Approximate Inference in Graphical Models using LP Relaxations

by

David Alexander Sontag

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2010

Author .....
Department of Electrical Engineering and Computer Science
September 3, 2010

Certified by .....
Tommi S. Jaakkola
Professor
Thesis Supervisor

Accepted by .....
Terry P. Orlando
Chairman, Department Committee on Graduate Students

Approximate Inference in Graphical Models using LP Relaxations

by

David Alexander Sontag

Submitted to the Department of Electrical Engineering and Computer Science
on September 3, 2010, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Graphical models such as Markov random fields have been successfully applied to a wide variety of fields, from computer vision and natural language processing, to computational biology. Exact probabilistic inference is generally intractable in complex models having many dependencies between the variables.

We present new approaches to approximate inference based on linear programming (LP) relaxations. Our algorithms optimize over the cycle relaxation of the marginal polytope, which we show to be closely related to the first lifting of the Sherali-Adams hierarchy, and is significantly tighter than the pairwise LP relaxation.

We show how to efficiently optimize over the cycle relaxation using a cutting-plane algorithm that iteratively introduces constraints into the relaxation. We provide a criterion to determine which constraints would be most helpful in tightening the relaxation, and give efficient algorithms for solving the search problem of finding the best cycle constraint to add according to this criterion.

By solving the LP relaxations in the dual, we obtain efficient message-passing algorithms that, when the relaxations are tight, can provably find the most likely (MAP) configuration. Our algorithms succeed at finding the MAP configuration in protein side-chain placement, protein design, and stereo vision problems.

Thesis Supervisor: Tommi S. Jaakkola
Title: Professor

Acknowledgments

First and foremost, let me thank my adviser, Tommi Jaakkola, for guiding me through my graduate studies. He has been encouraging and supportive of my research, from beginning to end. I went into his office with nearly every crazy idea that I had (there were many), and he listened patiently and helped me formalize them into concrete directions to pursue, many of which made it into this thesis. He has tremendous insight into what approaches will succeed.

My thesis committee consisted of Tommi Jaakkola, David Karger, and Amir Globerson. All three provided valuable feedback on this thesis. It was David who, in my second year of graduate school, pointed me to the literature on cutting plane algorithms and polyhedral combinatorics. Thesis aside, working with David has been very rewarding. Each of our meetings results in a long list of new ideas to think about.

Amir Globerson has been both a good friend and a close colleague. I was very lucky to have started working with him early in my graduate career. He has been an important adviser, not just on research topics, but on how to balance work and family life, and many other things (for example, he helped me figure out how to propose to my wife).

I would also like to thank my collaborators Talya Meltzer and Yair Weiss. Working with Talya has been a lot of fun. I first met Yair when he showed up at Tommi's reading group when I was presenting a paper. When I found out who he was, I became very nervous! Turns out he's a nice guy. His support during my job search was invaluable.

There are many colleagues, faculty, and friends that have helped me immeasurably throughout my graduate studies. A few deserve special recognition. My officemate throughout all of graduate school, Dan Roy, was always there for me, whether it was helping me move between apartments or bringing a poster to NIPS when it didn't finish printing in time for my own flight. Our conversations were some of the highlights of my time at MIT.

Graduate school wouldn't have been the same without Krzysztof Onak, Alex Andoni, and Mihai Patrascu. Besides our many fun social activities, they were always a useful sounding board when tackling a theoretical question. The same goes for many other members of the theory group at MIT.

There were many other faculty members at MIT that helped shape my research directions. Particular thanks go to Regina Barzilay, Bonnie Berger, Michael Collins, Michel Goemans, Piotr Indyk, Leslie Kaelbling, and Tomas Lozano-Perez. Thanks also to my officemate Christy Sauper for so happily bearing with me these last couple of years, and to Tommi's group: John Barnett, Neha Gupta, Patrik Hoyer, Patrycja Missiuro, Jason Rennie, Russ Salakhutdinov, and Yu Xin.

I would also like to acknowledge the National Science Foundation who provided me with a graduate fellowship supporting the first three years of my Ph.D., and Google for a graduate fellowship that supported my last year of graduate studies.

Finally, my biggest thanks go to my loving and supportive family. This thesis is dedicated to my wife, Violeta. Thank you for sharing this journey with me.

Contents

List of Figures

Chapter 1

Introduction

In recent years, advances in science and low-cost permanent storage have resulted in the availability of massive data sets. Together with advances in machine learning, this data has the potential to lead to many new breakthroughs. For example, high-throughput genomic and proteomic experiments can be used to enable personalized medicine. Large data sets of search queries can be used to improve information retrieval. Historical climate data can be used to understand global warming and to better predict weather. However, to take full advantage of this data, we need models that are capable of explaining the data, and algorithms that can use the models to make predictions about the future.

The goal of this thesis is to develop theory and practical algorithms for probabilistic inference in very large statistical models. We focus on a class of statistical models called *graphical models* that describe multivariate probability distributions which factor according to a graph structure. Graphical models provide a useful abstraction for quantifying uncertainty, describing complex dependencies in data while making the model's structure explicit so that it can be exploited by algorithms. These models have been widely applied across diverse fields such as statistical machine learning, computational biology, statistical physics, communication theory, and information retrieval.

For example, consider the problem of predicting the relative orientations of a protein's side-chains with respect to its backbone structure, a fundamental question about protein folding. Given an appropriate energy function, the prediction can be made by finding the side-chain configuration that has minimal energy. This is equivalently formulated as inference in a graphical model, where the distribution is given in terms of compatibility functions between pairs of side-chains and the backbone that take into consideration attractive and repulsive forces.

Statistical models are powerful because, once estimated, they enable us to make predictions with previously unseen observations (e.g., to predict the folding of a newly discovered protein). The key obstacle to using graphical models is that exact inference is known to be NP-hard, or computationally intractable. Finding the most likely protein side-chain configuration, for example, is a difficult combinatorial optimization problem. Many applications of graphical models have hundreds to millions of variables and long-range dependencies. To be useful, probabilistic inference needs to be fast and accurate.

We next describe two applications of graphical models to stereopsis and protein design. These problems will serve as running examples in the thesis of graphical models for which we need efficient inference algorithms.

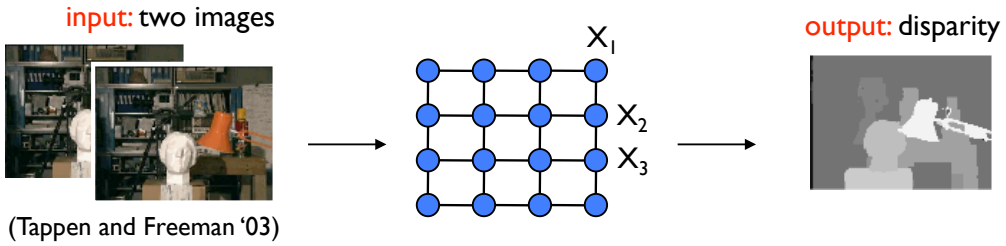


Figure 1-1: Example of how graphical models can be used for computer stereo vision. On the left we show the input, which are two images. We then construct a graphical model where we have one node for each pixel, and edges between neighboring pixels. Finally we perform inference in the model to find the most likely assignment of depth to the pixels, shown on the right (see text for more detail).

1.1 Examples of Graphical Models

1.1.1 Stereo Vision

The stereopsis problem, also called stereo vision, is as follows. We are given two images (e.g., one image coming from your left eye, and the other from your right eye), and the goal is to determine the depth of all the objects in the image from the viewer. The depth of any pixel is inversely proportional to its disparity, namely how much a pixel in the left image is horizontally shifted to obtain the corresponding pixel in the right image. Calculating these disparities is difficult for a number of reasons. First, there is a large amount of ambiguity: for any one pixel in the left image, there may be a large number of similar looking pixels in the right image. Second, in cases when there is occlusion, there may not even exist a corresponding pixel. Nonetheless, humans are able to very accurately estimate depth using stereopsis, and a natural question is whether we can automate the process on a computer.

Humans use a large amount of prior knowledge when performing stereopsis. For example, we may expect continuity of depth for two neighboring pixels with constant color (intensity), with changes in depth occurring near the edges of objects. We also have prior knowledge of what objects and shapes exist in the world. For example, if we see a person sitting on a couch, we know that the couch does not simply disappear behind the person. Thus, we have a prior *model* of the world that we use in interpreting what we see. The process of taking in evidence, accounting for prior beliefs, and making a prediction, is called *inference*. Humans do it pretty well, so why not computers? Designing algorithms for automated inference is a fundamental problem in artificial intelligence.

Graphical models provide a mathematical means of specifying prior beliefs in such a way that we can design algorithms for automated inference. A graphical model is specified by an undirected graph where we have one node for every random variable, and edges denote explicit dependencies between variables. We can model stereo vision as a graphical model (see Figure 1-1) where we have one variable per pixel location in the left image, whose states denote the disparity to the corresponding pixel in the right image (Tappen & Freeman, 2003). For each random variable and for each of its possible states (corresponding to a particular disparity), the model specifies a local cost that is based on the intensity differences between the two pixels. The model also specifies a pairwise cost that penalizes for neighboring pixels having different disparities. The penalty is larger for pairs of pixels

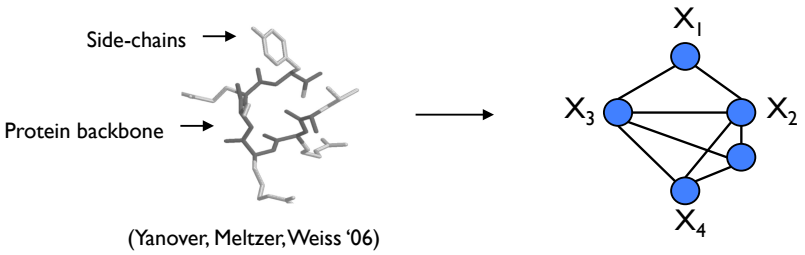


Figure 1-2: Example of a graphical model used for protein design.

that have a small intensity difference, as these are intuitively more likely to be from the same object (with identical depth). The graphical model then uses these costs to specify a probability distribution over the states of the random variables.

1.1.2 Protein Design

Many problems in structural biology can be formulated as graphical models. The protein design problem is to design a protein that will fold into some desired structure. This is important for antibody design, for improving the stability and shelf-life of drugs, and for designing new drugs. A protein consists of a sequence of amino acids that are joined together to form what is called the backbone structure. While part of each amino acid is used to form the backbone, another part, called the *side-chain*, protrudes from the backbone. It is the interactions between the side-chains that gives the protein its overall conformation.

We show in Figure 1-2 a simple graphical model that can be used to solve the protein design problem (Yanover *et al.*, 2006). Assume that we are handed the target backbone structure, and our goal is to choose amino acids for each position along the backbone such that the overall fold, taking into consideration the side-chain placements, is as stable as possible. We have one variable for each position, where each state specifies both a choice of amino acid and the angular orientation of its side-chain relative to the backbone structure. Our model corresponds to an energy function, taking into consideration both attractive and repulsive forces, that tells us the likelihood of any particular configuration of the side-chains. A frequently made assumption is that the energy function decomposes as the sum of pairwise energy terms that only consider the interaction of two side-chains at a time. Amino acids that are far from one another with respect to the backbone are assumed not to interact, and thus we do not have an edge between them in the graphical model.

1.2 The Challenge of Inference

To perform inference means to reason about the underlying state of the random variables in our model. There are many different ways to quantify the model's uncertainty in any particular assignment. For example, one important inference problem is to compute the marginal probability of a variable being in any one of its states, ignoring the states of all the other variables. Computing marginals involves summing over the probability assigned by the model to all of the exponentially many assignments to the other variables.

Another fundamental inference problem underlying many applications of graphical models is to find the most likely configuration of the probability distribution, called the maximum a posteriori (MAP) assignment. This is typically the inference problem that we

are most interested in when trying to solve the stereo vision and protein design problems described in the previous section. This thesis is primarily focused on solving the MAP inference problem.

The difficulty in designing algorithms for inference is in finding an efficient way to reason about the large number of possible assignments to the variables in the model. Although inference can be exactly performed in polynomial time for some graphical models that have particularly simple structure, such as low treewidth, most graphical models arising from real-world applications – for example, the stereo vision and the protein design problems – do not have such structure. In terms of worst-case theoretical guarantees, inference is NP-hard, and in many cases cannot even be approximated (Shimony, 1994).

However, as we motivated earlier with regards to stereo vision, it is not at all clear that real-world inference problems are as difficult as the theoretical worst case. Humans can estimate depth from stereo images both accurately and quickly. Although the graph structure of a graphical model may appear complex, the parameters that define the model may have significant structure that can be exploited to efficiently perform inference. Thus, a key problem is to design algorithms that can take advantage of this structure in performing inference.

The graphical models that we consider in this thesis involve discrete random variables. In this setting, MAP inference can be cast as an integer linear program. There are many ways to try to solve the MAP inference problem, from local search in the space of assignments, to optimization. One particularly successful class of approximate inference algorithms are called *message passing* algorithms (Yedidia *et al.*, 2005). Message passing algorithms solve the inference problem by passing messages along edges of the graph that summarize each variable's beliefs. After a node receives messages from its neighbors, it updates its belief about the uncertainty in its own assignment, and then propagates this to the rest of the graph. An assignment can then be *decoded* from the beliefs by choosing the most likely state according to each node's belief. The algorithms are very simple to implement and run quickly, making them applicable to very large-scale inference tasks. However, these algorithms can have trouble converging, and in difficult inference problems tend not to give good results.

1.3 Inference using Linear Programming Relaxations

This thesis focuses on a class of approximate inference algorithms based on linear programming (LP) relaxations. The LP relaxation is obtained by formulating inference as an integer linear program and then *relaxing* the integrality constraints on the variables. The linear program has variables for each node that specify its marginal distribution, and variables for each edge in the graphical model that specify the edge's marginal distribution.

Most previous linear programming approaches to approximate inference optimize over the *pairwise LP relaxation*, which enforces that the edge marginals are consistent with the single node marginals. If the solution to the pairwise LP relaxation is integral then it can be shown to be the MAP solution. In this case, we say that the LP relaxation is *tight*. Otherwise, since the relaxation optimizes over a larger solution space, its optimum provides an upper bound on the value of the MAP assignment.

For the protein design problem, and many others, the complex model and repulsive energy terms generally result in fractional, or inexact, solutions. These situations arise because there are *frustrated* cycles where a fractional solution can obtain a higher objective

value by letting the edge marginals along the cycle be globally inconsistent.

We can try to avoid these fractional solutions by instead optimizing over a *tighter* LP relaxation. In particular, we would like to construct a relaxation that is close to the *marginal polytope*, the exact (but very large) polyhedral formulation of the MAP inference problem. There are well-known methods for *tightening* relaxations, such as the Sherali-Adams hierarchy of relaxations which uses cluster consistency constraints to enforce the consistency of all edge marginals in a cluster, for all clusters of some fixed size. However, these approaches are typically computationally infeasible to solve because they tighten the relaxation uniformly across all of the problem. For graphical models of even moderate size, even the first lifting of the Sherali-Adams hierarchy is too slow to optimize over. The difficulty is finding the right trade-off between computation and accuracy. Stated another way, the whole game is in finding the right relaxation to solve.

The LP relaxation only needs to be tight in the vicinity of the optimal solution, not for all possible problem instances. Thus, if we had a way of efficiently searching for constraints to use in tightening the relaxation, we could use this to tighten the relaxation only where it matters.

In this thesis, we develop theory and algorithms that allow us to tighten the relaxation in a problem-specific way, using additional computation just for the hard parts of each instance. The general approach is to iteratively tighten the relaxation, adding valid constraints one-by-one to improve accuracy. After solving the initial LP relaxation, if the solution is fractional, we look for constraints that are violated by the solution and add these constraints to the relaxation. Then we re-solve using the tighter relaxation, and repeat. For this to succeed, we need to solve the *search* problem of finding which constraints to add to the relaxation. We also need a fast way of solving each of the LPs.

In summary, the key problems are: finding the right set of constraints to use in tightening the relaxation; giving algorithms to quickly solve the LPs; and solving the search problem of finding good constraints from within this set to use in tightening the relaxation.

1.4 Summary of Contributions

In what follows, we summarize the main contributions of this thesis, addressing many of the key problems raised in the previous section.

1. **We introduce the cycle relaxation of the marginal polytope. (Chapter 3)**

   The cycle relaxation is analogous to the pairwise relaxation, except that it enforces the consistency of cycles with edges, rather than edges with nodes. The cycle relaxation resolves frustration along cycles of the graph, and is significantly tighter than the pairwise relaxation. For graphical models on binary variables, we show that the cycle relaxation, the cycle inequalities, and the first lifting of the Sherali-Adams hierarchy are all equivalent.

   We give a method of deriving valid constraints for the marginal polytope from any constraint that is known for the cut polytope. Our key realization is that valid constraints can be constructed by a series of projections onto the cut polytope. Using this technique, we derive the k-ary cycle inequalities, an exponentially large class of valid constraints for the marginal polytope of non-binary graphical models.

   We show how to efficiently solve the separation problem of finding the most violated k-ary cycle inequality, by computing shortest paths in a graph. We also show how

these constraints are related to, but slightly weaker than, the cycle relaxation.

2. **We show how to solve the LP relaxations using message passing algorithms. (Chapter 4)**

   We give a general framework for understanding and deriving message passing algorithms based on block coordinate descent in the dual of the LP relaxation. We introduce a new technique for proving that many dual formulations are equivalent, by using monotonic transformations. We give a new interpretation of the MPLP algorithm (Globerson & Jaakkola, 2007b) using this framework, providing insight into why it solves inference problems faster than previous approaches such as max-sum diffusion (Schlesinger *et al.*, 1976). Also using this framework, we show how a slight change to the TRW algorithm (Wainwright *et al.*, 2005a) makes it monotonic and convergent. The resulting algorithm is well-suited for distributed computation of inference.

   We show how to use such combinatorial algorithms within message passing algorithms to solve tractable subproblems. For example, a spanning tree in a graphical model could be solved using dynamic programming. Our formulation allows one to make use of all tractable subproblems (e.g., all spanning trees) during inference, rather than having to fix a decomposition into subproblems before performing inference.

   We also characterize when it is possible to decode the MAP assignment from the beliefs of any LP-based message passing algorithm. Our results provide insight into the limitations of algorithms even outside of this class. In particular, we show that convex BP (Weiss *et al.*, 2007), which does not explicitly solve a LP relaxation, can only find the MAP assignment if the pairwise LP relaxation is tight.

3. **We introduce a criterion to use together with message passing to decide *which* clusters to use in tightening the relaxation. (Chapters 5 & 6)**

   We give a greedy bound minimization criterion to solve the cluster selection problem. The criterion chooses to add the cluster that would decrease the dual objective the most on the next coordinate-descent step. We use this within an algorithm that iteratively tightens the LP relaxation directly in the dual LP, alternating between message passing and choosing new clusters to add to the relaxation.

   Our approach makes tightening the pairwise relaxation practical for the first time. Surprisingly, we find that for many real-world problems, adding a few well-chosen constraints allows us to *exactly* find the MAP assignment. Using these techniques, we are able to solve protein side-chain placement, protein design, and stereo vision problems.

   We also propose a new class of cluster consistency constraints that are suitable for graphical models where the variables have large state spaces, and show how to use them together with this algorithm. By partitioning the state space of a cluster and enforcing consistency only across partitions, we obtain a class of constraints that, although less tight, are computationally feasible for large problems.

4. **We solve the search problem for cycle constraints. (Chapter 7)**

   We give an algorithm that, in nearly linear time, finds the best cycle of arbitrary length to add to the relaxation, according to the greedy bound criterion suggested above. We obtain the result by considering the problem of finding a frustrated cycle using the criterion applied to k-ary cycle inequalities, rather than consistency constraints.

This turns out to be substantially easier, allowing us to efficiently solve the search problem. We show how to use this search algorithm within the message passing algorithms described earlier.

We also consider the computational complexity of the search problem for consistency constraints. We show that, for graphical models involving just binary variables, and when the LP relaxation with the existing constraints has been solved to optimality, the bound criterion for both the cycle inequalities and for the consistency constraints are equivalent. Thus, under these conditions, we also obtain an efficient search algorithm for consistency constraints. We show that, in all other cases, the search problem with respect to consistency constraints is NP-hard.

Chapter 2

Background

2.1 Markov Random Fields

Let $\mathbf{x} \in \chi^n$ denote an assignment to n random variables $X_1, \ldots X_n$, where each variable X_i has k discrete states, $\chi_i = \{0, 1, \ldots, k-1\}$. A Markov random field is a joint distribution on $X_1, \ldots, X_n$ that is specified by a vector of d real-valued *sufficient statistics* $\phi_l(\mathbf{x})$ for $l = 1, \ldots, d$, and a parameter vector $\boldsymbol{\theta} \in \mathbb{R}^d$:

$$\Pr(\mathbf{x}; \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \exp\Big(\langle \boldsymbol{\theta}, \phi(\mathbf{x}) \rangle\Big), \quad Z(\boldsymbol{\theta}) = \sum_{\mathbf{x} \in \chi^n} \exp\Big(\langle \boldsymbol{\theta}, \phi(\mathbf{x}) \rangle\Big) \tag{2.1}$$

where $\langle \boldsymbol{\theta}, \phi(\mathbf{x}) \rangle$ denotes the dot product of the parameters and the sufficient statistics. $Z(\boldsymbol{\theta})$ is the normalization constant, also called the *partition function*.

In this thesis, we focus on *pairwise* Markov random fields (MRFs), given by a graph $G = (V, E)$ with vertices V and edges E. In pairwise MRFs, the sufficient statistics are restricted to be only over the nodes and edges of the graph. In the most general form, they are indicator functions denoting local assignments to nodes and edges,[1] i.e. $\phi_{i,x_i}(\mathbf{x}) = 1[X_i = x_i]$ and $\phi_{ij;x_i x_j}(\mathbf{x}) = 1[X_i = x_i, X_j = x_j]$. We can combine the parameter vectors and the sufficient statistics to obtain the *potential function* for an edge, $\theta_{ij}(x_i, x_j)$, which is a function $\chi_i \times \chi_j \to \mathbb{R}$. Using this notation, the joint distribution for a pairwise MRF can be written more simply as

$$\Pr(\mathbf{x}; \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \exp\left(\sum_{ij \in E} \theta_{ij}(x_i, x_j) + \sum_{i \in V} \theta_i(x_i)\right). \tag{2.2}$$

For pairwise MRFs, the dimension d of the sufficient statistic vector $\phi(\mathbf{x})$ is $d = \sum_{i \in V} |\chi_i| + \sum_{ij \in E} |\chi_i||\chi_j|$, or simply $k|V| + k^2|E|$ when each variable has exactly k states.

Although we mainly consider pairwise MRFs, we note that any graphical model can be converted into a pairwise MRF by introducing new variables and enforcing that they are consistent with the original variables (Wainwright & Jordan, 2008, p.289). However, the state spaces of the new variables can be very large. In some cases, more efficient algorithms can be obtained by taking advantage of special structure in higher-order MRFs (see Section 2.4).

[1] Notation: Although the edges are undirected, we interpret the edge $ij \in E$ as having a canonical ordering of the variables. Also, the indicator function $1[X_i = x_i]$ takes value 1 if $X_i = x_i$, and is 0 otherwise.

This thesis concerns algorithms for solving the *maximum a posteriori* (MAP) inference problem of finding the most likely assignment to the variables in the graphical model. Both the protein design and stereo vision problems that we discussed in Chapter 1 use MAP inference to make predictions. Since the partition function is a constant and the exponential is monotonic, finding the MAP assignment is equivalent to finding the assignment $\mathbf{x}_M$ that maximizes

$$\theta(\mathbf{x}) = \sum_{ij \in E} \theta_{ij}(x_i, x_j) + \sum_{i \in V} \theta_i(x_i). \tag{2.3}$$

This is a difficult discrete optimization problem and, in general, is NP-complete (see Section 2.3).

As we mentioned in the introduction, there are many other types of inference questions. For example, we may want to compute the probability of an assignment $\mathbf{x}$ under the model. This is important for evaluating the *likelihood* of data under the model, and is useful for model selection, hypothesis testing, and learning. Although the numerator of Eq. 2.2 is easily calculated, the partition function $Z(\boldsymbol{\theta})$ is substantially more difficult, corresponding to a summation over all possible assignments to the model. Calculating $Z(\boldsymbol{\theta})$ can be shown to be #P-hard, although fully-polynomial approximation algorithms do exist for some special cases (Jerrum & Sinclair, 1993).

We may also be interested in computing *marginal probabilities* for the variables in the model. We use the following notation to refer to the single node and edge marginals:

$$\mu_i(x_i) = \mathbb{E}_{\boldsymbol{\theta}}[\phi_{i;x_i}(\mathbf{x})] = \Pr(X_i = x_i; \boldsymbol{\theta}) \tag{2.4}$$

$$\mu_{ij}(x_i, x_j) = \mathbb{E}_{\boldsymbol{\theta}}[\phi_{ij;x_i x_j}(\mathbf{x})] = \Pr(X_i = x_i, X_j = x_j; \boldsymbol{\theta}). \tag{2.5}$$

We show in Section 2.2.2 how the techniques that we develop in the thesis are also applicable to the inference problems of estimating the partition function and marginals.

2.1.1 Ising Models

The *Ising model* is a special case of Markov random fields where the variables have only two states and the model is parameterized in terms of edge agreements and disagreements. Here, the sufficient statistics are given by the indicator function $x_{ij} = 1$ if $x_i = x_j$, and $x_{ij} = -1$ if $x_i \neq x_j$. We have one parameter $\lambda_{ij} \in \mathbb{R}$ for each edge of the model. The joint distribution of an Ising model with no external field is then

$$\Pr(\mathbf{x}; \boldsymbol{\lambda}) = \frac{1}{Z(\boldsymbol{\lambda})} \exp\left(\sum_{ij \in E} \lambda_{ij} x_{ij} \right). \tag{2.6}$$

Ising models and their non-binary extensions (called *Potts models*) were originally used in statistical mechanics to study magnetism in materials (beginning in the early 1920s). Their study led to the development of many of the algorithmic techniques that we will discuss in this thesis, such as the work of Barahona. It is straightforward to show that any binary pairwise MRF can be transformed into an equivalent Ising model (Sontag, 2007). Also, finding the most likely assignment in an Ising model is equivalent to a cut problem. For example, if $\boldsymbol{\lambda} \leq 0$, then the MAP assignment for graph $G = (V, E)$ corresponds to the maximum cut of G with edge weights $\boldsymbol{\lambda}$.

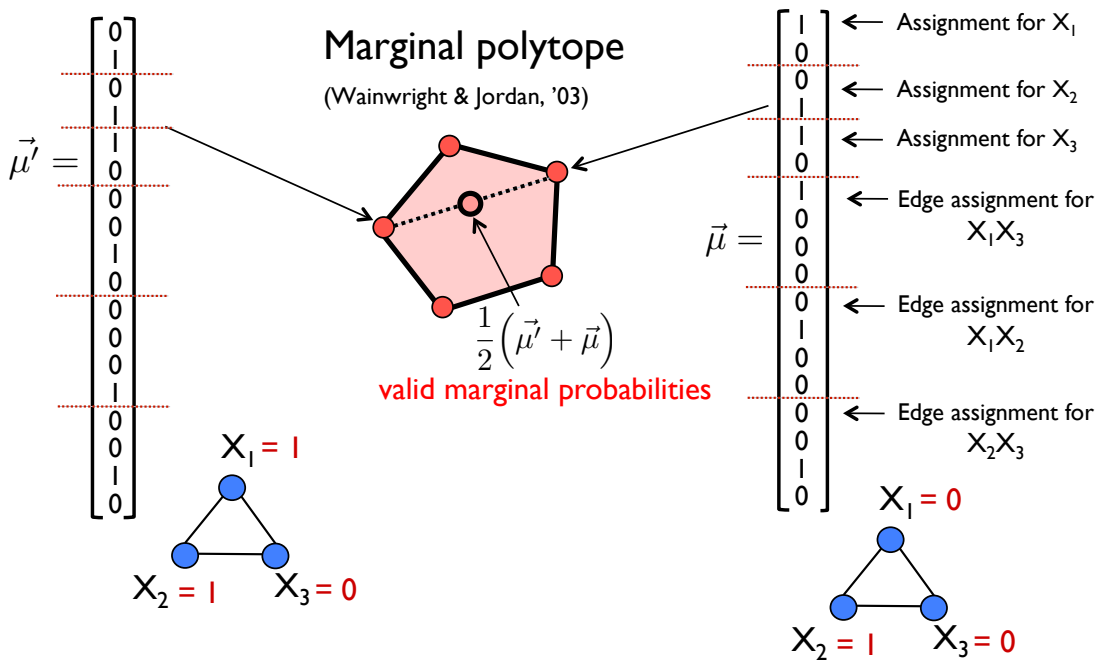


Figure 2-1: Illustration of the marginal polytope for a Markov random field with three nodes that have states in $\{0, 1\}$. The vertices correspond one-to-one with global assignments to the variables in the MRF. The marginal polytope is alternatively defined as the convex hull of these vertices, where each vertex is obtained by stacking the node indicator vectors and the edge indicator vectors for the corresponding assignment.

2.2 The Marginal Polytope

At the core of our approach is an equivalent formulation of inference problems in terms of an optimization over the *marginal polytope*. The marginal polytope is the set of realizable mean vectors $\boldsymbol{\mu}$ that can arise from some joint distribution on the graphical model:

$$\mathcal{M}(G) \quad = \quad \left\{ \boldsymbol{\mu} \in \mathbb{R}^d \mid \exists\, \boldsymbol{\theta} \in \mathbb{R}^d \text{ s.t. } \boldsymbol{\mu} = \mathbb{E}_{\Pr(\mathbf{x};\boldsymbol{\theta})}[\phi(\mathbf{x})] \right\} \tag{2.7}$$

Said another way, the marginal polytope is the convex hull of the $\phi(\mathbf{x})$ vectors, one for each assignment $\mathbf{x} \in \chi^n$ to the variables of the Markov random field. The dimension d of $\phi(\mathbf{x})$ is a function of the particular graphical model. In pairwise MRFs where each variable has k states, each variable assignment contributes k coordinates to $\phi(\mathbf{x})$ and each edge assignment contributes k^2 coordinates to $\phi(\mathbf{x})$. Thus, $\phi(\mathbf{x})$ will be of dimension $k|V| + k^2|E|$.

We illustrate the marginal polytope in Figure 2-1 for a binary-valued Markov random field on three nodes. In this case, $\phi(\mathbf{x})$ is of dimension $2 \cdot 3 + 2^2 \cdot 3 = 18$. The figure shows two vertices corresponding to the assignments $\mathbf{x} = (1, 1, 0)$ and $\mathbf{x}' = (0, 1, 0)$. The vector $\phi(\mathbf{x})$ is obtained by stacking the node indicator vectors for each of the three nodes, and then the edge indicator vectors for each of the three edges. $\phi(\mathbf{x}')$ is analogous. There should be a total of 9 vertices (the 2-dimensional sketch is inaccurate in this respect), one for each assignment to the MRF.

Any point inside the marginal polytope corresponds to the vector of node and edge marginals for some graphical model with the same sufficient statistics. By construction, the

vertices, also called extreme points, are one-to-one with assignments to the MRF. These are marginal vectors arising from delta distributions. We call a point $\boldsymbol{\mu}$ *integral* if $\boldsymbol{\mu} \in \mathbb{Z}^d$. The only integral points in the marginal polytope are its vertices.

2.2.1 Maximum a Posteriori Estimation

We can now formulate the MAP problem as a linear program over the marginal polytope,

$$\max_{\mathbf{x} \in \chi^n} \langle \boldsymbol{\theta}, \phi(\mathbf{x}) \rangle = \max_{\boldsymbol{y} \in \left\{ \phi(\mathbf{x}) \,:\, \mathbf{x} \in \chi^n \right\}} \langle \boldsymbol{\theta}, \boldsymbol{y} \rangle = \max_{\boldsymbol{\mu} \in \mathcal{M}(G)} \langle \boldsymbol{\theta}, \boldsymbol{\mu} \rangle. \tag{2.8}$$

The first equality holds because the sufficient statistics vector $\phi(\mathbf{x})$ is 1-1 with assignments $\mathbf{x}$. The marginal polytope $\mathcal{M}(G)$ is the convex hull of the discrete set $\left\{ \phi(\mathbf{x}) \;:\; \mathbf{x} \in \chi^n \right\}$. The second equality holds because the optimal value of a linear program over a polytope can be shown to be obtained by one of its vertices. When the MAP assignment $\mathbf{x}^*$ is *unique*, the maximizing $\boldsymbol{\mu}^*$ is equal to $\phi(\mathbf{x}^*)$.

We show in Section 2.3 that a large number of NP-hard combinatorial optimization problems (e.g., maximum cut) can be posed as finding the MAP assignment in a Markov random field. Thus, unless P=NP, in general there will be a superpolynomial number of constraints that define the marginal polytope (clearly there are also an exponential number of vertices, one for each assignment), and we cannot hope to optimize over it efficiently.

2.2.2 Variational Inference

The inference task of calculating marginals is to evaluate the marginal vector $\boldsymbol{\mu} = \mathbb{E}_{\boldsymbol{\theta}}[\phi(\mathbf{x})]$. The log-partition function $\log Z(\boldsymbol{\theta})$, a convex function of $\boldsymbol{\theta}$, plays a critical role in these calculations. In particular, we can write the log-partition function in terms of its Fenchel-Legendre conjugate (Wainwright & Jordan, 2008):

$$\log Z(\boldsymbol{\theta}) = \sup_{\boldsymbol{\mu} \in \mathcal{M}(G)} \left\{ \langle \boldsymbol{\theta}, \boldsymbol{\mu} \rangle + H(\boldsymbol{\mu}) \right\}, \tag{2.9}$$

where $H(\boldsymbol{\mu})$ is the entropy of the maximum entropy distribution with marginals $\boldsymbol{\mu}$, and is also convex. The value $\boldsymbol{\mu}^* \in \mathcal{M}(G)$ that maximizes Eq. 2.9 is precisely the desired marginal vector corresponding to $\Pr(\mathbf{x}; \boldsymbol{\theta})$.

It is illustrative to compare Eq. 2.9, the optimization problem that calculates both the partition function and the marginals, with Eq. 2.8, the optimization problem to find the MAP assignment. The only difference is that Eq. 2.9 has an additional non-linear entropy term in the objective. Both $\mathcal{M}(G)$ and the entropy $H(\boldsymbol{\mu})$ are difficult to characterize in general and have to be approximated. We call the resulting approximate marginal vectors – which may not be in the marginal polytope – *pseudomarginals*.

Many well-known approximate inference algorithms can be interpreted as making some approximation to Eq. 2.9. For example, mean field algorithms optimize over a non-convex *inner bound* on the marginal polytope by restricting the marginal vectors to those coming from simpler, e.g., fully factored, distributions. The entropy can be evaluated exactly in this case (the distribution is simple).

Alternatively, we can relax the optimization to be over an *outer bound* on the marginal polytope and also bound the entropy function. Most message passing algorithms for computing marginal probabilities optimize over pseudomarginals that are locally consistent,

enforcing only that the edge marginals are consistent with the single node marginals. In Section 2.5.1 we introduce the pairwise LP relaxation which has precisely these constraints.

Belief propagation can be seen as optimizing pseudomarginals over the pairwise relaxation with the (non-convex) Bethe approximation to the entropy (Yedidia *et al.*, 2001). The tree-reweighted sum-product algorithm (Wainwright *et al.*, 2005b), on the other hand, uses a concave upper bound on the entropy, expressed as a convex combination of entropies corresponding to the spanning trees of the original graph. The log-determinant relaxation (Wainwright & Jordan, 2006) is instead based on a semi-definite outer bound on the marginal polytope combined with a Gaussian approximation to the entropy function.

For the remainder of this thesis we will focus on the MAP problem, giving both efficient algorithms for solving its linear programming relaxation and methods for tightening the relaxation. However, all of the techniques that we develop are also applicable to the problems of approximating marginals and the partition function, with the distinction that the convex optimization would involve an objective which is non-linear (including the additional entropy term) instead of linear. See Sontag (2007) and Wainwright & Jordan (2008) for more details.

2.3 Inference as Combinatorial Optimization

A large number of well-studied combinatorial optimization problems can be posed as MAP inference in a graphical model. In this section we discuss a few of these related problems, illustrating the hardness of the general MAP problem, inapproximability results, and techniques that may be more broadly useful. The algorithms that we develop in this thesis to solve the MAP inference problem also apply to all of the combinatorial optimization problems that we discuss here, and thus have implications well beyond machine learning and probabilistic inference.

We begin by considering the MAP inference problem in pairwise Markov random fields with binary variables. This problem is equivalent to quadratic boolean optimization; see Boros & Hammer (2002) for a survey of pseudo-boolean optimization. One particularly important special case of quadratic boolean optimization is the maximum cut problem, which corresponds to finding the MAP assignment in an Ising model with antiferromagnetic potentials, or negative edge weights. Maximum cut is known to be NP-complete, and, furthermore, is hard to approximate better than $\frac{16}{17}$OPT (Trevisan *et al.*, 2000). Goemans & Williamson (1995) give a .878-approximation algorithm which is based on solving a semi-definite relaxation of the marginal polytope. Assuming the unique games conjecture, this is the best possible polynomial time approximation ratio, unless P=NP (Khot *et al.*, 2004). We discuss integrality gaps for the relaxations and more in Section 2.5.2.

Another special case of MAP inference in pairwise MRFs with binary variables is correlation clustering (Bansal *et al.*, 2002). Given a graph and similarity or dissimilarity measurements along each edge, correlation clustering attempts to partition the vertices into clusters. This task is motivated by clustering problems in machine learning, and was further studied in the theoretical computer science community. The difficulty with applying the approximation algorithm for maximum cut to this problem is that the edge weights can be both positive and negative, which affects the analysis of the rounding algorithm.

Partially motivated by the correlation clustering problem, Charikar & Wirth (2004) study MaxQP, which is the same as the MAP problem in Ising models with arbitrary potentials, and thus is much more closely aligned with real-world MAP inference problems.

The authors give an $\Omega(1/\log n)$ approximation algorithm, where n is the number of variables in the MRF, also based on semi-definite programming. This approach is closely related to the study of Grothendieck's inequality (Alon & Naor, 2004; Alon *et al.*, 2005).

On the positive side, MAP inference in Ising models with ferromagnetic potentials, or positive edge weights, is equivalent to the *minimum cut* problem, which can be solved in nearly linear time (Karger, 2000).[2] This special case of Markov random fields was first observed by Greig *et al.* (1989), and has become extremely influential in computer vision where such edge potentials are commonly used image segmentation, among other problems (Boykov & Kolmogorov, 2004). There has been significant interest in the computer vision and machine learning communities in using these efficient combinatorial algorithms for non-binary MRFs. For example, Boykov *et al.* (2001) showed how to use local search together with graph cuts to approximately find the MAP assignment in non-binary MRFs.

The metric labeling problem (Kleinberg & Tardos, 1999) is an instance of MAP in a non-binary MRF where there is a metric space on the variables' states and the edge potentials are proportional to the distance according to this metric. Metric labeling generalizes several combinatorial optimization problems such as multiway cut (Dahlhaus *et al.*, 1994). Metric labeling has a $O(1/\log k)$-approximation, where k is the number of states per variable, which can be obtained using either combinatorial algorithms or an LP relaxation (Kleinberg & Tardos, 1999; Chekuri *et al.*, 2005). Two interesting special cases of metric labeling, corresponding to linearly ordered states and tree metrics, can be solved exactly in polynomial time (Ishikawa, 2003; Felzenszwalb *et al.*, 2010).

In the quadratic assignment problem (QAP) we are given two $n \times n$ matrices $A = (a_{ij})$ and $B = (b_{ij})$, and the goal is to find a permutation π of $1, \ldots, n$ that minimizes $\sum_{i=1}^{n} \sum_{j=1}^{n} a_{\pi(i),\pi(j)} b_{ij}$. QAP can be posed as MAP in a non-binary MRF with n variables $X_1, \ldots, X_n$, each having n states. To enforce a valid permutation the potential $\theta_{ij}(x_i, x_j) = -\infty$ if $x_i = x_j$, and otherwise is $\theta_{ij}(x_i, x_j) = a_{x_i,x_j} b_{ij}$. Generalizations of QAP have many applications in machine learning, such as statistical machine translation where it has been used as a model of word alignment (Lacoste-Julien *et al.*, 2006). Closely related is the facility location problem; see Lazic *et al.* (2010) for a recent paper on facility location that applies many of the techniques developed in this thesis.

The protein side-chain placement problem, which we discuss much further in this thesis, can be formulated as a non-binary pairwise MRF. Methods to solve this optimization problem have been studied in the computational chemistry literature. One of the most effective methods is called *dead-end elimination*, and corresponds to iteratively applying local rules that attempt to rule out variables' states as not being part of the MAP assignment (Goldstein, 1994; Pierce *et al.*, 2000).

Finally, the decoding problem for low-density parity-check codes (LDPCs), and other codes, corresponds to MAP inference in a non-pairwise Markov random field with binary variables (see, e.g., Mceliece *et al.*, 1998). An iterative algorithm called belief propagation, discussed further in the next section, was found to be surprisingly effective at these inference problems, and spurred a significant amount of research in the theoretical computer science community. Linear programming relaxations can also be shown to give provably good information-theoretic rates (Feldman *et al.*, 2005; Vontobel & Koetter, 2006; Arora *et al.*,

[2] MAP inference in Ising models with arbitrary external fields, or node potentials, can be transformed into an s-t minimum cut problem, solvable in polynomial time, where the new terminals s and t are connected to all of the original nodes. The weights of edges to s are given by the positive fields, and the weights of edges to t are given by the (absolute value of the) negative fields.

2009). Several authors have considered the problem of tightening these LP relaxations; see, for example, Dimakis *et al.* (2009). We note, however, that the constraints used by Dimakis *et al.* (2009) are specifically for this coding problem and are not more broadly applicable to other MRFs.

As we mentioned, the Goemans-Williamson approximation algorithm for maximum cut is based on semi-definite programming. Recently, several authors have considered fast algorithms for obtaining the same approximation guarantee, without having to solve the SDP to optimality (Arora & Kale, 2007; Trevisan, 2009). Arora *et al.* (2005) give algorithms based on multiplicative weights to solve the SDP relaxation of MaxQP. Some MAP inference problems can be transformed into packing problems, where the fractional packing LP can be shown to be equivalent to the pairwise LP relaxation (see Section 2.5.1). In these cases, the fractional packing problem can be approximately solved using combinatorial algorithms (Plotkin *et al.*, 1995; Awerbuch & Khandekar, 2008). It would be extremely interesting to obtain combinatorial algorithms that could efficiently solve LP or SDP relaxations for non-binary pairwise MRFs more generally.

