pattern Analysis Machine Intelligence*, 2006.

V. Kolmogorov. Submodularity on a tree: Unifying $L^\natural$-convex and bisubmodular functions. *ArXiv e-prints*, 2010.

V. Kolmogorov. Minimizing a sum of submodular functions. *Discrete Applied Mathematics*, 160(15), 2012.

V. Kolmogorov and T. Schoenemann. Generalized sequential tree-reweighted message passing. *ArXiv e-prints*, 2012.

V. Kolmogorov, Y. Boykov, and C. Rother. Applications of parametric maxflow in computer vision. In *Proceedings of International Conference on Computer Vision*, 2007.

V. Kolmogorov, T. Pock, and M. Rolinek. Total variation on a tree. *ArXiv e-prints*, 2015.

N. Komodakis, Gi. Tziritas, and N. Paragios. Performance vs computational efficiency for optimizing single and dynamic mrfs: Setting the state of the art with primal-dual strategies. *Computer Vision Image Understanding*, 112(1):14–29, October 2008.

N. Komodakis, N. Paragios, and G. Tziritas. MRF energy minimization and beyond via dual decomposition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2011.

A. Krause and C. Guestrin. Near-optimal nonmyopic value of information in graphical models. In *Proceedings of Uncertainity in Artificial Intelligence*, 2005.

J. B. Kruskal. On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. In *Proceedings of the American Mathematical Society*, 1956.

L. Landrieu and G. Obozinski. Cut Pursuit: fast algorithms to learn piecewise constant functions. In *In Proc. AISTATS*, 2016.

S. L. Lauritzen. *Graphical Models*. Oxford University Press, Oxford, 1996.

H. Lin and J. Bilmes. A class of submodular functions for document summarization. In *Proceedings of North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2011.

M. Lorea. Hypergraphes et matroides. In *cahiers du centre d'études de recherche opérationnelle*, 1975.

L. Lovász. Submodular functions and convexity. *Mathematical programming: the state of the art, Bonn*, pages 235–257, 1982.

F.M. Malvestuto. Approximating discrete probability distributions with decomposable models. *IEEE Transactions on Systems, Man, Cybernetics*, 21(5), 1991.

K. Murota. *Discrete Convex Analysis: Monographs on Discrete Mathematics and Applications 10*. Society for Industrial and Applied Mathematics, 2003.

K. P. Murphy. *Machine learning: a probabilistic perspective*. MIT Press, 2012.

K. Nagano, Y. Kawahara, and K. Aihara. Size-constrained submodular optimization through minimum norm base. In *International Conference on Machine Learning*, 2011.

M. Narasimhan and J. Bilmes. PAC-learning bounded tree-width graphical models. In *Proceedings of Uncertainity in Artificial Intelligence*, 2004.

M. Narasimhan and J. Bilmes. A submodular-supermodular procedure with applications to discriminative structure learning. In *Proceedings of Uncertainity in Artificial Intelligence*, 2005.

A. Nedic and A. Ozdaglar. Approximate primal solutions and rate analysis for dual subgradient methods. *SIAM Journal on Optimization*, 19(4):1757–1780, 2009.

R. Nishihara, S. Jegelka, and M.I. Jordan. On the linear convergence rate of decomposable submodular function minimization. In *Advances in Neural Information Processing Systems*, 2014.

J. Nocedal and S. J. Wright. *Numerical optimization*. Springer Series in Operations Research and Financial Engineering. Springer, Berlin, 2006.

J. Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, New York, NY, USA, 2000. ISBN 0-521-77362-8.

J.C. Picard and H.D. Ratliff. Minimum cuts and related problems. *Neworks*, 5(4), 1975.

P. Ravikumar, M.J. Wainwright, and J. Lafferty. High-dimensional ising model selection using l1-regularized logistic regression. *The Annals of Statistics*, 2010.

P. Richtárik and M. Takáč. Parallel coordinate descent methods for big data optimization. *ArXiv e-prints*, 2012.

R. T. Rockafellar. *Convex Analysis*. Princeton University Publishers, 1997.

C. Rother, V. Kolmogorov, V. Lempitsky, and M. Szummer. Optimizing binary MRFs via extended roof duality. In *Proceedings of Computer Vision and Pattern Recognition*, 2007.

L. Saul and M.I. Jordan. Exploiting tractable substructures in intractable networks. In *Advances in Neural Information Processing Systems*, 1995.