2.4 Belief Propagation

We saw in the previous section that inference in graphical models corresponds to solving a large class of integer linear programs. Many special cases can be solved efficiently, such as the minimum cut and maximum weight matching problems. One desirable property for an approximate inference algorithm is that it should be able to solve simple cases exactly, such as graphical models that are tree-structured.

Belief propagation (BP) is a heuristic for approximate inference that is simple to code and scales very well with problem size. BP is an example of a *message-passing algorithm*, and works by iteratively passing messages along edges of the graph. On tree-structured graphical models, BP can be seen to be equivalent to a dynamic programming algorithm for inference, and thus solves the inference problem exactly.[3] We will be primarily concerned with the *max-product* belief propagation algorithm in this thesis, which approximates the MAP inference problem.

Although belief propagation is not guaranteed to give exact results on graphical models with cycles, it has often been observed empirically to give excellent results. Much research has been devoted to understanding its empirical success. For example, Weiss (1997) shows that, for a graphical model that only has a single cycle, when BP converges (it might not) it gives exact results. Bayati *et al.* (2008) show that BP gives exact results when applied to maximum weight bipartite matching problems that have a unique solution.

However, in many problems, belief propagation may not even converge to a solution. Murphy *et al.* (1999) observe that convergence problems are highly correlated with BP not finding a good solution to the MAP problem. Intuitively the problem is that of "double counting," in which evidence is passed around the graph multiple times without being recognized as the same evidence already considered in previous belief updates. The problem is only compounded in graphs with short cycles. There have been several attempts to improve BP's accuracy, most notably by the generalized Belief Propagation (GBP) algorithm (Yedidia *et al.*, 2005).

[3]The following two-pass schedule corresponds to the dynamic programming algorithm: Send messages from the leaves to the root, and then from the root back to the leaves.

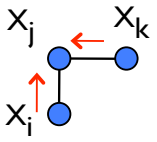


Figure 2-2: The pairwise consistency constraints ensure that two edges that have a node in common must have edge marginal distributions that are consistent on that node.

The approach pursued in this thesis is to directly solve a linear programming relaxation of the MAP inference problem. For all of the examples above where BP is known to do well – tree structured graphical models and inference problems arising from minimum cut and matchings – the LP relaxation also gives provably exact results. In Chapter 4 we show that the LP relaxation can be solved by a message-passing algorithm that is very similar to belief propagation. Another advantage of the BP heuristic is that it can be used together with combinatorial algorithms (Duchi *et al.*, 2007; Gupta *et al.*, 2007). We will show that the same is true for the linear programming approaches discussed in this thesis.

2.5 Linear Programming Relaxations

We discussed in Section 2.2.2 how the problems of approximating the partition function, estimating marginal probabilities, and finding the MAP assignment in graphical models can be formulated as (non-)linear optimization over the marginal polytope. Since optimizing over the marginal polytope is difficult, we could instead try *relaxing* the marginal polytope, optimizing over only a small number of its constraints. As we discussed in Section 2.3, linear programming relaxations have long been studied in combinatorial optimization, and often lead to approximation guarantees.

The marginal polytope is defined by the difficult global constraint that the edge marginals in $\boldsymbol{\mu}$ must arise from some common joint distribution. The LP relaxations presented in this section relax this global constraint, instead enforcing it only over some subsets of the variables.

2.5.1 Pairwise Relaxation

The pairwise LP relaxation, also known as the first-order relaxation, has constraints that enforce that the marginals for every pair of edges that share a variable are consistent with each other on that variable (see Figure 2-2). These *pairwise consistency constraints*, also called local consistency constraints, are given by:

$$\text{LOCAL}(G) = \left\{ \boldsymbol{\mu} \in \mathbb{R}^d \;\middle|\; \begin{array}{ll} \sum_{x_j} \mu_{ij}(x_i, x_j) = \mu_i(x_i) & \forall ij \in E, x_i \\ \sum_{x_i} \mu_{ij}(x_i, x_j) = \mu_j(x_j) & \forall ij \in E, x_j \\ \sum_{x_i} \mu_i(x_i) = 1 & \forall i \in V \\ \mu_i(x_i) \geq 0, \quad \mu_{ij}(x_i, x_j) \geq 0 & \end{array} \right\} \tag{2.10}$$

The pairwise LP relaxation is then given by the following optimization problem:

$$\max_{\boldsymbol{\mu} \in \text{LOCAL}(G)} \langle \boldsymbol{\theta}, \boldsymbol{\mu} \rangle \; . \tag{2.11}$$

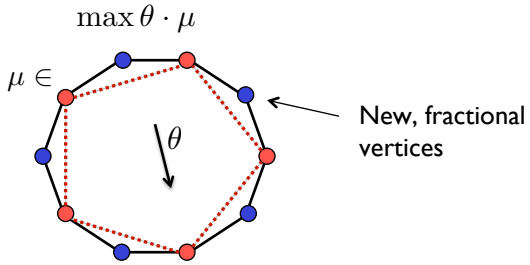


Figure 2-3: Sketch of the local consistency polytope, which is a relaxation of the marginal polytope (shown with dashed lines). Whereas the marginal polytope had only integral vertices (red), the relaxation introduces new fractional vertices (blue).

LOCAL(G) is an outer bound on $\mathcal{M}(G)$, i.e. $\mathcal{M}(G) \subseteq$ LOCAL(G), because every marginal vector $\boldsymbol{\mu}$ that is consistent with some joint distribution must satisfy these marginalization constraints. Another way to verify that these constraints are *valid* is to observe that they are satisfied by each of the integral vertices $\phi(\mathbf{x}) \in \mathcal{M}(G)$. Then, since the constraints are linear, they must also hold for any convex combination of the vertices of $\mathcal{M}(G)$, i.e. all points in $\mathcal{M}(G)$. Since we are now maximizing over a larger space, we have that $\max_{\boldsymbol{\mu} \in \mathcal{M}(G)} \langle \boldsymbol{\theta}, \boldsymbol{\mu} \rangle \leq \max_{\boldsymbol{\mu} \in \text{LOCAL}(G)} \langle \boldsymbol{\theta}, \boldsymbol{\mu} \rangle$, i.e. the relaxation provides an upper bound on the value of the MAP assignment.

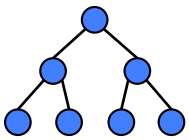
In general, the local consistency polytope, LOCAL(G), has both integral and fractional vertices (see Figure 2-3). In particular, all of the vertices of the marginal polytope are also vertices of the local consistency polytope. It is straightforward to show that there are no new integer points that satisfy the constraints of LOCAL(G). Thus, we obtain a *certificate of optimality* for this relaxation: if we solve the pairwise LP relaxation and obtain an integer solution, it is guaranteed to be a MAP assignment.

There are several classes of graphical models – corresponding both to the choice of graph structure, and the parameters – where the pairwise LP relaxation is known to be *tight*, meaning that on these instances the LP relaxation is guaranteed to have an integer solution. For example, if the graphical model has a tree structure (see Figure 2-4(a)), the junction tree theorem can be used to prove that the local consistency polytope has only integer extreme points.

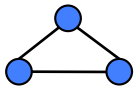
As we mentioned earlier, several well-known combinatorial optimization problems can be posed as inference in a graphical model. Consider, for example, a graphical model for finding the maximum matching on a bipartite graph. In formulating the graphical model for matching, some of the potentials $\theta_{ij}(x_i, x_j)$ have value $-\infty$. The corresponding local assignments will clearly never be optimal, so we can consider the projection of the marginal polytope onto the assignment $\mu_{ij}(x_i, x_j) = 0$. In this case, the local consistency polytope, after projection, is isomorphic to the matching polytope, and has only integer solutions.

Much is known about the MAP problem in binary-valued pairwise MRFs because this special case is equivalent to the maximum cut problem. In this case, the marginal polytope is isomorphic to the *cut polytope*, the convex hull of all valid graph cuts (Deza & Laurent, 1997; Sontag, 2007). The pairwise LP relaxation can be shown to have the following properties in this setting:

- The fractional vertices are *half integral* (Deza & Laurent, 1997). Each edge marginal



(a) Tree structured MRF



(b) MRF with attractive potentials

Figure 2-4: Examples of Markov random fields where the pairwise LP relaxation is known to have integer solutions. (a) When the graphical model decomposes according to a tree structure. (b) When the variables are binary and the potential functions are submodular, or attractive.

$\mu_{ij}(x_i, x_j)$ is either integral, or is equal to

	$x_j = 0$	$x_j = 1$
$x_i = 0$	.5	0
$x_i = 1$	0	.5

or

	$x_j = 0$	$x_j = 1$
$x_i = 0$	0	.5
$x_i = 1$	.5	0

.

See Example 1 below for a MRF where the LP relaxation has a fractional solution.

- *Persistency*, which guarantees that there exists a MAP assignment that extends the integer parts of a fractional vertex (Nemhauser & Trotter, 1975; Boros & Hammer, 2002). This can be applied to some non-binary MRFs by using a clever transformation of the problem into a binary MRF (Kohli *et al.*, 2008).

- Gives a 2-approximation when applied to the maximum cut problem.[4]

- The relaxation is tight when the edge potentials are *submodular*, meaning that there exists some labeling of the states of each variable as 0 or 1 such that, for all edges,

$$\theta_{ij}(0,0) + \theta_{ij}(1,1) \geq \theta_{ij}(1,0) + \theta_{ij}(0,1) \tag{2.12}$$

(Johnson, 2008, p.119). In Ising models, this property is typically referred to as the potentials being *attractive* or ferromagnetic. We discussed non-binary generalizations of this in Section 2.3 with regards to the metric labeling problem.

Inconsistencies cause fractional solutions

In graphs with cycles, inconsistencies can easily arise that lead to fractional solutions to the pairwise LP relaxation.

Example 1. Consider the simple three node MRF shown in Figure 2-4(b). Suppose that instead of having attractive potentials, the edge potentials were *repulsive*, e.g.

$$\theta_{ij}(x_i, x_j) = 1 \text{ if } x_i \neq x_j, \text{ and } 0 \text{ otherwise.} \tag{2.13}$$

[4]To put this in perspective: A 2-approximation for max-cut could also be obtained using a randomized algorithm that simply labels each variable 0 or 1 uniformly at random.

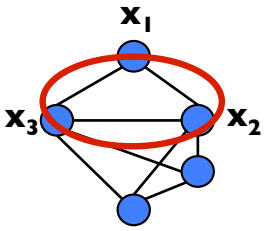


Figure 2-5: Example of a triplet cluster. A higher-order relaxation might enforce that μ_{12}, μ_{23}, and μ_{13} are consistent, meaning that they are the marginals of some joint distribution on X_1, X_2, X_3.

In this case, the pairwise LP relaxation can be shown to have the following optimal solution for each of the edge marginals:

$$\mu_{ij}(0,1) = \mu_{ij}(1,0) = .5, \qquad \mu_{ij}(0,0) = \mu_{ij}(1,1) = 0. \tag{2.14}$$

It is easy to see that the edge consistency constraints are satisfied – each of the node marginals has values $\mu_i(0) = \mu_i(1) = .5$. The pairwise relaxation has value 3, whereas any integer solution will have value at most 2.

In contrast, there are also fractional points in *in the interior* of the marginal polytope that are neither vertices of the marginal polytope nor of the local consistency polytope. For example, the point that for all edges has

$$\mu_{ij}(x_i, x_j) = \begin{array}{c|cc} & x_j = 0 & x_j = 1 \\ \hline x_i = 0 & .25 & .25 \\ x_i = 1 & .25 & .25 \end{array} \tag{2.15}$$

is fractional, but is equal to the average of all the integral vertices and is thus in the marginal polytope. Except for degenerate cases where the LP relaxation has many solutions, these benign fractional points are never optimal – one of the vertices will obtain a higher objective value.

2.5.2 Higher-Order Relaxations

Higher-order relaxations tighten the LP relaxation by enforcing that the edge marginals are consistent with one another on larger subsets of the variables. Consider, for example, the graphical model shown in Figure 2-5. The pairwise LP relaxation only enforced consistency of any pair of edges. We can tighten the relaxation by enforcing the joint consistency of edges in a *cluster* of variables, such as X_1, X_2, X_3.

The approach that we consider is an example of a *lift-and-project* method. Rather than explicitly enforcing the joint consistency of the edge marginals $\boldsymbol{\mu}_{12}, \boldsymbol{\mu}_{23}, \boldsymbol{\mu}_{13}$ using inequalities, we introduce new variables into the LP that together represent the joint distribution of X_1, X_2, X_3, and then enforce that the edge marginals are consistent with these new variables. The *lifting* refers to our introducing new variables into the relaxation, whereas the *projection* refers to the objective function only making use of the original variables, and so the actual polytope that is optimized can be understood as a projection of the lifted one.

Continuing the example, we do this by introducing new variables $\tau_{123}(x_1, x_2, x_3)$ – corresponding to the joint distribution of X_1, X_2, X_3 – into the LP, and enforcing the constraints

$$\sum_{x_1,x_2,x_3} \tau_{123}(x_1, x_2, x_3) = 1, \tag{2.16}$$

$$\sum_{x_1} \tau_{123}(x_1, x_2, x_3) = \mu_{23}(x_2, x_3), \quad \forall x_2, x_3 \tag{2.17}$$

$$\sum_{x_2} \tau_{123}(x_1, x_2, x_3) = \mu_{13}(x_1, x_3), \quad \forall x_1, x_3 \tag{2.18}$$

$$\sum_{x_3} \tau_{123}(x_1, x_2, x_3) = \mu_{12}(x_1, x_2), \quad \forall x_1, x_2 \tag{2.19}$$

in addition to the constraints that $\tau_{123}(x_1, x_2, x_3) \geq 0$ for all x_1, x_2, x_3. The non-negativity and normalization constraints together imply that $\boldsymbol{\tau}_{123}$ is some valid joint distribution over variables X_1, X_2, X_3. The last three equality constraints enforce that the edge marginals are consistent with this joint distribution.

More generally, we can consider a sequence of tighter and tighter relaxations obtained by using these *cluster consistency constraints* on all clusters of increasingly large size. Defining

$$\text{LOCAL}_t(G) = \left\{ \boldsymbol{\mu} \geq 0 \;\middle|\; \exists \{\boldsymbol{\tau}_c : c \subseteq V, \; |c| \leq t\} \begin{array}{ll} \sum_{\mathbf{x}_c} \tau_c(\mathbf{x}_c) = 1, \quad \boldsymbol{\tau}_c \geq 0 & \\ \sum_{\mathbf{x}_{c \setminus h}} \tau_c(\mathbf{x}_c) = \tau_h(\mathbf{x}_h), & \forall c, h \subseteq c, \mathbf{x}_h, \\ \sum_{\mathbf{x}_{c \setminus \{i,j\}}} \tau_c(\mathbf{x}_c) = \mu_{ij}(x_i, x_j), & \forall c \supseteq \{i, j\}, x_i, x_j \end{array} \right\}$$

to be the relaxation using clusters of size at most t, we have the containment

$$\text{LOCAL}(G) \supseteq \text{LOCAL}_3(G) \supseteq \text{LOCAL}_4(G) \supseteq \ldots \supseteq \mathcal{M}(G),$$

where the tightest relaxation explicitly represents the joint distribution $\boldsymbol{\tau}_V$ over all of the variables, and is thus equivalent to the marginal polytope. For any graph of treewidth t, $\text{LOCAL}_{t+1}(G)$ can be shown to have only integral vertices, and so this sequence will be exact earlier (Wainwright & Jordan, 2008, p.227). This sequence of relaxations is known in the polyhedral combinatorics and theoretical computer science literature as the *Sherali-Adams hierarchy* (Sherali & Adams, 1990; Wainwright & Jordan, 2004, 2008).

This thesis will be largely devoted to a study of the first lifting of the Sherali-Adams hierarchy, which we can state more simply as

$$\text{TRI}(G) = \left\{ \boldsymbol{\mu} \geq 0 \;\middle|\; \exists \boldsymbol{\tau} \geq 0, \begin{array}{ll} \sum_{x_j} \mu_{ij}(x_i, x_j) = \mu_i(x_i) & \forall ij \in E, x_i \\ \sum_{x_i} \mu_{ij}(x_i, x_j) = \mu_j(x_j) & \forall ij \in E, x_j \\ \sum_{x_i} \mu_i(x_i) = 1 & \forall i \in V \\ \sum_{x_k} \tau_{ijk}(x_i, x_j, x_k) = \mu_{ij}(x_i, x_j), & \forall i, j, k \end{array} \right\}. \tag{2.20}$$

We show in Chapter 3 that $\text{TRI}(G)$ rules out the fractional solution from Example 1.

Relationship to Sherali-Adams, Lovász-Schrijver, and Lasserre Hierarchies

Two other lift-and-project techniques that are frequently studied in integer programming are the Lovász-Schrijver and Lasserre hierarchies (Laurent, 2003; Wainwright & Jordan, 2004). In contrast to the Sherali-Adams hierarchy, these correspond to *semi-definite* outer bounds on the marginal polytope. Alizadeh (1993) gives an efficient interior-point algorithm

for solving semidefinite programs.

Let $\boldsymbol{\mu}$ be a marginal vector on K_n, the complete graph with n nodes. By definition, $\boldsymbol{\mu}$ must be the marginals of some joint distribution $\Pr(\mathbf{x}; \boldsymbol{\theta})$. Thus, $M_1(\boldsymbol{\mu}) = E_{\boldsymbol{\theta}}[(1\ \mathbf{x})^T(1\ \mathbf{x})]$, the matrix of second moments (i.e., edge marginals) for the vector $(1\ \mathbf{x})$, must be positive semi-definite. We then obtain the following outer bound on the marginal polytope of complete graphs: $\text{SDEF}_1(K_n) = \{\boldsymbol{\mu} \in \mathbb{R}^+ \mid M_1(\boldsymbol{\mu}) \succeq 0\}$. We can use this outer bound on a sparse MRF by adding variables for the remaining pairwise marginals.

Higher-order moment matrices must also be positive semi-definite, leading to a sequence of tighter and tighter relaxations known as the Lasserre hierarchy. One difficulty with using these semi-definite relaxations is that it is difficult to take advantage of the sparsity of the graph.[5] For example, Rendl *et al.* (2009) commented, with regards to their state-of-the-art SDP-based method:

> "*for sparse problems it is not advisable to use our approach. Since linear programming based methods are capable of exploiting sparsity, solutions might be obtained much faster when applying these methods to sparse data.*"

Since MAP inference is NP-hard, if lifting a constant number of levels in any of these hierarchies were guaranteed to result in a tight relaxation, this would imply that P=NP. However, this does not rule out better approximation guarantees as a result of using a tighter relaxation. LP and SDP-based approximation algorithms typically work by solving the relaxation and then rounding the optimal fractional solution into an (integer) assignment in such a way that one can guarantee that the integer assignment has value close to that of the relaxation. We can rule out better approximation algorithms using this technique by illustrating an *integrality gap*, i.e. an objective function such that the fractional solution is far from the best integer solution. We show in the next section how to construct a fractional solution that satisfies the first lifting of the Sherali-Adams hierarchy (see Example 3). However, illustrating such fractional solutions for higher levels of the hierarchy is substantially more difficult. de la Vega & Kenyon-Mathieu (2007) and Charikar *et al.* (2009) give integrality gaps for different levels of the Sherali-Adams hierarchy, lower bounding the approximation guarantees achievable using these relaxations together with this proof technique. In the process, they show how to construct a fractional point for each level of the hierarchy, which may be of independent interest. Khot & Saket (2009) show that even when augmented with the basic SDP relaxation, any constant number of rounds of Sherali-Adams still has an integrality gap that corresponds to the Goemans-Williamson approximation ratio.

It is important to note that these are theoretical *worst case* results. We will show in this thesis that, despite the existence of integrality gaps, the first few levels of the Sherali-Adams hierarchy suffice to give integer solutions for many real-world inference problems.

2.6 Cutting-Plane Algorithms

The difficulty with using these higher-order relaxations is that the size of the LPs, both in terms of the number of constraints and the variables, grows exponentially in the size of the clusters considered. Even optimizing over $\text{TRI}(G)$, the first lifting of the Sherali-Adams hierarchy, is impractical for all but the smallest of graphical models. Our approach

[5] This is related to the semi-definite matrix completion problem.

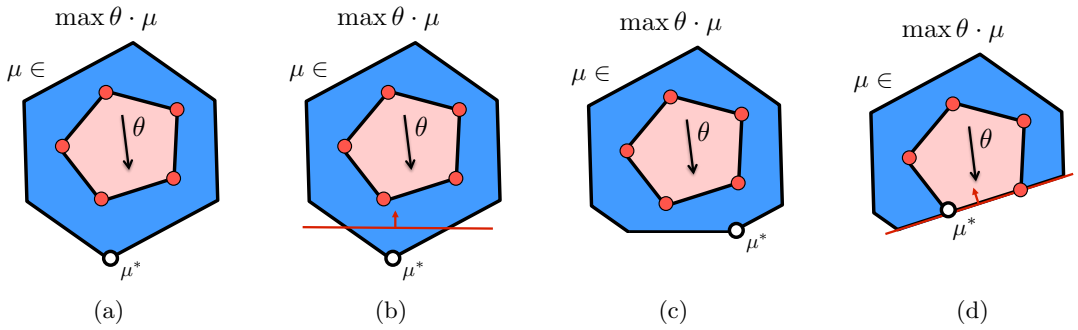


Figure 2-6: Illustration of the cutting-plane algorithm. (a) Solve the LP relaxation. (b) Find a violated constraint, add it to the relaxation, and repeat. (c) Result of solving the tighter LP relaxation. (d) Finally, we find the MAP assignment.

is motivated by the observation that it may not be necessary to add *all* of the constraints that make up a higher-order relaxation such as TRI(G). In particular, it possible that the pairwise LP relaxation alone is close to being tight, and that only a few carefully chosen constraints would suffice to obtain an integer solution.

Our algorithms tighten the relaxation in a problem-specific way, using additional computation just for the hard parts of each instance. We illustrate the general approach in Figure 2-6. This is an example of a *cutting-plane* algorithm. We first solve the pairwise LP relaxation. If we obtain an integer solution, then we have found the MAP assignment and can terminate. Otherwise, we look for a *valid* constraint to add to the relaxation. By valid, we mean that the constraint should not cut off any of the integral vertices. For example, we show in Figure 2-7 an example of an invalid constraint that happens to cut off the MAP assignment (so it could never be found by solving the new LP). Once we find a violated valid constraint, we add it to the relaxation and then repeat, solving the tighter relaxation.

Cutting-plane algorithms have a long history in combinatorial optimization. Gomory (1958) invented a generic recipe for constructing valid inequalities for integer linear programming problems. Gomory cuts play a major role in commercial ILP solvers, such as CPLEX's branch-and-cut algorithm. However, for many combinatorial optimization problems it is possible to construct special purpose valid inequalities that are more effective than Gomory cuts. For example, the *cycle inequalities* are known to be valid for the cut polytope, and have been studied in polyhedral combinatorics because of its relevance to max cut and Ising models. There is a huge literature in the operations research community on cutting-plane algorithms for max cut that use the cycle inqualities (Barahona & Anbil, 2000; Liers *et al.*, 2004; Frangioni *et al.*, 2005).

To apply the cutting-plane approach, we must answer several key questions:

1. **What are valid constraints for the marginal polytope?**

   We already discussed the pairwise and higher-order relaxations. In Chapter 3 we introduce the *cycle relaxation* and the k-ary cycle inequalities, which will be more efficient to optimize over.

2. **How do we efficiently solve the linear program, even for the pairwise LP relaxation?**

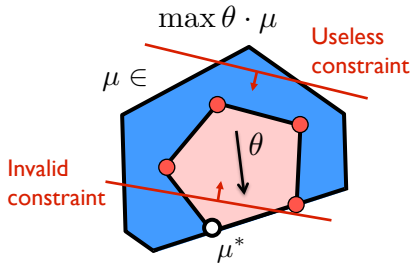


Figure 2-7: We want to choose constraints that are both *valid* and *useful*. A valid constraint is one that does not cut off any of the integer points. One way to guarantee that the constraints added are useful is to use them within the cutting-plane algorithm. Constraints are added if they *separate* the current fractional solution from the rest of the integer solutions. Note that the tightest valid constraints to add are the facets of the marginal polytope.

We address this in Chapter 4, showing how to use the technique of dual decomposition to solve the *dual* of the LP relaxation efficiently.

3. **How do we efficiently find violated constraints?**

   Such an algorithm is called a *separation algorithm*, and must be designed with respect to any class of constraints. We show how to design a separation algorithm for the k-ary cycle inequalities in Chapter 3. In Chapters 5 and 7 we give separation algorithms that work directly in the dual of the LP relaxation.

The above problems are shared by the cutting-plane approaches for max cut, and in many cases have not yet been solved. For example, in their conclusions, Liers *et al.* (2004) comment that

> "*In practical computations, around 90% of the total running time is spent in solving the linear programs by the simplex algorithm. Therefore, a topic of current research is to study the performance of branch-and-cut by replacing the simplex algorithm with fast approximate linear program solvers. The rationale for using an approximate solver is that especially in the beginning of the optimization process the current relaxation is not a "tight" relaxation of the cut polytope anyway.*"

Frangioni *et al.* (2005) study Lagrangian relaxation approaches to solving the max cut problem, which bear some similarity to the dual algorithms that we propose in Chapter 4. Our dual algorithms for tightening the LP relaxation, given in Chapter 5 and Chapter 7, are a *delayed column generation* method, where variables rather than constraints are iteratively added to the LP (Bertsimas & Tsitsiklis, 1997). By solving and tightening the LP relaxations completely in the dual, our algorithms resolve many of the problems raised by Liers *et al.* (2004), and thus may also be of interest to the operations research community.

In the next chapter we will describe the class of constraints that we will use in tightening the relaxation. Broadly speaking, these constraints all enforce that the edge marginals for every *cycle* of the graph are consistent with one another.

Chapter 3

Tightening using the Cycle Relaxation

In the previous chapter we saw how the pairwise LP relaxation, although having some theoretical guarantees, can often have fractional solutions. In these cases, it may only provide a loose upper bound on the value of the MAP assignment. In this chapter we propose using the *cycle relaxation* of the marginal polytope, which will be shown to be significantly tighter than the pairwise relaxation, yet tractable to optimize over. We begin by defining the cycle relaxation, and then discuss connections between it and other well-studied relaxations of the marginal polytope.

At a high level, the cycle relaxation enforces that the edge pseudomarginals along every cycle of the graph should be consistent with some joint distribution on the variables of the cycle. For binary MRFs, we will show that the cycle relaxation is *equivalent* to the first lifting of the Sherali-Adams hierarchy, i.e. using cluster consistency constraints on all triplet clusters. The cycle relaxation formulation is particularly convenient for showing how many seemingly different approaches are actually solving the same relaxation. The main computational advantage of using the cycle relaxation is that we will be able to formulate efficient separation algorithms for it, enabling us to use the cutting-plane methodology. Besides having powerful theoretical guarantees, we show empirically that the cycle relaxation is tight for many real-world inference problems.

3.1 Cycle Relaxation

Given a graphical model $G = (V, E)$, and set of edges $C \subseteq E$ that form a cycle in G, we say that the pseudomarginals $\hat{\boldsymbol{\mu}}$ satisfy the *cycle consistency* constraints for C if $\hat{\boldsymbol{\mu}} \in \text{CYCLE}(C)$, where the feasible space $\text{CYCLE}(C)$ is defined as:[1]

$$\text{CYCLE}(C) = \left\{ \boldsymbol{\mu} \in \mathbb{R}^d \;\middle|\; \exists \boldsymbol{\tau}_C \geq 0, \; \begin{array}{ll} \sum_{\mathbf{x}_{C \setminus i,j}} \tau_C(\mathbf{x}_C) = \mu_{ij}(x_i, x_j) & \forall ij \in C, x_i, x_j \\ \sum_{\mathbf{x}_C} \tau_C(\mathbf{x}_C) = 1 & \end{array} \right\}.$$

[1] We use C to refer to both a set of edges (e.g., with notation $ij \in C$), and the variables involved in these edges. The notation $\mathbf{x}_C$ refers to an assignment to all of the variables in the cycle C, and $C \backslash \{i, j\}$ refers to the set of variables in C except for i or j. Also, $\sum_{\mathbf{x}_{C \setminus i,j}} \tau_C(\mathbf{x}_C)$ means the sum over all assignments $\mathbf{x}_{C \setminus i,j}$ to the variables in $C \backslash \{i, j\}$ of $\tau_C(\mathbf{x}_{C \setminus i,j}, x_i, x_j)$, where x_i and x_j are instantiated outside of the sum.

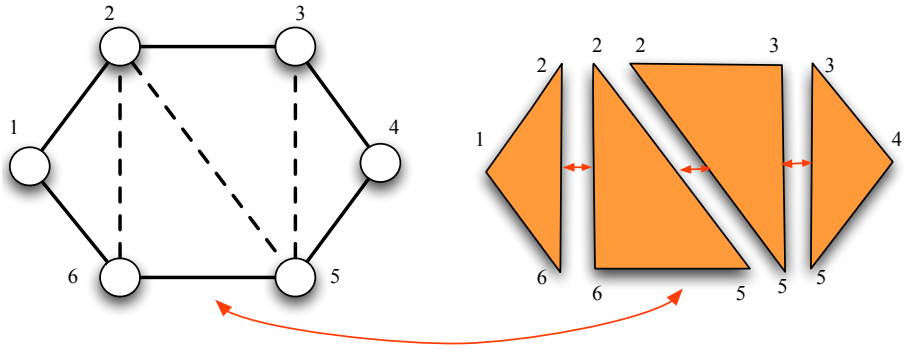


Figure 3-1: Illustration of how cycle consistency can be equivalently enforced by triangulating the cycle and using triplet clusters. The original cycle is shown on the left (solid edges). The dashed edges denote the edges that are added during triangulation. Each orange triangle corresponds to one triplet cluster, with the red arrows denoting consistency constraints between the triplet clusters and between the triplet clusters and the original edges (only one shown).

These constraints are analogous to the pairwise consistency constraints introduced in the previous chapter (c.f. Eq. 2.10), except that they enforce consistency between a cycle and its constituent edges, rather than between an edge and its constituent nodes. Although we defined the polytope by introducing new variables $\tau_C(\mathbf{x}_C)$, one for each assignment $\mathbf{x}_C$ (i.e., as the projection of a lifting), there are equivalent compact formulations.

For example, since cycles are of treewidth two, the cycle consistency constraints for C could be also described by introducing cycle-specific triplet clusters along a triangulation of the cycle, and then enforcing consistency between every pair of triplet clusters that share an edge (see Figure 3-1). Let $\text{TRI}(C)$ denote the triplet clusters in a triangulation of cycle C. For example, $\text{TRI}(C) = \big\{\{1,2,6\},\{2,6,5\},\{2,3,5\},\{3,4,5\}\big\}$ for the cycle in Figure 3-1. Then, an equivalent definition is:

$$\text{CYCLE}(C) = \left\{ \boldsymbol{\mu} \;\middle|\; \begin{array}{ll} \exists\big\{\boldsymbol{\tau}_c \geq 0 : & \sum_{\mathbf{x}_{c\setminus i,j}} \tau_c(\mathbf{x}_c) = \mu_{ij}(x_i,x_j) \quad \forall c,\ i,j \in c, x_i, x_j \\ c \in \text{TRI}(C)\big\} & \sum_{\mathbf{x}_c} \tau_c(\mathbf{x}_c) = 1 \qquad\qquad\qquad\quad \forall c \in \text{TRI}(C) \end{array} \right\} \tag{3.1}$$

We use the notation c to denote clusters, and C to denote cycles. To show that these are equivalent, we can apply the junction tree theorem. The junction tree theorem guarantees that every joint distribution on the variables in the cycle can be equivalently factored into triplet distributions along the triangulation that are consistent with one another, and vice-versa. In Section 3.2 we will give another equivalent formulation of $\text{CYCLE}(C)$, for models with binary variables, using *cycle inequalities*.

We define the *cycle relaxation* to be the intersection of the cycle consistency constraints for every cycle of the graph, in addition to the pairwise consistency constraints:

$$\text{CYCLE}(G) = \text{LOCAL}(G) \bigcap_{C \subseteq E\,:\,C \text{ is a cycle}} \text{CYCLE}(C)\ . \tag{3.2}$$

Rather than including all cycle consistency constraints, it can be shown that the cycle consistency constraints for *chordless* cycles are sufficient to define the cycle relaxation.

Consider the formulation of CYCLE(C) given in Eq. 3.1. If, in addition, we enforce that the triplet clusters are identical across cycles, rather than being cycle-specific, we would obtain the relaxation TRI(G) (the first lifting of the Sherali-Adams hierarchy). Thus, we have TRI(G) $\subseteq$ CYCLE(G). In Section 3.3 we show that, for the special case of binary-valued graphical models, TRI(G) = CYCLE(G). However, the precise relationship is unknown for non-binary models, and an interesting open problem is to show that TRI(G) is strictly tighter than CYCLE(G).

3.2 Cycle Inequalities

In this section we describe a *polyhedral* approach to tightening the relaxation, where we use new constraints but no new variables. We begin by describing the *cycle inequalities* (Barahona & Mahjoub, 1986; Barahona, 1993; Deza & Laurent, 1997) for graphical models with binary variables. Let $\mathcal{M}_{\{0,1\}}$ denote the marginal polytope of a binary pairwise MRF.

Given an assignment $\mathbf{x} \in \{0,1\}^n$, edge $ij \in E$ is *cut* if $x_i \neq x_j$. The cycle inequalities arise from the observation that a cycle must have an even (possibly zero) number of cut edges. Suppose we start at node i, where $x_i = 0$. As we traverse the cycle, the assignment changes each time we cross a cut edge. Since we must return to $x_i = 0$, the assignment can only change an even number of times. For a cycle C and any $F \subseteq C$ such that $|F|$ is odd, this constraint can be written as $\sum_{ij \in C \setminus F} 1[x_i \neq x_j] + \sum_{ij \in F} 1[x_i = x_j] \geq 1$. Since this constraint is valid for all assignments $\mathbf{x} \in \{0,1\}^n$, it holds also in expectation. Thus, the constraint

$$\sum_{ij \in C \setminus F} \Big(\mu_{ij}(1,0) + \mu_{ij}(0,1) \Big) + \sum_{ij \in F} \Big(\mu_{ij}(0,0) + \mu_{ij}(1,1) \Big) \geq 1, \tag{3.3}$$

for $|F|$ odd, is valid for any $\boldsymbol{\mu} \in \mathcal{M}_{\{0,1\}}$. All cycle inequalities based on chordless cycles are known to be *facets* of $\mathcal{M}_{\{0,1\}}$, meaning that they are $(d-1)$ dimensional faces of the polytope (Barahona & Mahjoub, 1986).

Example 2. Consider the fractional solution given in Example 1. Let F be all three edges. Then, $\mu_{ij}(0,0) = \mu_{ij}(1,1) = 0$ for $ij \in F$, and $C \setminus F = \emptyset$. The left side of Eq. 3.3 is thus equal to 0, and we have found a violated cycle inequality.

We next show that only the chordless cycle inequalities are necessary to define the cycle relaxation.

Theorem 3.2.1. *Any violated cycle inequality on a cycle with a chord can be decomposed into two shorter cycle inequalities, one of which must be violated.*

Proof. Suppose that the cycle inequality defined on C, $F \subseteq C$ with $|F|$ odd is violated by $\boldsymbol{\mu}$. We assume only that $\boldsymbol{\mu}$ satisfies the edge marginalization constraints. Let the cycle be $(a, b, c, \ldots, d, e, f, \ldots)$, where a could equal f, and c could equal d, and suppose that the edge be is the chord that splits the cycle in two.

Consider the cycles $C_1 = (e, f, \ldots, a, b)$ and $C_2 = (e, d, \ldots, c, b)$. Since $|F|$ is odd, either C_1 has an odd number of edges in F or C_2 does (but not both). Suppose that it is C_1. Then, for C_1, let $F_1 = C_1 \cap F$, and for C_2, let $F_2 = \Big(C_2 \cap F \Big) \cup \{eb\}$. Suppose for contradiction

that neither of the cycle inequalities C_1, F_1 nor C_2, F_2 are violated, i.e.

$$A = \sum_{ij \in C_1 \setminus F_1} \Big(\mu_{ij}(1,0) + \mu_{ij}(0,1)\Big) + \sum_{ij \in F_1} \Big(\mu_{ij}(0,0) + \mu_{ij}(1,1)\Big) \geq 1, \tag{3.4}$$

$$B = \sum_{ij \in C_2 \setminus F_2} \Big(\mu_{ij}(1,0) + \mu_{ij}(0,1)\Big) + \sum_{ij \in F_2} \Big(\mu_{ij}(0,0) + \mu_{ij}(1,1)\Big) \geq 1. \tag{3.5}$$

Summing these together, we get $A + B \geq 2$. Moreover, since $eb \in C_1 \setminus F_1$ and $eb \in F_2$, we get that

$$A + B = \sum_{ij \in C \setminus F} \Big(\mu_{ij}(1,0) + \mu_{ij}(0,1)\Big) + \sum_{ij \in F} \Big(\mu_{ij}(0,0) + \mu_{ij}(1,1)\Big) + 1 \geq 2, \tag{3.6}$$

where we used that $\mu_{eb}(1,0) + \mu_{eb}(0,1) + \mu_{eb}(1,1) + \mu_{eb}(0,0) = 1$. This contradicts our assumption that the cycle inequality on C, F was violated by $\boldsymbol{\mu}$. Thus, either C_1, F_1 or C_2, F_2 must be a violated cycle inequality. □

In Chapter 7, we consider the dual of the LP relaxation consisting of all cycle inequalities, and also the dual of the cycle relaxation. Using these, we prove that in binary pairwise MRFs, the cycle inequalities give exactly the same relaxation as the cycle relaxation (Corollary 7.2.2). Clearly, however, the cycle relaxation is not going to be tight for all instances. We next give an example of a MAP inference problem for a binary pairwise MRF for which the cycle relaxation (given by all cycle inequalities) is loose.

Example 3. Consider the fully connected MRF on five binary nodes, K_5, and let $\mu_{ij}(0,1) = \mu_{ij}(1,0) = \frac{1}{3}$ and $\mu_{ij}(1,1) = \mu_{ij}(0,0) = \frac{1}{6}$ for all $ij \in K_5$. One can verify, by enumeration, that all of the cycle inequalities are satisfied by $\boldsymbol{\mu}$. To see that $\boldsymbol{\mu}$ is not in the marginal polytope, consider the following objective function: $\theta_{ij}(0,1) = \theta_{ij}(1,0) = 1$, and $\theta_{ij}(0,0) = \theta_{ij}(1,1) = 0$. Then, $\boldsymbol{\mu}$ achieves objective value $|E| \cdot \frac{2}{3} = \frac{20}{3}$. On the other hand, all integer solutions have objective value at most $6 \cdot \frac{2}{3} = \frac{18}{3}$ (to maximize the number of cut edges, let exactly two of the x_i be 1). The maximum of any linear objective over the marginal polytope must be achieved by one of its vertices, which are all integer (Vanderbei, 2007). Since $\boldsymbol{\mu}$ is a fractional vertex that achieves a strictly better objective value than the integral vertices, we conclude that $\boldsymbol{\mu}$ is not in the marginal polytope.

3.3 Equivalent Formulations for Binary Models

We showed in Section 3.1 that, for general graphical models, $\text{TRI}(G) \subseteq \text{CYCLE}(G)$. In the special case of graphical models with binary variables, we obtain an equality.

Theorem 3.3.1. *For pairwise MRFs with binary variables,* $\text{TRI}(G) = \text{CYCLE}(G)$*, i.e. the first lifting on the Sherali-Adams hierarchy is equivalent to the cycle relaxation.*

Proof. Our proof will use the fact that, for binary variables, the cycle consistency constraints can be shown to be equivalent to the cycle inequalities (Corollary 7.2.2). We already showed that $\text{TRI}(G) \subseteq \text{CYCLE}(G)$. Letting $|V| = n$, we denote the complete graph on n nodes as K_n. We prove that $\text{CYCLE}(G) \subseteq \text{TRI}(G)$ by induction on the number of edges $|K_n \setminus E|$ that the graph G is missing, compared to the complete graph.

We first note that, by Theorem 3.2.1, the constraints in the definition of CYCLE(G) only need to be given for *chordless* cycles, since the consistency of a cycle with a chord is implied by the consistency of the two smaller cycles using the chord. Thus, if $G = K_n$, we would trivially have TRI(G) = CYCLE(G). This proves the base case of $|K_n \backslash E| = 0$.

Consider $|K_n \backslash E| > 0$, and suppose that $\boldsymbol{\mu} \in \text{CYCLE}(G)$. Let st be some edge in $K_n \backslash E$. Note that $\boldsymbol{\mu}$ is only defined for the edges in E. Suppose that there exists $\mu_{st}(x_s, x_t) \geq 0$ satisfying $\sum_{x_s, x_t} \mu_{st}(x_s, x_t) = 1$ such that $\overline{\boldsymbol{\mu}}$ (the extended $\boldsymbol{\mu}$, now including st) $\in \text{CYCLE}(G \cup \{st\})$. Then, by the inductive hypothesis, $\overline{\boldsymbol{\mu}} \in \text{TRI}(G \cup \{st\})$ (i.e., $\exists \tau$ that satisfies the triplet consistency constraints). The same τ also shows that $\boldsymbol{\mu} \in \text{TRI}(G)$.