B. Savchynskyy, S. Schmidt, J. Kappes, and C. Schnörr. A study of Nesterov's scheme for Lagrangian decomposition and MAP labeling. In *Proceedings of Computer Vision and Pattern Recognition*, 2011.

M. Schmidt, A. Niculescu-Mizil, and K. Murphy. Learning graphical model structure using l1-regularization paths. In *Proceedings of the 22Nd National Conference on Artificial Intelligence - Volume 2*, 2007.

A. Schrijver. *Combinatorial Optimization - Polyhedra and Efficiency*. Springer, 2003.

K S Sesh Kumar and F. R. Bach. Maximizing submodular functions using probabilistic graphical models. *ArXiv e-prints*, 2013a.

K. S. Sesh Kumar and F. R. Bach. Convex relaxations for learning bounded treewidth decomposable graphs. In *Proceedings of International Conference on Machine Learning*, 2013b.

K. S. Sesh Kumar and F.R. Bach. Active-set Methods for Submodular Optimization. *ArXiv e-prints*, June 2015.

K. S. Sesh Kumar, A. Barbero, S. Jegelka, S. Sra, and F. R. Bach. Convex optimization for parallel energy minimization. *ArXiv e-prints*, 2015.

D. Shahaf, A. Chechetka, and C. Guestrin. Learning thin junction trees via graph cuts. In *Proceedings of Artificial Intelligence and Statistics (AISTATS)*, 2009.

H. D. Sherali and W. P. Adams. A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM Journal Discrete Mathematics*, 1990.

X. Shusheng. Estimation of the convergence rate of Dykstra's cyclic projections algorithm in polyhedral case. *Acta Mathematicae Applicatae Sinica (English Series)*, 16(2):217–220, 2000.

D. Sontag and T. Jaakkola. Tree block coordinate descent for MAP in graphical models. In *Proceedings of Artificial Intelligence and Statistics*, 2009.

D. Sontag, A. Globerson, and T. Jaakkola. Introduction to dual decomposition for inference. In *Optimization for Machine Learning*. MIT Press, 2011.

P. Spirtes, C. Glymour, and R. Scheines. *Causation, prediction, and search*, volume 81. MIT press, 2001.

N. Srebro. Maximum likelihood bounded tree-width Markov networks. In *Proceedings of Uncertainity in Artifial Intelligence*, 2002.

P. Stobbe and A. Krause. Efficient minimization of decomposable submodular functions. In *Advances in Neural Information Processing Systems*, 2010.

P. Strandmark and F. Kahl. Parallel and distributed graph cuts by dual decomposition. In *Proceedings of Computer Vision and Pattern Recognition*, 2010.

M. Studeny. *Probabilistic Conditional Independence Structures*. Springer Publishing Company, Incorporated, 1st edition, 2010. ISBN 1849969485, 9781849969482.

T. Szántai and E. Kovács. Discovering a junction tree behind a Markov network by a greedy algorithm. *ArXiv e-prints*, 2011.

T. Szántai and E. Kovács. Hypergraphs as a mean of discovering the dependence structure of a discrete multivariate probability distribution. *Annals of Operations Research*, 193(1), 2012.

R. Tarjan, J. Ward, B. Zhang, Y. Zhou, and J. Mao. Balancing applied to maximum network flow problems. In *European Symposium on Algorithms (ESA)*, pages 612–623, 2006.

M. Teyssier and D. Koller. Ordering-based search: A simple and effective algorithm for learning bayesian networks. In *Proceedings of Uncertainity in Artifial Intelligence*, 2001.

I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *International Conference on Machine Learning*, 2004.

N.K. Vishnoi. *Lx = B - Laplacian Solvers and Their Algorithmic Applications*. Now Publishers, 2013.

J. Vondrák, C. Chekuri, and R. Zenklusen. Submodular function maximization via the multilinear relaxation and contention resolution schemes. In *Proceedings on ACM Symposium Theory of Computing*, pages 783–792, 2011.

M.J. Wainwright and M.I. Jordan. Treewidth-based conditions for exactness of the sherali-adams and lasserre relaxations. Technical report, University of California, Berkeley, Technical Report, 2004.

M.J. Wainwright and M.I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2), 2008.

M.J. Wainwright, T.S. Jaakkola, and A.S. Willsky. A new class of upper bounds on the log partition function. *IEEE Transactions on Information Theory*, 51(7), 2005.