We now consider the case where no such μ_{st} exists. Let $c = \mu_{st}(0,1) + \mu_{st}(1,0)$, and consider all cycles that include edge st. Every cycle inequality where $st \notin F$ (edge st is not cut) gives a lower bound on c,

$$c \geq \quad 1 - \sum_{ij \in C_1 \backslash F_1, ij \neq st} \Big(\mu_{ij}(1,0) + \mu_{ij}(0,1)\Big) - \sum_{ij \in F_1} \Big(\mu_{ij}(0,0) + \mu_{ij}(1,1)\Big), \tag{3.7}$$

while every cycle inequality where $st \in F$ (edge st is cut) gives an upper bound on c,

$$c \leq \quad \sum_{ij \in C_2 \backslash F_2} \Big(\mu_{ij}(1,0) + \mu_{ij}(0,1)\Big) + \sum_{ij \in F_2, ij \neq st} \Big(\mu_{ij}(0,0) + \mu_{ij}(1,1)\Big). \tag{3.8}$$

If no such μ_{st} exists, then there must be two cycle inequalities C_1, F_1 (corresponding to the upper bound in Eq. 3.7) and C_2, F_2 (corresponding to the lower bound in Eq. 3.8) such that $A > B$, where A denotes the right-hand side of Eq. 3.7 and B denotes the right-hand side of Eq. 3.8. Consider the cycle inequality $C = C_1 \cup C_2 \backslash \{st\}$ and $F = F_1 \cup F_2 \backslash \{st\}$. Note that $|F|$ must be odd because $|F_1|$ and $|F_2|$ were odd and $st \in F_2, st \notin F_1$. Since we just showed that $A - B > 0$, we have that the cycle inequality C, F is violated. However, this contradicts our assumption that $\boldsymbol{\mu} \in \text{CYCLE}(G)$. □

We discussed in Section 2.1.1 a special case of binary pairwise Markov random fields called *Ising models* in which the edge potential functions only depend on whether the variables' states agree or differ. In this case, one can show the following result:

Theorem 3.3.2 (Barahona, 1993)**.** *The cycle relaxation is exact, i.e.* $\mathcal{M}(G) = \text{CYCLE}(G)$*, for Ising models of planar graphs (*G *has no* K_5 *or* $K_{3,3}$ *minor).*

This result applies *regardless* of the treewidth of G, as long as G is planar. Since inference in Ising models is equivalent to finding the maximum cut in a graph, this result also shows that the maximum cut problem can be tractably solved in planar graphs.

A natural question is whether this theorem provides any non-trivial guarantees for other binary graphical models. Any binary pairwise Markov random field G can be transformed into an Ising model G' by including an additional variable X_{n+1} and adding edges from all other variables to X_{n+1}. *Outer-planar graphs* are graphs G such that the transformed graph G' is planar. It can be shown that all outer-planar graphs are of treewidth two. However, since we already know that TRI(G) exactly defines the marginal polytope for graphs of treewidth two, the theorem does not provide additional insight.

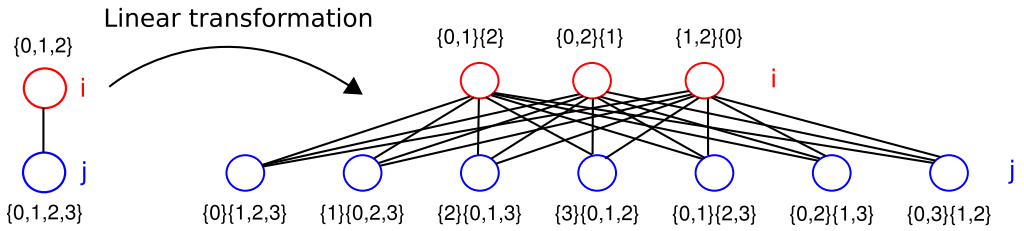


Figure 3-2: Illustration of the projection for one edge $ij \in E$ where $\chi_i = \{0, 1, 2\}$ and $\chi_j = \{0, 1, 2, 3\}$. The projection graph (shown on right), has 3 partitions for i and 7 for j.

3.4 k-ary Cycle Inequalities

In this section, we show how to derive valid inequalities for the marginal polytope of non-binary pairwise MRFs from any inequality that is known for the binary marginal polytope. As a consequence, we obtain a new class of k-ary cycle inequalities, constraints that are valid for non-binary MRFs. We describe the k-ary cycle inequalities, relate them to the cycle relaxation, and give empirical results showing that they allow us to exactly find the MAP assignment in protein side-chain placement problems. This section is based in part on material previously published in (Sontag, 2007; Sontag & Jaakkola, 2008).

The simplest way to understand the new constraints is to equivalently reformulate the non-binary graphical model as a graphical model with only binary variables. For example, we could introduce one binary variable X_{i,x_i} for each variable X_i and state x_i in the original model. The new pairwise potentials would then be $\theta^{\text{new}}(X_{i,x_i} = 1, X_{j,x_j} = 1) = \theta_{ij}(x_i, x_j)$, and zero otherwise. We also have a higher-order factor, one for each i, on the variables X_{i,x_i} for all x_i, that enforces that exactly one of these should be 1. We now have a binary-valued graphical model, and its LP relaxation can take advantage of all the constraints that we know to be valid for the binary marginal polytope.

However, there are many different ways of formulating an equivalent binary graphical model, and each results in a different LP relaxation. A better way to understand the new constraints is via projections of the marginal polytope onto different binary marginal polytopes. Aggregation and projection are well-known techniques in polyhedral combinatorics for obtaining valid inequalities (Deza & Laurent, 1997). Given a linear projection $\Phi(\mathbf{x}) = A\mathbf{x}$, any valid inequality $\mathbf{c}'\Phi(\mathbf{x}) \leq \mathbf{b}$ for $\Phi(\mathbf{x})$ also gives the valid inequality $\mathbf{c}'A\mathbf{x} \leq \mathbf{b}$ for $\mathbf{x}$. We obtain new inequalities for the marginal polytope by aggregating the *states* of each non-binary variable into just two states.

For each variable i, let π_i^q be a *partition* of its states into two non-empty sets, i.e., the map $\pi_i^q : \chi_i \to \{0, 1\}$ is surjective (χ_i denotes the states of node i). Let $\pi_i = \{\pi_i^1, \pi_i^2, \ldots\}$ be a *collection of partitions* of variable i. Define the *projection graph* $G_\pi = (V_\pi, E_\pi)$ so that there's a node for each partition in each collection π_i and such nodes are fully connected across adjacent variables:

$$V_\pi = \bigcup_{i \in V} \pi_i, \qquad E_\pi \subseteq \{(\pi_i^q, \pi_j^r) \mid (i, j) \in E, q \leq |\pi_i|, r \leq |\pi_j|\}. \tag{3.9}$$

We obtain a different projection graph depending on the quantity and type of partitions that we choose for each node, $\pi = \{\pi_i\}$. We call the graph consisting of *all* possible

variable partitions the *full projection graph*. In Figure 3-2 we show the part of the full projection graph that corresponds to edge ij, where x_i has three states and x_j has four states. Intuitively, a partition for a variable splits its states into two clusters, resulting in a binary variable. For example, the (new) variable corresponding to the partition $\{0,1\}\{2\}$ of x_i is 1 if $x_i = 2$, and 0 otherwise.

Given any pseudomarginal $\boldsymbol{\mu}$, its projection onto the binary marginal polytope of the projection graph G_π, which has binary variables $x_m \in \{0,1\}$ for each partition, is

$$\mu_m^\pi(x_m) = \sum_{s_i \in \chi_i \,:\, \pi_i^q(s_i) = x_m} \mu_i(s_i) \qquad \forall m = \pi_i^q \in V_\pi \tag{3.10}$$

$$\mu_{mn}^\pi(x_m, x_n) = \sum_{\substack{s_i \in \chi_i \,:\, \pi_i^q(s_i) = x_m, \\ s_j \in \chi_j \,:\, \pi_j^r(s_j) = x_n}} \mu_{ij}(s_i, s_j) \qquad \forall mn = (\pi_i^q, \pi_j^r) \in E_\pi. \tag{3.11}$$

It is straightforward to show that, for any marginal vector $\boldsymbol{\mu} \in \mathcal{M}(G)$, the corresponding $\boldsymbol{\mu}^\pi$ is in $\mathcal{M}_{\{0,1\}}(G_\pi)$, the binary marginal polytope of the projection graph (Sontag & Jaakkola, 2008). Thus, valid inequalities for $\mathcal{M}_{\{0,1\}}(G_\pi)$ carry over to $\mathcal{M}(G)$.

These projections yield a new class of cycle inequalities for the marginal polytope. Consider a projection graph G_π, a cycle C in G_π, and any $F \subseteq C$ such that $|F|$ is odd. We obtain the following k-ary cycle inequality for $\boldsymbol{\mu} \in \mathcal{M}(G)$ after applying the projection to a binary cycle inequality:

$$\sum_{mn \in C \setminus F} \Big(\mu_{mn}^\pi(0,1) + \mu_{mn}^\pi(1,0)\Big) + \sum_{mn \in F} \Big(\mu_{mn}^\pi(0,0) + \mu_{mn}^\pi(1,1)\Big) \geq 1. \tag{3.12}$$

Notice that since we are now considering cycles in the *projection graph*, it is possible to have a simple cycle that uses more than one partition of a variable (see Figure 3-6 for an example). Because of how we derived the constraints, these inequalities are guaranteed to be valid for the marginal polytope. Nonetheless, a natural question to ask is whether these cycles provide any tighter of a relaxation than if we had just restricted ourselves to inequalities involving just one partition per variable. We show that the answer is no:

Theorem 3.4.1. *Given a violated cycle inequality $C, F \in G_\pi$, if C contains two nodes $a = \pi_i^q$ and $a' = \pi_i^r$ corresponding to different partitions of i, there exists a shorter violated cycle inequality $C' \subset C$ that excludes either a or a'.*

Proof. By induction on the cycle length $|C|$. Suppose that the violated cycle inequality is on the cycle $C = (a, b, \ldots, c, a', d, \ldots)$, where a and a' refer to two different partitions of the same variable. Since a and a' have the same neighbors, there must also be an edge between c and a. Having shown that C has a chord, we now apply Theorem 3.2.1 to give a violated cycle inequality on either $C_1 = (a, b, \ldots, c)$ (with associated F_1) or $C_2 = (c, a', d, \ldots, a)$ (with associated F_2). If C_1, F_1 is violated, we are finished. Otherwise, we repeat the argument on the shorter cycle inequality C_2, F_2. Our base case is the length-3 cycle inequality (a, b, a'): assuming $\boldsymbol{\mu}$ satisfies the edge marginalization constraints, this inequality cannot be violated. □

In the projection graph from Figure 3-2, we used only three partitions for each variable X_i. For example, we had the partition $\{0,1\}\{2\}$ but not the partition $\{2\}\{0,1\}$. We next show that this is without loss of generality: having a partition and its complement in the projection graph can never result in a tighter relaxation.

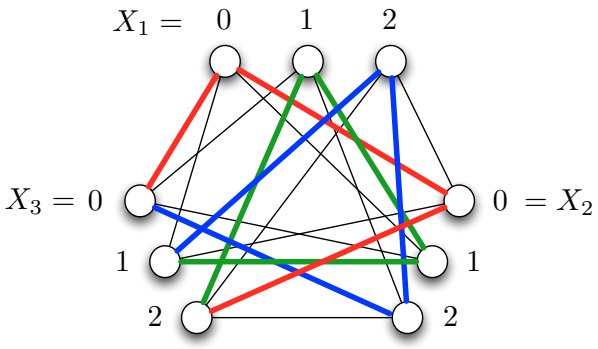


Figure 3-3: Illustration of the fractional point μ and the objective θ used to prove that the k-ary cycle inequalities are strictly weaker than the cycle relaxation (Theorem 3.4.3). Shown are the three states (one node per state) of the variables X_1, X_2, X_3, where $X_i \in \{0, 1, 2\}$. An edge between $X_i = x_i$ and $X_j = x_j$ denotes $\mu_{ij}(x_i, x_j) = 1/6$; no edge denotes $\mu_{ij}(x_i, x_j) = 0$. Thick lines between $X_i = x_i$ and $X_j = x_j$ indicate that $\theta_{ij}(x_i, x_j) = 1$, whereas for the non-thick lines, $\theta_{ij}(x_i, x_j) = 0$, and when there is no line, $\theta_{ij}(x_i, x_j) = -2$. The red, green, and blue colors are used to make clear that all assignments to X_1, X_2, X_3 (corresponding to a length-3 cycle which uses one node from each of X_1, X_2, X_3) use at most one of the $\theta_{ij}(x_i, x_j) = 1$ edges.

Theorem 3.4.2. *Given a violated cycle inequality $C, F \in G_\pi$ for any cycle C' having the complement partition for some of the nodes in C, $\exists F'$ such that the cycle inequality C', F' is violated.*

Proof. By induction on the number of complement partitions. The base case, zero, is trivial. Suppose that $C = (\ldots, \pi_i^q, \ldots)$ and $C' = (\ldots, a, \pi_i^{\overline{q}}, b, \ldots)$. Let F'' be given by the inductive hypothesis applied to C and C'', where C'' is the same as C' except with π_i^q instead of $\pi_i^{\overline{q}}$. Start with $F' = F'' \backslash \{a\pi_i^q, \pi_i^q b\}$. If $a\pi_i^q \notin F''$, add $a\pi_i^{\overline{q}}$ to F'. Similarly, if $\pi_i^q b \notin F''$, add $\pi_i^{\overline{q}} b$ to F'. The left hand side of the cycle inequality on C', F' is equivalent in value to the cycle inequality on C'', F'', and thus is also violated. □

Thus, if one were to try to construct the full projection graph for a binary-valued graphical model, it would simply be the same as the original graph.

Related work

Althaus *et al.* (2000) analyze the *GMEC polyhedron*, which is equivalent to the marginal polytope. They use a similar state-aggregation technique to derive valid constraints from the triangle inequalities. Koster *et al.* (1998) investigate the *Partial Constraint Satisfaction Problem polytope*, which is also equivalent to the marginal polytope. They used state-aggregation to show that a class of cycle inequalities (corresponding to Eq. 3.12 for $|F| = 1$) are valid for this polytope, and give an algorithm to separate the inequalities for a single cycle. Both papers showed that these constraints are facet-defining.

3.4.1 Relationship to Cycle Relaxation

Having introduced the k-ary cycle inequalities, a natural question is whether these are as strong as the cycle relaxation. Although this is true for *binary* cycle inequalities, the answer is no for $k \geq 3$. We prove this by demonstrating a point that satisfies all of the k-ary cycle inequalities yet is not in the marginal polytope.

Theorem 3.4.3. *The k-ary cycle inequalities are strictly weaker than the cycle relaxation when $k \geq 3$.*

Proof. (See Figure 3-3 for an illustration of the quantities used in the proof.) Let G be a graphical model with three nodes, X_1, X_2, X_3, where $X_i \in \{0, 1, 2\}$. Since G is a cycle, the cycle relaxation is exactly equal to the marginal polytope. Consider the following fractional point:

$\mu_{12}(x_1, x_2) =$

	$x_2 = 0$	$x_2 = 1$	$x_2 = 2$
$x_1 = 0$	1/6	1/6	0
$x_1 = 1$	0	1/6	1/6
$x_1 = 2$	1/6	0	1/6

(3.13)

$\mu_{13}(x_1, x_3) =$

	$x_3 = 0$	$x_3 = 1$	$x_3 = 2$
$x_1 = 0$	1/6	1/6	0
$x_1 = 1$	1/6	0	1/6
$x_1 = 2$	0	1/6	1/6

(3.14)

$\mu_{23}(x_2, x_3) =$

	$x_3 = 0$	$x_3 = 1$	$x_3 = 2$
$x_2 = 0$	0	1/6	1/6
$x_2 = 1$	1/6	1/6	0
$x_2 = 2$	1/6	0	1/6

(3.15)

It is easy to verify that μ satisfies all of the pairwise consistency constraints and the k-ary cycle inequalities. Now consider the following objective function:

$\theta_{12}(x_1, x_2) =$

	$x_2 = 0$	$x_2 = 1$	$x_2 = 2$
$x_1 = 0$	1	0	-2
$x_1 = 1$	-2	1	0
$x_1 = 2$	0	-2	1

(3.16)

$\theta_{13}(x_1, x_3) =$

	$x_3 = 0$	$x_3 = 1$	$x_3 = 2$
$x_1 = 0$	1	0	-2
$x_1 = 1$	0	-2	1
$x_1 = 2$	-2	1	0

(3.17)

$\theta_{23}(x_2, x_3) =$

	$x_3 = 0$	$x_3 = 1$	$x_3 = 2$
$x_2 = 0$	-2	0	1
$x_2 = 1$	0	1	-2
$x_2 = 2$	1	-2	0

(3.18)

In the objective, $\theta_{ij}(x_i, x_j) = -2$ wherever $\mu_{ij}(x_i, x_j) = 0$, thereby ruling out any optimal solutions $\mu^*(\theta)$ with $\mu^*_{ij}(x_i, x_j)$ non-zero. It can be shown by enumeration that for all integer assignments x, $\theta(x) \leq 1$ (the maximum is achieved, e.g., by $x = (0, 0, 1)$). On the other hand, $\mu \cdot \theta$ has value 1.5. The maximum of any linear objective over the marginal polytope must be achieved by one of its vertices, which are all integer. Since μ is a fractional point

that achieves a strictly better objective value than the integral vertices, we conclude that μ is not in the marginal polytope (yet satisfies all of the k-ary cycle inequalities). $\square$

Although the k-ary cycle inequalities do not suffice to obtain an integer solution for the objective function used in the proof of Theorem 3.4.3 and illustrated in Figure 3-3, they do provide a significantly tighter bound on the value of the MAP assignment: the pairwise LP relaxation gives an upper bound of 3, the k-ary cycle inequalities[2] give an upper bound of 1.5, and the MAP assignment has value 1.

We next looked to see how strong the k-ary cycle inequalities are relative to the pairwise relaxation, again on three node MRFs with three states. For comparison, the cycle relaxation would give an exact solution on 100% of the examples, since this graphical model is a single cycle. We constructed 10,000 random objective functions by sampling $\theta'_{ij}(x_i, x_j) \sim U[-1, 1]$, and solved the LP relaxation for each using the (a) pairwise relaxation, and (b) pairwise relaxation plus all k-ary cycle inequalities. We found that the pairwise relaxation gives an integral solution 88% of the time, while the k-ary cycle inequalities gave an integral solution 100% of the time. If we instead added on the objective function used in the previous example, i.e. optimizing over $\theta + \theta'$, the pairwise relaxation is exact less than 7% of the time, while the k-ary cycle inequalities give an exact solution on over 99% of the examples.

3.4.2 Separation Algorithm

Although there are exponentially many cycles and cycle inequalities for a graph, Barahona & Mahjoub (1986) give a simple algorithm to separate the whole class of cycle inequalities for binary models. In this section, we describe our generalization of this algorithm to the k-ary cycle inequalities.

The algorithm is given in Figure 3-4, and takes as input a projection graph G_π and the current pseudomarginals $\boldsymbol{\mu}_\pi$. To see whether any cycle inequality is violated, we construct an undirected graph $G' = (V', E')$ that consists of two copies of G_π, with edges between each copy. For each $i \in V_\pi$, V' contains nodes i_1 and i_2, corresponding to the two copies. For each edge $(i, j) \in E_\pi$, the edges in E' are: (i_1, j_1) and (i_2, j_2) with weight $\mu^\pi_{ij}(0, 1)+\mu^\pi_{ij}(1, 0)$, and (i_1, j_2) and (i_2, j_1) with weight $\mu^\pi_{ij}(0, 0) + \mu^\pi_{ij}(1, 1)$. See Figure 3-5 for an illustration. Then, for each node $i \in V_\pi$ we find the shortest path in G' from i_1 to i_2. We assume that the shortest path algorithm returns the shortest length path with the minimal weight.[3] The shortest of all the paths found, P_s, will not use both copies of any node j (otherwise the path j_1 to j_2 would be shorter, since all edge weights are non-negative).

Thus, we have a path P_s where the first node is s_1 and the last node is s_2. The cycle $C^* \subseteq E_\pi$ is obtained by merging s_1 and s_2 and dropping the subscripts from P_s. All edges traversed by P_s of the form (i_1, j_2) or (i_2, j_1) (i.e., that cross between the two graph copies) are added to the set F^*. The resulting cycle inequality gives the minimum value of $\sum_{mn \in C \setminus F} \left(\mu^\pi_{mn}(0, 1) + \mu^\pi_{mn}(1, 0)\right) + \sum_{mn \in F} \left(\mu^\pi_{mn}(0, 0) + \mu^\pi_{mn}(1, 1)\right)$. If this is less than 1, we have found a violated cycle inequality; otherwise, $\boldsymbol{\mu}_\pi$ satisfies all cycle inequalities.

[2]Using the cutting-plane approach and adding violated inequalities one at a time, we found that we needed to add 12 k-ary cycle inequalities before the remaining inequalities were satisfied.

[3]This can be achieved by changing the comparison operator used by the shortest paths algorithm to prefer shorter length paths of the same total weight. Alternatively, one can add a small weight $\epsilon > 0$ to each of the edges prior to running the shortest paths algorithm.

Algorithm SeparateCycles(G_π, $\boldsymbol{\mu}_\pi$)

```
// Initialize the auxilliary graph used in the shortest path computation.
let G' = (V', E'), where
    V' = ∪_{i∈V_π} {i_1, i_2}
    E' = ∪_{(i,j)∈E_π} {(i_1, i_2), (i_1, j_2), (i_2, j_1), (j_2, j_2)}

// Setup the edge weights.
for each edge ij ∈ E_π
    w(i_1, j_2) = μ^π_ij(0,0) + μ^π_ij(1,1)    // Cut
    w(i_2, j_1) = μ^π_ij(0,0) + μ^π_ij(1,1)    // Cut
    w(i_1, j_1) = μ^π_ij(0,1) + μ^π_ij(1,0)    // Not cut
    w(i_2, j_2) = μ^π_ij(0,1) + μ^π_ij(1,0)    // Not cut

// Run the shortest path algorithm, once for each node.
for each node i ∈ V_π
    // Find shortest path P_i from i_1 to i_2 on graph G' with weights w
    P_i = ShortestPath(i_1, i_2, G', w)

    // Make sure that this is a simple cycle in G_π.
    if ∃j ≠ i such that j_1, j_2 ∈ P_i
        Discard P_i.

return {P_i : w(P_i) < 1}
```

Figure 3-4: Given the projection graph $G_\pi = (V_\pi, E_\pi)$ and edge pseudomarginals $\boldsymbol{\mu}_\pi$, find the most violated cycle inequality.

Using Dijkstra's shortest paths algorithm with a Fibonacci heap to implement the priority queue (Cormen *et al.*, 2001), each shortest paths computation takes time $O(n \log n + |E'|)$, where $n = |V'|$. Thus, the overall running time of the separation algorithm from Figure 3-4 is $O(n^2 \log n + n|E'|)$.

Implementation Details

We use the separation algorithm together with the cutting-plane methodology described in Section 2.6. The overall algorithm is as follows:

1. Solve the LP relaxation (in Iteration 1, use the pairwise relaxation).
2. Construct the projection graph G_π and pseudomarginals $\boldsymbol{\mu}_\pi$ using Eqs. 3.9-3.11.
3. Run **SeperateCycles**(G_π, $\boldsymbol{\mu}_\pi$) to see if there are any violated cycle inequalities.
4. Add all cycle inequalities returned by Step 3 to the LP relaxation.
5. Return to Step 1, but now solve using the tighter relaxation.

From a practical perspective, it would be slow to repeatedly solve the new linear program each time a constraint is added. However, there are a number of optimization methods that

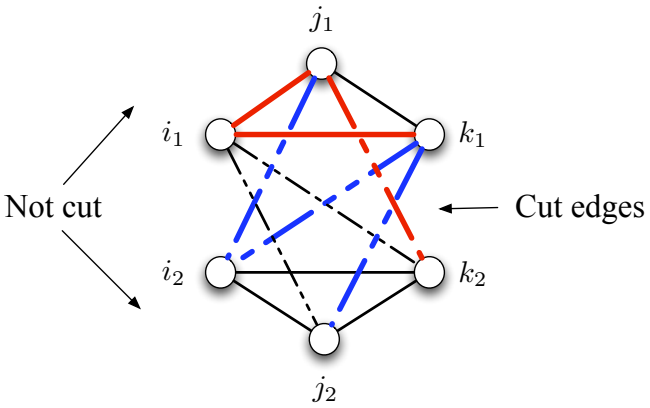


Figure 3-5: Illustration of graph used in shortest path algorithm for finding the most violated cycle inequality. The dashed edges denote *cut* edges (i.e., if used in the shortest path, they are assigned to F), while the solid edges denote edges that are *not cut*. The algorithm is as follows: To find the most violated cycle inequality on a cycle involving node j, find the shortest path from j_1 to j_2 in the graph (edge weights are discussed in Section 3.4.2). To find the most violated cycle inequality overall, considering *all* cycles, repeat this for every node (e.g., also look for the shortest path from k_1 to k_2). The red and blue paths demonstrate two different cycle inequalities. The red path, from k_1 to k_2, denotes the cycle inequality $C = \{ki, ij, jk\}$, $F = \{jk\}$. The blue path, from j_1 to j_2, denotes the cycle inequality $C = \{ji, ik, kj\}$, $F = C$, i.e. all three edges are cut. Since the paths begin in the top component and end in the bottom component, each path must have an odd number of cut edges. Thus, $|F|$ is always odd, as required to obtain a valid inequality.

allow you to "warm start" after tightening the LP with new constraints, using the optimal solution from the previous LP. For example, when using the dual simplex algorithm, one can set the basis (the initial vertex) using the previous solution's basis. We found this method to be extremely effective, and discuss it further in Section 3.4.3.

There has also been a significant amount of work on designing specialized LP solvers to solve LP relaxations that arise from graphical models. We briefly mention how to use the k-ary cycle inequalities within these. One such algorithm is the subgradient method (e.g., Komodakis *et al.* (2010)), which solves the dual of the LP relaxation. Although it is straightforward to include new inequalities in the dual subgradient algorithm, it can be difficulty to obtain a primal solution, which is necessary to run the separation algorithm. Nedic & Ozdaglar (2007) show that one way to obtain a primal solution is by averaging the subgradient vectors. Another such algorithm is the proximal point method, which directly solves the primal LP relaxation. Ravikumar *et al.* (2008) give a proximal point algorithm where the inner loop uses Bregman projections onto the constraint set. It is straightforward to derive the corresponding projections for the cycle inequalities.[4]

An alternative approach is to make use of algorithms specifically designed for constraint sets defined by separation oracles. For example, the *ellipsoid* algorithm at every iteration calls a separation oracle to either certify that a point is within the feasible space, or to pro-

[4]To give the closed-form update for the k-ary cycle inequalities using entropic regularization, one must make use of Theorem 3.4.1 to first construct a violated inequality where only one partition per variable is used. This then ensures that all of the variables' coefficients are 1.

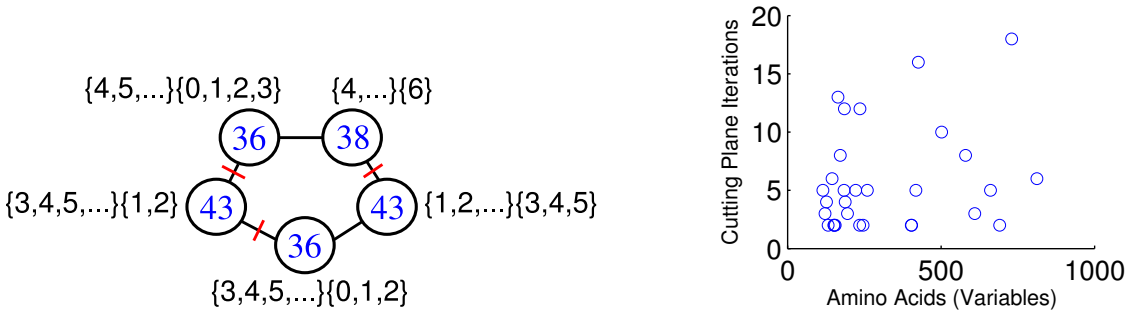


Figure 3-6: MAP for protein side-chain prediction with the Rosetta energy function. Left: One of the violated k-ary cycle inequalities that we found using the full projection graph. Right: Number of cutting-plane iterations it took us to find the MAP assignment of each protein, as a function of the protein length. In each iteration we tighten the relaxation by adding k-ary cycle inequalities.

vide a hyperplane that separates it from the feasible space. However, ellipsoid is generally not practical for even medium-sized problems. Interior point algorithms for linear programming, on the other hand, have better theoretical and practical running times. Recent work has focused on *interior point cutting plane* algorithms that can make use of a separation oracle (Mitchell, 2005).

Finally, another possibility is to directly look for violated cycle inequalities in the dual, rather than attempting to recover a primal solution before running the separation algorithm. We show how to do this in Chapter 7.

3.4.3 Experiments on Protein Side-Chain Placement

We next applied our algorithm to the problem of predicting protein side-chain configurations. Given the 3-dimensional structure of a protein's backbone, the task is to predict the relative angle of each amino acid's side-chain. The angles are discretized into at most 45 values. Yanover *et al.* (2006) showed that minimization of the Rosetta energy function corresponds to finding the MAP assignment of a non-binary pairwise MRF. They also showed that the tree-reweighted max-product algorithm (Wainwright *et al.*, 2005a) can be used to solve the pairwise LP relaxation, and that this succeeds in finding the MAP assignment for 339 of the 369 proteins in their data set. However, the optimal solution to the LP relaxation for the remaining 30 proteins, arguably the most difficult of the proteins, is fractional.

We use the following strategy to efficiently find violated k-ary cycle inequalities. First, we begin by using the k-projection graph, which simply has a partition $\pi_i^{x_i} = \{x_i\}$ (versus all other states) for every variable i and state x_i. We only consider the full projection graph after satisfying all of the k-ary cycle inequalities that can be obtained from the k-projection graph. Second, we exclude partitions $m = \pi_i^q$ from the projection graphs for which the corresponding single-node marginals $\mu_m^\pi(x_m)$ are integral. This is without making the relaxation looser, as it can be shown that there cannot be any violated cycle inequality involving such a partition m.

Using the k-ary cycle inequalities found with just the k-projection graph, we succeeded in finding the MAP assignment for all proteins except for the protein '1rl6'. We show in Figure 3-6 the number of cutting-plane iterations needed for each of the 30 proteins. In

each iteration, we solve the LP relaxation, and, if the solution is not integral, run the separation algorithm to find violated inequalities. For the protein '1rl6', after 12 cutting-plane iterations, the solution was not integral, and we could not find any violated cycle inequalities using the k-projection graph. We then tried using the full projection graph, and found the MAP after just one (additional) iteration.[5] Figure 3-6 shows one of the cycle inequalities Eq. 3.12 in the full projection graph that was found to be violated. By Theorem 3.4.1, there must also be a shorter violated inequality. The cut edges indicate the 3 edges in F. The violating μ had $\mu_{36}(s) = \frac{1}{6}$ for $s \in \{0, 1, 2, 3, 4, 5\}$, $\mu_{38}(6) = \frac{1}{3}$, $\mu_{38}(4) = \frac{2}{3}$, $\mu_{43}(s) = \frac{1}{6}$ for $s \in \{1, 2, 4, 5\}$, $\mu_{43}(3) = \frac{1}{3}$, and zero for all other values of these variables. This example shows that the relaxation given by the full projection graph is strictly tighter than that of the k-projection graph.

The commercial linear programming solver CPLEX 11.2 solves the pairwise LP relaxation using the dual simplex algorithm in under 34 seconds per protein.[6] After calling the separation algorithm, we re-run dual simplex using the previous basis as the starting point.[7] Each subsequent LP took under 1.8 seconds to solve, showing that this "warm start" technique was extremely effective for protein side-chain placement. Also, the separation algorithm never took more than 1.3 seconds. We found each protein's MAP assignment in under 45 seconds (average: 8 seconds, median: 4 seconds) and, on average, the running time subsequent to solving the pairwise relaxation (i.e., for tightening the relaxation) was only 26% of the total time to find the MAP assignment.

Kingsford *et al.* (2005) found, and we also observed, that CPLEX's branch-and-cut algorithm for solving integer linear programs also works well for these problems. In particular, CPLEX succeeds in finding these proteins' MAP assignments in under 59 seconds (average: 12 seconds, median: 7 seconds). Although we found the dual simplex algorithm to be very fast for these protein side-chain placement problems (whose LPs are relatively small), we will see in Section 5.4 that it is slow on larger MRFs, such as those arising from the protein design problem. Thus, in the subsequent chapters we will develop new algorithms for solving the LP relaxations.

3.5 Related Work

Recently, several authors have suggested seemingly different approaches to tightening the pairwise LP relaxation (Globerson & Jaakkola, 2007a; Werner, 2008; Komodakis & Paragios, 2008; Schraudolph & Kamenetsky, 2009; Batra *et al.*, 2010; Schraudolph, 2010). We show in this section that these approaches can all be viewed as enforcing either cluster or cycle consistency, and thus correspond to optimizing over a relaxation that is equivalent to the ones discussed in this chapter.

Werner (2008) proposes tightening the pairwise relaxation with cluster consistency constraints, described in Section 2.5.2, and which we show how to efficiently optimize over in Chapter 5. In their experiments, they illustrate how to tighten the relaxation for a binary graphical model by adding clusters over length-4 cycles and enforcing consistency between these clusters and the corresponding edges. This is equivalent to enforcing cycle consistency

[5] Only three variables had non-integral values, and each of these three had fewer than 6 non-zero values, so the projection graph remained small.

[6] All reported running times are using a single CPU of a machine with an Intel Xeon 3 GHz processor.

[7] In particular, we initialize the slacks (artificial variables) for the new constraints as basic, since they were previously violated.

over the length-4 cycles. As we showed in Sections 3.1 and 3.2, this corresponds to the same relaxation given by the cycle inequalities (restricted to length-4 cycles). Werner (2008) does not address the search problem of finding which clusters to use in tightening the relaxation.

Komodakis & Paragios (2008) give an algorithm for tightening the relaxation by a sequence of cycle-repairing operations. It is straightforward to show that this algorithm optimizes over the cycle relaxation. In fact, we show in Chapter 7 that, for graphical models with binary variables, a sequence of two cycle repair operations is equivalent to a coordinate-descent step on one cycle inequality in the *dual* of the LP relaxation. For non-binary graphical models, cycle repair operations are more similar to the method that we propose in Chapter 5. Komodakis & Paragios (2008) do not address the search problem.

As we discussed in Section 3.1, finding the most likely assignment in a planar Ising model is well-known to be tractable. Kasteleyn in 1961 showed how to transform this inference problem into a maximum-weight perfect matching problem that can be solved in polynomial time using Edmonds' blossom-shrinking algorithm (Schraudolph & Kamenetsky, 2009). The cycle relaxation is also exact for planar Ising models, and thus the cutting-plane algorithm given in Section 3.4.2, which uses cycle inequalities, will give equivalent results.

Globerson & Jaakkola (2007a) consider the problem of approximating the log-partition function of non-planar binary graphical models by using a decomposition of the graph into planar graphs that cover the original graph, and then optimizing over the planar graphs' weightings. In recent work, Schraudolph (2010) gives a similar reweighting approach for the problem of finding the most likely assignment in non-planar binary graphical models. Both approaches can be shown to be equivalent to optimizing over a relaxation with one cycle consistency constraint for every cycle that appears in at least one of the planar graphs used in the decomposition. Thus, as discussed in Schraudolph (2010), if the planar graphs in the decomposition make up a *cycle basis* for the original graph, then this approach is equivalent to optimizing over the cycle relaxation.

Batra *et al.* (2010) also consider a decomposition approach, but use only outer-planar graphs in their decomposition. Thus, they may need many more graphs before obtaining a cycle basis and being equivalent to the cycle relaxation. We also note that, since outer-planar graphs have tree-width 2, the maximization over outer-planar graphs (which needs to be repeatedly performed) is possible to do with dynamic programming, and the resulting algorithm is very similar to optimizing with triplet clusters (see Chapter 5).

None of the planar graph decomposition approaches address the question of how to choose a good decomposition. In addition, these approaches were applied only to graphical models with binary variables. These methods could likely be extended to non-binary models by using our projection graph technique (Section 3.4). However, this may result in many more planar graphs being needed in the decomposition.

3.6 Discussion

Cycle consistency is powerful, shown by both our experimental results in Section 3.4.3, and by the theoretical results of Barahona & Mahjoub (1986) which show that it exactly gives the marginal polytope for planar Ising models. For graphical models with only binary variables, the cycle inequalities provide a tractable alternative to explicitly optimizing over $\mathrm{TRI}(G)$, which has $O(n^3)$ variables and constraints. Often the pairwise relaxation is close to tight, with only a few cycles inconsistent in the solution to the pairwise LP relaxation. In these cases, using the cutting-plane approach and introducing violated cycle inequalities

one-by-one may be significantly more efficient than either optimizing over $\mathrm{TRI}(G)$ or using planar graph decompositions.

Although the k-ary cycle inequalities are not as tight as the cycle consistency constraints, the results in this section motivate an approach whereby the k-ary cycle inequalities are used to find inconsistent cycles, and then we add a constraint enforcing cycle consistency along these cycles (e.g., by triangulating and adding triplet clusters. See Figure 3-1). It would be interesting to show that whenever the pseudomarginals along a cycle are inconsistent (prior to adding any cycle inequalities along this cycle), there exists some violated k-ary cycle inequality.

An open problem arising from this work is whether it is possible to design a faster separation algorithm for the k-ary cycle inequalities. Our algorithm constructs the full projection graph which has, in the worst case, $O(n2^k)$ vertices, where k is the number of states for each variable. Is it possible to find the most violated k-ary cycle inequality with a running time that is polynomial in k and n? One direction worth exploring is whether the separation algorithm from Section 3.4.2 could be modified so that it only implicitly uses the projection graph.

Chapter 4

Solving LP Relaxations via Dual Decomposition

In the previous chapters, we introduced various LP relaxations for the MAP problem. Although linear programming can be solved in polynomial time using interior point methods or the ellipsoid algorithm, it is often faster to use heuristic solvers that can take advantage of the special structure in each LP instance. We observed in Chapter 3.4.3 that we could quickly solve the pairwise LP relaxation using the dual simplex algorithm. The LPs arising from the protein side-chain placement problem, however, are relatively small – we want to efficiently solve inference problems on the scale of millions of variables. Yanover *et al.* (2006) showed, and we also observed, that off-the-shelf LP solvers (such as dual simplex) have difficulty solving LPs that arise from these larger graphical models, such as those used for protein design or stereo vision.

However, inference problems in graphical models have significant structure, such as sparsity of the graphs, that can be exploited by LP solvers. This chapter addresses an alternative approach to solving LP relaxations which uses the technique of dual decomposition, or Lagrangian relaxation. With dual decomposition, the original problem is broken up into smaller subproblems that can be solved exactly using combinatorial algorithms, thus taking advantage of the structure of the LP. One simple decomposition into subproblems is given by the individual nodes and edges, each of which can be independently maximized efficiently. In models where the variables have large state spaces, this decomposition takes advantage of the fact that the non-negativity and normalization constraints can be optimized over in closed form.

In recent years, a number of dual LP relaxation algorithms have been proposed, and these have been demonstrated to be useful tools for solving large MAP problems (Kolmogorov, 2006; Werner, 2007; Globerson & Jaakkola, 2008; Komodakis & Paragios, 2008). These algorithms can all be understood as dual coordinate descent, but operate in different duals of the same pairwise LP relaxation, arising from different approaches to dual decomposition.

We place these dual algorithms under a common framework so that they can be understood as optimizing the same objective, and demonstrate how to change from one representation to another in a monotone fashion relative to the common objective. This framework permits us to analyze and extend all the methods together as a group. One of the key goals in introducing the new framework is to facilitate the design of new algorithms and modifications that can be used broadly across the different dual formulations.

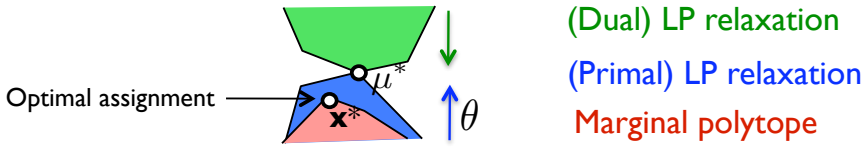


Figure 4-1: Sketch of the dual of the LP relaxation. By LP duality, the original maximization problem (primal) turns into a minimization problem (dual). The dual LP always upper bounds the primal LP optimum, and is equal to it at optimality. Since the pairwise LP relaxation is itself an upper bound on the MAP assignment, we obtain the guarantee that any dual feasible point provides an upper bound on the value of the MAP assignment.

Many graphical models have larger components that can be exactly solved using combinatorial algorithms. One example is a spanning tree of a Markov random field, which can be solved via dynamic programming. Another example is of a subgraph containing just attractive binary potentials, which can be solved via graph cuts. In these cases, rather than using the local updates, we can do global coordinate descent steps using the combinatorial algorithm as a black box. We demonstrate this for spanning trees, where we show that a combinatorial algorithm can be used to perform a block coordinate descent step for all of the dual variables involved in the spanning tree. The resulting block update takes linear time in the size of the tree, and closely resembles max-product.

We exemplify the flexibility of the common framework by providing a monotone version of the TRW algorithm that makes use of the new tree-block update. Moreover, we discuss parallel yet monotone update schemes for the distributed coordinate descent steps.

Finally, the algorithms are only as good as their ability to reconstruct a MAP assignment (primal solution) from the dual optimal solution. We provide conditions for when the MAP assignment can and cannot be found from the dual solutions. The analysis applies to all of the dual algorithms.

This chapter is based in part on material previously published in Sontag & Jaakkola (2009).

4.1 Dual Decomposition for LP Relaxations

In this section we introduce a common framework for understanding several dual linear programs corresponding to LP relaxations. All of the dual LPs that we will discuss in Section 4.1.1 can be viewed as minimizing the following function,

$$J(f) = \sum_i \max_{x_i} f_i(x_i) + \sum_{ij \in E} \max_{x_i, x_j} f_{ij}(x_i, x_j) \tag{4.1}$$

over possible decompositions of the original potential function $\theta(\mathbf{x})$ to single node $f_i(x_i)$ and pairwise $f_{ij}(x_i, x_j)$ potentials. The necessary constraint on these potentials is that they define *sup-reparameterizations*,

$$F(\theta) = \left\{ f : \forall \mathbf{x}, \quad \sum_i f_i(x_i) + \sum_{ij \in E} f_{ij}(x_i, x_j) \geq \sum_{ij \in E} \theta_{ij}(x_i, x_j) \right\}, \tag{4.2}$$

as this guarantees that $J(f)$ provides an upper bound on the value of the MAP assignment, $\theta(\mathbf{x}^*)$. In this chapter, we use the notation $f(\mathbf{x})$ to denote $f_i(x_i) + \sum_{ij \in E} f_{ij}(x_i, x_j)$, and $\theta(\mathbf{x})$ to denote $\sum_{ij \in E} \theta_{ij}(x_i, x_j)$.

Without any other constraints on $F(\theta)$, the optimum of this LP would give the MAP value, i.e.

$$\max_{\mathbf{x}} \theta(\mathbf{x}) = \min_{f \in F(\theta)} J(f). \tag{4.3}$$

To see this, first note that $J(f) \geq \max_{\mathbf{x}} f(\mathbf{x})$. The constraints given by $F(\theta)$ guarantee that $f(\mathbf{x}) \geq \theta(\mathbf{x})$, which implies that $J(f) \geq \max_{\mathbf{x}} \theta(\mathbf{x})$, for all f. We then show that the inequality is tight by demonstrating a feasible solution $f^* \in F(\theta)$ that attains the MAP value. One example is given by $f^*_{ij}(x_i, x_j) = \frac{1}{|E|} \max_{\mathbf{x} \setminus \{x_i, x_j\}} \left\{ \sum_{ij \in E} \theta_{ij}(x_i, x_j) \right\}$ and $f^*_i(x_i) = 0$.

The optimization problem $\min_{f \in F(\theta)} J(f)$, used by Komodakis & Paragios (2008), has one constraint for every assignment $\mathbf{x}$ ensuring that the reparameterization's value on $\mathbf{x}$ is at least as large as $\theta(\mathbf{x})$. Not surprisingly, finding the optimum of this LP is NP-hard.

The key to understanding the different LP formulations are the *additional constraints* that are imposed on $F(\theta)$. It can be shown that simply changing the inequality constraint in Eq. (4.2) to an equality constraint would result in this being the dual of the pairwise relaxation. In the remainder of this section, we will specify three different but related constrained classes, each of which is known to correspond to a (different) dual of the pairwise LP relaxation. Thus, all of these have the same optimal value when maximizing $J(f)$ over them. In Section 4.1.1 we give a proof of these equivalences using a different approach, that of monotonic transformations between the classes.

The first class, $F_L(\theta)$, is a simple reparameterization in terms of single node potentials:

$$F_L(\theta) = \left\{ f : \ \exists \{\delta_{ji}(x_i)\} \text{ s.t. } \begin{array}{l} f_i(x_i) = \sum_{j \in N(i)} \delta_{ji}(x_i), \\ f_{ij}(x_i, x_j) = \theta_{ij}(x_i, x_j) - \delta_{ji}(x_i) - \delta_{ij}(x_j), \end{array} \right\} \tag{4.4}$$

The single node "messages" $\delta_{ji}(x_i)$ and $\delta_{ij}(x_j)$ are subtracted from the edge terms and added to the node terms so as to maintain a valid reparameterization: $\sum_i f_i(x_i) + \sum_{ij \in E} f_{ij}(x_i, x_j) = \sum_{ij \in E} \theta_{ij}(x_i, x_j)$ for all $\mathbf{x}$. Note that, although the graph is undirected, we assume that the edges $ij \in E$ have a canonical ordering of the variables so that, e.g., $\delta_{ij}(x_j)$ refers to a different quantity than $\delta_{ji}(x_i)$.

We show in Appendix A.1 that $\min_{f \in F_L(\theta)} J(f)$ is the dual of the pairwise LP relaxation given by Eq. 2.11 and Eq. 2.10. $F_L(\theta)$ is the same dual linear program formulation introduced by Schlesinger *et al.* in 1976 and optimized by the *max-sum diffusion algorithm* (see Werner, 2007, and references within).

We also introduce a restricted version of $F_L(\theta)$ where the single node potentials are identically zero: $f_i(x_i) = \sum_{j \in N(i)} \delta_{ji}(x_i) = 0$. This corresponds to an additional constraint on how $\delta_{ji}(x_i)$ can be chosen, i.e., they must sum to zero around each node. We call this set of single node (zero) and pairwise potentials $F_{L,E}(\theta)$ as the objective only depends on the edges. Clearly, $F_{L,E}(\theta) \subseteq F_L(\theta)$. An algorithm similar to max-sum diffusion can be given to optimize this representation.

Finally, we introduce a complementary class where the edge terms are zero and the

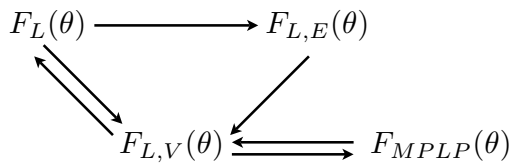


Figure 4-2: Monotone transformations between different representations.

$f_i(x_i)$ are defined in terms of constrained "messages" $\delta_{ji}(x_i)$ as follows:

$$F_{L,V}(\theta) = \left\{ f : \begin{array}{l} f_i(x_i) = \sum_{j \in N(i)} \delta_{ji}(x_i) \\ f_{ij}(x_i, x_j) = 0 \\ \delta_{ji}(x_i) + \delta_{ij}(x_j) \geq \theta_{ij}(x_i, x_j) \end{array} \right\} \tag{4.5}$$

It is easy to see that $F_{L,V}(\theta) \subseteq F(\theta)$. However, in general, potentials in $F_{L,V}(\theta)$ are not members of $F_L(\theta)$. Minimizing $J(f)$ subject to $f \in F_{L,V}(\theta)$ is the dual formulation given by Komodakis & Paragios (2008). It is also closely related to the dual given by Globerson & Jaakkola (2008).

4.1.1 Relating the Different Dual Formulations

We now relate each of the dual LPs in a monotone fashion, showing constructively that we can move from one representation to another while not increasing the common objective $J(f)$. We will show another example of this in Section 4.3.2 for the TRW dual. One of our goals is to clarify the relationship between the different dual formulations and algorithms. In particular, in designing new algorithms one often comes up with a new dual formulation, and it is important to be able to quantify how tight the upper bound provided by the new dual is compared to other well-known relaxations. The monotone transformations presented here can be viewed as a new proof technique for showing this.

Some algorithms, such as the tree block update that we present in Section 4.3, can be conveniently formulated for one of the representations, but because of these results, are applicable to the others as well. In addition, we could smoothly transition between the different representations throughout the optimization of the dual.

These transformations are easiest to illustrate by referring to the messages $\delta = \{\delta_{ji}(x_i)\}$ used in the definitions of each of the classes $F_L(\theta)$, $F_{L,E}(\theta)$, and $F_{L,V}(\theta)$. Recall that $F_L(\theta)$ puts no constraints on the messages δ, while $F_{L,E}$ requires that $\sum_{j \in N(i)} \delta_{ji}(x_i) = 0$, and $F_{L,V}$ requires that $\delta_{ji}(x_i) + \delta_{ij}(x_j) \geq \theta_{ij}(x_i, x_j)$.

The same messages, when used to construct potentials for two different classes (assuming the messages satisfy the constraints for both of these classes), can be used to identify members of both classes. However, the potentials will be different. For example, $f_\delta \in F_{L,V}(\theta)$ has pairwise terms identically zero, while the pairwise terms of $f'_\delta \in F_L(\theta)$ are of the form $f'_{ij}(x_i, x_j) = \theta_{ij}(x_i, x_j) - \delta_{ji}(x_i) - \delta_{ij}(x_j)$.

The transformations are specified in the following propositions, and are illustrated in Figure 4-2. The transformations resemble the sequential updates found in various message passing algorithms, but are generally weaker in terms of the effect on the objective. We begin with a simple transformation that removes the single node functions $f_i(x_i)$. The

notation $f_i^\delta(x_i)$ refers to the single node potential for the reparameterization f_δ.

Proposition 4.1.1. *Consider any $f_\delta \in F_L(\theta)$ with messages $\delta_{ji}(x_i)$. Then $f'_{\delta'} \in F_{L,E}(\theta)$, defined by messages*

$$\delta'_{ji}(x_i) = \delta_{ji}(x_i) - \frac{1}{|N(i)|} \sum_{k \in N(i)} \delta_{ki}(x_i) \tag{4.6}$$

satisfies $J(f_\delta) \geq J(f'_{\delta'})$.

Proof. The constraint $\sum_{j \in N(i)} \delta'_{ji}(x_i) = 0$ is satisfied. The transformation is monotone because

$$\begin{aligned} J(f_\delta) &= \sum_i |N(i)| \max_{x_i} \frac{f_i^\delta(x_i)}{|N(i)|} + \sum_{ij \in E} \max_{x_i, x_j} f_{ij}^\delta(x_i, x_j) \\ &\geq \sum_{ij \in E} \max_{x_i, x_j} \left\{ \frac{f_i^\delta(x_i)}{|N(i)|} + \frac{f_j^\delta(x_j)}{|N(j)|} + f_{ij}^\delta(x_i, x_j) \right\} = J(f'_{\delta'}) \end{aligned}$$

where we split up the node potentials, then combined maximizations (monotonic by convexity of max). □

We can also push all the information into the single node terms, effectively removing the edge functions $f_{ij}(x_i, x_j)$, as shown below.

Proposition 4.1.2. *Consider any $f_\delta \in F_L(\theta)$ or $f_\delta \in F_{L,E}(\theta)$ with messages $\delta_{ji}(x_i)$. Then $f'_{\delta'} \in F_{L,V}(\theta)$, defined by*

$$\delta'_{ji}(x_i) = \frac{1}{2}\delta_{ji}(x_i) + \frac{1}{2} \max_{x_j} \left\{ \theta_{ij}(x_i, x_j) - \delta_{ij}(x_j) \right\} \tag{4.7}$$

satisfies $J(f_\delta) \geq J(f'_{\delta'})$.

Proof. We first show that δ' satisfies the constraint for $F_{L,V}(\theta)$. For any $ij \in E, \hat{x}_i, \hat{x}_j$,

$$\begin{aligned} \delta'_{ji}(x_i) &\geq \frac{1}{2}\delta_{ji}(x_i) + \frac{1}{2}\theta_{ij}(x_i, \hat{x}_j) - \frac{1}{2}\delta_{ij}(\hat{x}_j) \ , \\ \delta'_{ij}(x_j) &\geq \frac{1}{2}\delta_{ij}(x_j) + \frac{1}{2}\theta_{ij}(\hat{x}_i, x_j) - \frac{1}{2}\delta_{ji}(\hat{x}_i) \ . \end{aligned}$$

Thus, by summing these and considering the setting $x_i = \hat{x}_i$, $x_j = \hat{x}_j$, we obtain $\delta'_{ji}(\hat{x}_i) + \delta'_{ij}(\hat{x}_j) \geq \theta_{ij}(\hat{x}_i, \hat{x}_j)$, as desired. To show that $J(f_\delta) \geq J(f'_{\delta'})$, first define

$$g_i^\delta(x_i) = \sum_{j \in N(i)} \left(\frac{1}{2} \max_{x_j} \{\theta_{ij}(x_i, x_j) - \delta_{ij}(x_j)\} - \frac{1}{2}\delta_{ji}(x_i) \right).$$

We then split the edge potential in two, giving

$$\begin{aligned}
J(f_\delta) &= \sum_i \max_{x_i} f_i^\delta(x_i) + \sum_i \sum_{j \in N(i)} \frac{1}{2} \max_{x_i, x_j} f_{ij}^\delta(x_i, x_j) \\
&= \sum_i \max_{x_i} f_i^\delta(x_i) + \sum_i \sum_{j \in N(i)} \max_{x_i} \left(\frac{1}{2} \max_{x_j} \{\theta_{ij}(x_i, x_j) - \delta_{ij}(x_j)\} - \frac{1}{2} \delta_{ji}(x_i) \right) \\
&\geq \sum_i \max_{x_i} f_i^\delta(x_i) + \sum_i \max_{x_i} g_i^\delta(x_i) \\
&\geq \sum_i \max_{x_i} \left\{ f_i^\delta(x_i) + g_i^\delta(x_i) \right\} \\
&= \sum_i \max_{x_i} \left\{ \sum_{j \in N(i)} \delta_{ji}(x_i) + \sum_{j \in N(i)} \left(\frac{1}{2} \max_{x_j} \{\theta_{ij}(x_i, x_j) - \delta_{ij}(x_j)\} - \frac{1}{2} \delta_{ji}(x_i) \right) \right\} \\
&= J(f_{\delta'}).
\end{aligned}$$

Proposition 4.1.3. *Consider any $f_\delta \in F_{L,V}(\theta)$ with messages $\delta_{ji}(x_i)$. Then $f'_\delta \in F_L(\theta)$ defined in terms of the same messages δ, now also modifying edges, satisfies $J(f_\delta) \geq J(f'_\delta)$.*

Proof. The old messages must satisfy the constraint for $F_{L,V}(\theta)$, that $\theta_{ij}(x_i, x_j) - \delta_{ji}(x_i) - \delta_{ij}(x_j) \leq 0$. Thus, the new edge terms for $f'_\delta \in F_L(\theta)$ are all ≤ 0, so maximizing over them only decreases the objective. The node terms stay the same. □

While the transformation given in Proposition 4.1.3 may decrease the objective value, one can show that adding a constant to each message, in particular $\delta'_{ij}(x_j) =$

$$\delta_{ij}(x_j) + \frac{1}{2} \max_{x_i, x_j} \{\theta_{ij}(x_i, x_j) - \delta_{ji}(x_i) - \delta_{ij}(x_j)\}, \tag{4.8}$$

results in a $f'_{\delta'} \in F_L(\theta)$ such that $J(f_\delta) = J(f'_{\delta'})$. This now gives an *exact* mapping from $F_{L,V}(\theta)$, the dual given by Komodakis & Paragios (2008), to $F_L(\theta)$.

4.1.2 MPLP

A pairwise LP relaxation can also be obtained from the point of view of enforcing consistency in a directional manner, considering each edge in two different directions. The associated dual LP corresponds to dividing each edge potential into the associated nodes (Globerson & Jaakkola, 2008). More precisely, the objective is $\min_{f \in F_{MPLP}(\theta)} J(f)$, where the class $F_{MPLP}(\theta)$ is given by

$$\left\{ f : \begin{array}{l} f_i(x_i) = \sum_{j \in N(i)} \max_{x_j} \beta_{ji}(x_j, x_i) \\ f_{ij}(x_i, x_j) = 0 \\ \beta_{ji}(x_j, x_i) + \beta_{ij}(x_i, x_j) = \theta_{ij}(x_i, x_j) \end{array} \right\} \tag{4.9}$$

where each edge potential $\theta_{ij}(x_i, x_j)$ is divided into $\beta_{ji}(x_j, x_i)$ and $\beta_{ij}(x_i, x_j)$ for nodes i and j, respectively.

It is straightforward to show that $f_\beta \in F_{MPLP}(\theta)$ gives a valid sup-reparameterization similar to $F_{L,V}(\theta)$. The two formulations are indeed closely related. We show below how to move from one to the other.

Proposition 4.1.4. *Consider any $f_\beta \in F_{MPLP}(\theta)$ given by the dual variables $\beta_{ji}(x_j, x_i)$. Then $f'_\delta \in F_{L,V}(\theta)$, defined by*

$$\delta_{ji}(x_i) = \max_{x_j} \beta_{ji}(x_j, x_i) \tag{4.10}$$

satisfies $J(f_\beta) = J(f'_\delta)$.

Proof. The objectives are the same because $f_i^\beta(x_i) = f_i^\delta(x_i)$. Also, the constraint is satisfied, since $\delta_{ji}(\hat{x_i}) + \delta_{ij}(\hat{x_j}) \geq \beta_{ji}(\hat{x_j}, \hat{x_i}) + \beta_{ij}(\hat{x_i}, \hat{x_j}) = \theta_{ij}(\hat{x_i}, \hat{x_j})$. □

Proposition 4.1.5. *Consider any $f_\delta \in F_{L,V}(\theta)$ given by the dual variables $\delta_{ji}(x_i)$. Then $f_\beta \in F_{MPLP}(\theta)$, defined by*

$$\beta_{ji}(x_j, x_i) = \theta_{ij}(x_i, x_j) - \delta_{ij}(x_j) \tag{4.11}$$
$$\beta_{ij}(x_i, x_j) = \delta_{ij}(x_j) \tag{4.12}$$

satisfies $J(f_\delta) \geq J(f_\beta)$.

Proof. For any $f_\delta \in F_{L,V}(\theta)$, $\delta_{ji}(x_i)+\delta_{ij}(x_j) \geq \theta_{ij}(x_i, x_j)$. Re-arranging, we have $\theta_{ij}(x_i, x_j)-\delta_{ij}(x_j) \leq \delta_{ji}(x_i)$. Since we defined $\beta_{ji}(x_j, x_i) = \theta_{ij}(x_i, x_j)-\delta_{ij}(x_j)$, we get that $\beta_{ji}(x_j, x_i) \leq \delta_{ji}(x_i)$. Since this holds for all x_j, we conclude that $\max_{x_j} \beta_{ji}(x_j, x_i) \leq \delta_{ji}(x_i)$.

Also, $\delta_{ij}(x_j) \geq \max_{x_i} \beta_{ij}(x_i, x_j)$ trivially. Therefore, for all i and x_i, $\sum_{j \in N(i)} \delta_{ji}(x_i) \geq \sum_{j \in N(i)} \max_{x_j} \beta_{ji}(x_j, x_i)$. □

4.2 Coordinate Descent Algorithms

In this section, we describe two algorithms for performing block coordinate descent in the dual of the pairwise LP relaxation, max-sum diffusion (Schlesinger *et al.*, 1976) and MPLP (Globerson & Jaakkola, 2008). Although MPLP was originally derived as a block coordinate-descent algorithm in F_{MPLP} (see Eq. 4.9), it is equivalently viewed as coordinate-descent in F_L. This alternative view allows us to contrast the choices made by these two different algorithms.

The MPLP updates correspond to taking a block coordinate descent step on $J(f)$ with respect to all dual variables associated to an edge $ij \in E$, i.e. $\delta_{ij}(x_j)$ and $\delta_{ji}(x_i)$ for all values x_i, x_j. Consider the terms in the dual objective where these variables appear,

$$J(\delta_{ij}, \delta_{ji}) = \max_{x_i} \sum_{k \in N(i)} \delta_{ki}(x_i) + \max_{x_j} \sum_{k \in N(j)} \delta_{kj}(x_j) + \max_{x_i, x_j} \Big[\theta_{ij}(x_i, x_j) - \delta_{ji}(x_i) - \delta_{ij}(x_j)\Big].$$

For convenience we define $m_i^{-j}(x_i) = \sum_{k \in N(i) \setminus \{j\}} \delta_{ki}(x_i)$ and $m_j^{-i}(x_j) = \sum_{k \in N(j) \setminus \{i\}} \delta_{kj}(x_j)$. By Jensen's inequality, we have that

$$\begin{aligned} J(\delta_{ij}, \delta_{ji}) &\geq \max_{x_i, x_j} \Big[m_i^{-j}(x_i) + \delta_{ji}(x_i)\Big] + \Big[m_j^{-i}(x_j) + \delta_{ij}(x_j)\Big] + \Big[\theta_{ij}(x_i, x_j) - \delta_{ji}(x_i) - \delta_{ij}(x_j)\Big] \\ &= \max_{x_i, x_j} \Big[m_i^{-j}(x_i) + m_j^{-i}(x_j) + \theta_{ij}(x_i, x_j)\Big] = J_{\min}. \end{aligned}$$

If the three maximizations in $J(\delta_{ij}, \delta_{ji})$ are such that there exists some setting of x_i, x_j that are in the argmax of all three, then it is easy to see that $J(\delta_{ij}, \delta_{ji}) = J_{\min}$, i.e. decrease in the dual will be zero. Otherwise, we will show that there is a way to set the dual variables such that $J(\hat{\delta}_{ij}, \hat{\delta}_{ji}) = J_{\min}$. In particular, we set

$$\hat{\delta}_{ji}(x_i) = -\frac{1}{2} m_i^{-j}(x_i) + \frac{1}{2} \max_{x_j} \left[m_j^{-i}(x_j) + \theta_{ij}(x_i, x_j) \right] \tag{4.13}$$

$$\hat{\delta}_{ij}(x_j) = -\frac{1}{2} m_j^{-i}(x_j) + \frac{1}{2} \max_{x_i} \left[m_i^{-j}(x_i) + \theta_{ij}(x_i, x_j) \right]. \tag{4.14}$$

These updates must be done for all values of x_i, x_j. The first term of $J(\hat{\delta}_{ij}, \hat{\delta}_{ji})$ is then equal to

$$\begin{aligned}
\max_{x_i} \sum_{k \in N(i)} \hat{\delta}_{ki}(x_i) &= \max_{x_i} \left[m_i^{-j}(x_i) + \hat{\delta}_{ji}(x_i) \right] \\
&= \max_{x_i} \left[m_i^{-j}(x_i) - \frac{1}{2} m_i^{-j}(x_i) + \frac{1}{2} \max_{x_j} \left[m_j^{-i}(x_j) + \theta_{ij}(x_i, x_j) \right] \right] \\
&= \max_{x_i} \left[\frac{1}{2} m_i^{-j}(x_i) + \frac{1}{2} \max_{x_j} \left[m_j^{-i}(x_j) + \theta_{ij}(x_i, x_j) \right] \right] \\
&= \frac{1}{2} \max_{x_i, x_j} \left[m_i^{-j}(x_i) + m_j^{-i}(x_j) + \theta_{ij}(x_i, x_j) \right] = \frac{1}{2} J_{\min}
\end{aligned}$$

and the second term of $J(\hat{\delta}_{ij}, \hat{\delta}_{ji})$ can be shown analogously. We now show that the third term $\max_{x_i, x_j} \left[\theta_{ij}(x_i, x_j) - \hat{\delta}_{ji}(x_i) - \hat{\delta}_{ij}(x_j) \right] = 0$. First, note that it can be simplified to be:

$$\begin{aligned}
\frac{1}{2} \max_{x_i, x_j} \Big[\quad & \left(\theta_{ij}(x_i, x_j) + m_i^{-j}(x_i) \right) - \max_{\hat{x}_i} \left(m_i^{-j}(\hat{x}_i) + \theta_{ij}(\hat{x}_i, x_j) \right) + \\
& \left(\theta_{ij}(x_i, x_j) + m_j^{-i}(x_j) \right) - \max_{\hat{x}_j} \left(m_j^{-i}(\hat{x}_j) + \theta_{ij}(x_i, \hat{x}_j) \right) \Big]
\end{aligned}$$

Let $g(x_i, x_j) = \theta_{ij}(x_i, x_j) + m_i^{-j}(x_i)$ and $h(x_i, x_j) = \theta_{ij}(x_i, x_j) + m_j^{-i}(x_i)$. By convexity, we have that $\max_{x_i, x_j} \left[g(x_i, x_j) - \max_{\hat{x}_i} g(\hat{x}_i, x_j) + h(x_i, x_j) - \max_{\hat{x}_j} h(x_i, \hat{x}_j) \right]$

$$\begin{aligned}
&\leq \max_{x_i, x_j} \left[g(x_i, x_j) - \max_{\hat{x}_i} g(\hat{x}_i, x_j) \right] + \max_{x_i, x_j} \left[h(x_i, x_j) - \max_{\hat{x}_j} h(x_i, \hat{x}_j) \right] \\
&= \max_{x_j} \left[\max_{x_i} g(x_i, x_j) - \max_{\hat{x}_i} g(\hat{x}_i, x_j) \right] + \max_{x_i} \left[\max_{x_j} h(x_i, x_j) - \max_{\hat{x}_j} h(x_i, \hat{x}_j) \right] = 0.
\end{aligned}$$

We have thus derived the MPLP updates as block coordinate descent on the dual objective. The MPLP updates keep all of the dual objective in the single node terms $f_i(x_i)$, and each update to edge ij results in $\max_i f_i(x_i) = \max_j f_j(x_j)$, so that, at convergence, the dual objective is equally shared between all of the single node terms. In the subsequent chapters of the thesis, we use the MPLP algorithm to solve the dual LP relaxations.

The max-sum diffusion (MSD) algorithm, in contrast, performs block coordinate descent only for one direction of the dual variables defined on edge ij, e.g. $\delta_{ji}(x_i)$ for all x_i. Consider

1. Obtain max-marginals μ by running max-product on $f_T = \{f_i(x_i), f_{ij}(x_i, x_j) : ij \in T\}$.

2. Update the parameters for $ij \in T$ as follows:

$$f_i^{(t+1)}(x_i) = \frac{1}{n} \log \mu_i(x_i) \tag{4.16}$$

$$f_{ij}^{(t+1)}(x_i, x_j) = \log \mu_{ij}(x_i, x_j) - \frac{n_{j \to i}}{n} \log \mu_i(x_i) - \frac{n_{i \to j}}{n} \log \mu_j(x_j) \tag{4.17}$$

Figure 4-3: Max-product tree block update algorithm. n is the number of nodes in tree, and $n_{j \to i}$ is the number of nodes in the subtree of node i with parent j.

the dual objective

$$\begin{aligned} J(\delta_{ji}) &= \max_{x_i} \sum_{k \in N(i)} \delta_{ki}(x_i) + \max_{x_i, x_j} \Big[\theta_{ij}(x_i, x_j) - \delta_{ji}(x_i) - \delta_{ij}(x_j)\Big] \\ &= \max_{x_i} \Big[m_i^{-j}(x_i) + \delta_{ji}(x_i)\Big] + \max_{x_i, x_j} \Big[\theta_{ij}(x_i, x_j) - \delta_{ji}(x_i) - \delta_{ij}(x_j)\Big]. \end{aligned}$$

As before, by convexity and Jensen's inequality we have that

$$J(\delta_{ji}) \geq \max_{x_i} \Big[m_i^{-j}(x_i) + \max_{x_j} \Big[\theta_{ij}(x_i, x_j) - \delta_{ij}(x_j)\Big]\Big] = J_{\min}.$$

The max-sum diffusion update performs the following, for all x_i:

$$\hat{\delta}_{ji}(x_i) = -\frac{1}{2} m_i^{-j}(x_i) + \frac{1}{2} \max_{x_j} \Big[\theta_{ij}(x_i, x_j) - \delta_{ij}(x_j)\Big]. \tag{4.15}$$

As before, this update can be shown to result in $J(\hat{\delta}_{ji}) = J_{\min}$, with both terms of $J(\hat{\delta}_{ji})$ having value $\frac{1}{2} J_{\min}$. With max-sum diffusion, unlike MPLP, at convergence the dual objective is shared between the edge terms in addition to the single node terms. This difference results in MSD requiring significantly more iterations to converge compared to MPLP.

Finally, we note that although the dual objective $J(f)$ is convex, it is not strictly convex. For this reason, although every update results in a monotonic improvement to the dual objective, it is possible that the dual coordinate descent algorithms converge to a solution that is not dual optimal. We note, however, that there are variants of these dual coordinate descent algorithms that can guarantee convergence to the dual optimum. We refer the reader to Globerson & Jaakkola (2008) and Jojic *et al.* (2010) for further discussion.

4.3 Block Coordinate Descent using Spanning Trees

Most coordinate-descent algorithms for solving the dual LPs perform local operations on the messages, updating edge reparameterizations or messages around each node. This is advantageous for simplicity. However, even when the model is a tree, a large number of

local operations may be needed to reach a dual optimal solution. In this section, we provide block update algorithms for trees, analogous to exact collect-distribute computation on a tree, but leading to a reparameterization rather than max-marginals.

In each step of the algorithm we isolate a tree out of the current reparameterization objective and perform a block update for this tree. Such a tree block update can lead to faster convergence for appropriately chosen trees, and may also help in decoding, as discussed in Section 4.4.

We give two tree block update algorithms. The first algorithm is shown in Figure 4-3. Consider any tree structured model specified by node parameters $f_i(x_i), i \in T$, and edge parameters $f_{ij}(x_i, x_j)$, $ij \in T$. This tree may have been extracted from the current LP dual to perform a block update. The algorithm works by running a forward and backward pass of max-product to compute the max-marginals $\mu_{ij}(x_i, x_j) = \max_{\mathbf{x} \setminus \{i,j\}} \Pr(\mathbf{x})$ for the tree distribution

$$\Pr(\mathbf{x}) = \frac{1}{Z(f_T)} \exp\left\{ \sum_{i \in T} f_i(x_i) + \sum_{ij \in T} f_{ij}(x_i, x_j) \right\},$$

and then uses the max-marginals to construct a reparameterization of the original tree model. After each update, the constant $c = \log(Z(f_T^{(t)})/Z(f_T^{(t+1)}))$ should be added to the dual objective. This can be found by evaluating $f_T^{(t)}(\mathbf{x}) - f_T^{(t+1)}(\mathbf{x})$ for any assignment $\mathbf{x}$.

This approach uses only the standard max-product algorithm to solve the LP relaxation. If applied, for example, with stars around individual nodes rather than spanning trees, this results in one of the simplest dual-coordinate descent algorithms given to date. However, since the tree block updates are used as part of the overall dual LP algorithm, it is important to ensure that the effect of the updates is distributed to subsequent operations as effectively as possible. We found that by instead performing tree block updates directly within the class of $F_L(\theta)$ reparameterizations, we obtain significantly faster running times (see Section 4.3.1).

The second block update algorithm, shown in Figure 4-4, finds a set of $\delta_{ji}(x_i)$ messages such that $f_T^{(t+1)} \in F_L(f_T)$, defined by the $\delta_{ji}(x_i)$, minimizes $J(f_T)$. This algorithm makes use of *directed* trees. We found that choosing a random root works well. The first step is to send max-product messages from the leaves to the root. Then, on the downward pass, we do a series of edge reparameterizations, constructing the $\delta_{ji}(x_i)$ to ensure that each term in the objective is maximized by the MAP assignment. ($c = 0$ for this algorithm.)

Proposition 4.3.1. *The tree block procedures given in Figures 4-3 and 4-4 attain the MAP value for a tree,*

$$\max_{\mathbf{x}} \sum_{i \in T} f_i^{(t)}(x_i) + \sum_{ij \in T} f_{ij}^{(t)}(x_i, x_j) = J(f_T^{(t+1)}) + c,$$

Proof. **Max-product algorithm.** First we show this returns a reparameterization. Recall that n is the number of nodes in the tree, and $n_{j \to i}$ is the number of nodes in the subtree of node i with parent j. Using the fact that $(\sum_{j \in N(i)} n_{j \to i} - 1)/n = |N(i)| - 1$, we get that

$$\exp\{\sum_{i \in T} f_i^{(t+1)}(x_i) + \sum_{ij \in T} f_{ij}^{(t+1)}(x_i, x_j)\} = \prod_{ij \in T} \mu_{ij}(x_i, x_j) \prod_{i \in T} \mu_i(x_i)^{1-|N(i)|} , \qquad (4.21)$$

which, by a special case of the junction-tree theorem, can be shown to be proportional to $\Pr(\mathbf{x})$.

1. Select a root, and orient the edges away from it.

2. Evaluate max-marginal messages toward the root (collect operation). For each $j \in ch(i)$,

$$m_{j\to i}(x_i) = \max_{x_j} \left[f_{ij}(x_i, x_j) + f_j(x_j) + \sum_{k \in ch(j)} m_{k\to j}(x_j) \right]. \tag{4.18}$$

3. Reparameterize away from the root (distribute operation). Set $\tilde{n}_i = n_i$. For each $j \in ch(i)$:

   (a) Define $m^{-j}(x_i) = \sum_{k \in N(i)\setminus j} m_{k\to i}(x_i)$.

   (b) Set

$$\delta_{ji}(x_i) = -\frac{n_j}{\tilde{n}_i}\left[m^{-j}(x_i) + f_i(x_i)\right] + \frac{\tilde{n}_i - n_j}{\tilde{n}_i} \max_{x_j} \left[f_{ij}(x_i, x_j) + f_j(x_j) + m^{-i}(x_j) \right],$$

$$\delta_{ij}(x_j) = \frac{n_j - \tilde{n}_i}{\tilde{n}_i}\left[m^{-i}(x_j) + f_j(x_i)\right] + \frac{n_j}{\tilde{n}_i} \max_{x_i} \left[f_{ij}(x_i, x_j) + f_i(x_i) + m^{-j}(x_i) \right].$$

   (c) Re-define upward $m_{j\to i}(x_i) = \delta_{ji}(x_i)$.
   Define downward $m_{i\to j}(x_j) = \delta_{ij}(x_j)$. Change size of subtree: $\tilde{n}_i = \tilde{n}_i - n_j$.

4. Update the parameters for $ij \in T$ as follows:

$$f_i^{(t+1)}(x_i) = f_i(x_i) + \sum_{j \in N(i)} \delta_{ji}(x_i) \tag{4.19}$$

$$f_{ij}^{(t+1)}(x_i, x_j) = f_{ij}(x_i, x_j) - \delta_{ji}(x_i) - \delta_{ij}(x_j) \tag{4.20}$$

Figure 4-4: Sequential tree block update algorithm. $ch(i)$ denotes the children of i in the tree, and $N(i)$ all neighbors of i in the tree. Also, n_i is the number of nodes in the subtree rooted at i.

Let $\mathbf{x}^*$ be the maximizing assignment of $f_T^{(t)}(\mathbf{x})$. Since $f_i^{(t+1)}(x_i)$ was defined as a max-marginal, we have that x_i^* is a maximizer of $f_i^{(t+1)}(x_i)$. We now show that x_i^*, x_j^* also maximizes $f_{ij}^{(t+1)}(x_i, x_j)$. Note that $f_{ij}^{(t+1)}(x_i^*, x_j^*) = 0$ since $\mu_{ij}(x_i^*, x_j^*) = \mu_i(x_i^*)$ and $n_{j\to i} + n_{i\to j} = n$. However, $\mu_{ij}(x_i, x_j) \le \mu_i(x_i)$ for any x_i, x_j, which implies that $f_{ij}^{(t+1)}(x_i, x_j) \le 0$.

Sequential algorithm. For a tree, the pairwise LP relaxation is tight, so the dual optimal reparameterization necessarily attains the MAP value. Let $\delta_i = \{\delta_{ji}(x_i)\}_{j \in N(i)}$ denote the reparameterization around node i. We can write the objective as a function of the messages as $J(f_T^{(t+1)}) = J(\delta_1, \ldots, \delta_n)$. Set node 1 as the root. The collect operation in the algorithm corresponds to evaluating $\min_{\delta_2,\ldots,\delta_n} J(\delta_1, \ldots, \delta_n)$, which can be done recursively and corresponds to evaluating max-marginal messages. Note that the LP relaxation is also tight for each subtree. δ_1 is then solved exactly at the root node via a series of edge reparameterizations. In the distribute phase of the algorithm, we iteratively instantiate each reparameterization by minimizing $\min_{\delta_{i+1},\ldots,\delta_n} J(\hat{\delta}_1, \ldots, \hat{\delta}_{i-1}, \delta_i, \delta_{i+1}, \ldots, \delta_n)$ with respect to

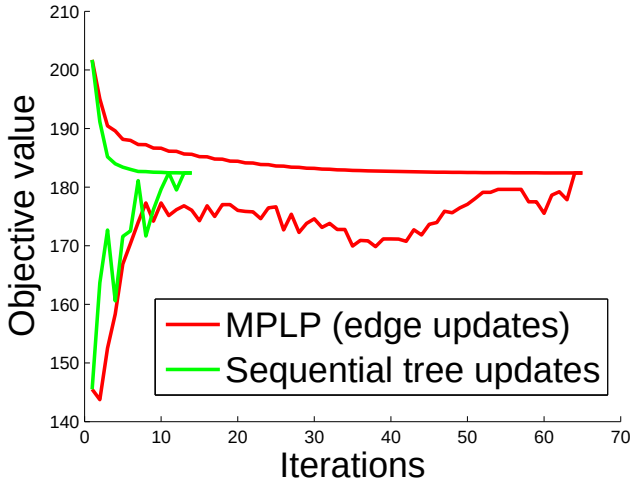


Figure 4-5: Number of iterations to find MAP assignment on a randomly generated 10×10 Ising model with attractive potentials, comparing the sequential tree block algorithm (Figure 4-4) to the edge based MPLP algorithm (Globerson & Jaakkola, 2008).

δ_i when $\hat{\delta}_1, \ldots, \hat{\delta}_{i-1}$ are already fixed. □

4.3.1 Empirical Evaluation

In this section we give an empirical evaluation of the sequential tree block update algorithm on Ising models, a special case of Markov random fields with variables $x_i \in \{-1, +1\}$, where the joint distribution is given by

$$\log \Pr(x_1, \ldots, x_N) \propto \sum_i \lambda_i x_i + \sum_{ij \in E} \lambda_{ij} x_i x_j. \tag{4.22}$$

We considered grid structured Ising models with 100 variables (i.e., a 10×10 grid) and *attractive* couplings, where we fix $\lambda_{ij} = 1$ for all edges. It can be shown that the pairwise LP relaxation is tight whenever $\lambda_{ij} \geq 0$. The single node field terms were randomly generated according to $\lambda_i \sim U[-.8, .8]$.

We performed the tree block updates using spanning trees that are combs of the grid: either all of the rows plus one column (left/right), or all of the columns plus one row (top/down). We cycled through all possible combs, alternating between tree updates on left/right and top/down combs. Each update used a random root. Each iteration corresponds to one tree block update on a comb. For comparison, we solved the same problems using MPLP. One iteration for MPLP corresponds to performing the edge updates once for each edge in the comb.

Averaged over ten trials, we found that the tree block update algorithm solves the LP relaxation (up to an integrality gap of $2e^{-4}$) in 16.5 iterations, compared to 58.3 iterations for MPLP. If we look at the number of iterations for each of the algorithms to first find the MAP assignment, we see that the tree block update algorithm takes only 10 iterations compared to MPLP's 55.5 iterations.

In Figure 4-5 we show the convergence of both algorithms on one of the randomly generated examples. The top line shows the dual objective, and the bottom line shows the value of the integer assignment that is decoded (using a heuristic single node decoding) at each iteration. The tree block update significantly improved the convergence time.

4.3.2 A Monotone Version of TRW

The tree-reweighted max-product algorithm (TRW) (Wainwright *et al.*, 2005a) for MAP problems iteratively combines inference operations carried out on trees of the graph to solve the pairwise LP relaxation. In this section, we show that by using the tree block updates, the TRW algorithm becomes monotone in the dual LP.

Consider a collection of trees T of G, and a distribution over these trees given by $\rho(T)$. For example, we could give equal weight to a few trees that together cover all of the edges in the graph. The dual variables are parameter vectors $\theta^T(\mathbf{x}) = \sum_{i\in T} \theta_i^T(x_i) + \sum_{ij\in T} \theta_{ij}^T(x_i, x_j)$ for each tree T. The TRW dual problem is to minimize $\sum_T \rho(T) \max_{\mathbf{x}} \theta^T(\mathbf{x})$ subject to the reparameterization constraint $\sum_T \rho(T)\theta^T(\mathbf{x}) = \theta(\mathbf{x})$.

The tree-based update algorithm for TRW given by Wainwright *et al.* (2005a) is shown below. ρ_{ij} denotes the probability that an edge $ij \in E$ is part of a tree drawn from the distribution $\rho(T)$, and ρ_i denotes the probability that a node $i \in V$ is part of a tree drawn from the distribution $\rho(T)$ (this is less than 1 if some trees are not spanning).

1. For each tree T, set $\theta_i^T(x_i) = \theta_i(x_i)/\rho_i$ and $\theta_{ij}^T(x_i, x_j) = \theta_{ij}(x_i, x_j)/\rho_{ij}$.

2. Reparameterize each tree distribution. TRW does this by computing max-marginals μ for $\Pr(\mathbf{x}; \theta^T)$ using max-product, then setting

$$\begin{aligned} \hat{\theta}_i^T(x_i) &= \log \mu_i(x_i) \\ \hat{\theta}_{ij}^T(x_i, x_j) &= \log \frac{\mu_{ij}(x_i, x_j)}{\mu_i(x_i)\mu_j(x_j)} \end{aligned} \tag{4.23}$$

3. Average the solutions, and return to Step 1:

$$\begin{aligned} \hat{\theta}_i(x_i) &= \sum_{T: i\in T} \rho(T)\hat{\theta}_i^T(x_i) \\ \hat{\theta}_{ij}(x_i, x_j) &= \sum_{T: ij\in T} \rho(T)\hat{\theta}_{ij}^T(x_i, x_j). \end{aligned} \tag{4.24}$$

Kolmogorov (2006) showed that TRW does not monotonically solve the dual LP. However, if in Step 2 we replace max-product with either of our tree block update algorithms (Figure 4-3 or 4-4) applied to θ^T, we obtain a monotone algorithm. With the max-product tree block update, the new algorithm looks nearly identical to TRW. The following proposition shows that these modified TRW steps are indeed valid and monotone with respect to the common objective $J(\theta)$.

Proposition 4.3.2. *Steps 1-3 described above, using block tree updates, satisfy*

$$\begin{aligned} J(\theta) &\overset{1}{=} \sum_T \rho(T) J(\theta^T) \geq \sum_T \rho(T) \max_{\mathbf{x}} \theta^T(\mathbf{x}) \\ &\overset{2}{=} \sum_T \rho(T) J(\hat{\theta}^T) \overset{3}{\geq} J(\hat{\theta}) \end{aligned} \tag{4.25}$$

where $\hat{\theta} = \sum_T \rho(T)\hat{\theta}^T \in F_L(\theta)$. *Equality in Step 3 would correspond to achieving weak tree agreement (shared maximizing assignment).*

Proof. The first equality $J(\theta) = \sum_T \rho(T) J(\theta^T)$ follows directly by substitution. Each $J(\theta^T) \geq \max_{\mathbf{x}} \theta^T(\mathbf{x})$ since $J(\theta^T)$, for $\theta^T \in F_L(\theta^T)$, is a dual LP relaxation and therefore its value upper bounds the corresponding MAP value for the tree T with parameters θ^T. The pairwise LP relaxation is exact for any tree and therefore the dual objective attains the MAP value $\max_{\mathbf{x}} \theta^T(\mathbf{x}) = \min_{\theta \in F_L(\theta^T)} J(\theta) = J(\hat{\theta}^T)$. $J(\theta)$ is a convex function as a point-wise maximum of potentials and therefore the last inequality corresponds to Jensen's inequality. Jensen's inequality is tight only if all the tree models being averaged have at least one common maximizing local assignment for each pairwise and single node potential terms. This is weak tree agreement. The fact that $\hat{\theta} \in F_L(\theta)$ follows from Eq. (4.24) and because the block tree updates return a $\hat{\theta}^T$ that is a reparameterization of θ^T. □

A monotone version of the algorithm, known as TRW-S, was introduced by Kolmogorov (2006). One key difference is that in Step 3, only one node or one edge is averaged, and then max-product is run again on each tree. TRW-S is monotone in $\sum_T \rho(T) \max_{\mathbf{x}} \theta^T(\mathbf{x})$, but may not be for $J(\theta)$, depending on the reparameterization used. By using a slightly different weighting scheme, the algorithm in Figure 4-3 can be used to give an optimal reparameterization where one edge st has $\hat{\theta}^T_{st}(x_s, x_t) = \log \mu^T_{st}(x_s, x_t)$ (analogously for one node). Both TRW-S and this algorithm give the same solution for $\hat{\theta}_{st}(x_s, x_t)$. However, TRW-S would stop here, while our algorithm also averages over the other edges, obtaining a possibly larger (and never smaller) decrease in the dual objective. When the trees are monotonic chains, Kolmogorov (2006) shows how TRW-S can be implemented much more efficiently.

We observed empirically that using the tree block update algorithms sequentially to solve $F_L(\theta)$ converges more quickly than using them in parallel and averaging. However, the modified TRW algorithm is ideally suited for parallel processing, allowing us in Step 2 to independently find the optimal reparameterizations for each tree. By modifying the particular choice of trees, reparameterizations, and averaging steps, this framework could be used to derive various new parallel coordinate descent algorithms.