J. Ward and S. Zivny. Maximizing k-submodular functions and beyond. *ArXiv e-prints*, 2014.

H. Whitney. On the abstract properties of linear dependence. *American Journal of Mathematics*, 1935.

A. L. Yuille and A. Rangarajan. The concave-convex procedure. *Neural Computation*, 2003.

Z. Zhang and R. W. Yeung. On characterization of entropy function via information inequalities. *IEEE Transactions on Information Theory*, pages 1440–1452, 1998.

List of Figures

List of Tables

Résumé

L'entropie d'une distribution sur un ensemble de variables aléatoires discrètes est toujours bornée par l'entropie de la distribution factorisée correspondante. Cette propriété est due à la sous-modularité de l'entropie. Par ailleurs, les fonctions sous-modulaires sont une généralisation des fonctions de rang des matroides ; ainsi, les fonctions linéaires sur les polytopes associés peuvent être minimisées exactement par un algorithme glouton.

Dans ce manuscrit, nous exploitons ces liens entre les structures des modèles graphiques et les fonctions sous-modulaires. Nous utilisons des algorithmes gloutons pour optimiser des fonctions linéaires sur des polytopes liés aux matroides graphiques et hypergraphiques pour apprendre la structure de modèles graphiques, tandis que nous utilisons des algorithmes d'inférence sur les graphes pour optimiser des fonctions sous-modulaires.

La première contribution de cette thèse consiste à approcher par maximum de vraisemblance une distribution de probabilité par une distribution factorisable et de complexité algorithmique contrôlée. Comme cette complexité est exponentielle dans la largeur arborescente du graphe, notre but est d'apprendre un graphe décomposable avec une largeur arborescente bornée, ce qui est connu pour être NP-difficile. Nous posons ce problème comme un problème d'optimisation combinatoire et nous proposons une relaxation convexe basée sur les matroides graphiques et hypergraphiques. Ceci donne lieu à une solution approchée avec une bonne performance pratique.

Pour la seconde contribution principale, nous utilisons le fait que l'entropie d'une distribution est toujours bornée par l'entropie de sa distribution factorisée associée, comme conséquence principale de la sous-modularité, permettant une généralisation à toutes les fonctions sous-modulaires de bornes basées sur les concepts de modèles graphiques. Un algorithme est développé pour maximiser les fonctions sous-modulaires, un autre problème NP-difficile, en maximisant ces bornes en utilisant des algorithmes d'inférence vibrationnels sur les graphes.

En troisième contribution, nous proposons et analysons des algorithmes visant à minimiser des fonctions sous-modulaires pouvant s'écrire comme somme de fonctions plus simples. Nos algorithmes n'utilisent que des oracles de ces fonctions simple basés sur minimisation sous-modulaires et de variation totale de telle fonctions.

Mots Clés

Modéles graphiques probabilistes, arbres du maximum de vraisemblance, optimisation discréte, optimisation submodular, variation totale, optimisation convexe.

Abstract

The entropy of a probability distribution on a set of discrete random variables is always bounded by the entropy of its factorisable counterpart. This is due to the submodularity of entropy on the set of discrete random variables. Submodular functions are also generalisation of matroid rank function; therefore, linear functions may be optimised on the associated polytopes exactly using a greedy algorithm.

In this manuscript, we exploit these links between the structures of graphical models and submodular functions: we use greedy algorithms to optimise linear functions on the polytopes related to graphic and hypergraphic matroids for learning the structures of graphical models, while we use inference algorithms on graphs to optimise submodular functions.

The first main contribution of the thesis aims at approximating a probabilistic distribution with a factorisable tractable distribution under the maximum likelihood framework. Since the tractability of exact inference is exponential in the treewidth of the decomposable graph, our goal is to learn bounded treewidth decomposable graphs, which is known to be NP-hard. We pose this as a combinatorial optimisation problem and provide convex relaxations based on graphic and hypergraphic matroids. This leads to an approximate solution with good empirical performance.

In the second main contribution, we use the fact that the entropy of a probability distribution is always bounded by the entropy of its factorisable counterpart mainly as a consequence of submodularity. This property of entropy is generalised to all submodular functions and bounds based on graphical models are proposed. We refer to them as *graph-based bounds*. An algorithm is developped to maximise submodular functions, which is NP-hard, by maximising the graph-based bound using variational inference algorithms on graphs.

As third contribution, we propose and analyse algorithms aiming at minimizing submodular functions that can be written as sum of simple functions. Our algorithms only make use of submodular function minimisation and total variation oracles of simple functions.

Keywords

Probabilistic graphical models, maximum likelihood trees, discrete optimisation, submodular optimisation, total variation, convex optimisation.