Our approach is also related to Komodakis *et al.* (2010), who use a subgradient method to optimize the TRW dual problem. The subgradient approach also alternates between solving the MAP problem for each tree and reparameterizing the distribution. However, although the subgradient approach uses max-product to solve the MAP problem, it only uses the most likely assignment within the update, instead of the full dual solution as we do here. Our approach could be considered the coordinate-descent algorithm that corresponds to their subgradient method.

4.4 Primal Recovery

If the pairwise LP relaxation has a unique optimal solution and it is the MAP assignment, we could find it simply by solving the primal linear program. We use dual algorithms for reasons of *efficiency*. However, the dual solution is only useful if it helps us *find* the MAP assignment. In this section, we characterize when it is possible to decode the MAP assignment from the dual solution, using the common framework given by the dual $\min_{f \in F_L(\theta)} J(f)$. By using the transformations of the previous section, these results can be shown to apply to all of the algorithms discussed in this chapter.

Duality in linear programming specifies *complementary slackness* conditions that every primal and dual optimal solution must satisfy. In particular, it can be shown that for any optimal μ^* for the pairwise LP relaxation given by Eq. 2.11 and any optimal f^* for the dual $\min_{f \in F_L(\theta)} J(f)$,

$$\begin{aligned} \mu_i^*(\hat{x}_i) > 0 \;\; &\Rightarrow \;\; f_i^*(\hat{x}_i) = \max_{x_i} f_i^*(x_i), \\ \mu_{ij}^*(\hat{x}_i, \hat{x}_j) > 0 \;\; &\Rightarrow \;\; f_{ij}^*(\hat{x}_i, \hat{x}_j) = \max_{x_i, x_j} f_{ij}^*(x_i, x_j). \end{aligned} \tag{4.26}$$

We will use these complementary slackness conditions to give conditions under which we can recover the MAP assignment from the dual optimal solution.

Definition 4.4.1. *We say that $f \in F_L(\theta)$* locally supports $\mathbf{x}^*$ *if $f_{ij}(x_i^*, x_j^*) \geq f_{ij}(x_i, x_j)$ for all $ij \in E$, x_i, x_j, and $f_i(x_i^*) \geq f_i(x_i)$ for all $i \in V, x_i$.*

Our claims refer to the pairwise LP relaxation being *tight* for some θ. By this, we mean that the dual value $\min_{f \in F_L(\theta)} J(f) = J(f^*)$ equals the MAP value. As a result, each MAP assignment (there can be more than one) represents a primal optimal solution. In addition, there may be fractional solutions that are also primal optimal, i.e., attain the MAP value.

Lemma 4.4.2. *When the pairwise LP relaxation is tight, every optimal $f^* \in F_L(\theta)$ locally supports every MAP assignment $\mathbf{x}^*$. Conversely, if any dual feasible $f \in F_L(\theta)$ supports an assignment $\mathbf{x}^*$, then f is optimal, the LP is tight, and $\mathbf{x}^*$ is a MAP assignment.*

Proof. The lemma is a simple consequence of complementary slackness. To show the first part, apply Eq. (4.26) to each MAP assignment $\mathbf{x}^*$. Since $\mu_{ij}^*(x_i^*, x_j^*) = 1$ for the corresponding primal solution, x_i^*, x_j^* must maximize $f_{ij}^*(x_i, x_j)$. The second part follows from the fact any primal solution that $f \in F_L(\theta)$ supports attains the same value. □

Lemma 4.4.2 is closely related to decodability results for convex max-product BP (Weiss *et al.*, 2007). The beliefs at the fixed-points of convex max-product BP can be shown to give a dual feasible (not necessarily optimal) $f \in F_L(\theta)$ with the property that, if the beliefs support an assignment, f does too. Thus, this must be a MAP assignment. Our result also characterizes when it is possible to find the MAP assignment with convex max-product BP by looking for supporting assignments: only when the LP relaxation is tight.

The search for a locally supporting assignment may be formulated as a satisfiability problem, satisfiable only when the LP is tight. If the variables are binary, the corresponding 2SAT problem can be solved in linear time (Johnson, 2008). However, when some variables are non-binary, finding a satisfying assignment may be intractable. We now look at a setting where reading off the solution from f^* is indeed straightforward.

Definition 4.4.3. *We say that $f \in F_L(\theta)$ is node* locally decodable *to $\mathbf{x}^*$ if $f_i(x_i^*) > f_i(x_i)$ for all $i, x_i \neq x_i^*$.*

The definition for edge local decodability is analogous. If solving the dual problem results in a locally decodable solution, then we can easily construct the MAP assignment from each node or edge (cf. Lemma 4.4.2). However, in many cases this cannot happen.

Lemma 4.4.4. *A dual optimal $f^* \in F_L(\theta)$ can be node or edge locally decodable only if the MAP assignment is unique and the pairwise LP is tight.*

Proof. Either node or edge local decodability suffices to uniquely determine a supporting assignment. If any dual feasible $f \in F_L(\theta)$ supports an assignment, then the assignment attains the dual value, thus the LP must be tight. When the LP is tight, the optimal f^* has to support all the MAP assignments by Lemma 4.4.2. Thus f^* can be locally decodable only if the MAP assignment is unique. □

But, how can we find a locally decodable solution when one exists? If the MAP assignment $\mathbf{x}^*$ is unique, then evaluating max-marginals is one way to get a locally decodable solution $f^* \in F(\theta)$. Under some conditions, this holds for a dual optimal $f^* \in F_L(\theta)$ as well.

Theorem 4.4.5. *Assume the MAP assignment $\mathbf{x}^*$ is unique. Then,*

1. *if the pairwise LP is tight and has a unique solution, there exists $f^* \in F_L(\theta)$ that is locally decodable to $\mathbf{x}^*$.*

2. *for a tree structured model, the tree block updates given in Section 4.3 construct $f^* \in F_L(\theta)$ which is node locally decodable.*

Proof. The first claim follows from strict complementary slackness (Vanderbei, 2007). We can always find a primal-dual pair (μ^*, f^*) that satisfies the implication in Eq. (4.26) both ways. Since μ^* is unique, it corresponds to the unique MAP assignment $\mathbf{x}^*$, and the strict complementary slackness guarantees that $f^* \in F_L(\theta)$ is locally decodable.

The second claim trivially holds for the max-product tree block update since each $f_i^{(t+1)}(x_i)$ is given by the single node max-marginals and MAP is unique.

We now show the second claim for the sequential algorithm. Assume without loss of generality that the node potentials $f_i(x_i) = 0$. The block tree update contracts a tree into an edge by propagating max-marginals towards edge ij. Let $\hat{\theta}_{ij}(x_i, x_j)$ be $\sum_{k \in N(i) \setminus j} m_{k \to i}(x_i) + f_{ij}(x_i, x_j) + \sum_{k \in N(j) \setminus i} m_{k \to j}(x_j)$. Since the MAP assignment is unique, $\hat{\theta}_{ij}(x_i, x_j)$ must have the unique maximizer x_i^*, x_j^*. The edge reparameterization sets $\delta_{ij}(x_j)$ and $\delta_{ji}(x_i)$ so that the updated single node term $\hat{\theta}_i(x_i) = \sum_{k \in N(i) \setminus j} m_{k \to i}(x_i) + \delta_{ji}(x_i) \propto \max_{x_j} \hat{\theta}_{ij}(x_i, x_j)$. Thus, $\hat{\theta}_i(x_i)$ uniquely decodes to x_i^*.

Next we show that the subtrees associated with i and j, after setting $\delta_{ji}(x_i)$ and $\delta_{ij}(x_j)$, also uniquely decode to $\mathbf{x}^*$. The subtree rooted at i has a max-marginal of $\hat{\theta}_i(x_i)$ for node i. Thus, the MAP assignment for this subtree must have $x_i = x_i^*$. The remaining variables' maximizers are independent of how we set $\delta_{ji}(x_i)$ once the assignment to x_i is fixed, and so must also be maximized at $\mathbf{x}^*$. We can now apply this argument recursively. After the last edge incident on node i is updated, $\hat{\theta}_i(x_i)$ equals $f_i^{(t+1)}(x_i)$, maximized at x_i^*. □

A slightly different tree block update constructs a solution that is edge locally decodable. The above theorem shows that we can efficiently construct locally decodable dual solutions for trees. This could also be useful for non-tree models if repeated applications of the tree block updates move towards solutions that are locally decodable. An interesting open problem is to design algorithms that are guaranteed to return a solution that is locally decodable, for general graphs.

4.5 Discussion

We have placed several dual formulations of the pairwise LP relaxation for MAP under a unified functional view. As a result, algorithms for these can be understood as optimizing a common objective, and analyzed theoretically as a group.

There are a number of immediate generalizations of this work. For example, the generalization to non-pairwise models is straightforward. Also, if the pairwise LP relaxation is not tight, we may wish to include higher-order cluster consistency constraints. The functional characterization can be extended to this setting, with similar equivalence transformations as presented here. The tree-block update scheme would then be given for hyper-trees.

Cycles are typically difficult to fully optimize using local block coordinate-descent algorithms. We believe that efficient block updates, similar to those given in this chapter, can be derived directly for cycles instead of trees. Finally, much of our work here contributes to inference problems involving marginals as well.

This framework has recently been used by Yarkony *et al.* (2010) to give a new dual coordinate-descent algorithm for solving the LP relaxation. The main insight for the new algorithm is that one can duplicate some of the nodes in the graph (a "splitting" operation), thereby obtaining a spanning tree involving *all edges* for which we can do a block coordinate-descent step using dynamic programming. The algorithm can be viewed as moving back and forth between $J(f)$ and this spanning tree representation.

Chapter 5

Tightening LP Relaxations for MAP using Message Passing

We showed in the previous chapter how the pairwise LP relaxation can be efficiently solved using message-passing algorithms that operate in the *dual* of the LP relaxation. Using these algorithms, we can now solve LP relaxations of large-scale problems where standard, off-the-shelf LP solvers could not previously be used. However, despite the success of LP relaxations, there are many real-world problems for which the pairwise LP relaxation is of limited utility in solving the MAP problem. For example, in a database of 97 protein design problems studied in Yanover *et al.* (2006), the pairwise LP relaxation allowed finding the MAP in only 2 cases.

One way to obtain tighter relaxations is to use *cluster-based* LP relaxations, where local consistency is enforced between cluster marginals (c.f. Section 2.5.2). As the size of the clusters grow, this leads to tighter and tighter relaxations. Furthermore, message-passing algorithms can still be used to solve these cluster-based relaxations, with messages now being sent between clusters and not individual nodes. Unfortunately, the computational cost increases exponentially with the size of the clusters, and for many real-world problems this severely limits the number of large clusters that can be feasibly incorporated into the approximation. For example, in the protein design database studied in Yanover *et al.* (2006), each node has around 100 states, so even a cluster of only 3 variables would have 10^6 states. Clearly we cannot use too many such clusters in our approximation.

In this chapter we propose a cluster-pursuit method where clusters are incrementally added to the relaxation, and where we only add clusters that are guaranteed to improve the approximation. Similar to the work of Welling (2004) who worked on region-pursuit for sum-product generalized belief propagation (Yedidia *et al.*, 2005), we show how to use the messages from a given cluster-based approximation to decide which cluster to add next. In addition, by working with a message-passing algorithm based on dual coordinate descent, we monotonically decrease an upper bound on the MAP value. Unlike the cutting-plane algorithm from Chapter 3, which incrementally added constraints to the relaxation, the new dual algorithm is a *column generation* method, incrementally adding new variables as well as constraints.

This chapter highlights a key advantage of the LP-relaxation based message-passing algorithms over max-product belief propagation (BP). Although BP is known to give good results in many cases when the pairwise LP relaxation is tight, such as for bipartite matching (Bayati *et al.*, 2008), it gives much worse results in models where there is frustration

(Murphy *et al.*, 1999). These are precisely the cases when the pairwise LP relaxation is not tight. By using a message-passing algorithm that directly solves the LP relaxation, we have a clear way of *improving* the approximation, by tightening the relaxation. The striking result of this chapter is that, for many real-world problems, adding a few well-chosen constraints allows us to *exactly* find the MAP assignment. Using these techniques, we are able to solve nearly all of the protein design problems considered in Yanover *et al.* (2006). Such a result would have been inconceivable with BP.

This chapter is based in part on material previously published in Sontag *et al.* (2008).

5.1 MAP and its LP Relaxation

As in the earlier chapters, we consider graph-structured functions over n discrete variables $\{X_1, \ldots, X_n\}$ of the form

$$\theta(\mathbf{x}) = \sum_{ij \in E} \theta_{ij}(x_i, x_j) + \sum_{i \in V} \theta_i(x_i) \ . \tag{5.1}$$

Our goal is to find the MAP assignment, $\mathbf{x}_M$, that maximizes the function $\theta(\mathbf{x})$.

We showed in Section 2.2 that the MAP problem can be equivalently formulated as a linear program,

$$\max_{\mathbf{x}} \theta(\mathbf{x}) = \max_{\boldsymbol{\mu} \in \mathcal{M}(G)} \boldsymbol{\mu} \cdot \boldsymbol{\theta} \ , \tag{5.2}$$

where $\boldsymbol{\mu} \cdot \boldsymbol{\theta} = \sum_{ij \in E} \sum_{x_i, x_j} \theta_{ij}(x_i, x_j) \mu_{ij}(x_i, x_j) + \sum_i \sum_{x_i} \mu_i(x_i) \theta_i(x_i)$ and $\mathcal{M}(G)$ is the marginal polytope, the set of $\boldsymbol{\mu}$ that arise from some joint distribution:

$$\mathcal{M}(G) = \left\{ \boldsymbol{\mu} \mid \exists \Pr(\mathbf{x}) \text{ s.t. } \begin{array}{l} \Pr(x_i, x_j) = \mu_{ij}(x_i, x_j) \\ \Pr(x_i) = \mu_i(x_i) \end{array} \right\}. \tag{5.3}$$

The idea in LP relaxations is to relax the difficult global constraint that the marginals in $\boldsymbol{\mu}$ arise from some common joint distribution. Instead, we enforce this only over some subsets of variables that we refer to as clusters. More precisely, we introduce auxiliary distributions over clusters of variables and constrain the edge distributions $\mu_{ij}(x_i, x_j)$ associated with each cluster to arise as marginals from the cluster distribution.[1] Let $\mathcal{C}$ be a set of clusters such that each $c \in \mathcal{C}$ is a subset of $\{1, \ldots, n\}$, and let $\tau_c(\mathbf{x}_c)$ be any distribution over the variables in c. Define $\mathcal{M}_{\mathcal{C}}(G)$ as

$$\mathcal{M}_{\mathcal{C}}(G) = \left\{ \boldsymbol{\mu} \geq 0 \ \middle| \ \exists \boldsymbol{\tau} \geq 0, \ \begin{array}{ll} \sum_{x_j} \mu_{ij}(x_i, x_j) = \mu_i(x_i) & \forall ij \in E, x_i \\ \sum_{x_i} \mu_{ij}(x_i, x_j) = \mu_j(x_j) & \forall ij \in E, x_j \\ \sum_{x_i} \mu_i(x_i) = 1 & \forall i \in V \\ \sum_{\mathbf{x}_{c \setminus i,j}} \tau_c(\mathbf{x}_c) = \mu_{ij}(x_i, x_j) & \forall c, \ \ i, j \in c \text{ s.t. } ij \in E, \ x_i, x_j \end{array} \right\}$$

It is easy to see that $\mathcal{M}_{\mathcal{C}}(G)$ is an outer bound on $\mathcal{M}(G)$, namely $\mathcal{M}_{\mathcal{C}}(G) \supseteq \mathcal{M}(G)$. As we add more clusters to $\mathcal{C}$ the relaxation of the marginal polytope becomes tighter (c.f. Section 2.5.2). Note that similar constraints should be imposed on the cluster marginals, i.e., they themselves should arise as marginals from some joint distribution. To exactly represent the marginal polytope, such a hierarchy of auxiliary clusters would require clusters

[1] Each edge may participate in multiple clusters.

of size equal to the treewidth of the graph. In this chapter, we will not generate such a hierarchy but instead use the clusters to constrain only the associated edge marginals.

5.1.1 Choosing Clusters in the LP Relaxation

Adding a cluster to the relaxation $\mathcal{M}_{\mathcal{C}}(G)$ requires computations that scale with the number of possible cluster states. The choice of clusters should therefore be guided by both how much we are able to constrain the marginal polytope, as well as the computational cost of handling larger clusters. We will consider a specific scenario where the clusters are selected from a pre-defined set of possible clusters $\mathcal{C}_0$ such as triplet clusters. However, we will ideally not want to use all of the clusters in $\mathcal{C}_0$, but instead add them gradually based on some ranking criterion.

The best ranking of clusters is problem dependent. In other words, we would like to choose the subset of clusters which will give us the best possible approximation to a particular MAP problem. We seek to *iteratively* improve the approximation, using our current beliefs to guide which clusters to add. The advantage of iteratively selecting the clusters is that we add them only up to the point that the relaxed LP has an integral solution.

In Chapter 3 we suggested an approach for incrementally adding valid inequalities for the marginal polytope using a cutting-plane algorithm in the primal LP. However, the message-passing algorithms that we described in Chapter 4 solve the *dual* of the LP relaxation, and it is often difficult to obtain a primal solution to use within the (primal-based) separation algorithms.

In the next section we present a method that incrementally adds cluster consistency constraints, but which works exclusively within the dual LP. The key idea is that the dual LP provides an upper bound on the MAP value, and we seek to choose clusters to most effectively minimize this bound. This differs from the primal-based separation algorithms, which attempted to find the most *violated* inequality. An analogous bound minimization strategy for cluster consistency constraints is problematic in the primal where we would have to assess how much *less* the maximum of the tighter relaxation is due to including additional variables and constraints. In other words, obtaining a certificate for improvement is difficult in the primal.

Finally, we can "warm start" our optimization scheme after each cluster addition in order to avoid re-solving the dual LP. We do this by reusing the dual variables calculated in the previous iterations which did not have the new clusters.

5.2 Dual LP Relaxation

The obstacles to working in the primal LP lead us to consider the dual of the LP relaxation. Different formulations of the primal LP have lead to different dual LPs, each with efficient message-passing algorithms for solving them (c.f. Chapter 4). In this chapter we focus on a particular dual formulation by Globerson & Jaakkola (2008) which has the advantage that the message-passing algorithm corresponds to performing coordinate-descent in the dual LP. Our dual algorithm will address many of the problems that were inherent in the primal approaches, giving us:

1. Monotonically decreasing upper bound on MAP.
2. Choosing clusters which give a guaranteed bound improvement.

3. Simple "warm start" of tighter relaxation.

4. An efficient algorithm that scales to very large problems.

Although we use the dual formulation given by Globerson & Jaakkola (2008), it is straightforward to give the corresponding algorithm for any of the other duals discussed in Chapter 4. In Appendix A.2 we describe a different dual formulation that results in a slightly simpler algorithm than the one described here.

5.2.1 The Generalized MPLP Algorithm

The generalized Max-Product LP (MPLP) message-passing algorithm, introduced in Globerson & Jaakkola (2008), decreases the dual objective of the cluster-based LP relaxation at every iteration. This monotone property makes it ideal for adding clusters since we can initialize the new messages such that the dual value is monotonically decreased.

Another key advantage of working in the dual is that the dual objective gives us a certificate of optimality. Namely, if we find an assignment $\mathbf{x}$ such that $\theta(\mathbf{x})$ is equal to the dual objective, we are guaranteed that $\mathbf{x}$ is the MAP assignment (since the dual objective upper bounds the MAP value). Indeed, using this property we show in our experiments that MAP assignments can be found for nearly all of the problems we consider.

We next describe the generalized MPLP algorithm for the special case of clusters comprised of three nodes. Although the algorithm applies to general clusters, we focus on triplets for simplicity, and because these are the clusters used in our experimental results.

MPLP passes the following types of messages:

- **Edge to Node**: For every edge $e \in E$ (e denotes two indices in V) and every node $i \in e$, we have a message $\lambda_{e \to i}(x_i)$.
- **Edge to Edge**: For every edge $e \in E$, we have a message $\lambda_{e \to e}(\mathbf{x}_e)$ (where $\mathbf{x}_e$ is shorthand for x_i, x_j, and i and j are the nodes in the edge).
- **Triplet to Edge**: For every triplet cluster $c \in \mathcal{C}$, and every edge $e \in c$, we have a message $\lambda_{c \to e}(\mathbf{x}_e)$.

The updates for these messages are given in Figure 5-1. To guarantee that the dual objective decreases, all messages from a given edge must be sent simultaneously, as well as all messages from a triplet to its three edges.

The dual objective[2] in the Globerson & Jaakkola (2008) formulation that is decreased in every iteration is $g(\boldsymbol{\lambda}) =$

$$\sum_{i \in V} \max_{x_i} \left[\theta_i(x_i) + \sum_{k \in N(i)} \lambda_{ki \to i}(x_i) \right] + \sum_{e \in E} \max_{\mathbf{x}_e} \left[\theta_e(\mathbf{x}_e) + \lambda_{e \to e}(\mathbf{x}_e) + \sum_{c : e \in c} \lambda_{c \to e}(\mathbf{x}_e) \right] , \quad (5.4)$$

[2] We give a slightly simpler – but equivalent – dual than that originally presented in Globerson & Jaakkola (2008). The first change was to explicitly put the edge potential $\theta_e(x_e)$ in the objective. To see this, substitute $\lambda'_{ij \to ij}(x_i, x_j) = \theta_{ij}(x_i, x_j) + \lambda_{ij \to ij}(x_i, x_j)$ in Eq. 5.4. The constraint in Eq. 5.5 becomes $\lambda_{ij \to i}(x_i) + \lambda_{ij \to j}(x_j) + \lambda'_{ij \to ij}(x_i, x_j) \geq \theta_{ij}(x_i, x_j)$. We then obtained inequalities by substituting $\lambda_{ij \to i}(x_i)$ – which in Globerson & Jaakkola (2008) was defined as $\max_{x_j} \beta_{ji}(x_j, x_i)$ – for $\beta_{ji}(x_j, x_i)$, and similarly for the other variables. These two duals can be seen to be equivalent using transformations similar to those of Section 4.1.2. Also, the updates given in Figure 5-1 can be shown to correspond to coordinate descent in the new dual using a derivation similar to that of Section 4.2.

- **Edge to Node**: For every edge $ij \in E$ and node i (or j) in the edge:

$$\lambda_{ij\to i}(x_i) \leftarrow -\frac{2}{3}\Big(\lambda_i^{-j}(x_i) + \theta_i(x_i)\Big) + \frac{1}{3}\max_{x_j}\Big[\sum_{c:ij\in c} \lambda_{c\to ij}(x_i,x_j) + \lambda_j^{-i}(x_j) + \theta_{ij}(x_i,x_j) + \theta_j(x_j)\Big]$$

  where $\lambda_i^{-j}(x_i)$ is the sum of edge-to-node messages into i that are not from edge ij, namely: $\lambda_i^{-j}(x_i) = \sum_{k\in N(i)\setminus j} \lambda_{ik\to i}(x_i)$.

- **Edge to Edge**: For every edge $ij \in E$:

$$\lambda_{ij\to ij}(x_i,x_j) \leftarrow -\frac{2}{3}\sum_{c:ij\in c} \lambda_{c\to ij}(x_i,x_j) + \frac{1}{3}\Big[\lambda_j^{-i}(x_j) + \lambda_i^{-j}(x_i) + \theta_{ij}(x_i,x_j) + \theta_i(x_i) + \theta_j(x_j)\Big]$$

- **Triplet to Edge**: For every triplet $c \in \mathcal{C}$ and every edge $e \in c$:

$$\lambda_{c\to e}(\mathbf{x}_e) \leftarrow -\frac{2}{3}\Big(\lambda_{e\to e}(\mathbf{x}_e) + \sum_{\substack{c'\neq c\\ e\in c'}} \lambda_{c'\to e}(\mathbf{x}_e)\Big) + \frac{1}{3}\max_{\mathbf{x}_{c\setminus e}}\Big[\sum_{e'\in c\setminus e}\Big(\lambda_{e'\to e'}(\mathbf{x}_{e'}) + \sum_{\substack{c'\neq c\\ e'\in c'}} \lambda_{c'\to e'}(\mathbf{x}_{e'})\Big)\Big]$$

Figure 5-1: The generalized MPLP updates for an LP relaxation with three node clusters.

subject to the constraints:

$$\lambda_{ij\to i}(x_i) + \lambda_{ij\to j}(x_j) + \lambda_{ij\to ij}(x_i,x_j) \geq 0 \quad \forall ij \in E, x_i, x_j \ , \tag{5.5}$$

$$\sum_{e\in c} \lambda_{c\to e}(\mathbf{x}_e) \geq 0 \quad \forall \text{ clusters } c, \mathbf{x}_c \ . \tag{5.6}$$

In our experiments, we initialize all of the dual variables (messages) to zero, which satisfies the dual feasibility constraints.

By LP duality, there exists a value of $\boldsymbol{\lambda}$ such that $g(\boldsymbol{\lambda})$ is equal to the optimum of the corresponding primal LP. Although the MPLP updates decrease the objective at every iteration, they may converge to a $\boldsymbol{\lambda}$ that is not dual optimal, as discussed in Globerson & Jaakkola (2008). However, as we will show in the experiments, our procedure often finds the exact MAP solution, and therefore also achieves the primal optimum in these cases.

5.2.2 Choosing Clusters in the Dual LP Relaxation

In this section we provide a very simple procedure that allows adding clusters to MPLP, while satisfying the algorithmic properties in the beginning of Section 5.2.

Assume we have a set of triplet clusters $\mathcal{C}$ and now wish to add a new triplet. Denote the

messages before adding the new triplet by $\boldsymbol{\lambda}_t$. Two questions naturally arise. The first is: assuming we decide to add a given triplet, how do we set $\boldsymbol{\lambda}_{t+1}$ such that the dual objective retains its previous value $g(\boldsymbol{\lambda}_t)$. The second question is how to choose the new triplet to add.

The initialization problem is straightforward. Simply set $\boldsymbol{\lambda}_{t+1}$ to equal $\boldsymbol{\lambda}_t$ for all messages from triplets and edges in the previous run, and set $\boldsymbol{\lambda}_{t+1}$ for the messages from the new triplet to its edges to zero. The new dual variables immediately satisfy the feasibility constraints given in Eq. 5.6. This also clearly results in $g(\boldsymbol{\lambda}_{t+1}) = g(\boldsymbol{\lambda}_t)$.

In order to choose a *good* triplet, one strategy would be to add different triplets and run MPLP until convergence to find the one that decreases the objective the most. However, this may be computationally costly and, as we show in the experiments, is not necessary. Instead, the criterion we use is to consider the decrease in value that results from just sending messages from the triplet c to its edges (while keeping all other messages fixed).

The decrease in $g(\boldsymbol{\lambda})$ resulting from such an update has a simple form, as we show next. Assume we are considering adding a triplet c. For every edge $e \in c$, define $b_e(\mathbf{x}_e)$ to be

$$b_e(\mathbf{x}_e) = \lambda_{e \to e}(\mathbf{x}_e) + \sum_{c' : e \in c'} \lambda_{c' \to e}(\mathbf{x}_e) \ , \tag{5.7}$$

where the summation over clusters c' does not include c (those messages are initially zero). The decrease in $g(\boldsymbol{\lambda})$ corresponding to updating only messages from c to the edges $e \in c$ can be shown to be

$$d(c) = \sum_{e \in c} \max_{\mathbf{x}_e} b_e(\mathbf{x}_e) - \max_{\mathbf{x}_c} \left[\sum_{e \in c} b_e(\mathbf{x}_e) \right] \ . \tag{5.8}$$

The above corresponds to the difference between independently maximizing each edge and jointly maximizing over the three edges. Thus $d(c)$ is a lower bound on the improvement in the dual objective if we were to add triplet c. Our algorithm will therefore add the triplet c that maximizes $d(c)$.

5.2.3 The Dual Algorithm

We now present the complete algorithm for adding clusters and optimizing over them. Let $\mathcal{C}_0$ be the predefined set of triplet clusters that we will consider adding to our relaxation, and let $\mathcal{C}_L$ be the initial relaxation consisting of only edge clusters (pairwise local consistency).

1. Run MPLP until convergence using the $\mathcal{C}_L$ clusters.

2. Find an integral solution $\mathbf{x}$ by locally maximizing the single node beliefs $b_i(x_i)$, where $b_i(x_i) = \theta_i(x_i) + \sum_{k \in N(i)} \lambda_{ki \to i}(x_i)$. Ties are broken arbitrarily.

3. If the dual objective $g(\boldsymbol{\lambda}_t)$ is sufficiently close to the primal objective $\theta(\mathbf{x})$, terminate (since $\mathbf{x}$ is approximately the MAP).

4. Add the cluster $c \in \mathcal{C}_0$ with the largest guaranteed bound improvement, $d(c)$, to the relaxation.

5. Construct "warm start" messages $\boldsymbol{\lambda}_{t+1}$ from $\boldsymbol{\lambda}_t$.

6. Run MPLP for N iterations, and return to **2**.

Note that we obtain (at least) the promised bound improvement $d(c)$ within the first iteration of step **6**. By allowing MPLP to run for N iterations, the effect of adding the cluster will be propagated throughout the model, obtaining an additional decrease in the bound. Since the MPLP updates correspond to coordinate-descent in the dual LP, every

step of the algorithm decreases the upper bound on the MAP. The monotonicity property holds even if MPLP does not converge in step **6**, giving us the flexibility to choose the number of iterations N. In Section 5.4 we show results corresponding to two different choices of N.

In the case where we run MPLP to convergence before choosing the next cluster, we can show that the greedy bound minimization corresponds to a cutting-plane algorithm, as stated below.

Theorem 1. *Given a dual optimal solution, if we find a cluster for which we can guarantee a bound decrease, all primal optimal solutions were inconsistent with respect to this cluster.*

Proof. By duality both the dual optimum and the primal optimum will decrease. Suppose for contradiction that in the previous iteration there was a primal feasible point that was cluster consistent and achieved the LP optimum. Since we are maximizing the LP, after adding the cluster consistency constraint, this point is still feasible and the optimal value of the primal LP will not change, giving our contradiction. □

This theorem does not tell us *how much* the given cluster consistency constraint was violated, and the distinction remains that a typical cutting-plane algorithm would attempt to find the constraint which is most violated. In Chapter 7 we will precisely state the connection between the dual criterion and the primal-based cutting-plane algorithms discussed in Chapter 3. In particular, we will see that for binary graphical models, if for a dual optimal solution there is a cluster c such that $d(c) > 0$, then there exists a cycle inequality that is *very* violated by all primal optimal solutions.

5.3 Related Work

Since MPLP is closely related to the max-product generalized belief propagation (GBP) algorithm, our work can be thought of as a region-pursuit method for GBP. This is closely related to the work of Welling (2004) who suggested a region-pursuit method for sum-product GBP. Similar to our work, he suggested greedily adding from a candidate set of possible clusters. At each iteration, the cluster that results in the largest change in the GBP free energy is added. He showed excellent results for 2D grids, but on fully connected graphs the performance actually started deteriorating with additional clusters. In Welling *et al.* (2005), a heuristic related to maxent normality (Yedidia *et al.*, 2005) was used as a stopping criterion for region-pursuit to avoid this behavior. In our work, in contrast, since we are working with the dual function of the LP, we can guarantee monotonic improvement throughout the running of the algorithm.

Our work is also similar to Welling's in that we focus on criteria for determining the utility of adding a cluster, not on *finding* these clusters efficiently. We found in our experiments that a simple enumeration over small clusters proved extremely effective. For problems where triplet clusters alone would not suffice to find the MAP, we could triangulate the graph and consider larger clusters. This approach is reminiscent of the bounded join-graphs described in Dechter *et al.* (2002).

There is a large body of recent work describing the relationship between message-passing algorithms such as belief propagation, and LP relaxations (Kolmogorov & Wainwright, 2005; Weiss *et al.*, 2007; Yanover *et al.*, 2006). Although we have focused here on using one particular message-passing algorithm, MPLP, we emphasize that similar region-pursuit algorithms can be derived for other message-passing algorithms as well. In particular, for all

the convex max-product BP algorithms described in Weiss *et al.* (2007), it is easy to design region-pursuit methods. The main advantage of using MPLP is its guaranteed decrease of the dual value at each iteration, a guarantee that does not exist for general convex BP algorithms.

Region-pursuit algorithms are also conceptually related to the question of message scheduling in BP, as in the work of Elidan *et al.* (2006). One way to think of region-pursuit is to consider a graph where all the clusters are present all the time, but send and receive non-informative messages. The question of which cluster to add to an approximation, is thus analogous to the question of which message to update next.

Finally, related approaches were proposed by Johnson (2008) and Werner (2008), where the authors also suggest tightening the pairwise LP relaxation using higher-order clusters. However, one of the major distinctions of our approach is that we provide a criterion for deciding *which* clusters to add to the relaxation.

5.4 Experiments

Due to the scalable nature of our message-passing algorithm, we can apply it to cases where standard LP solvers cannot be applied to the primal LP (see also Yanover *et al.* (2006)). Here we report applications to problems in computational biology and machine vision.[3]

We use the algorithm from Section 5.2.3 for all of our experiments. We first run MPLP with edge clusters until convergence or for at most 1000 iterations, whichever comes first. All of our experiments, except those intended to show the difference between schedules, use $N = 20$ for the number of MPLP iterations run after adding a cluster. While running MPLP we use the messages to decode an integral solution, and compare the dual objective to the value of the integral solution. If these are equal, we have found the MAP solution.[4] Otherwise, we keep adding triplets.

Our results will show that we often find the MAP solution to these hard problems by using only a small number of triplet clusters. This indicates both that triplets are sufficient for characterizing $\mathcal{M}(G)$ near the MAP solution of these problems, and that our algorithm can efficiently find the informative triplets.

5.4.1 Side-Chain Prediction

The side-chain prediction problem involves finding the three-dimensional configuration of rotamers given the backbone structure of a protein (Yanover *et al.*, 2006). This problem can be posed as finding the MAP configuration of a pairwise model, and in Yanover *et al.* (2006) the tree-reweighted belief propagation (TRBP) algorithm (Wainwright *et al.*, 2005a) was used to find the MAP solution for most of the models studied. However, for 30 of the models, TRBP could not find the MAP solution.

In Chapter 3.4.3 we used a cutting-plane algorithm to solve these side-chain problems and found the MAP solution for all 30 models. Here, we applied our dual algorithm to the same 30 models and found that it also results in the MAP solution for all of them (up to a 10^{-4} integrality gap). This required adding between 1 and 27 triplets per model. The running time was between 1 minute and 1 hour to solve each problem, with over half

[3] Graphical models for these are given in Yanover *et al.* (2006).

[4] In practice, we terminate when the dual objective is within 10^{-4} of the decoded assignment, so these are approximate MAP solutions. Note that the objective values are significantly larger than this threshold.

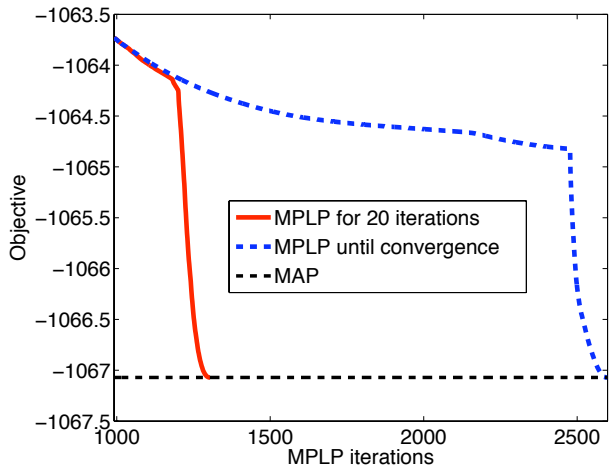


Figure 5-2: Comparison of different schedules for adding clusters to tighten the LP relaxation on a side-chain prediction problem.

solved in under 9 minutes. On average we added only 7 triplets (median was 4.5), another indication of the relative ease with which these techniques can solve the side-chain prediction problem.

We also used these models to study different update schedules. One schedule (which gave the results in the previous paragraph) was to first run a pairwise model for 1000 iterations, and then alternate between adding triplets and running MPLP for 20 more iterations. In the second schedule, we run MPLP to convergence after adding each triplet. Figure 5-2 shows the two schedules for the side-chain protein '1gsk', one of the side-chain proteins which took us the longest to solve (30 minutes). Running MPLP to convergence results in a much larger number of overall MPLP iterations compared to using only 20 iterations. This highlights one of the advantages of our method: adding a new cluster does not require solving the earlier problem to convergence.

5.4.2 Protein Design

The protein design problem is the inverse of the protein folding problem. Given a particular 3D shape, we wish to find a sequence of amino-acids that will be as stable as possible in that 3D shape. Typically this is done by finding a set of amino-acids and rotamer configurations that minimizes an approximate energy.

While the problem is quite different from side-chain prediction, it can be solved using the same graph structure, as shown in Yanover *et al.* (2006). The only difference is that now the nodes do not just denote rotamers, but also the identity of the amino-acid at that location. Thus, the state-space here is significantly larger than in the side-chain prediction problem (up to 180 states per variable for most variables).

In contrast to the side-chain prediction problems, which are often easily solved by general purpose integer linear programming packages such as CPLEX's branch-and-cut algorithm (Kingsford *et al.*, 2005), the sheer size of the protein design problems immediately limits the techniques by which we can attempt to solve them. Algorithms such as our earlier cutting-plane algorithm (c.f. Section 3.4) or CPLEX's branch-and-cut algorithm require

solving the primal LP relaxation at least once, but solving the primal LP on all but the smallest of the design problems is intractable (Yanover *et al.*, 2006). Branch and bound schemes have been recently used in conjunction with a message passing algorithm (Hong & Lozano-Pérez, 2006) and applied to similar protein design problems, although not the ones we solve here.

We applied our method to the 97 protein design problems described in Yanover *et al.* (2006), adding 5 triplets at a time to the relaxation. The key striking result of these experiments is that our method found the exact MAP configuration for all but one of the proteins[5] (up to a precision of 10^{-4} in the integrality gap). This is especially impressive since, as reported in Yanover *et al.* (2006), only 2 of these problems were solvable using TRBP, and the primal problem was too big for commercial LP solvers such as CPLEX. For the problem where we did not find the MAP, we did not reach a point where all the triplets in the graph were included, since we ran out of memory beforehand.

Among the problems that were solved exactly, the mean running time was 9.7 hours with a maximum of 11 days and a minimum of a few minutes. We note again that most of these problems could not be solved using LP solvers, and when LP solvers could be used, they were typically at least 10 times slower than message-passing algorithms similar to ours (see Yanover *et al.* (2006) for detailed timing comparisons).

Note that the main computational burden in the algorithm is processing triplet messages. Since each variable has roughly 100 states, passing a triplet message requires 10^6 operations. Thus the number of triplets added is the key algorithmic complexity issue. For the models that were solved exactly, the median number of triplets added was 145 (min: 5, max: 735). As mentioned earlier, for the unsolved model this number grew until the machine's memory was exhausted. We believe however, that by optimizing our code for speed and memory we will be able to accommodate a larger number of triplets, and possibly solve the remaining model as well. Our current code is written mostly in Matlab, so significant optimization may be possible.

5.4.3 Stereo Vision

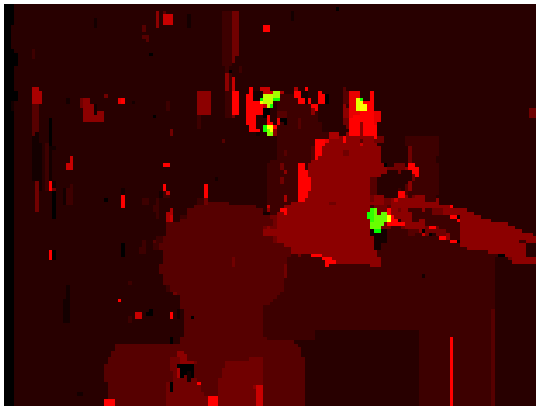
Given a stereo pair of images, the stereo problem is to find the disparity of each pixel in a reference image (c.f. Figure 1-1). This disparity can be straightforwardly translated into depth from the camera. The best algorithms currently known for the stereo problem are those that minimize a global energy function (Scharstein & Szeliski, 2002), which is equivalent to finding a MAP configuration in a pairwise model.

For our experiments we use the pairwise model described in Yanover *et al.* (2006), and apply our procedure to the "Tsukuba" sequence from the standard Middlebury stereo benchmark set (Scharstein & Szeliski, 2002), reduced in size to contain 116x154 pixels. The images in this sequence were taken from the same height, each horizontally displaced from one another. One image from the sequence is shown in Figure 5-3(a). We use the same energy function that was used by Tappen & Freeman (2003).

Since there are no connected triplets in the grid graph, we use our method with square clusters. We calculate the bound decrease using square clusters, but rather than add them directly, we triangulate the cycle and add two triplet clusters. Although this results in an equivalent relaxation, it has the consequence that we may have to wait until MPLP convergence to achieve the guaranteed bound improvement. This is because we are only performing

[5] We could not solve '1fpo', the largest protein.

(a) Original "Tsukuba" image



(b) MAP assignment

Figure 5-3: (a) "Tsukuba" image used in stereopsis experiments. (b) Visualization of the MAP assignment found by the dual algorithm and of the clusters used in tightening the relaxation. Depth is shown in shades of red: darker pixels denote greater depth. The green pixels show the square clusters that were used to tighten the relaxation.

coordinate-descent on triplet clusters, whereas the new bound criterion corresponds to the dual decrease due to one coordinate-descent step on a square cluster.

In the first experiment, we varied the parameters of the energy function to create several different instances. We tried to find the MAP using TRBP, resolving ties using the methods proposed in (Meltzer *et al.*, 2005). In 4 out of 10 cases those methods failed. Using our algorithm, we managed to find the MAP for all 4 cases.[6]

Figure 5-4 shows the dual objective and the decoded integer solution after each MPLP iteration, for one setting of the parameters.[7] In Figure 5-3(b) we show the MAP assignment found by our algorithm for the same setting of the parameters. Depth is shown in shades of red: darker pixels denote greater depth. As can be seen by comparing Figure 5-3(b) with Figure 5-3(a), the model correctly predicts the relative depth for a large fraction of the image, but is far from perfect.

We then looked to see *which* clusters were added to the relaxation, across all iterations. Since our schedule adds a fixed 20 clusters per iteration, some of these clusters could have been superfluous. Thus, to visualize this, we consider only the set S of clusters c where the guaranteed bound improvement $d(c) > 0.01$. In Figure 5-3(b) we display this set S in green, overlaid on top of the MAP assignment. These green areas correspond to the frustrated parts of the model. Interestingly, many of the sites of frustration are adjacent to close-by object boundaries. These are the locations where occlusion happens between the pair of images, i.e. where there is no 1-1 correspondence between pairs of pixels in the left and right images.

In the results above, we added 20 squares at a time to the relaxation. We next contrasted it with two strategies: one where we pick 20 random squares (not using our bound

[6]For one of these models, a few single node beliefs at convergence were tied, and we used the junction tree algorithm to decode the tied nodes (see Meltzer *et al.* (2005)).

[7]The parameters used were $(T, s, P) = (4, 20, 2)$; c.f. Tappen & Freeman (2003).

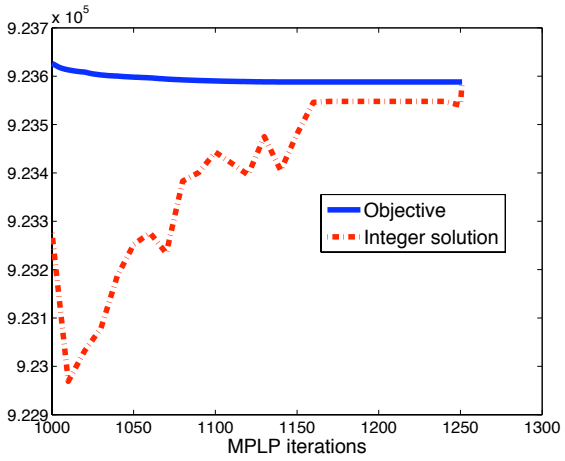


Figure 5-4: Dual objective and value of decoded integer solution for one of the reduced "Tsukuba" stereo models, as a function of MPLP iterations. It can be seen that both curves converge to the same value, indicating that the MAP solution was found.

improvement criterion) and one where we pick the single best square according to the bound criterion. Figure 5-5 shows the resulting bound per iteration for one of the models. It can be seen that the random method is much slower than the bound criterion based one, and that adding 20 squares at a time is better than just one. We ended up adding 1060 squares when adding 20 at a time, and 83 squares when adding just one. Overall, adding 20 squares at a time turned out to be faster.

We also tried running MPLP with all of the square clusters. Although fewer MPLP iterations were needed, the cost of using all squares resulted in an overall running time of about four times longer.

5.5 Bound Criterion in Sparse Graphs

In our experiments we only considered clusters of size 3 or 4, of which there are only polynomially many, and thus we were able to do this by enumeration. However, a sparse graphical model may not have *any* short cycles. In this setting, it may seem natural to first triangulate the graph before running our algorithm. In this section we show that the bound criterion can be non-informative in these settings, and thus triangulation may not be helpful.

Consider a binary-valued MRF on four variables X_1, X_2, X_3, X_4 which has edges in the form of a square: $E = \{(1,2),(2,3),(3,4),(1,4)\}$. We now define the edge potentials. For $(i,j) \in \{(1,2),(2,3),(3,4)\}$, let $\theta_{ij}(x_i, x_j) = 1$ if $x_i \neq x_j$, and 0 if $x_i = x_j$. We do the opposite for edge $(1,4)$, letting $\theta_{1,4}(x_1, x_4) = 1$ if $x_1 = x_4$, and 0 if $x_1 \neq x_4$.

All of the MAP assignments have value 3. For example, one MAP assignment is $(X_1, X_2, X_3, X_4) = (1,0,1,0)$. The pairwise LP relaxation, on the other hand, has value 4, obtained by $\mu_i(x_i) = 0.5$ for $i \in \{1,2,3,4\}$ and $x_i \in \{0,1\}$, $\mu_{ij}(0,1) = \mu_{ij}(1,0) = 0.5$ for $(i,j) \in \{(1,2),(2,3),(3,4)\}$, and $\mu_{1,4}(0,0) = \mu_{1,4}(1,1) = 0.5$. One way to triangulate the graph is to add the edge $(1,3)$. However, as can be seen by setting the edge marginal for

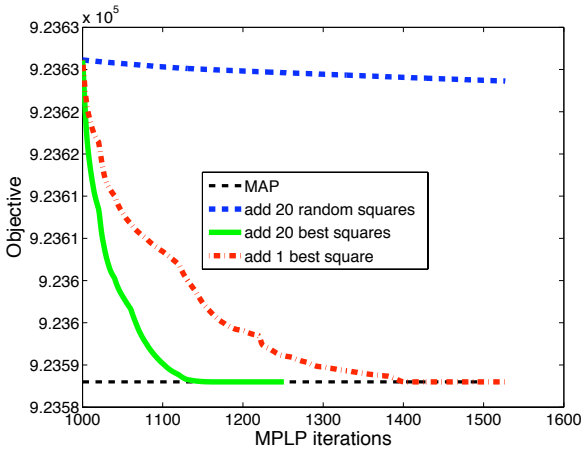


Figure 5-5: Comparison of different schedules for adding squares in one of the stereo problems.

$(1,3)$ to $\mu_{1,3}(1,1) = \mu_{1,3}(0,0) = 0.5$, the pairwise LP relaxation still has value 4.

Now consider adding a triplet consistency constraint for the cluster $c = \{1,2,3\}$. Solving the new LP relaxation, we find that it again has value 4. The solution is the same as earlier, with the new triplet marginal taking value $\mu_{1,2,3}(1,0,1) = \mu_{1,2,3}(0,1,0) = 0.5$ (note that this is consistent with the edge marginals already given, as it should be). Let's see what this corresponds to in the dual. Suppose that we use MPLP to solve the dual of the pairwise LP relaxation to optimality. By LP duality, the dual objective has value 4. Also by LP duality, we know that the optimal dual objective *after* adding the triplet cluster must also be 4. Recall that the bound criterion $d(c)$ corresponds to the amount that the dual objective will decrease after one block coordinate descent step involving the new cluster. Since the dual objective is lower bounded by 4 (its value at optimality), we conclude that $d(c)$ must be zero for the triplet cluster $c = \{1,2,3\}$.

The same can be shown for the triplet cluster $c = \{1,3,4\}$. A generalization of the argument shows that, for a MRF that consists of a single cycle of length larger than 3, and for any dual optimal solution of the pairwise LP relaxation (after triangulation), $d(c) = 0$ for all of the triplets in the triangulation. In contrast, if we had evaluated $d(c)$ for c corresponding to the *whole cycle* – as we did in the stereo vision experiments – we would see that it is non-zero. Note that it is possible to evaluate $d(c)$ for a cycle cluster in $O(nk^3)$ running time, where n is the length of the cycle and k is the number of states per node. However, we will show in Chapter 7 that *searching* for arbitrary length cycle clusters where $d(c) > 0$ is substantially more difficult.

An alternative approach, that we describe in Chapter 7, is to directly look for violated k-ary cycle inequalities (see Chapter 3) in the *dual* of the LP relaxation. The resulting algorithm is very similar to the one described here, and solves the search problem.

5.6 Discussion

In order to apply LP relaxations to real-world problems, one needs to find an efficient way of adding clusters to the basic relaxation such that the problem remains tractable but yields

a better approximation of the MAP value.

In this chapter we presented a greedy bound-minimization algorithm on the dual LP to solve this problem, and showed that it has all the necessary ingredients: an efficient message-passing algorithm, "warm start" of the next iteration using current beliefs, and a monotonically decreasing bound on the MAP.

We showed that the algorithm works well in practice, finding the MAP configurations for many real-world problems that were previously thought to be too difficult for known methods to solve. While in this chapter we focused primarily on adding triplet clusters, our approach is general and can be used to add larger clusters as well, as long as as the messages in the dual algorithm can be efficiently computed.

There are various algorithmic choices that could be explored in future work. For example, rather than adding a fixed number of clusters per iteration, we could choose to include only those clusters c where the bound criterion $d(c)$ is sufficiently large. The difficulty with such an approach is deciding an appropriate cut-off point. Also, rather than running a fixed number of iterations of MPLP after adding each new set of clusters, we could have an adaptive stopping criterion, e.g. stopping as soon as the dual objective decreases some multiple of the guaranteed bound improvement (or earlier, if MPLP converges). In our experiments, we chose $N = 20$ iterations by balancing the running time of MPLP with the time required to find new clusters to use in tightening the relaxation. Finally, one could consider different message-passing schedules. For example, after adding a new cluster, we could first pass messages near the cluster, and then propagate the effect to parts of the model further away.

One difficulty that we observed was that the computational cost of sending even one triplet message could be prohibitive when the state spaces of the variables are large, as in the protein design problem. We will show in Chapter 6 that it is possible to solve protein design problems by using a much weaker class of cluster consistency constraints for which the maximization over the clusters' messages in the dual algorithm is fast. Also, recent work by McAuley & Caetano (2010) has shown how to more efficiently compute the triplet messages in pairwise MRFs.

Finally, while here we focused on the MAP problem, similar ideas may be applied to approximating the marginals in graphical models.

Chapter 6

Clusters and Coarse Partitions in LP Relaxations

We saw in Chapter 5 how relaxations can be made increasingly tight by introducing LP variables that correspond to clusters of variables in the original model. In fact, by adding a set of clusters, each over just three variables, complex problems such as protein-design and stereo-vision could be solved exactly. The problem with adding clusters over variables is that the computational cost scales exponentially with the cluster size. Consider, for example, a problem where each variable has 100 states (cf. protein-design). Using clusters of s variables means adding 100^s LP variables, which is computationally demanding even for clusters of size three.

In this chapter we give a new class of consistency constraints over clusters that have reduced computational cost. We achieve this by representing clusters at a coarser level of granularity. The key observation is that it may not be necessary to represent all the possible joint states of a cluster of variables. Instead, we partition the cluster's assignments at a coarser level, and enforce consistency only across such partitions. This removes the number of states per variable from consideration, and instead focuses on resolving currently ambiguous settings of the variables.

Following the approach described in Chapter 4, we formulate a dual LP for the partition-based LP relaxations and derive a message passing algorithm for optimizing the dual LP based on block coordinate descent. Unlike standard message passing algorithms, the algorithm we derive involves passing messages between coarse and fine representations of the same set of variables. We show how to use the new constraints within the region-pursuit algorithm given in the previous chapter.

This chapter is based in part on material previously published in Sontag *et al.* (2009). Also, in this chapter we assume that the graphical model does not have potentials on single nodes, $\theta_i(x_i)$, as these can be folded into the edge potentials $\theta_{ij}(x_i, x_j)$.

6.1 Coarsened Constraints in the Primal LP Relaxation

We begin with an illustrative example. Suppose we have a graphical model that is a triangle with each variable taking k states. We can recover the exact marginal polytope in this case by forcing the pairwise marginals $\mu_{ij}(x_i, x_j)$ to be consistent with some distribution $\mu_{123}(x_1, x_2, x_3)$. However, when k is large, introducing the corresponding k^3 variables to our LP may be too costly and perhaps unnecessary, if a weaker consistency constraint

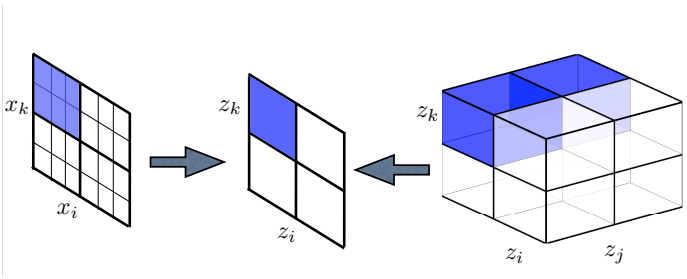


Figure 6-1: A graphical illustration of the consistency constraint between the original (fine granularity) edge (x_i, x_k) and the coarsened triplet (z_i, z_j, z_k). The two should agree on the marginal of z_i, z_k. For example, the shaded area in all three figures represents the same probability mass.

would already lead to an integral extreme point. To this end, we will use a coarse-grained version of $\mu_{123}(x_1, x_2, x_3)$ where the joint states are partitioned into larger collections, and consistency is enforced over the partitions.

The simplest partitioning scheme builds on coarse-grained versions of each variable X_i. Let Z_i denote a disjoint collection of sets covering the possible values of variable X_i. For example, if variable X_i has five states, Z_i might be defined as $\big\{\{1,2\},\{3,5\},\{4\}\big\}$. We simultaneously interpret Z_i as a random variable with three states, each state denoted by z_i. We will use z_i to index the sets in Z_i so that, for example, z_i may take a value $\{1,2\}$. Then, for each of the three coarse-grained states z_i, we have

$$\Pr(Z_i = z_i) \quad = \quad \sum_{x_i \in z_i} \Pr(X_i = x_i) \ . \tag{6.1}$$

Given such a partitioning scheme, we can introduce a higher-order distribution over coarsened variables, e.g. $\mu_{123}(z_1, z_2, z_3)$, and constrain it to agree with the fine-grained edge distributions, $\mu_{ik}(x_i, x_k)$, in the sense that they both yield the same marginals for z_i, z_k. This is illustrated in Figure 6-1. The corresponding constraints are:

$$\sum_{x_i \in z_i,\ x_k \in z_k} \mu_{ik}(x_i, x_k) = \mu_{ik}(z_i, z_k) = \sum_{z_j} \mu_{ijk}(z_i, z_j, z_k), \quad \forall z_i, z_k \ . \tag{6.2}$$

Note that the $\mu_{ik}(z_i, z_k)$ variables are not actually needed to enforce consistency between $\mu_{ik}(x_i, x_k)$ and $\mu_{ijk}(z_i, z_j, z_k)$, and so we omit them from the remainder of the chapter. In the case when Z_i individuates each state, i.e., $\big\{\{1\},\{2\},\{3\},\{4\}\big\}$, we recover the usual cluster consistency constraints used in Chapter 5.

We use the above idea to construct tighter outer bounds on the marginal polytope and incorporate them into the MAP-LP relaxation. We assume that we are given a set of clusters $\mathcal{C}$. For each cluster $c \in \mathcal{C}$ and variable $i \in c$ we also have a partition Z_i^c as in the above example (the choice of clusters and partitions will be discussed later). We use a superscript of c to highlight the fact that different clusters may use different partitionings for X_i. Also, there can be multiple clusters on the same set of variables, each using a different partitioning. We introduce marginals over the coarsened clusters, $\mu_c(\mathbf{z}_c^c)$, and constrain

them to agree with the edge variables $\mu_{ij}(x_i, x_j)$ for all edges $ij \in c$:

$$\sum_{x_i \in z_i^c,\, x_j \in z_j^c} \mu_{ij}(x_i, x_j) = \sum_{\mathbf{z}^c_{c \setminus i,j}} \mu_c(\mathbf{z}_c^c), \qquad \forall z_i^c, z_j^c \; . \tag{6.3}$$

The key idea is that the coarsened cluster represents higher-order marginals albeit at a lower resolution, whereas the edge variables represent lower-order marginals but at a finer resolution. The constraint in Eq. 6.3 implies that these two representations should agree.

We can now state the LP that we set out to solve. Our LP optimizes over the following marginal variables: $\mu_{ij}(x_i, x_j), \mu_i(x_i)$ for the edges and nodes of the original graph, and $\mu_c(\mathbf{z}_c^c)$ for the coarse-grained clusters. We would like to constrain these variables to belong to the following outer bound on the marginal polytope:

$$\mathcal{M}_{\mathcal{C}}(G) = \left\{ \boldsymbol{\mu} \geq 0 \;\middle|\; \begin{array}{rcll} \sum_{x_j} \mu_{ij}(x_i, x_j) & = & \mu_i(x_i) & \forall ij \in E, x_i \\ \sum_{x_i} \mu_{ij}(x_i, x_j) & = & \mu_j(x_j) & \forall ij \in E, x_j \\ \sum_{x_i} \mu_i(x_i) & = & 1 & \forall i \in V \\ \sum\limits_{x_i \in z_i^c,\, x_j \in z_j^c} \mu_{ij}(x_i, x_j) & = & \sum\limits_{\mathbf{z}^c_{c \setminus i,j}} \mu_c(\mathbf{z}_c^c) & \forall c,\;\; i,j \in c \text{ s.t. } ij \in E, z_i^c, z_j^c \end{array} \right\}$$

Note that $\sum_{\mathbf{z}_c^c} \mu_c(\mathbf{z}_c^c) = 1$ is implied by the above constraints. The corresponding MAP-LP relaxation is then:

$$\max_{\boldsymbol{\mu} \in \mathcal{M}_{\mathcal{C}}(G)} \boldsymbol{\mu} \cdot \boldsymbol{\theta} \tag{6.4}$$

This LP could in principle be solved using generic LP optimization tools. However, a more efficient and scalable approach is to solve it via message passing in the dual LP, which we show how to do in the next section. In addition, for this method to be successful, it is critical that we choose *good* coarsenings, meaning that it should have few partitions per variable, yet still sufficiently tightens the relaxation. Our approach for choosing the coarsenings is to iteratively solve the LP using an initial relaxation (beginning with the pairwise consistency constraints), then to introduce additional cluster constraints, letting the current solution guide how to coarsen the variables. As we showed in Chapter 5, solving with the dual LP gives us a simple method for "warm starting" the new LP (the tighter relaxation) using the previous solution, and also results in an algorithm for which every step monotonically decreases an upper bound on the MAP assignment. We will give further details of the coarsening scheme in Section 6.3.

6.2 Dual LP and Message Passing Algorithm

In this section we give the dual of the partition-based LP from Eq. 6.4, and use it to obtain a message passing algorithm to efficiently optimize this relaxation. Our approach extends earlier work by Globerson & Jaakkola (2008) who gave the generalized max-product linear programming (MPLP) algorithm to solve the usual (non-coarsened) cluster LP relaxation in the dual (c.f. Section 5.2.1).

The dual formulation in Globerson & Jaakkola (2008) was derived by adding auxiliary variables to the primal. We followed a similar approach to obtain the LP dual of Eq. 6.4. The dual variables are as follows: $\lambda_{ij \to i}(x_i), \lambda_{ij \to j}(x_j), \lambda_{ij \to ij}(x_i, x_j)$ for every edge $ij \in E$, and $\lambda_{c \to ij}(z_i^c, z_j^c)$ for every coarsened cluster c and edge $ij \in c$.

As we show below, the variables $\boldsymbol{\lambda}$ correspond to the messages sent in the message

- **Edge to Node**: For every edge $ij \in E$ and node i (or j) in the edge:

$$\lambda_{ij\to i}(x_i) \leftarrow -\frac{2}{3}\lambda_i^{-j}(x_i) + \frac{1}{3}\max_{x_j}\Big[\sum_{c:ij\in c}\lambda_{c\to ij}(z_i^c[x_i], z_j^c[x_j]) + \lambda_{ij\to ij}(x_i,x_j) + \lambda_j^{-i}(x_j) + \theta_{ij}(x_i,x_j)\Big]$$

  where $\lambda_i^{-j}(x_i) = \sum_{k\in N(i)\setminus j}\lambda_{ik\to i}(x_i)$.

- **Edge to Edge**: For every edge $ij \in E$:

$$\lambda_{ij\to ij}(x_i,x_j) \leftarrow -\frac{2}{3}\sum_{c:ij\in c}\lambda_{c\to ij}(z_i^c[x_i], z_j^c[x_j]) + \frac{1}{3}\Big[\lambda_j^{-i}(x_j) + \lambda_i^{-j}(x_i) + \theta_{ij}(x_i,x_j)\Big]$$

- **Cluster to Edge**: First define

$$b_{ij}(z_i^c, z_j^c) = \max_{\substack{x_i \in z_i^c \\ x_j \in z_j^c}} \left[\lambda_{ij\to ij}(x_i,x_j) + \sum_{c'\neq c: ij\in c'}\lambda_{c'\to ij}(z_i^{c'}[x_i], z_j^{c'}[x_j])\right] \tag{6.5}$$

  The update is then:

$$\lambda_{c\to ij}(z_i^c, z_j^c) \leftarrow -b_{ij}(z_i^c, z_j^c) + \frac{1}{|\mathcal{S}(c)|}\max_{\mathbf{z}_{c\setminus i,j}^c}\sum_{st\in c} b_{st}(z_s^c, z_t^c) \tag{6.6}$$

Figure 6-2: The message passing updates for solving the dual LP given in Eq. 6.7.

passing algorithm that we use for optimizing the dual. Thus $\lambda_{ij\to i}(x_i)$ should be read as the message sent from edge ij to node i, and $\lambda_{c\to ij}(z_i^c, z_j^c)$ is the message from the coarsened cluster to one of its intersection edges. Finally, $\lambda_{ij\to ij}(x_i,x_j)$ is the message sent from an edge to itself. The dual of Eq. 6.4 is the following minimization problem over $\boldsymbol{\lambda}$,

$$\sum_i \max_{x_i}\sum_{k\in N(i)}\lambda_{ik\to i}(x_i) + \sum_{ij\in E}\max_{x_i,x_j}\Big[\theta_{ij}(x_i,x_j) + \lambda_{ij\to ij}(x_i,x_j) + \sum_{c:ij\in c}\lambda_{c\to ij}(z_i^c[x_i], z_j^c[x_j])\Big]$$

subject to the constraints:

$$\begin{aligned} \lambda_{ij\to i}(x_i) + \lambda_{ij\to j}(x_j) + \lambda_{ij\to ij}(x_i,x_j) &\geq 0 \qquad \forall ij\in E, x_i, x_j \\ \sum_{ij\in c}\lambda_{c\to ij}(z_i^c, z_j^c) &\geq 0 \qquad \forall c, \mathbf{z}_c^c \end{aligned} \tag{6.7}$$

The notation $z_i^c[x_i]$ refers to the mapping from $x_i \in X_i$ to the coarse state $z_i^c \in Z_i^c$ such that $x_i \in z_i^c$. By convex duality, the dual objective evaluated at a dual feasible point upper bounds the primal LP optimum, which in turn upper bounds the value of the MAP assignment. It is illustrative to compare this dual LP with the dual given in Eq. 5.4 from Chapter 5, where the cluster dual variables were $\lambda_{c\to ij}(x_i,x_j)$. Our dual corresponds to

introducing the additional constraint that $\lambda_{c\to ij}(x_i, x_j) = \lambda_{c\to ij}(x'_i, x'_j)$ whenever $z^c_i[x_i] = z^c_i[x'_i]$ and $z^c_j[x_j] = z^c_j[x'_j]$.

The advantage of the above dual is that it can be optimized via a simple message passing algorithm that corresponds to block coordinate descent. The key idea is that it is possible to fix the values of the $\boldsymbol{\lambda}$ variables corresponding to all clusters except one, and to find a closed form solution for the non-fixed $\boldsymbol{\lambda}$s. Figure 6-2 provides the form of the updates for all three message types. $\mathcal{S}(c)$ is the set of edges in cluster c (e.g. ij, jk, ik). Importantly, all messages outgoing from a cluster or edge must be sent simultaneously.

Here we derive the cluster to edge updates, which differ from Globerson & Jaakkola (2008). Assume that all values of $\boldsymbol{\lambda}$ are fixed except for $\lambda_{c\to ij}(z^c_i, z^c_j)$ for all $ij \in c$ for some cluster c. The term in the dual objective that depends on $\lambda_{c\to ij}(z^c_i, z^c_j)$ can be written equivalently as

$$\begin{aligned} & \max_{x_i, x_j} \Big[\lambda_{ij\to ij}(x_i, x_j) + \sum_{c': c' \neq c, ij \in c'} \lambda_{c'\to ij}(z^{c'}_i[x_i], z^{c'}_j[x_j]) + \lambda_{c\to ij}(z^c_i[x_i], z^c_j[x_j]) \Big] \\ = \; & \max_{z^c_i, z^c_j} \Big[b_{ij}(z^c_i, z^c_j) + \lambda_{c\to ij}(z^c_i[x_i], z^c_j[x_j]) \Big] \;, \end{aligned} \tag{6.8}$$

where $b_{ij}(z^c_i, z^c_j)$ is defined in Eq. 6.5. Due to the constraint $\sum_{ij\in c} \lambda_{c\to ij}(z^c_i, z^c_j) \geq 0$, all of the $\lambda_{c\to ij}(z^c_i, z^c_j)$ variables need to be updated simultaneously. It can be shown (using an equalization argument as in Globerson & Jaakkola (2008)) that the $\lambda_{c\to ij}(z^c_i, z^c_j)$ that minimize the objective subject to the constraint are given by Eq. 6.6 in Figure 6-2.

Note that none of the cluster messages involve the original cluster variables $\mathbf{x}_c$, but rather only $\mathbf{z}^c_c$. Thus, we have achieved the goal of both representing higher-order clusters and doing so at a reduced computational cost.

The algorithm in Figure 6-2 solves the dual for a given choice of coarsened clusters. As mentioned in Section 6.1, we would like to add such clusters gradually, as in Chapter 5. Our overall algorithm is thus similar in structure to the one from Section 5.2.2 and proceeds as follows (we denote the message passing algorithm from Figure 6-2 by MPLP): **1.** Run MPLP until convergence using the pairwise relaxation, **2.** Find an integral solution $\mathbf{x}$ by locally maximizing the single node beliefs $b_i(x_i) = \sum_{k\in N(i)} \lambda_{ki\to i}(x_i)$, **3.** If the dual objective given in Eq. 6.7 is sufficiently close to the primal objective $\theta(\mathbf{x})$, terminate, **4.** Add a new coarsened cluster c using the strategy given in Section 6.3, **5.** Initialize messages going out of the new cluster c to zero, and keep all the previous message values (this will not change the bound value), **6.** Run MPLP for N iterations, then return to **2**.

6.3 Choosing the Partitionings

Until now we have not discussed how to choose the clusters to add and their partitionings. Our strategy for doing so closely follows that used in Chapter 5. Given a set $\mathcal{C}$ of candidate clusters to add (e.g., the set of all triplets in the graph), we would like to add a cluster that would result in the maximum decrease of the dual bound on the MAP. In principle such a cluster could be found by optimizing the dual for each candidate cluster, then choosing the best one. However, this is computationally costly, so in Chapter 5 we instead used the bound decrease resulting from just once sending messages from the candidate cluster to its intersection edges.

If we were to add the full (un-coarsened) cluster, this bound decrease would be:

$$d(c) = \sum_{ij \in c} \max_{x_i, x_j} b_{ij}(x_i, x_j) - \max_{\mathbf{x}_c} \sum_{ij \in c} b_{ij}(x_i, x_j), \tag{6.9}$$

where $b_{ij}(x_i, x_j) = \lambda_{ij \to ij}(x_i, x_j) + \sum_{c: ij \in c} \lambda_{c \to ij}(z_i^c[x_i], z_j^c[x_j])$.

Our strategy now is as follows: we add the cluster c that maximizes $d(c)$, and then choose a partitioning Z_i^c for all $i \in c$ that is guaranteed to achieve a decrease that is *close* to $d(c)$. This can clearly be achieved by using the trivial partition $Z_i^c = X_i$ (which achieves $d(c)$). However, in many cases it is also possible to achieve it while using much coarser partitionings. To be precise, we would like to find the coarse-grained variables Z_i^c for $i \in c$ that minimizes $\prod_{i \in c} |Z_i^c|$ subject to the constraint

$$\begin{aligned} \max_{\mathbf{x}_c} \sum_{ij \in c} b_{ij}(x_i, x_j) &= \max_{\mathbf{z}_c^c} \sum_{ij \in c} b_{ij}(z_i^c, z_j^c) & (6.10) \\ &= \max_{\mathbf{z}_c^c} \sum_{ij \in c} \max_{x_i \in z_i^c,\ x_j \in z_j^c} b_{ij}(x_i, x_j). & (6.11) \end{aligned}$$

The set of all possible partitionings Z_i^c is too large to explicitly enumerate. Instead, we use the following heuristic. Consider just $|X_i|$ candidate partitions that are generated based on the beliefs $b_i(x_i)$. Intuitively, the states with lower belief values $b_i(x_i)$ are less likely to be part of the MAP assignment, and can thus be bundled together. We will therefore consider partitions where the states with lowest belief values are put into the same "catch-all" coarse state s_i^c, and all other states of x_i get their own coarse state. Formally, a partition Z_i^c is characterized by a value κ_i such that s_i^c is the set of all x_i with $b_i(x_i) < \kappa_i$.

The next question is how big we can make the catch-all state without sacrificing the bound decrease. We employ a greedy strategy whereby each $i \in c$ (in arbitrary order) is partitioned separately, while the other partitions are kept fixed. The process starts with $Z_i^c = X_i$ for all $i \in c$. We would like to choose s_i^c such that it is sufficiently separated from the state that achieves $d(c)$. Formally, given a margin parameter γ (we will discuss γ in a moment; for now, assume that $\gamma = 0$) we choose κ_i to be as large as possible such that the following constraint still holds[1]:

$$\max_{\substack{\mathbf{z}_c^c \text{ s.t.} \\ z_i^c = s_i^c}} \sum_{st \in c} b_{st}(z_s^c, z_t^c) \ \leq \ \max_{\mathbf{x}_c} \sum_{st \in c} b_{st}(x_s, x_t) - \gamma, \tag{6.12}$$

where the first maximization is over the coarse variables $Z_{c \setminus i}$, and Z_i^c is fixed to the catch-all state s_i^c (note that the partitioning for Z_i^c is a function of κ_i). We can find the optimal κ_i in time $O(|X_i|^{|c|})$ by starting with $\kappa_i = -\infty$ and increasing it until the constraint is violated. Since each subsequent value of s_i^c differs by one additional state x_i, we can re-use the maximizations over $\mathbf{z}_{c \setminus i}^c$ for the previous value of s_i^c in evaluating the constraint for the current s_i^c.

It can be shown by induction that this results in a coarsening that has a guaranteed bound decrease of at least $d(c) + \min(0, \gamma)$. Setting $\gamma < 0$ would give a partitioning with fewer coarse states at the cost of a smaller guaranteed bound decrease. On the other hand, setting $\gamma > 0$ results in an *over-partitioning* of the variables' states. This results in a *margin*

[1]The constraint may be infeasible for $\gamma > 0$, in which case we simply choose $Z_i^c = X_i$.

between the value of the dual objective (after sending the coarsened cluster message) and its value if we were to fix x_i to one of the states in s_i^c for all of the edge maximization terms ($\max_{x_i,x_j}$) in the dual objective corresponding to $ij \in c$. Intuitively, this makes it less likely that a state in s_i^c will become important again in subsequent message passing iterations. Thus, this partitioning will be good even after a few more iterations of coordinate-descent. For the experiments in this chapter we use $\gamma = 3d(c)$, scaling γ with the value of the guaranteed bound decrease for the full cluster.

Note that this greedy algorithm does not necessarily find the partitioning with the fewest number of coarse states that achieves the bound decrease. An interesting open problem is to design an efficient partitioning algorithm with such a guarantee.

6.4 Experiments

We report results on the protein design problem, originally described in Yanover *et al.* (2006). The protein design problem is the inverse of the protein folding problem. Given a desired backbone structure for the protein, the goal is to construct the sequence of amino-acids that results in a low energy, and thus stable, configuration. We can use an approximate energy function to guide us towards finding a set of amino-acids and rotamer configurations with minimal energy. In Yanover *et al.* (2006) the design problem was posed as finding a MAP configuration in a pairwise MRF. The models used there (which are also available online) have a number of states per variable that is between 2 and 158, and contain up to 180 variables per model. The models are also quite dense so that exact calculation is not feasible.

In Chapter 5 we showed that all but one of the problems described in Yanover *et al.* (2006) can be solved exactly by using a LP relaxation with clusters on three variables. However, since each individual state has roughly 100 possible values, processing triplets required 10^6 operations, making the optimization costly. In what follows we describe two sets of experiments that show that, by coarsening, we can both significantly reduce the computation time and achieve similar performance as if we had used un-coarsened triplets (see Section 5.4). The experiments differ in the strategy for adding triplets, and illustrate two performance regimes. In both experimental setups we first run the standard edge-based message passing algorithm for 1000 iterations.

We call the single node belief $b_i(x_i)$ *tied* if there are at least two states x_i and x_i' such that $b_i(x_i) = b_i(x_i') = \max_{\hat{x}_i} b_i(\hat{x}_i)$. In the first experiment, we add all triplets that correspond to variables whose single node beliefs are tied (within 10^{-5}) at the maximum after running the edge-based algorithm. Since tied beliefs correspond to fractional LP solutions (c.f. Section 4.4), it is natural to consider these in tighter relaxations. The triplets correspond to partitioned variables, as explained in Section 6.1. The partitioning is guided by the ties in the single node beliefs. Specifically, for each variable X_i we find states whose single node beliefs are tied at the maximum. Denote the number of states maximizing the belief by r. Then, we partition the states into r subsets, each containing a different maximizing state. The other (non-maximizing) states are split randomly among the r subsets. The triplets are then constructed over the coarsened variables Z_i^c and the message passing algorithm of Section 6.2 is applied to the resulting structure. After convergence of the algorithm, we recalculate the single node beliefs. These may result in a different partition scheme, and hence new variables Z_i^c. We add new triplets corresponding to the new variables and re-run. We repeat until the dual-LP bound is sufficiently close to the value of the integral

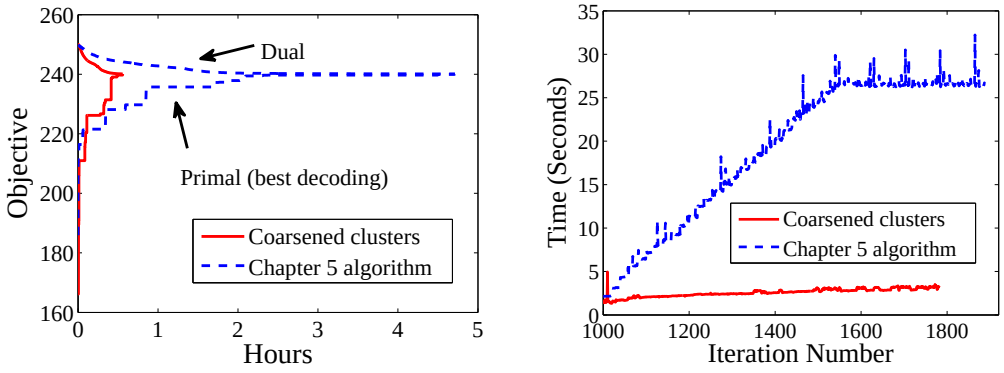


Figure 6-3: Comparison with algorithm from Chapter 5 for the protein "1aac", after the first 1000 iterations. **Left**: Dual objective as a function of time. **Right**: The running time per iteration of message-passing (one pass through the entire graph).

assignment obtained from the messages (note that these values would not coincide if the relaxation were not tight; in these experiments they do, so the final relaxation is tight).

We applied the above scheme to the ten smallest proteins in the dataset (for the larger proteins we used a different strategy described next). We were able to solve all ten exactly, as in Chapter 5. The mean running time was six minutes. The gain in computational efficiency as a result of using coarsened-triplets was considerable: The average state space size for coarsened triplets was on average 3000 times smaller than that of the original triplet state space, resulting in a factor 3000 speed gain over a scheme that uses the complete (uncoarsened) triplets.[2] This big factor comes about because a very small number of states are tied per variable, thus increasing the efficiency of our method where the number of partitions is equal to the number of tied states. While running on full triplets was completely impractical, the coarsened message passing algorithm is very practical and achieves the exact MAP assignments.

Our second set of experiments follows the setup of Chapter 5 (see Section 6.2), alternating between adding 5 triplets to the relaxation and running MPLP for 20 more iterations. The only difference is that, after deciding to add a cluster, we use the algorithm from Section 6.3 to partition the variables. We tried various settings of γ, including $\gamma = 0$ and .01, and found that $\gamma = 3d(c)$ gave the best overall runtimes.

We applied this second scheme to the 15 largest proteins in the dataset.[3] Of these, we found the exact MAP in 7 of the cases (according to the criterion used in Chapter 5), and in the rest of the cases were within 10^{-2} of the known optimal value. For the cases that were solved exactly, the mean running time was 1.5 hours, and on average the proteins were solved 8.1 times faster than with the uncoarsened constraints.[4] To compare the running times on all 15 proteins, we checked how long it took for the difference between the dual and primal objectives to be less than $.01\theta(\mathbf{x}_M)$, where $\mathbf{x}_M$ is the MAP assignment. This revealed that our method is faster by an average factor of 4.3. The reason why these factors

[2]These timing comparisons do not apply to Chapter 5 since that algorithm did not use all the triplets.
[3]We do not run on the protein 1fpo, which was not solved in Chapter 5.
[4]We made sure that differences were not due to different processing powers or CPU loads.

are less than the 3000 in the previous setup is that, for the larger proteins, the number of tied states is typically much higher than that for the small ones.

Results for one of the proteins that we solved exactly are shown in Figure 6-3. The running time per iteration increases very little after adding each triplet, showing that our algorithm significantly coarsened the clusters. The total number of iterations and number of triplets added were roughly the same. Two triplet clusters were added twice using different coarsenings, but otherwise each triplet only needed to be added once, demonstrating that our algorithm chose the right coarsenings.

6.5 Discussion

We presented an algorithm that enforces higher-order consistency constraints on LP relaxations, but at a reduced computational cost. Our technique further explores the trade-offs of representing complex constraints on the marginal polytope while keeping the optimization tractable. In applying the method, we chose to cluster variables' states based a bound minimization criterion after solving using a looser constraint on the polytope.

A class of approaches related to ours are the "coarse-to-fine" applications of belief propagation (Felzenszwalb & Huttenlocher, 2006; Raphael, 2001). In those, one solves low-resolution versions of a MRF, and uses the resulting beliefs to initialize finer resolution versions. Although they share the element of coarsening with our approach, the goal of coarse-to-fine approaches is very different from our objective. Specifically, the low-resolution MRFs only serve to speed-up convergence of the full resolution MRF via better initialization. Thus, one typically should not expect it to perform better than the finest granularity MRF. In contrast, our approach is designed to strictly improve the performance of the original MRF by introducing additional (coarse) clusters. One of the key technical differences is that in our formulation the setting of coarse and fine variables are refined iteratively whereas in Felzenszwalb & Huttenlocher (2006), once a coarse MRF has been solved, it is not revisited.

This coarsening strategy approach is also closely related to the method that we introduced in Section 3.4 for deriving valid constraints on the marginal polytope through projections onto the cut polytope. The partitioned-based consistency constraints can also be derived by a projection of the marginal polytope, although here we chose a different presentation for simplicity. Earlier we used projected cycle inequalities, not consistency constraints, to tighten LP relaxations for MAP. Focusing on consistency constraints allows us to derive a particularly simple message passing algorithm for solving the dual LP, while our use of partitionings helps to maintain sparsity in the constraints.

The first time that we add a cluster and perform coordinate-descent with respect to it, we in effect create a higher-order potential function involving the cluster variables – even though the original graphical model was edge-structured. The cluster partitioning ensures that this higher-order potential has extra structure (e.g., low rank) that allows us to efficiently compute outgoing messages from the cluster. Viewed in this way, our approach is related to other works that show how message-passing algorithms can efficiently compute messages for higher-order factors that are *sparse* (Duchi *et al.*, 2007; Gupta *et al.*, 2007; Tarlow *et al.*, 2010). This view also illustrates how our techniques may be helpful for the problem of finding the MAP assignment in graphical models with structured potentials, such as context-specific Bayesian networks.

As we mentioned at the end of Section 6.4, our algorithm sometimes add more than one

cluster involving the same variables, but with different partitionings. Rather than add a new cluster, we could instead *revise* an existing cluster's partitioning. For example, we could apply the heuristic from Section 6.3 to an existing cluster c, further partitioning the catch-all state s_i^c of each variable $i \in c$. This would result in a new partitioning that is strictly more fine-grained than the earlier one. In contrast, it seems to be much more difficult to *coarsen* a cluster's partitioning while improving the dual objective monotonically.

There are a number of interesting directions to explore. Using the same ideas as in this chapter, one can introduce coarsened pairwise consistency constraints in addition the full pairwise consistency constraints. Although this would not tighten the relaxation, by passing messages more frequently in the coarsened space, and only occasionally revisiting the full edges, this could give significant computational benefits when the nodes have large numbers of states. This would be much more similar to the coarse-to-fine approach described above. A related approach would be to use a Dantzig–Wolfe decomposition for the edge variables (Bertsimas & Tsitsiklis, 1997).

With the coarsening strategy used here, the number of variables still grows exponentially with the cluster size, albeit at a lower rate. One way to avoid the exponential growth is to build a hierarchy of coarsened variables. After adding a fine-grained cluster, we partition its states (corresponding to the cross-product of the states of its constituent variables) into a fixed number of coarsened states (e.g., two). Then, we introduce a new cluster that enforces the consistency of two or more coarsened clusters. Such a process can be repeated recursively. The key advantage of this approach is that it represents progressively larger clusters, but with no exponential growth. An interesting open question is to understand how these hierarchies should be constructed such that they are both efficient and succeed in tightening the relaxation.

Chapter 7

Solving the Cycle Relaxation in the Dual

In this chapter we address the most significant shortcoming of the message-passing algorithm given in Chapter 5: the difficulty of finding where to tighten the relaxation in sparse graphs. Our earlier approach was to explicitly enumerate the short cycles (e.g., of size 3 or 4) in the graph, using the bound criterion to select the most promising cycle to use in tightening the relaxation. However, when there are few short cycles, this explicit enumeration is not feasible. We show in Section 7.2 that, for non-binary graphical models (e.g., protein design or stereo vision), it is NP-hard to find the best cycle according to the bound criterion. Thus we need an alternative approach to address the search problem of finding where to tighten the relaxation.

We describe a new approach where, as before, we solve the *dual* of the pairwise LP relaxation, but where we now add k-ary cycle inequalities to tighten the relaxation rather than cluster consistency constraints. We consider the same greedy bound minimization criterion used earlier for cluster consistency constraints, corresponding to the amount that the dual objective would decrease with just one coordinate descent step on the new dual variable(s). We show that one can, in near-linear time (in the size of the projection graph), find the best k-ary cycle inequality according to this criterion. The resulting algorithm is similar to the separation algorithm given in Chapter 3 for finding the most violated cycle inequality in the primal, but is not based on shortest paths.

Somewhat surprisingly, in Section 7.2 we show that, for *binary* graphial models and when the dual has been solved to optimality, the two bound criterions (the one given in Chapter 5 and this one) coincide. Thus, at least for binary graphical models, the algorithm that we present completely solves the open problem from Chapter 5 of how to efficiently search over cycle clusters. Although we use the cycle inequalities as a means of obtaining an efficient search algorithm, we could instead add to the relaxation the cycle consistency constraint corresponding to the cycle, rather than the violated cycle inequality.

The cycle inequalities for the marginal polytope are similar in form to the cycle inequalities that we used to enforce acyclicity in LP relaxations for learning the structure of Bayesian networks (Jaakkola *et al.*, 2010). The dual coordinate descent and separation algorithms presented here closely mirror the techniques that we used in this other work.

7.1 Cycle Inequalities in the Dual

Recall the primal k-ary cycle inequalities from Eq. 3.3, where we denote C as a cycle in G, π specifies a (binary) partitioning of the states of the variables in C, and $F \subseteq C$, $|F|$ odd:

$$\sum_{ij \in C \setminus F} \sum_{\substack{x_i, x_j : \\ \pi_i(x_i) \neq \pi_j(x_j)}} \mu_{ij}(x_i, x_j) + \sum_{ij \in F} \sum_{\substack{x_i, x_j : \\ \pi_i(x_i) = \pi_j(x_j)}} \mu_{ij}(x_i, x_j) \quad \geq \quad 1 \tag{7.1}$$

We show in Appendix A.3 how to derive the dual of the LP relaxation which includes all of the k-ary cycle inequalities. The dual LP is $\min_{\lambda \geq 0, \delta} J(\delta, \lambda)$, where

$$\begin{aligned} J(\delta, \lambda) \quad = \quad & \sum_i \max_{x_i} \Big(\theta_i(x_i) + \sum_{j \in N(i)} \delta_{ji}(x_i) \Big) \\ & + \sum_{ij} \max_{x_i, x_j} \Big(\theta_{ij}(x_i, x_j) - \delta_{ji}(x_i) - \delta_{ij}(x_j) \\ & \qquad + \sum_{C, F, \pi \,:\, ij \in C \setminus F} \lambda_{C,F,\pi} 1[\pi_i(x_i) \neq \pi_j(x_j)] \\ & \qquad + \sum_{C, F, \pi \,:\, ij \in F} \lambda_{C,F,\pi} 1[\pi_i(x_i) = \pi_j(x_j)] \Big) \\ & - \sum_{C, F, \pi} \lambda_{C,F,\pi} \ . \end{aligned} \tag{7.2}$$

The node and edge beliefs (as a function of the dual variables) are:

$$\begin{aligned} b_i(x_i) \quad &= \quad \theta_i(x_i) + \sum_{j \in N(i)} \delta_{ji}(x_i), \\ b_{ij}(x_i, x_j) \quad &= \quad \theta_{ij}(x_i, x_j) - \delta_{ji}(x_i) - \delta_{ij}(x_j) + \sum_{C, F, \pi \,:\, ij \in F} \lambda_{C,F,\pi} 1[\pi_i^q(x_i) = \pi_j^r(x_j)] \\ & \quad + \sum_{C, F, \pi \,:\, ij \in C \setminus F} \lambda_{C,F,\pi} 1[\pi_i^q(x_i) \neq \pi_j^r(x_j)]. \end{aligned} \tag{7.3}$$

When the dual variables $\boldsymbol{\lambda}, \delta$ are optimal, we can use the complementary slackness conditions (c.f. Section 4.4, with $b_{ij}(x_i, x_j)$ replacing $f_{ij}(x_i, x_j)$) to find a primal optimal solution.

7.1.1 Using Cycle Inequalities within Message Passing

We next show how to take a coordinate descent step with respect to one $\lambda_{C,F,\pi}$ variable. The corresponding update can be used together with the message passing algorithms that we described in the earlier chapters (which, recall, correspond to performing block coordinate descent in the dual of the pairwise LP relaxation).

The dual objective $J(\delta, \lambda)$ is a piecewise linear function of $\lambda_{C,F,\pi}$ (see Figure 7-1). As long as $\lambda_{C,F,\pi}$ does not play a role in the edge terms (we will make this precise in a moment), we can *increase* the value of $\lambda_{C,F,\pi}$, thereby decreasing $J(\delta, \lambda)$. On the other hand, if $\lambda_{C,F,\pi}$ is active in two or more edge terms (which outweighs the single $-\lambda_{C,F,\pi}$ term), *decreasing* its value will decrease $J(\delta, \lambda)$.

For $ij \in C$, consider the inside of the edge maximization terms, ignoring the terms that

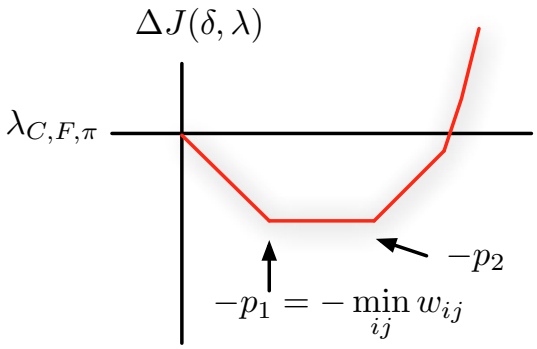


Figure 7-1: Illustration of decrease in dual objective (see Eq. 7.2) as a function of $\lambda_{C,F,\pi}$. p_1 refers to the minimal value in $\{w_{ij}\}$, while p_2 is the next smallest value. Clearly we must have $w_{ij} > 0$ for all edges, as otherwise the decrease would be zero.

involve $\lambda_{C,F,\pi}$. Defining

$$b_{ij}^{-C,F,\pi}(x_i, x_j) = \theta_{ij}(x_i, x_j) - \delta_{ji}(x_i) - \delta_{ij}(x_j) + \sum_{\substack{(C',F',\pi') \neq (C,F,\pi) \\ : ij \in F'}} \lambda_{C',F',\pi'} 1[\pi'_i(x_i) = \pi'_j(x_j)] + \sum_{\substack{(C',F',\pi') \neq (C,F,\pi) \\ : ij \in C' \setminus F'}} \lambda_{C',F',\pi'} 1[\pi'_i(x_i) \neq \pi'_j(x_j)] \ ,$$

we can rewrite the relevant terms of the dual objective in Eq. 7.2 as

$$\begin{aligned} J(\lambda_{C,F,\pi}) \quad = \quad & \sum_{ij \in F} \max_{x_i, x_j} \left(b_{ij}^{-C,F,\pi}(x_i, x_j) + \lambda_{C,F,\pi} 1[\pi_i(x_i) = \pi_j(x_j)] \right) \\ & + \sum_{ij \in C \setminus F} \max_{x_i, x_j} \left(b_{ij}^{-C,F,\pi}(x_i, x_j) + \lambda_{C,F,\pi} 1[\pi_i(x_i) \neq \pi_j(x_j)] \right) \\ & - \lambda_{C,F,\pi} \ . \end{aligned}$$

If $ij \in F$, we call $\lambda_{C,F,\pi}$ *active* for edge ij when $\max_{x_i,x_j \,:\, \pi_i(x_i)=\pi_j(x_j)} \left(b_{ij}^{-C,F,\pi}(x_i, x_j) + \lambda_{C,F,\pi} \right) \geq \max_{x_i,x_j} b_{ij}^{-C,F,\pi}(x_i, x_j)$, in which case further increasing $\lambda_{C,F,\pi}$ results in a linear increase in the corresponding edge term of $J(\lambda_{C,F,\pi})$. Similarly, if $ij \notin F$, we call $\lambda_{C,F,\pi}$ active for edge ij when $\max_{x_i,x_j \,:\, \pi_i(x_i)\neq\pi_j(x_j)} \left(b_{ij}^{-C,F,\pi}(x_i, x_j) + \lambda_{C,F,\pi} \right) \geq \max_{x_i,x_j} b_{ij}^{-C,F,\pi}(x_i, x_j)$. We define w_{ij} to be the largest that $\lambda_{C,F,\pi}$ can be before becoming active for edge ij:

$$\begin{aligned} w_{ij} = & \max_{x_i,x_j \,:\, \pi_i(x_i)\neq\pi_j(x_j)} b_{ij}^{-C,F,\pi}(x_i, x_j) - \max_{x_i,x_j \,:\, \pi_i(x_i)=\pi_j(x_j)} b_{ij}^{-C,F,\pi}(x_i, x_j) \qquad \text{if } ij \in F, \\ & \max_{x_i,x_j \,:\, \pi_i(x_i)=\pi_j(x_j)} b_{ij}^{-C,F,\pi}(x_i, x_j) - \max_{x_i,x_j \,:\, \pi_i(x_i)\neq\pi_j(x_j)} b_{ij}^{-C,F,\pi}(x_i, x_j) \qquad \text{if } ij \notin F. \end{aligned} \tag{7.4}$$

When $\min_{ij \in C} w_{ij} > 0$, the dual objective $J(\delta, \lambda)$ decreases as $\lambda_{C,F,\pi}$ increases, until $\lambda_{C,F,\pi} = \min_{ij \in C} w_{ij} = p_1$ (let ij^* denote the argmin). At this point, the function has zero slope, and remains constant until $\lambda_{C,F,\pi} = \min_{ij \neq ij^*} w_{ij} = p_2$. Thus, by setting

$\lambda_{C,F,\pi} = \frac{p_1+p_2}{2}$ we obtain the maximal decrease. When $p_2 \neq p_1$, there are a range of values for $\lambda_{C,F,\pi}$ that achieve the maximal decrease in the dual objective. We choose the midpoint because it leads to dual optimal solutions for which "decoding", or finding the corresponding primal optimal solution, is easier (we illustrate this in Example 4).

The amount that the dual objective decreases with one coordinate descent step on $\lambda_{C,F,\pi}$, assuming that $\lambda_{C,F,\pi}$ was previously zero, is

$$d(C, F, \pi) = \max(0, \min_{ij \in C} w_{ij}). \tag{7.5}$$

When the cycle inequality on C, F, π is already in the relaxation (i.e., $\lambda_{C,F,\pi} > 0$), the overall change in $J(\delta, \lambda)$ can be zero (although $\lambda_{C,F,\pi}$ might have a different value due to the midpoint moving).

Example 4. Consider a triangle on three edges ($C = \{12, 23, 31\}$), with $x_i \in \{0, 1\}$, where $\theta_i(x_i) = 0\ \forall i, x_i$ and $\theta_{ij}(x_i, x_j) = 1$ if $x_i \neq x_j$, and 0 otherwise.[1] Let all of the dual variables $\delta_{ji}(x_i)$ be 0, and assume that initially there are no cycle inequalities. The best integer solution has value 2, while the pairwise LP relaxation gives only a loose upper bound of 3 (note: δ as defined can be shown to be optimal for the dual of the pairwise LP relaxation, i.e. Eq. 7.2 with $\boldsymbol{\lambda} = 0$).

Consider the problem of finding the best cycle inequality according to $\arg\max_{C,F} d(C, F)$. First, note that $b_{ij}(x_i, x_j) = \theta_{ij}(x_i, x_j)$, so $w_{ij} = 1$ for $ij \in F$ and $w_{ij} = -1$ for $ij \notin F$. If $F = \emptyset$, then $w_{ij} = -1$ for all edges, and so $d(C, F) = 0$. On the other hand, if $F = C$, then $w_{ij} = 1$ for all edges, which gives a bound decrease of $d(C, F) = 1$, corresponding to $\lambda_{C,F} = 1$.

After this one coordinate descent step involving the new cycle inequality, the dual objective has value 2, which is optimal. Note, however, that finding the optimal *primal* solution is non-trivial because no assignments can be ruled out by the complementary slackness conditions on the edges. Any primal feasible point that satisfies Eq. 7.1 for $F = C$ with equality is optimal.

If, instead, we had $\theta_{12}(x_1 \neq x_2) = .95$, then (still for $F = C$) we would have $w_{12}(C, F) = .95$ so $d(C, F) = .95$ and $\lambda_{C,F} = \frac{.95+1}{2}$. Consider edge $(1, 2)$. We have that $b_{12}(1, 0) = b_{12}(0, 1) = 0.95$ and $b_{12}(1, 1) = b_{12}(0, 0) = 0.975$, so the complementary slackness conditions tell us that the MAP assignment has x_1^M, x_2^M either equal to $1, 1$ or $0, 0$ (i.e., the edge is not cut). Thus, finding the primal optimal solution (decoding) is much easier.

7.1.2 Separation Algorithm

In this section we will show how to efficiently find k-ary cycle inequalities to use in tightening the relaxation. We use a similar greedy bound minimization criteria to the one suggested in Chapter 5 for cluster consistency,

$$\max_{C,F \subseteq C \text{ s.t. } |F| \text{ odd}, \pi} d(C, F, \pi). \tag{7.6}$$

We show that this can be computed efficiently using a variation on the shortest-path based separation algorithm for cycle inequalities presented in Section 3.4.2.

[1] Since this example is binary, we let the projection be the identity, i.e. $\pi_i(x_i) = x_i$. We omit the π notation from the rest of the example.

Let G_π denote the projection graph, where the variables and edges are as defined in Eq. 3.9. All subsequent quantities will use the partitions for the variables specified by the edges $(\pi_i^q, \pi_j^r) \in E_\pi$ in the projection graph. For now, assume that the cycle inequality that we are going to add is not already in the relaxation.

The separation algorithm takes as input the projection graph $G_\pi = (V_\pi, E_\pi)$ and the current beliefs $b_{ij}(x_i, x_j)$. For each edge $mn = (\pi_i^q, \pi_j^r) \in E_\pi$, define the weight

$$s_{mn} = \max_{x_i, x_j \,:\, \pi_i^q(x_i) = \pi_j^r(x_j)} b_{ij}(x_i, x_j) - \max_{x_i, x_j \,:\, \pi_i^q(x_i) \neq \pi_j^r(x_j)} b_{ij}(x_i, x_j).$$

Remove all edges with $s_{mn} = 0$. We then have that

$$\max_{C, F \subseteq C \text{ s.t. } |F| \text{ odd}, \pi} d(C, F, \pi) = \max\left(0, \max_{C, F \subseteq C \text{ s.t. } |F| \text{ odd}, \pi} \min_{mn \in C} w_{mn}\right) \tag{7.7}$$

$$= \max\left(0, \max_{\substack{C \subseteq E_\pi \text{ s.t.} \\ \prod_{mn \in C} \text{sign}(s_{mn}) = -1}} \min_{mn \in C} |s_{mn}|\right), \tag{7.8}$$

where the change from Eq. 7.7 to Eq. 7.8 is because for $d(C, F, \pi)$ to be positive, we need $mn \in F$ when $s_{mn} < 0$, and $mn \notin F$ when $s_{mn} > 0$.

Thus, the maximum bound decrease is achieved by the cycle in G_π with an odd number of edges in F that maximizes the minimum weight along the cycle. Suppose that $s_{mn} \in \{+1, -1\}$. Then, $\min_{mn \in C} |s_{mn}| = 1$ and the optimization problem simplifies to finding a cycle with an odd number of -1 edges. This can be solved in linear time by the following algorithm (if the graph is not connected, do this for each component): First, construct a rooted spanning tree. Assign the root node r the value $v_r = +1$. Then, propagate labels towards the leaves, assigning each node the value $v_{p(m)} s_{m,p(m)}$, where $p(m)$ denotes the parent of node m. Finally, for each edge mn that is not in the tree, check to see if $v_m = v_n s_{mn}$. If the answer is ever "no", then we have found a cycle with an odd number of negative edges.[2]

Now consider the case of general s_{mn}. We can solve the optimization in Eq. 7.8 by doing a binary search on $|s_{mn}|$. There are only $|E_\pi|$ possible edge weights, so to do this search we first sort the values $\{|s_{mn}| : mn \in E_\pi\}$. At each step, we consider the subgraph G' consisting of all $mn \in E_\pi$ such that $|s_{mn}| > R$, where R is the threshold used in the binary search. We then let $s_{m'n'} = \text{sign}(s_{mn})$ for $m'n' \in G'$, and search for a cycle with an odd number of negative edges using the algorithm described in the previous paragraph. The binary search will find the largest R such that there is a negative-signed cycle $C \in G'$, if one exists. We then add the cycle inequality C, F, π to the relaxation, where F corresponds to the negative edges in C and π is given by the partitions used by the cycle C. The total running time is only $O(|E_\pi| \log |E_\pi|)$.

If the existing dual variables $\lambda_{C,F,\pi}$ are not optimal for $\{\delta_{ij}(x_j)\}$ then it is possible that this separation algorithm would return a cycle inequality which is already included in the relaxation. In this case, we can simply increment the value of the existing $\lambda_{C,F,\pi}$. Note that the separation algorithm cannot *decrease* the value of an existing $\lambda_{C,F}$, so to obtain the dual optimum we must make sure to repeatedly re-optimize the new dual variables, e.g. using the coordinate descent scheme described in the previous section.

[2] In the case when there is more than one cycle with an odd number of -1 edges, the particular cycle that we find depends on the choice of spanning tree. However, the algorithm is always guaranteed to find *some* cycle with an odd number of -1 edges, when one exists, regardless of the choice of spanning tree.

7.2 Separating the Cycle Relaxation

Consider the restricted set of clusters $\mathcal{C}_{\text{cycles}}(G)$ corresponding to cycles of arbitrary length,

$$\mathcal{C}_{\text{cycles}}(G) = \Big\{ C \subseteq E \mid C \text{ forms a cycle in } G \Big\}. \tag{7.9}$$

If we enforce that the pseudomarginals along each cycle in $\mathcal{C}_{\text{cycles}}(G)$ arise from some joint distribution, and enforce consistency between this distribution and the edge marginals, we recover the cycle relaxation CYCLE(G) introduced in Section 3.1.

A natural question is whether it is possible to find the best cycle *cluster* to add to the relaxation, according to the greedy bound minimization criteria

$$d(C) = \sum_{e \in C} \max_{\mathbf{x}_e} b_e(\mathbf{x}_e) - \max_{\mathbf{x}_C} \left[\sum_{e \in C} b_e(\mathbf{x}_e) \right] , \tag{7.10}$$

where $b_e(\mathbf{x}_e)$ is defined in Eq. A.6. The bound criterion $d(C)$ is the amount that the dual objective would decrease following one block coordinate descent step on all of the dual edge variables in the cycle. Its form is very intuitive: it is the difference between independently maximizing each of the edge beliefs in the cycle (one can think of this as a heuristic to find the MAP assignment) and jointly maximizing over the edge beliefs of the cycle.

We show the following, where k refers to the number of states per node.

1. For $k = 2$, when the beliefs $b_e(\mathbf{x}_e)$ are dual optimal, maximizing Eq. 7.10 is *equivalent* to finding the best cycle inequality in the dual.[3]

2. For $k = 2$, maximizing Eq. 7.10 is NP-hard when the beliefs $b_e(\mathbf{x}_e)$ are not dual optimal.

3. For $k > 2$, maximizing Eq. 7.10 is always NP-hard.

By dual optimal, we mean that the beliefs correspond to a dual optimal solution of the current LP relaxation. Note that, before solving the dual LP to optimality, $b_e(\mathbf{x}_e)$ can be almost anything. For example, we can set $\theta_e(\mathbf{x}_e)$ arbitrarily and consider the separation problem at the first iteration.

Theorem 7.2.1. *When $k = 2$ and the beliefs $b_{ij}(x_i, x_j)$ correspond to a dual optimal solution,* $\max_{C \in \mathcal{C}_{cycles}(G)} d(C) = \max_{C,F:|F| \text{ odd}} d(C, F)$.

Proof. First, defining $w_e(\mathbf{x}_e) = \max_{\hat{\mathbf{x}_e}} b_e(\hat{\mathbf{x}_e}) - b_e(\mathbf{x}_e)$, note that

$$\begin{aligned} d(C) &= \sum_{e \in C} \max_{\mathbf{x}_e} b_e(\mathbf{x}_e) - \max_{\mathbf{x}_C} \left[\sum_{e \in C} b_e(\mathbf{x}_e) \right] && (7.11) \\ &= \sum_{e \in C} \max_{\mathbf{x}_e} b_e(\mathbf{x}_e) + \min_{\mathbf{x}_C} \left[- \sum_{e \in C} b_e(\mathbf{x}_e) \right] && (7.12) \\ &= \min_{\mathbf{x}_C} \sum_{e \in C} \left[\left(\max_{\hat{\mathbf{x}_e}} b_e(\hat{\mathbf{x}_e}) \right) - b_e(\mathbf{x}_e) \right] = \min_{\mathbf{x}_C} \sum_{e \in C} w_e(\mathbf{x}_e) \ . && (7.13) \end{aligned}$$

[3]This result is for binary variables only, so we let the projection be $\pi_i(x_i) = x_i$ and omit the π notation.

Our proof proceeds as follows. In part (a) we show that for any cycle C where $d(C) > 0$, for all edges $ij \in C$, either $\arg\max_{x_i,x_j} b_{ij}(x_i, x_j) = \{(0,0),(1,1)\}$ or $\arg\max_{x_i,x_j} b_{ij}(x_i, x_j) = \{(1,0),(0,1)\}$.[4] Then, calling the edges with maximizing assignments equal to $\{(1,0),(0,1)\}$ "cut" and the edges with maximizing assignments equal to $\{(0,0),(1,1)\}$ "not cut", we show in part (b) that a cycle C has $d(C) > 0$ if and only if it has an *odd* number of cut edges. In part (c) we show that, when $d(C) > 0$, $d(C) = \min_{e \in C, \mathbf{x}_e \text{ s.t. } w_e(\mathbf{x}_e) > 0} w_e(\mathbf{x}_e)$.

Recall that, by part (a), $b_{ij}(0,0) = b_{ij}(1,1)$ for an edge that is not cut and $b_{ij}(0,1) = b_{ij}(1,0)$ for an edge that is cut. Let the cost of "cutting" an edge ij refer to the smallest value t such that either of $b_{ij}(0,1)+t$ or $b_{ij}(1,0)+t$ is equal in value to $b_{ij}(0,0)$ and $b_{ij}(1,1)$ Similarly, let the cost of "un-cutting" an edge ij refer to the smallest value t such that either of $b_{ij}(0,0)+t$ or $b_{ij}(1,1)+t$ is equal in value to $b_{ij}(0,1)$ and $b_{ij}(1,0)$. It follows from part (c) that, when $d(C) > 0$, $d(C)$ is the minimal cost, over all edges in C, of cutting an edge that is not cut, or un-cutting an edge that is cut. Thus, letting F' be the set of cut edges in C, when $d(C) > 0$ we have

$$\min_{e \in C, \mathbf{x}_e \text{ s.t. } w_e(\mathbf{x}_e) > 0} w_e(\mathbf{x}_e) = \max_{F \subseteq C : |F| \text{ odd}} \min_{ij \in C} w_{ij} \ , \tag{7.14}$$

where w_{ij} was defined in Eq. 7.4. The equality is because $\min_{e \in C, \mathbf{x}_e \text{ s.t. } w_e(\mathbf{x}_e) > 0} w_e(\mathbf{x}_e) = \min_{ij \in C} w_{ij}$ for $F = F'$ (and, by part (b), $|F'|$ is odd), and $\min_{ij \in C} w_{ij} < 0$ for $F \neq F'$ (whereas the left hand side of Eq. 7.14 is positive). By part (b), when $d(C) = 0$ we have that $\min_{ij \in C} w_{ij} < 0$ for *all* $|F|$ odd, so $\max_{F \subseteq C : |F| \text{ odd}} d(C, F) = 0$. We conclude that $d(C) = \max_{F \subseteq C : |F| \text{ odd}} d(C, F)$, which shows the claim.

(a) We first show that either $\{(0,0),(1,1)\} \subseteq \arg\max_{x_i,x_j} b_{ij}(x_i, x_j)$ or $\{(1,0),(0,1)\} \subseteq \arg\max_{x_i,x_j} b_{ij}(x_i, x_j)$, or both (these are not mutually exclusive). By the definition of $w_e(\mathbf{x}_e)$, every edge ij has at least one assignment x_i, x_j such that $w_{ij}(x_i, x_j) = 0$. Suppose for contradiction that there is an edge ij such that $\{(0,0),(1,1)\} \not\subseteq \arg\max_{x_i,x_j} b_{ij}(x_i, x_j)$ and $\{(1,0),(0,1)\} \not\subseteq \arg\max_{x_i,x_j} b_{ij}(x_i, x_j)$. We will show just the following case, with the others following by similar arguments:

$$(1,1) \notin \arg\max_{x_i,x_j} b_{ij}(x_i, x_j) \ , \tag{7.15}$$

$$(1,0) \notin \arg\max_{x_i,x_j} b_{ij}(x_i, x_j) \ , \text{ and} \tag{7.16}$$

$$(0,0) \in \arg\max_{x_i,x_j} b_{ij}(x_i, x_j) \ . \tag{7.17}$$

Let μ be any primal optimal solution corresponding to b. By complementary slackness, we have that $\mu_{ij}(0,0) > 0$, which implies that $\mu_i(0) > 0$. Let j and k denote the two neighbors of node i in the cycle. By complimentary slackness and the pairwise consistency constraints, for any x_i, if there exists x_k such that $w_{ki}(x_k, x_i) = 0$, then there exists x_j such that $w_{ij}(x_i, x_j) = 0$. Using this property, we can construct an assignment $\mathbf{x}_C$ for the variables of the cycle such that $\sum_{e \in C} w_e(\mathbf{x}_e) = 0$ by starting with $x_i = 0$ and consecutively setting each neighbor's assignment (starting with k, and continuing in the same direction along the cycle). Importantly, we must return to $x_i = 0$ because we have assumed that $w_{ij}(1,0) > 0$ and $w_{ij}(1,1) > 0$. We have thus contradicted our assumption that $d(C) > 0$.

To show the equality, suppose for contradiction that there is an edge ij such that

[4] We use the notation $\arg\max_{x_i,x_j} b_{ij}(x_i, x_j)$ to refer to the *set* of assignments $\hat{x}_i, \hat{x}_j$ such that $b_{ij}(\hat{x}_i, \hat{x}_j) = \max_{x_i,x_j} b_{ij}(x_i, x_j)$.

$\{(0,0),(1,1),(0,1)\} \subseteq \arg\max_{x_i,x_j} b_{ij}(x_i,x_j)$. Then, we can construct an assignment using the same technique, starting at $x_i = 0$ and going in the direction of k, which shows that $d(C) = 0$, again contradicting our assumption.

(b) Since $w_e(\mathbf{x}_e) \geq 0$, $\sum_{e \in C} w_e(\mathbf{x}_e) = 0$ if and only if every edge assignment $\mathbf{x}_e$ is consistent with our definition of "cut" and "not cut". However, any assignment to the variables in a cycle must correspond to an even number of cut edges (c.f. Section 3.2). Thus, $d(C) = 0$ if and only if the cycle has an even number of cut edges.

(c) If $d(C) > 0$, then there are an odd number of cut edges. Cutting an uncut edge or un-cutting a cut edge would make the cycle have an even number of cut edges, and so there would be an assignment $\mathbf{x}_C$ such that $w_e(\mathbf{x}_e) = 0$ for all edges other than the modified edge, using the construction from part (a). Thus, for all edges e' and edge assignments $\mathbf{x}_{e'}$ such that $w_{e'}(\mathbf{x}_{e'}) > 0$, $\min_{\mathbf{x}_{C \setminus e'}} \sum_{e \in c} w_e(\mathbf{x}_e) = w_{e'}(\mathbf{x}_{e'})$. Since $w_e(\mathbf{x}_e) \geq 0$ always, if $d(C) > 0$ then for all assignments $\mathbf{x}_c$, there must be some e' and $\mathbf{x}_{e'}$ such that $w_{e'}(\mathbf{x}_{e'}) > 0$. However, we just showed that there exists a completion $\mathbf{x}_{C' \setminus e'}$ such that $w_e(\mathbf{x}_e) = 0$ for all edges $e \neq e'$. Thus, when $d(C) > 0$, the value of $d(C)$ is equal to $\min_{e \in C, \mathbf{x}_e \text{ s.t. } w_e(\mathbf{x}_e) > 0} w_e(\mathbf{x}_e)$. □

The proof of Theorem 7.2.1 uses the assumption that $b_{ij}(x_i, x_j)$ corresponds to a dual optimal solution only when applying the complementary slackness conditions for the edge variables. Thus, Theorem 7.2.1 holds for any LP relaxation of the MAP problem which is at least as tight as the pairwise relaxation. In particular, adding cycle inequalities does not change the premise of the theorem.

One conclusion that is immediate given Theorem 7.2.1 is that the cycle inequalities give at least as tight of a relaxation as the cycle relaxation. In fact, for a single cycle, just one cycle inequality suffices to make the LP relaxation tight for a given instance. This is precisely what we observed in Example 4.

Corollary 7.2.2. *For any binary pairwise MRF consisting of a single cycle, it is always possible to make the pairwise LP relaxation tight with just* one *additional cycle inequality.*

7.2.1 NP-Hardness Results

Recall from Chapter 5 that we obtained much better results by tightening the relaxation even before the dual was solved to optimality (see Figure 5-2). Unfortunately, although we showed in Section 7.1 that for cycle inequalities $\max_{C,F} d(C,F)$ can be computed efficiently, the corresponding problem for cycle consistency constraints is significantly more difficult:

Theorem 7.2.3. *The optimization problem* $\max_{C \in \mathcal{C}_{cycles}(G)} d(C)$ *is NP-hard for* $k = 2$ *and beliefs* $b_{ij}(x_i, x_j)$ *arising from a non-optimal dual feasible point.*

Proof. Our reduction is from the Hamiltonian cycle problem, which is known to be NP-hard. The Hamiltonian cycle problem is: Given a graph $G = (V, E)$, decide whether there exists a cycle in G that visits all vertices exactly once.

We show how to efficiently construct a Markov random field $G' = (V', E')$ with $x_i \in \{0, 1\}$ and beliefs $b_{ij}(x_i, x_j)$ for $ij \in E'$ such that there is a 1-1 mapping between cycles $C \in G$ and $C' \in G'$, and evaluating $d(C')$ for $C' \in G'$ gives the length of the corresponding cycle in G. As a result, we have that $\max_{C' \in \mathcal{C}_{\text{cycles}}(G')} d(C')$ gives the length of the longest cycle in G. Thus, if we could solve this optimization problem, then, simply by checking whether the solution is $|V|$, we answer the Hamiltonian cycle problem.

Let $V' = V \cup \{x_{ij}, \forall ij \in E\}$, where we introduce a new variable x_{ij} for every edge in E. The edges are $E' = \{(i, x_{ij}), (x_{ij}, j), \forall ij \in E\}$, where we replace every edge in G with a length-2 path in G'. For each $ij \in E$, denoting $k = x_{ij}$, we let the beliefs be:

$b_{ik}(x_i, x_{ij}) =$

	$x_{ij} = 0$	$x_{ij} = 1$
$x_i = 0$	$\lvert V\rvert$	0
$x_i = 1$	0	0

$b_{kj}(x_{ij}, x_j) =$

	$x_j = 0$	$x_j = 1$
$x_{ij} = 0$	0	0
$x_{ij} = 1$	0	1

Then, we have that:

$w_{ik}(x_i, x_{ij}) =$

	$x_{ij} = 0$	$x_{ij} = 1$
$x_i = 0$	0	$\lvert V\rvert$
$x_i = 1$	$\lvert V\rvert$	$\lvert V\rvert$

$w_{kj}(x_{ij}, x_j) =$

	$x_j = 0$	$x_j = 1$
$x_{ij} = 0$	1	1
$x_{ij} = 1$	1	0

As a result of our construction, every cycle $C \in G$ on nodes $i, j, k, \ldots, m$ corresponds 1-1 with the cycle $C' \in G'$ on nodes $i, x_{ij}, j, x_{jk}, \ldots, m, x_{mi}$. It can be verified that for all cycles $C' \in G'$, $d(C') = \min_{\mathbf{x}_{C'}} \sum_{e \in C'} w_e(\mathbf{x}_e) = |C'|/2 = |C|$, where C is the cycle corresponding to C' (the minimum is attained by the all zeros assignment). We thus have that $\max_{C' \in \mathcal{C}_{\text{cycles}}(G')} d(C') = \max_{C \in G} |C|$. □

We next show that, for $k > 2$, not even dual optimality helps:

Theorem 7.2.4. *The optimization problem* $\max_{C \in \mathcal{C}_{cycles}(G)} d(C)$ *is NP-hard for* $k \geq 3$ *even for beliefs* $b_{ij}(x_i, x_j)$ *corresponding to a dual optimal solution of the pairwise relaxation.*

Proof. As in the proof of Theorem 7.2.3, we reduce from the Hamiltonian cycle problem, for an input graph G. First, we show that the Hamiltonian cycle problem is NP-hard even when restricted to graphs with an *odd* number of nodes, by reducing from the general case. Suppose we are given a graph with an even number of nodes and we want to decide whether it has a Hamiltonian cycle. We repeat the following, once for each edge ij: construct a new graph which is identical to the original except that we introduce a new node n and replace the edge ij with the edges in and nj. We then check whether any of the new graphs (all of which now have an odd number of vertices) have a Hamiltonian cycle. If the answer is "yes", we have found a Hamiltonian cycle for the original graph. Otherwise, the original graph does not have a Hamiltonian cycle.

Assume for the rest of the proof that G has an odd number of vertices. We show how to efficiently construct a Markov random field $G' = (V', E')$ with $x_i \in \{0, 1, 2\}$ and beliefs $b_{ij}(x_i, x_j)$ for $ij \in E'$ such that there is a 1-1 mapping between cycles $C \in G$ and $C' \in G'$, and evaluating $d(C')$ for $C' \in G'$ gives the length of the corresponding cycle in G. As a result, we have that $\max_{C' \in \mathcal{C}_{\text{cycles}}(G')} d(C')$ gives the length of the longest cycle in G. Thus, if we could solve this optimization problem, then, simply by checking whether the solution is $|V|$, we answer the Hamiltonian cycle problem.

Let $V' = V \cup \{x_{ij}, \forall ij \in E\}$, where we introduce a new variable x_{ij} for every edge in E, also with 3 states. The edges are $E' = \{(i, x_{ij}), (x_{ij}, j), \forall ij \in E\}$, where we replace every edge in G with a length-2 path in G'. For each $ij \in E$, denoting $k = x_{ij}$, we let the beliefs

be:

$b_{ik}(x_i, x_{ij})$ =

	$x_{ij}=0$	$x_{ij}=1$	$x_{ij}=2$
$x_i=0$	$\|V\|$	0	0
$x_i=1$	0	$\|V\|$	0
$x_i=2$	0	0	$\|V\|-.5$

(7.18)

$b_{kj}(x_{ij}, x_j)$ =

	$x_j=0$	$x_j=1$	$x_j=2$
$x_{ij}=0$	0	$\|V\|$	0
$x_{ij}=1$	$\|V\|$	0	0
$x_{ij}=2$	0	0	$\|V\|-.5$

As a result of our construction, every cycle $C \in G$ on nodes $i, j, k, \ldots, m$ corresponds 1-1 with the cycle $C' \in G'$ on nodes $i, x_{ij}, j, x_{jk}, \ldots, m, x_{mi}$. Every cycle $C \in G$ where $|C|$ is even corresponds to a cycle $C' \in G'$ such that $\min_{\mathbf{x}_{C'}} \sum_{e \in C'} w_e(\mathbf{x}_e) = 0$ (the minimum is attained by the assignment $0011 \ldots 0011$). On the other hand, every cycle $C \in G$ where $|C|$ is *odd* corresponds to a cycle $C' \in G'$ such that $\min_{\mathbf{x}_{C'}} \sum_{e \in C'} w_e(\mathbf{x}_e) = .5|C'| = |C|$ (the minimum is attained by the assignment of 2 to every node). Thus, G (which has an odd number of nodes) has a Hamiltonian cycle if and only if $\max_{C' \in \mathcal{C}_{\text{cycles}}(G')} d(C') = |V|$.

What remains is to show that the beliefs $b_{ij}(x_i, x_j)$ that we constructed are dual optimal for some potentials $\theta(\mathbf{x})$. We do this by illustrating a primal and dual feasible point for which the primal objective is equal to the dual objective. Let $\theta_{ij}(x_i, x_j) = b_{ij}(x_i, x_j)$ and $\delta_{ij}(x_j) = 0$ for all edges $ij \in E$ and assignments x_j. Clearly $\delta_{ij}(x_j)$ is dual feasible, and it gives an objective value of $|E||V|$. Consider the following primal point μ:

$\mu_{ik}(x_i, x_{ij})$ =

	$x_{ij}=0$	$x_{ij}=1$	$x_{ij}=2$
$x_i=0$	.5	0	0
$x_i=1$	0	.5	0
$x_i=2$	0	0	0

(7.19)

$\mu_{kj}(x_{ij}, x_j)$ =

	$x_j=0$	$x_j=1$	$x_j=2$
$x_{ij}=0$	0	.5	0
$x_{ij}=1$	.5	0	0
$x_{ij}=2$	0	0	0

The point $\boldsymbol{\mu}$ satisfies the pairwise consistency constraints (the single node marginals are $\mu_i(x_i) = .5$ for $x_i \in \{0, 1\}$, and 0 otherwise), and has objective value $|E||V|$. Note that μ and δ also satisfy the complementary slackness conditions (as they must, since they are a primal-dual optimal pair). □

Remark 7.2.5. *The previous construction also shows that the pairwise relaxation* does not *guarantee* persistency *in the non-binary setting, since some of the* 0*'s in the optimal fractional solution are non-zero in the optimal integral solution.*

7.3 Discussion

Both of the constructions used in the hardness results of Theorems 7.2.3 and 7.2.4 were such that there were many cycles that were inconsistent. Thus, although we showed that finding the *best* of these according to the greedy bound minimization criteria is NP-hard for $k > 2$, for these examples we could have easily found some cycle which would make

the relaxation tighter. This motivates the following open problem: Is it possible to give an efficient algorithm to find any cycle C such that $d(C) > 0$, when one exists? The k-ary cycle inequalities can be understood as one such algorithm for doing this, in some cases. An interesting open question is to characterize precisely how much guidance is given by the k-ary cycle inequalities, particularly since they are known to be strictly weaker than the cycle relaxation (c.f. Section 3.4.1).

Komodakis & Paragios (2008) proposed to tighten the pairwise LP relaxation by a sequence of cycle repairing operations. For binary graphical models, when the dual is at optimality, two cycle repairs – corresponding to the two anchors of any variable (using their terminology) – can be seen to be equivalent to one coordinate descent step on a new cycle inequality for this cycle. One distinction is that the cycle repairs directly modify the edge potentials, having no memory, whereas we use the $\lambda_{C,F,\pi}$ dual variables to keep track of the modifications. Thus, we could later decide to do a coordinate descent step on some $\lambda_{C,F,\pi} > 0$ where we decrease its value, potentially to zero. The analogy for cycle repairs is that the corresponding $\lambda_{C,F,\pi}$ (if they had kept track of it) can only be increased, not decreased. We also solve the open problem of how to *find* the cycles where cycle repairs are necessary. In their experiments, Komodakis & Paragios (2008) explicitly enumerated over short cycles.

Johnson (2008) proposed an algorithm to find *inconsistent* cycles in the dual. His approach applies only to binary-valued graphical models, and only when the dual is close to optimality. In these cases, his algorithm can be shown to find a cycle C such that $d(C, F) > 0$ for some $F \subseteq C$, $|F|$ odd. His algorithm, which inspired our approach, constructs $s_{ij} \in \{+1, -1\}$ and looks for inconsistent cycles using the linear time method described in Section 7.1.2. Because of numerical difficulties, (Johnson, 2008, p.134) needed to use an edge-wise correlation measure, computed using the primal solution obtained from the smoothed dual. By drawing the connection to cycle inequalities, we obtain a *weighted* approach whereas his was *unweighted*. As a result, there are no numerical difficulties, and our algorithm can be applied long before solving the dual to optimality.

It may seem surprising that the dual separation algorithm is so much faster than the primal separation algorithm. However, this is because the dual searches over a smaller class of cycle inequalities. Consider the case where we have solved the dual to optimality for the current relaxation. Then, using the complementary slackness conditions one can show that, for any cycle inequality C, F, π such that $d(C, F, \pi) > 0$, and for any primal solution μ,

$$\sum_{mn \in C \setminus F} \Big(\mu^{\pi}_{mn}(0,1) + \mu^{\pi}_{mn}(1,0) \Big) + \sum_{mn \in F} \Big(\mu^{\pi}_{mn}(0,0) + \mu^{\pi}_{mn}(1,1) \Big) = 0. \tag{7.20}$$

The right hand side could be anything less than 1 for us to obtain a violated cycle inequality, and it is always non-negative because $\mu_{ij}(x_i, x_j) \geq 0$. But, since the right hand side of Eq. 7.20 is 0, we conclude that the dual separation algorithm is only able to find cycle inequalities to add to the relaxation that are *very violated*. By Theorem 7.2.1, the same conclusion holds for binary graphical models for the cluster-pursuit algorithm given in Chapter 5, when applied to a dual optimal solution – if a triplet cluster C is found such that $d(C) > 0$, then there exists a cycle inequality C, F that is very violated by any primal solution. It is also possible to give a $O(|E_\pi|)$ time separation algorithm in the primal that would separate this smaller class of cycle inequalities.

Our approach could be valuable even in graphs that are not sparse. We showed in Chapter 3 that – once the pairwise consistency constraints are satisfied – if a cycle with

a chord is inconsistent, one of its shorter cycles will also be inconsistent. Thus, a greedy strategy that tightens the relaxation one triplet cluster at a time can eventually succeed at making the cycle consistent, and we indeed observed this in our experiments. However, there are three potential wrinkles, all avoided with our new approach: First, the dual bound criterion may be zero for the triplet clusters, but non-zero for the larger cycle (c.f. Section 5.5).[5] Second, the inconsistency may not propagate to the triplet clusters until we are at the *optimum* of the dual LP, so it may not be possible to tighten the relaxation before solving the dual to optimality. Third, if a cycle is of length k and we only add triplet clusters one at a time, it would take $k-2$ iterations before we add all of the triplet clusters that triangulate the cycle and thus make it consistent.

The problem of finding a good cycle cluster to add to the relaxation can be equivalently viewed as that of finding a good treewidth 2 subgraph to use in tightening the relaxation. A natural generalization would be to search for larger treewidth subgraphs to use in tightening. One way to extend our approach to this setting would be to first introduce new higher-order variables into the relaxation that represent the cross-product of the states of the original variables, and to enforce consistency between these higher-order variables. For example, the triplet clusters that we used in Chapter 5 would enable us to search over treewidth 3 subgraphs. These higher-order variables are *non-binary* even if the original graphical model had only binary-valued variables. Thus, given our hardness results for the non-binary setting, the search problem formulated in this way is likely to be intractable. However, we could still use the k-ary cycle inequalities, as discussed in this chapter.

[5]This is a failure of the dual bound criterion; the primal cutting-plane algorithm would not have this difficulty.

Chapter 8

Discussion

There are many exciting directions to pursue using the techniques developed in this thesis. As we mentioned in Section 2.2.2, many of these techniques can be applied to the problem of estimating marginal probabilities and the partition function. Our empirical results demonstrate that the cycle relaxation of the marginal polytope is powerful, allowing us to find the MAP solution for nearly all of the MRFs we considered, and suggest that using them will also lead to significantly more accurate marginals for non-binary MRFs. Our outer bounds are well suited for marginals compared to other approaches. For example, it is difficult to apply Gomory cuts to these problems because the solution to the relaxed problem of the (now non-linear) variational objective will be in the interior of the polytope, and testing whether a point is in the marginal polytope is NP-hard (Sontag, 2007). There are also straightforward ways to generalize the message-passing algorithms from Chapter 4 to inference problems involving marginals (Meltzer *et al.*, 2009).

In many applications, such as protein design, it is important to be able to find the M most likely assignments, not just the MAP assignment ($M = 1$). Fromer & Globerson (2009) show how the LP relaxations described in this thesis can also be applied to the M-best problem. Their key insight is that a vertex v of the marginal polytope can be cut off from the rest of the marginal polytope by a set of *spanning tree inequalities*. For a tree-structured pairwise MRF, exactly one such inequality, together with pairwise consistency constraints, defines the v-excluded marginal polytope. This then gives an exact polyhedral formulation for the $M = 2$ problem. The authors then show how to extend the polyhedral approach to $M > 2$, and successfully applied it to the protein side-chain placement problem, using it together with the algorithm that we gave in Chapter 5. Interestingly, Fromer & Globerson (2009) found that the same triplet clusters that were added by our algorithm in finding the MAP assignment (c.f. Section 5.4) sufficed to make the M-best LP relaxations (obtained by adding the spanning tree inequalities) tight as well.

Many of the prediction problems arising from machine learning applications are *structured* prediction tasks. For example, in protein folding we are given a new protein sequence and the goal is to predict the most likely configuration of the protein's side-chains (Yanover *et al.*, 2008). In parsing for natural language processing (NLP), we are given a new sentence and the goal is to predict the most likely parse tree (Collins, 2002; Rush *et al.*, 2010). These prediction problems are typically solved by formulating an objective function (which is dependent on the input) and solving a combinatorial optimization problem, returning the optimal solution as the prediction. The methods that we developed in this thesis are broadly applicable to many structured prediction problems.

For example, we have recently had success applying a dual decomposition approach similar to that given in Chapter 4 to the NLP problem of non-projective dependency parsing with higher-order constraints (Koo *et al.*, 2010). Our algorithm uses a combinatorial algorithm to solve a first-order parsing subproblem as part of the overall optimization, analogous to how we used dynamic programming to solve spanning tree subproblems in Section 4.3. We found that the LP relaxations were nearly always tight.

These methods are also useful for the problem of *learning* how to do structured prediction. Most algorithms for learning structured prediction models, such as structured perceptron (Collins, 2002), require making a prediction at every iteration. One approach would be to use our algorithms to repeatedly solve LP relaxations of the prediction problems during learning. In Meshi *et al.* (2010), we propose an alternative approach to learning with LP relaxations. The main idea is to instead solve the dual of the structured prediction loss. We formulate the learning task as a convex minimization over both the weights and the dual variables corresponding to each data point. As a result, we can begin to optimize the weights even before completely solving any of the individual prediction problems. We showed that the dual variables can be efficiently optimized using the coordinate descent algorithms described in Chapter 4.

More generally, the polyhedral approach pursued here is widely applicable in machine learning. For example, Kawahara *et al.* (2009) consider a cutting-plane algorithm for the problem of maximizing submodular functions, which is important for feature selection among other applications. Nowozin & Jegelka (2009) consider LP relaxations of clustering and partitioning problems, using constraints related to the ones presented in this thesis and Sontag (2007). In Jaakkola *et al.* (2010), we show how Bayesian network structure learning can be formulated as an integer linear program and solved using LP relaxations. Just as in this thesis, finding the right set of constraints was critical to obtaining tight relaxations.

Our work raises several open questions, many of which are discussed at the end of the relevant chapters. Two of the most interesting questions raised by this thesis are:

- Why are the LP relaxations so often tight for the graphical models that we considered?
- Why are the block-coordinate descent algorithms so effective at solving the LPs?

Although we do not have complete answers to these questions, in what follows we suggest a couple of directions that would be worth exploring.

To address the first question, of why the LP relaxations were tight, we suggest an alternative view of the MAP inference problem as structured prediction with a mixture of experts. Consider the potential function for one edge, $\theta_{ij}(x_i, x_j)$. We can view this potential as a weighted vote for the assignment of variables X_1 and X_2. Each potential function gives a weighted vote over some subset of the variables. The MAP inference problem corresponds to finding the assignment which has the maximum vote according to the mixture of experts. Clearly, if each expert voted in the same direction, then the MAP assignment would be trivial to find. It is easy to see that the LP relaxation is also tight in this scenario. The mixture of experts formulation is most interesting and useful in the setting when some of the experts can be wrong, in which case taking the vote should provide a better prediction than looking at any one expert individually. Suppose that the experts are "mostly right" and are not adversarial. For example, they might vote with high confidence they have the correct assignment, otherwise giving low-weight votes. It may be possible to show that the pairwise LP relaxation is tight in these scenarios.

However, noise can prevent the pairwise LP relaxation from being tight, even in this non-adversarial setting. Consider a binary-valued Markov random field on a length-3 cycle. In the noiseless setting, if all three edge potentials voted for the assignments along the edge to agree, then the pairwise LP relaxation would be tight – the MAP assignment would be $0, 0, 0$ or $1, 1, 1$. However, if noise causes one of the three edges to flip its vote, i.e. to ask that the assignment along that edge disagree, then a fractional vertex is optimal. In this case, enforcing cycle consistency resolves the problem. There must be multiple instances of noise for the cycle consistency relaxation or $\text{TRI}(G)$ to be fractional. It may be possible to show that, under an appropriate noise model, the cycle relaxation is tight with high probability, whereas the pairwise relaxation is not.

A similar question was raised by Goldstein (1994), who observed that a heuristic that he developed, called dead-end elimination, was surprisingly effective at reducing the size of protein design problems. He applied the heuristic to random instances to better understand how the success depended on the structure of the instance. To match the distributions, he constructed the test cases by taking each protein design problem and randomly permuting the values within each edge potential function. The heuristic was significantly less effective on these examples. We expect similar results for the MAP inference problems considered in this thesis. To put this in context, consider an inference problem on a pairwise binary MRF. If the original potential functions had $\theta_{ij}(x_i = x_j) \geq 0$ and $\theta_{ij}(x_i \neq x_j) = 0$, then inference is equivalent to solving minimum cut and the pairwise LP relaxation is tight. However, after permuting the values of θ_{ij} we may have $\theta_{ij}(x_i \neq x_j) > 0$, making inference significantly more difficult.

As we discussed in Chapter 4, one possible reason for the dual block-coordinate descent algorithms being so fast compared to CPLEX, on graphical models where the variables have large state spaces, is because we treat the non-negativity and normalization constraints explicitly. Rather than take the Lagrangian of these constraints to obtain an unconstrained dual, we leave them as constraints and do all block coordinate descent steps with respect to them. This is discussed further in (Yanover *et al.*, 2006, p.17), where they show that the speedup of a similar message-passing algorithm over CPLEX grows larger and larger as the number of states per variable increases.

Finally, another explanation may have to do with the fact that the LP relaxations are typically tight, and that we use a node-factored dual decomposition. To give an extreme example, suppose that our decomposition involved two components and both components were such that, even before optimizing the dual, they both had the same assignment in their argmax. Then, we would obtain the MAP assignment in the first iteration of DD. Zhang *et al.* (2008) showed that protein side-chain placement with a fixed backbone (we also used a fixed backbone in our experiments) can be reasonably predicted at each location, *independently*, using just local backbone information. Although the prediction accuracy is improved by using graphical models, this shows that there is a strong local signal, which may help explain why the LP relaxations are solved quickly and exactly. One could make similar arguments about the stereo vision problem. For example, if there is only one orange pixel in the left image and one in the right image, this is strong local evidence that these two pixels correspond to one another. Thus, the dual decomposition algorithm is likely to reach agreement quickly.

Appendix A

Appendix

A.1 Derivation of Dual of Pairwise LP Relaxation

In this section we show how to derive the dual of the pairwise LP relaxation using the technique of Lagrangian relaxation, or dual decomposition. Consider the primal LP,

$$\max_{\mu \geq 0} \sum_i \sum_{x_i} \mu_i(x_i)\theta_i(x_i) + \sum_{ij} \sum_{x_i, x_j} \mu_{ij}(x_i, x_j)\theta_{ij}(x_i, x_j) \tag{A.1}$$

subject to:

$$\begin{aligned}
\sum_{x_j} \mu_{ij}(x_i, x_j) &= \mu_i(x_i), \ \ \forall ij \in E, x_i \\
\sum_{x_i} \mu_{ij}(x_i, x_j) &= \mu_j(x_j), \ \ \forall ij \in E, x_j \\
\sum_{x_i, x_j} \mu_{ij}(x_i, x_j) &= 1, \ \ \forall ij \in E \qquad \text{(A.2)} \\
\sum_{x_i} \mu_i(x_i) &= 1, \ \ \forall i \in V. \qquad \text{(A.3)}
\end{aligned}$$

We introduce the Lagrange multipliers $\delta_{ji}(x_i)$ and $\delta_{ij}(x_j)$ for the first two constraints. Leaving the last two equality constraints and the non-negativity constraints explicit, we obtain the following equivalent optimization problem:

$$\begin{aligned}
\min_{\delta} \max_{\mu \geq 0} \quad & \sum_i \sum_{x_i} \mu_i(x_i)\theta_i(x_i) + \sum_{ij \in E} \sum_{x_i, x_j} \mu_{ij}(x_i, x_j)\theta_{ij}(x_i, x_j) \\
& + \sum_{ij \in E} \sum_{x_i} \delta_{ji}(x_i)\Big(\mu_i(x_i) - \sum_{x_j} \mu_{ij}(x_i, x_j)\Big) \\
& + \sum_{ij \in E} \sum_{x_j} \delta_{ij}(x_j)\Big(\mu_j(x_j) - \sum_{x_i} \mu_{ij}(x_i, x_j)\Big)
\end{aligned}$$

subject to Eq. A.2 and Eq. A.3. Re-arranging the objective, we get

$$\min_{\delta} \max_{\mu \geq 0} \quad \sum_i \sum_{x_i} \mu_i(x_i) \Big(\theta_i(x_i) + \sum_{j \in N(i)} \delta_{ji}(x_i) \Big) + \sum_{ij \in E} \sum_{x_i, x_j} \mu_{ij}(x_i, x_j) \Big(\theta_{ij}(x_i, x_j) - \delta_{ji}(x_i) - \delta_{ij}(x_j) \Big) \tag{A.4}$$

Finally, we analytically solve the maximization with respect to $\mu \geq 0$ and the normalization constraints from Eq. A.2 and Eq. A.3 to obtain the dual objective:

$$J(\delta) = \sum_i \max_{x_i} \Big(\theta_i(x_i) + \sum_{j \in N(i)} \delta_{ji}(x_i) \Big) + \sum_{ij} \max_{x_i, x_j} \Big(\theta_{ij}(x_i, x_j) - \delta_{ji}(x_i) - \delta_{ij}(x_j) \Big)$$

The dual linear program is then: $\min_\delta J(\delta)$.

A.2 Dual Coordinate Descent with Triplet Clusters

We can analogously derive the dual of the LP relaxation which has both the pairwise consistency constraints and triplet consistency constraints. Our treatment differs from Chapter 5 in that we use the dual formulation of Section A.1 rather than the MPLP formulation. This will result in a simpler message-passing algorithm and bound criterion, replacing the one used originally in Chapter 5. Theorem 7.2.1 refers to the formulation described here.

After adding in triplet clusters c, the dual objective can be shown to be (we use δ in place of λ to be consistent with the new notation):

$$\begin{aligned} J(\delta) \quad = \quad & \sum_{i \in V} \max_{x_i} \Big[\theta_i(x_i) + \sum_{k \in N(i)} \delta_{ki}(x_i) \Big] \\ & + \sum_{ij \in E} \max_{x_i, x_j} \Big[\theta_{ij}(x_i, x_j) - \delta_{ji}(x_i) - \delta_{ij}(x_j) + \sum_{c : ij \in c} \delta_{c \to ij}(x_i, x_j) \Big] \\ & + \sum_c \max_{x_i, x_j, x_k} \Big[- \delta_{c \to ij}(x_i, x_j) - \delta_{c \to jk}(x_j, x_k) - \delta_{c \to ki}(x_k, x_i) \Big]. \end{aligned} \tag{A.5}$$

Unlike the MPLP dual $g(\lambda)$ described in Chapter 5, this formulation is unconstrained. In addition, we no longer have the $\lambda_{e \to e}(x_e)$ dual variables. We give the new message passing algorithm in Figure A-1. The edge to node messages are identical to the MPLP updates that we derived in Section 4.2. There are no longer any edge to edge messages. The new triplet to edge messages correspond to block coordinate descent on the triplet cluster dual variables $\delta_{c \to ij}, \delta_{c \to jk}, \delta_{c \to ki}$ for the objective $J(\delta)$.

Suppose that initially a triplet cluster c is not included in the relaxation. For every $ij \in E$, define:

$$b_{ij}(x_i, x_j) \quad = \quad \theta_{ij}(x_i, x_j) - \delta_{ji}(x_i) - \delta_{ij}(x_j) + \sum_{c' : ij \in c'} \delta_{c' \to ij}(x_i, x_j). \tag{A.6}$$

Then, the amount that the dual objective would decrease following one block coordinate

- **Edge to Node**: For every edge $ij \in E$ and node i (or j) in the edge:

$$\delta_{ij\to i}(x_i) \leftarrow -\frac{1}{2}\Big(\delta_i^{-j}(x_i) + \theta_i(x_i)\Big) + \frac{1}{2}\max_{x_j}\Big[\sum_{c:ij\in c}\delta_{c\to ij}(x_i,x_j) + \delta_j^{-i}(x_j) + \theta_{ij}(x_i,x_j) + \theta_j(x_j)\Big]$$

  where $\delta_i^{-j}(x_i)$ is the sum of edge-to-node messages into i that are not from edge ij, namely: $\delta_i^{-j}(x_i) = \sum_{k\in N(i)\setminus j}\delta_{ik\to i}(x_i)$.

- **Triplet to Edge**: For every triplet $c \in \mathcal{C}$ and every edge $ij \in c$:

$$\delta_{c\to ij}(x_i,x_j) \leftarrow -\frac{2}{3}\Big(\theta_{ij}(x_i,x_j) - \delta_{ji}(x_i) - \delta_{ij}(x_j) + \sum_{\substack{c'\neq c\\ ij\in c'}}\delta_{c'\to ij}(x_i,x_j)\Big) + \frac{1}{3}\max_{x_{c\setminus\{ij\}}}\Big[\sum_{st\in c\setminus\{ij\}}\Big(\theta_{st}(x_s,x_t) - \delta_{ts}(x_s) - \delta_{st}(x_t) + \sum_{\substack{c'\neq c\\ st\in c'}}\delta_{c'\to st}(x_s,x_t)\Big)\Big]$$

Figure A-1: The new coordinate descent updates for an LP relaxation with three node clusters. We could use these instead of the GMPLP updates originally given in Figure 5-1.

descent step on cluster c is

$$d(c) = \sum_{e\in c}\max_{x_e} b_e(x_e) - \max_{x_c}\left[\sum_{e\in c} b_e(x_e)\right] . \tag{A.7}$$

Thus, we can use the same greedy bound criterion described in Chapter 5, but with the beliefs $b_e(x_e)$ re-defined to take into consideration the new dual formulation.

A.3 Derivation of Dual with Cycle Inequalities

In this section we show how to derive the *dual* of the LP relaxation which includes all of the k-ary cycle inequalities. Consider the primal LP,

$$\max_{\mu\geq 0}\ \sum_i\sum_{x_i}\mu_i(x_i)\theta_i(x_i) + \sum_{ij}\sum_{x_i,x_j}\mu_{ij}(x_i,x_j)\theta_{ij}(x_i,x_j) \tag{A.8}$$

subject to:

$$\begin{aligned}
\sum_{x_j} \mu_{ij}(x_i, x_j) &= \mu_i(x_i), \;\; \forall ij \in E, x_i \\
\sum_{x_i} \mu_{ij}(x_i, x_j) &= \mu_j(x_j), \;\; \forall ij \in E, x_j \\
\sum_{ij \in C \setminus F} \sum_{\substack{x_i, x_j : \\ \pi_i(x_i) \neq \pi_j(x_j)}} \mu_{ij}(x_i, x_j) + \sum_{ij \in F} \sum_{\substack{x_i, x_j : \\ \pi_i(x_i) = \pi_j(x_j)}} \mu_{ij}(x_i, x_j) &\geq 1, \;\; \forall \text{cycles } C, F \subseteq C, |F| \text{ odd}, \pi \\
\sum_{x_i, x_j} \mu_{ij}(x_i, x_j) &= 1, \;\; \forall ij \in E \\
\sum_{x_i} \mu_i(x_i) &= 1, \;\; \forall i \in V
\end{aligned}$$

As in Chapter 4 we introduce the Lagrange multipliers $\delta_{ji}(x_i)$ an $\delta_{ij}(x_j)$ for the first two constraints. We also introduce Lagrange multipliers $\lambda_{C,F,\pi}$ for each k-ary cycle inequality. Leaving the last two equality constraints and the non-negativity constraints explicit, we obtain the following equivalent optimization problem:

$$\begin{aligned}
\min_{\lambda \geq 0, \delta} \max_{\mu \geq 0} \quad & \sum_i \sum_{x_i} \mu_i(x_i) \theta_i(x_i) + \sum_{ij \in E} \sum_{x_i, x_j} \mu_{ij}(x_i, x_j) \theta_{ij}(x_i, x_j) \\
& + \sum_{ij \in E} \sum_{x_i} \delta_{ji}(x_i) \Big(\mu_i(x_i) - \sum_{x_j} \mu_{ij}(x_i, x_j) \Big) \\
& + \sum_{ij \in E} \sum_{x_j} \delta_{ij}(x_j) \Big(\mu_j(x_j) - \sum_{x_i} \mu_{ij}(x_i, x_j) \Big) \\
& + \sum_{C, F, \pi} \lambda_{C,F,\pi} \Big(\sum_{ij \in C \setminus F} \sum_{\substack{x_i, x_j : \\ \pi_i(x_i) \neq \pi_j(x_j)}} \mu_{ij}(x_i, x_j) + \sum_{ij \in F} \sum_{\substack{x_i, x_j : \\ \pi_i(x_i) = \pi_j(x_j)}} \mu_{ij}(x_i, x_j) - 1 \Big)
\end{aligned}$$

subject to:

$$\sum_{x_i, x_j} \mu_{ij}(x_i, x_j) = 1, \;\; \forall ij \in E \tag{A.9}$$

$$\sum_{x_i} \mu_i(x_i) = 1, \;\; \forall i \in V. \tag{A.10}$$

Re-arranging the objective, we get

$$\begin{aligned}\min_{\lambda\geq 0,\delta}\max_{\mu\geq 0} \quad & \sum_i \sum_{x_i} \mu_i(x_i)\Big(\theta_i(x_i) + \sum_{j\in N(i)} \delta_{ji}(x_i)\Big) \\ & + \sum_{ij} \sum_{x_i,x_j} \mu_{ij}(x_i,x_j)\Big(\theta_{ij}(x_i,x_j) - \delta_{ji}(x_i) - \delta_{ij}(x_j) \\ & \qquad + \sum_{C,F,\pi\,:\,ij\in C\setminus F} \lambda_{C,F,\pi} 1[\pi_i(x_i)\neq\pi_j(x_j)] \\ & \qquad + \sum_{C,F,\pi\,:\,ij\in F} \lambda_{C,F,\pi} 1[\pi_i(x_i)=\pi_j(x_j)]\Big) \\ & - \sum_{C,F,\pi} \lambda_{C,F,\pi}\ .\end{aligned}$$

Finally, we analytically solve the maximization with respect to $\mu \geq 0$ and the normalization constraints from Eq. A.9 and Eq. A.10 to obtain the dual objective given in Eq. 7.2.

Bibliography

Alizadeh, Farid. 1993. Interior Point Methods in Semidefinite Programming with Applications to Combinatorial Optimization. *SIAM Journal on Optimization*, **5**, 13–51.

Alon, Noga, & Naor, Assaf. 2004. Approximating the cut-norm via Grothendieck's inequality. *Pages 72–80 of: STOC '04: Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*. New York, NY, USA: ACM.

Alon, Noga, Makarychev, Konstantin, Makarychev, Yury, & Naor, Assaf. 2005. Quadratic forms on graphs. *Invent. Math*, **163**, 486–493.

Althaus, E., Kohlbacher, O., Lenhof, H.-P., & Müller, P. 2000. A combinatorial approach to protein docking with flexible side-chains. *Pages 15–24 of: RECOMB '00*.

Arora, Sanjeev, & Kale, Satyen. 2007. A combinatorial, primal-dual approach to semidefinite programs. *Pages 227–236 of: STOC '07: Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*. New York, NY, USA: ACM.

Arora, Sanjeev, Hazan, Elad, & Kale, Satyen. 2005. Fast Algorithms for Approximate Semidefinite Programming using the Multiplicative Weights Update Method. *Pages 339–348 of: FOCS '05: Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*. Washington, DC, USA: IEEE Computer Society.

Arora, Sanjeev, Daskalakis, Constantinos, & Steurer, David. 2009. Message passing algorithms and improved LP decoding. *Pages 3–12 of: STOC '09: Proceedings of the 41st annual ACM symposium on Theory of computing*. New York, NY, USA: ACM.

Awerbuch, Baruch, & Khandekar, Rohit. 2008. Stateless distributed gradient descent for positive linear programs. *Pages 691–700 of: STOC '08: Proceedings of the 40th annual ACM symposium on Theory of computing*. New York, NY, USA: ACM.

Bansal, Nikhil, Blum, Avrim, & Chawla, Shuchi. 2002. Correlation Clustering. *Page 238 of: FOCS '02: Proceedings of the 43rd Symposium on Foundations of Computer Science*. Washington, DC, USA: IEEE Computer Society.

Barahona, F. 1993. On cuts and matchings in planar graphs. *Mathematical Programming*, **60**, 53–68.

Barahona, F., & Mahjoub, A. R. 1986. On the cut polytope. *Mathematical Programming*, **36**, 157–173.

Barahona, Francisco, & Anbil, Ranga. 2000. The volume algorithm: producing primal solutions with a subgradient method. *Mathematical Programming*, **87**, 385–399.

Batra, D., Gallagher, A., Parikh, D., & Chen, T. 2010. MRF Inference via Outer-Planar Decomposition. *In: IEEE Computer Vision and Pattern Recognition.*

Bayati, M., Shah, D., & Sharma, M. 2008. Max-Product for Maximum Weight Matching: Convergence, Correctness, and LP Duality. *IEEE Transactions on Information Theory*, **54**(3), 1241–1251.

Bertsimas, D., & Tsitsiklis, J. N. 1997. *Introduction to Linear Optimization.* Athena Scientific.

Boros, Endre, & Hammer, Peter L. 2002. Pseudo-boolean optimization. *Discrete Appl. Math.*, **123**(1-3), 155–225.

Boykov, Yuri, & Kolmogorov, Vladimir. 2004. An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision. *IEEE Trans. Pattern Anal. Mach. Intell.*, **26**(9), 1124–1137.

Boykov, Yuri, Veksler, Olga, & Zabih, Ramin. 2001. Fast Approximate Energy Minimization via Graph Cuts. *IEEE Trans. Pattern Anal. Mach. Intell.*, **23**(11), 1222–1239.

Charikar, Moses, & Wirth, Anthony. 2004. Maximizing Quadratic Programs: Extending Grothendieck's Inequality. *Foundations of Computer Science, Annual IEEE Symposium on*, **0**, 54–60.

Charikar, Moses, Makarychev, Konstantin, & Makarychev, Yury. 2009. Integrality gaps for Sherali-Adams relaxations. *Pages 283–292 of: STOC '09: Proceedings of the 41st annual ACM symposium on Theory of computing.* New York, NY, USA: ACM.

Chekuri, C., Khanna, S., Naor, J., & Zosin, L. 2005. A Linear Programming Formulation and Approximation Algorithms for the Metric Labeling Problem. *SIAM J. Discret. Math.*, **18**(3), 608–625.

Collins, M. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. *In: EMNLP.*

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. 2001. *Introduction to Algorithms.* 2nd edn. MIT Press.

Dahlhaus, E., Johnson, D. S., Papadimitriou, C. H., Seymour, P. D., & Yannakakis, M. 1994. The Complexity of Multiterminal Cuts. *SIAM J. Computing*, **23**(4), 864–894.

de la Vega, Wenceslas Fernandez, & Kenyon-Mathieu, Claire. 2007. Linear programming relaxations of maxcut. *Pages 53–61 of: SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms.* Philadelphia, PA, USA: Society for Industrial and Applied Mathematics.

Dechter, R., Kask, K., & Mateescu, R. 2002. Iterative Join-Graph Propagation. *In: UAI.*

Deza, M. M., & Laurent, M. 1997. *Geometry of Cuts and Metrics.* Algorithms and Combinatorics, vol. 15. Springer.

Dimakis, Alexandros G., Gohari, Amin A., & Wainwright, Martin J. 2009. Guessing facets: polytope structure and improved LP decoder. *IEEE Trans. Inf. Theor.*, **55**(8), 3479–3487.

Duchi, J., Tarlow, D., Elidan, G., & Koller, D. 2007. Using Combinatorial Optimization within Max-Product Belief Propagation. *In:* Schölkopf, B., Platt, J., & Hoffman, T. (eds), *Advances in Neural Information Processing Systems 19*. MIT Press.

Elidan, G., Mcgraw, I., & Koller, D. 2006. Residual belief propagation: informed scheduling for asynchronous message passing. *In: UAI*.

Feldman, J., Wainwright, M.J., & Karger, D.R. 2005. Using linear programming to Decode Binary linear codes. *IEEE Transactions on Information Theory*, **51**(3), 954 – 972.

Felzenszwalb, P. F., & Huttenlocher, D. P. 2006. Efficient Belief Propagation for Early Vision. *Int. J. Comput. Vision*, **70**(1), 41–54.

Felzenszwalb, Pedro F., Pap, Gyula, va Tardos, & Zabih, Ramin. 2010. Globally Optimal Pixel Labeling Algorithms for Tree Metrics. *In: Computer Vision and Pattern Recognition*.

Frangioni, A., Lodi, A., & Rinaldi, G. 2005. New approaches for optimizing over the semimetric polytope. *Math. Program.*, **104**(2), 375–388.

Fromer, Menachem, & Globerson, Amir. 2009. An LP View of the M-best MAP problem. *Pages 567–575 of:* Bengio, Y., Schuurmans, D., Lafferty, J., Williams, C. K. I., & Culotta, A. (eds), *Advances in Neural Information Processing Systems 22*.

Globerson, A., & Jaakkola, T. 2007a. Approximate Inference Using Planar Graph Decomposition. *In: Advances in Neural Information Processing Systems 20*.

Globerson, A., & Jaakkola, T. 2007b. Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations. *In:* Platt, J.C., Koller, D., Singer, Y., & Roweis, S. (eds), *Advances in Neural Information Processing Systems 21*. Cambridge, MA: MIT Press.

Globerson, A., & Jaakkola, T. 2008. Fixing Max-Product: Convergent Message Passing Algorithms for MAP LP-Relaxations. *In: NIPS 21*.

Goemans, Michel X., & Williamson, David P. 1995. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, **42**(6), 1115–1145.

Goldstein, R.F. 1994. Efficient rotamer elimination applied to protein side-chains and related spin glasses. *Biophysical Journal*, **66**(5), 1335 – 1340.

Gomory, R.E. 1958. Outline of an algorithm for integer solutions to linear programs. vol. 64.

Greig, D.M., Porteous, B.T., & Seheult, A.H. 1989. Exact Maximum a Posteriori Estimation for Binary Images. *J. Royal Statistical Soc. B*, **51**(2), 271–279.

Gupta, Rahul, Diwan, Ajit A., & Sarawagi, Sunita. 2007. Efficient inference with cardinality-based clique potentials. *Pages 329–336 of: ICML '07: Proceedings of the 24th international conference on Machine learning*. New York, NY, USA: ACM.

Hong, E.J., & Lozano-Pérez, T. 2006. Protein Side-Chain Placement Through MAP Estimation and Problem-Size Reduction. *In: WABI*.

Ishikawa, Hiroshi. 2003. Exact Optimization for Markov Random Fields with Convex Priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **25**, 1333–1336.

Jaakkola, Tommi, Sontag, David, Globerson, Amir, & Meila, Marina. 2010. Learning Bayesian Network Structure using LP Relaxations. *Pages 358–365 of: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AI-STATS)*, vol. 9. JMLR: W&CP.

Jerrum, Mark, & Sinclair, Alistair. 1993. Polynomial-time approximation algorithms for the Ising model. *SIAM J. Comput.*, **22**(5), 1087–1116.

Johnson, J. 2008. *Convex Relaxation Methods for Graphical Models: Lagrangian and Maximum Entropy Approaches.* Ph.D. thesis, EECS, MIT.

Jojic, V., Gould, S., & Koller, D. 2010. Fast and smooth: Accelerated dual decomposition for MAP inference. *In: Proceedings of International Conference on Machine Learning (ICML).*

Karger, David R. 2000. Minimum cuts in near-linear time. *J. ACM*, **47**(1), 46–76.

Kawahara, Yoshinobu, Nagano, Kiyohito, Tsuda, Koji, & Bilmes, Jeff. 2009. Submodularity Cuts and Applications. *Pages 916–924 of:* Bengio, Y., Schuurmans, D., Lafferty, J., Williams, C. K. I., & Culotta, A. (eds), *Advances in Neural Information Processing Systems 22.*

Khot, Subhash, & Saket, Rishi. 2009. SDP Integrality Gaps with Local ell_1-Embeddability. *Foundations of Computer Science, Annual IEEE Symposium on*, **0**, 565–574.

Khot, Subhash, Kindler, Guy, Mossel, Elchanan, & O'Donnell, Ryan. 2004. Optimal Inapproximability Results for Max-Cut and Other 2-Variable CSPs? *Pages 146–154 of: FOCS '04: Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science.* Washington, DC, USA: IEEE Computer Society.

Kingsford, C. L., Chazelle, B., & Singh, M. 2005. Solving and analyzing side-chain positioning problems using linear and integer programming. *Bioinformatics*, **21**(7), 1028–1039.

Kleinberg, Jon, & Tardos, Eva. 1999. Approximation Algorithms for Classification Problems with Pairwise Relationships: Metric Labeling and Markov Random Fields. *Foundations of Computer Science, Annual IEEE Symposium on*, **0**, 14.

Kohli, Pushmeet, Shekhovtsov, Alexander, Rother, Carsten, Kolmogorov, Vladimir, & Torr, Philip. 2008. On partial optimality in multi-label MRFs. *Pages 480–487 of: ICML '08: Proceedings of the 25th international conference on Machine learning.* New York, NY, USA: ACM.

Kolmogorov, V. 2006. Convergent Tree-Reweighted Message Passing for Energy Minimization. *IEEE Trans. Pattern Anal. Mach. Intell.*, **28**(10), 1568–1583.

Kolmogorov, V., & Wainwright, M. 2005. On the optimality of tree-reweighted max-product message-passing. *In: UAI.*

Komodakis, N., & Paragios, N. 2008. Beyond Loose LP-Relaxations: Optimizing MRFs by Repairing Cycles. *Pages 806–820 of: ECCV.*

Komodakis, Nikos, Paragios, Nikos, & Tziritas, Georgios. 2010. MRF Energy Minimization and Beyond via Dual Decomposition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **99**(PrePrints).

Koo, Terry, Rush, Alexander M., Collins, Michael, Jaakkola, Tommi, & Sontag, David. 2010. Dual Decomposition for Parsing with Non-Projective Head Automata. *In: EMNLP*.

Koster, A., van Hoesel, S.P.M., & Kolen, A.W.J. 1998. The partial constraint satisfaction problem: Facets and lifting theorems. *Operations Research Letters*, **23**, 89–97.

Lacoste-Julien, S., Taskar, B., Klein, D., & Jordan, M. I. 2006. Word alignment via quadratic assignment. *Pages 112–119 of: Proceedings of NAACL HLT*.

Laurent, Monique. 2003. A Comparison of the Sherali-Adams, Lovász-Schrijver, and Lasserre Relaxations for 0–1 Programming. *Math. Oper. Res.*, **28**(3), 470–496.

Lazic, Nevena, Frey, Brendan, & Aarabi, Parham. 2010. Solving the Uncapacitated Facility Location Problem Using Message Passing Algorithms. *Pages 429–436 of: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, vol. 9. JMLR: W&CP.

Liers, Frauke, Jünger, Michael, Reinelt, Gerhard, & Rinaldi, Giovanni. 2004. *Computing Exact Ground States of Hard Ising Spin Glass Problems by Branch-and-Cut*. Wiley. Pages 47–68.

McAuley, J. J., & Caetano, T. S. 2010. Exploiting data-independence for fast belief-propagation. *In: ICML*.

Mceliece, Robert J., Mackay, David J. C., & fu Cheng, Jung. 1998. Turbo decoding as an instance of Pearls "Belief Propagation" algorithm. *IEEE Journal on Selected Areas in Communications*, **16**, 140–152.

Meltzer, T., Yanover, C., & Weiss, Y. 2005. Globally optimal solutions for energy minimization in stereo vision using reweighted belief propagation. *In: ICCV*.

Meltzer, Talya, Globerson, Amir, & Weiss, Yair. 2009. Convergent message passing algorithms - a unifying view. *In: In Proc. Twenty-eighth Conference on Uncertainty in Artificial Intelligence (UAI 09*.

Meshi, Ofer, Sontag, David, Jaakkola, Tommi, & Globerson, Amir. 2010. Learning Efficiently with Approximate Inference via Dual Losses. *In: Proceedings of the International Conference on Machine Learning (ICML)*.

Mitchell, John E. 2005. Cutting Plane Methods and Subgradient Methods. *In: Tutorials in Operations Research. INFORMS*.

Murphy, Kevin, Weiss, Yair, & Jordan, Michael. 1999. Loopy Belief Propagation for Approximate Inference: An Empirical Study. *Pages 467–47 of: Proceedings of the Proceedings of the Fifteenth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-99)*. San Francisco, CA: Morgan Kaufmann.

Nedic, A., & Ozdaglar, A. 2007. Approximate primal solutions and rate analysis for dual subgradient methods. *Page 61853 of: Urbana*, vol. 51.

Nemhauser, G. L., & Trotter, L. E. 1975. Vertex packings: Structural properties and algorithms. *Mathematical Programming*, **8**, 232–248.

Nowozin, Sebastian, & Jegelka, Stefanie. 2009. Solution stability in linear programming relaxations: graph partitioning and unsupervised learning. *Pages 769–776 of: ICML '09: Proceedings of the 26th Annual International Conference on Machine Learning*. New York, NY, USA: ACM.

Pierce, Niles A., Spriet, Jan A., Desmet, J., & Mayo, Stephen L. 2000. Conformational splitting: A more powerful criterion for dead-end elimination. *Journal of Computational Chemistry*, **21**(11), 999–1009.

Plotkin, Serge A., Shmoys, David B., & Tardos, Éva. 1995. Fast approximation algorithms for fractional packing and covering problems. *Math. Oper. Res.*, **20**(2), 257–301.

Raphael, C. 2001. Coarse-to-Fine Dynamic Programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **23**(12), 1379–1390.

Ravikumar, Pradeep, Agarwal, Alekh, & Wainwright, Martin J. 2008. Message-passing for graph-structured linear programs: proximal projections, convergence and rounding schemes. *Pages 800–807 of: ICML '08: Proceedings of the 25th international conference on Machine learning*. New York, NY, USA: ACM.

Rendl, Franz, Rinaldi, Giovanni, & Wiegele, Angelika. 2009. Solving Max-Cut to optimality by intersecting semidefinite and polyhedral relaxations. *Math. Program.*, **121**(2), 307–335.

Rush, Alexander M., Sontag, David, Collins, Michael, & Jaakkola, Tommi. 2010. On Dual Decomposition and Linear Programming Relaxations for Natural Language Processing. *In: EMNLP*.

Scharstein, D., & Szeliski, R. 2002. A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms. *IJCV*.

Schraudolph, Nicol N. 2010. Polynomial-Time Exact Inference in NP-Hard Binary MRFs via Reweighted Perfect Matching. *Pages 717–724 of:* Teh, Yee Whye, & Titterington, Mike (eds), *13th Intl. Conf. Artificial Intelligence and Statistics (AIstats)*, vol. 9. Chia Laguna, Italy: JMLR: W&CP.

Schraudolph, Nicol N., & Kamenetsky, Dmitry. 2009. Efficient Exact Inference in Planar Ising Models. *In: Advances in Neural Information Processing Systems*, vol. 21. Cambridge, MA: MIT Press.

Sherali, H. D., & Adams, W. P. 1990. A Hierarchy of Relaxations between the Continuous and Convex Hull Representations for Zero-One Programming Problems. *SIAM Journal on Discrete Mathematics*, **3**(3), 411–430.

Shimony, Y. 1994. Finding the MAPs for belief networks is NP-hard. *Aritifical Intelligence*, **68(2)**, 399–410.

Sontag, David. 2007. *Cutting Plane Algorithms for Variational Inference in Graphical Models*. M.Phil. thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science.

Sontag, David, & Jaakkola, Tommi. 2008. New Outer Bounds on the Marginal Polytope. *Pages 1393–1400 of:* Platt, J.C., Koller, D., Singer, Y., & Roweis, S. (eds), *Advances in Neural Information Processing Systems 20.* Cambridge, MA: MIT Press.

Sontag, David, & Jaakkola, Tommi. 2009. Tree Block Coordinate Descent for MAP in Graphical Models. *Pages 544–551 of: Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics (AI-STATS)*, vol. 8. JMLR: W&CP.

Sontag, David, Meltzer, Talya, Globerson, Amir, Weiss, Yair, & Jaakkola, Tommi. 2008. Tightening LP Relaxations for MAP using Message-Passing. *Pages 503–510 of: 24th Conference in Uncertainty in Artificial Intelligence.* AUAI Press.

Sontag, David, Globerson, Amir, & Jaakkola, Tommi. 2009. Clusters and Coarse Partitions in LP Relaxations. *Pages 1537–1544 of:* Koller, D., Schuurmans, D., Bengio, Y., & Bottou, L. (eds), *Advances in Neural Information Processing Systems 21.* MIT Press.

Tappen, Marshall F., & Freeman, William T. 2003. Comparison of Graph Cuts with Belief Propagation for Stereo, using Identical MRF Parameters. *Page 900 of: ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision.* Washington, DC, USA: IEEE Computer Society.

Tarlow, Daniel, Givoni, Inmar, & Zemel, Richard. 2010. HOP-MAP: Efficient Message Passing with High Order Potentials. *In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics.*

Trevisan, Luca. 2009. Max cut and the smallest eigenvalue. *Pages 263–272 of: STOC '09: Proceedings of the 41st annual ACM symposium on Theory of computing.* New York, NY, USA: ACM.

Trevisan, Luca, Sorkin, Gregory B., Sudan, Madhu, & Williamson, David P. 2000. Gadgets, Approximation, and Linear Programming. *SIAM J. Comput.*, **29**(6), 2074–2097.

Vanderbei, R.J. 2007. *Linear Programming: Foundations and Extensions.* 3rd edn. Springer.

Vontobel, Pascal O., & Koetter, Ralf. 2006. Towards Low-Complexity Linear-Programming Decoding. *In: Proc. 4th Intern. Conf. on Turbo Codes and Related Topics.*

Wainwright, M., & Jordan, M. I. 2006. Log-determinant relaxation for approximate inference in discrete Markov random fields. *IEEE Transactions on Signal Processing*, **54**(6), 2099–2109.

Wainwright, M., & Jordan, M. I. 2008. *Graphical Models, Exponential Families, and Variational Inference.* Hanover, MA, USA: Now Publishers Inc.

Wainwright, M., Jaakkola, T., & Willsky, A. 2005a. MAP estimation via agreement on trees: message-passing and linear programming. *IEEE Trans. on Information Theory*, **51**(11), 3697–3717.

Wainwright, M., Jaakkola, T., & Willsky, A. 2005b. A new class of upper bounds on the log partition function. *IEEE Transactions on Information Theory*, **51**, 2313–2335.

Wainwright, Martin J., & Jordan, Michael I. 2004. *Treewidth-based conditions for exactness of the Sherali-Adams and Lasserre relaxations.* Technical Report 671. UC Berkeley, Dept. of Statistics.

Weiss, Y., Yanover, C., & Meltzer, T. 2007. MAP Estimation, Linear Programming and Belief Propagation with Convex Free Energies. *In: UAI.*

Weiss, Yair. 1997. *Belief Propagation and Revision in Networks with Loops.* Tech. rept. Cambridge, MA, USA.

Welling, M. 2004. On the Choice of Regions for Generalized Belief Propagation. *In: UAI.*

Welling, M., Minka, T., & Teh, Y. W. 2005. Structured Region Graphs: Morphing EP into GBP. *In: UAI.*

Werner, T. 2007. A Linear Programming Approach to Max-Sum Problem: A Review. *IEEE Trans. Pattern Anal. Mach. Intell.*, **29**(7), 1165–1179.

Werner, T. 2008. High-arity interactions, polyhedral relaxations, and cutting plane algorithm for soft constraint optimisation (MAP-MRF). *In: CVPR.*

Yanover, C., Meltzer, T., & Weiss, Y. 2006. Linear Programming Relaxations and Belief Propagation – An Empirical Study. *JMLR*, **7**, 1887–1907.

Yanover, C., Schueler-Furman, O., & Weiss, Y. 2008. Minimizing and Learning Energy Functions for Side-Chain Prediction. *Journal of Computational Biology*, **15**(7), 899–911.

Yarkony, Julian, Fowlkes, Charless, & Ihler, Alexander. 2010. Covering Trees and Lower-bounds on Quadratic Assignment. *In: Computer Vision and Pattern Recognition.*

Yedidia, J., Freeman, W., & Weiss, Y. 2001. *Bethe Free Energy, Kikuchi Approximations, and Belief Propagation Algorithms.* Technical Report 16. Mitsubishi Electric Research Lab.

Yedidia, J.S., Freeman, W.T., & Weiss, Y. 2005. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Trans. on Information Theory*, **51**(7), 2282– 2312.

Zhang, Jing, Gao, Xin, Xu, Jinbo, & Li, Ming. 2008. Rapid and Accurate Protein Side Chain Packing using Local Backbone Information. *In: RECOMB 2008.*