8 Introduction to Dual Decomposition for Inference

David Sontag dsontag@csail.mit.edu
Microsoft Research New England
Cambridge, MA

Amir Globerson gamir@cs.huji.ac.il
Hebrew University
Jerusalem, Israel

Tommi Jaakkola tommi@csail.mit.edu
CSAIL, MIT
Cambridge, MA

Many inference problems with discrete variables result in a difficult combinatorial optimization problem. In recent years, the technique of dual decomposition, also called Lagrangian relaxation, has proved to be a powerful means of solving these inference problems by decomposing them into simpler components that are repeatedly solved independently and combined into a global solution. In this chapter, we introduce the general technique of dual decomposition through its application to the problem of finding the most likely (MAP) assignment in graphical models. We discuss both subgradient and block coordinate descent approaches to solving the dual problem. The resulting message-passing algorithms are similar to max-product, but can be shown to solve a linear programming relaxation of the MAP problem. We show how many of the MAP algorithms are related to each other, and also quantify when the MAP solution can and cannot be decoded directly from the dual solution.

8.1 Introduction

Many problems in engineering and the sciences require solutions to challenging combinatorial optimization problems. These include traditional problems such as scheduling, planning, fault diagnosis, or searching for molecular conformations. In addition, a wealth of combinatorial problems arise directly from probabilistic modeling (graphical models). Graphical models (Koller and Friedman, 2009) have been widely adopted in areas such as computational biology, machine vision, and natural language processing, and are increasingly being used as frameworks expressing combinatorial problems.

Consider, for example, a protein side-chain placement problem where the goal is to find the minimum energy conformation of amino acid side-chains along a fixed carbon backbone. The orientations of the side-chains are represented by discretized angles called rotamers. The combinatorial difficulty arises here from the fact that rotamer choices for nearby amino acids are energetically coupled. For globular proteins, for example, such couplings may be present for most pairs of side-chain orientations. This problem is couched in probabilistic modeling terms by associating molecular conformations with the setting of discrete random variables corresponding to the rotamer angles. The interactions between such random variables come from the energetic couplings between nearby amino acids. Finding the minimum energy conformation is then equivalently solved by finding the most probable assignment of states to the variables.

We will consider combinatorial problems that are expressed in terms of structured probability models (graphical models). A graphical model is defined over a set of discrete variables $\boldsymbol{x} = \{x_j\}_{j \in V}$. Local interactions between the variables are modeled by functions $\theta_f(\boldsymbol{x}_f)$ which depend only on a subset of variables $\boldsymbol{x}_f = \{x_j\}_{j \in f}$. For example, if $\boldsymbol{x}_f = (x_i, x_j)$, $\theta_f(\boldsymbol{x}_f) = \theta_{ij}(x_i, x_j)$ may represent the coupling between rotamer angles x_i and x_j corresponding to nearby amino acids. The functions $\theta_f(\boldsymbol{x}_f)$ are often represented as small tables of numbers with an adjustable value for each local assignment $\boldsymbol{x}_f$. The joint probability distribution over all the variables $\boldsymbol{x}$ is then defined by combining all the local interactions,

$$\log P(\boldsymbol{x}) = \sum_{i \in V} \theta_i(x_i) + \sum_{f \in F} \theta_f(\boldsymbol{x}_f) + \text{const},$$

where we have included (singleton) functions biasing the states of individual variables. The local interactions provide a compact parametric description of the joint distribution, since we only need to specify the local functions (as small tables) rather than the full probability table involving one value for each complete assignment $\boldsymbol{x}$.

Despite their compact description, graphical models are capable of describing complex dependencies among the variables. These dependences arise from the combined effect of all the local interactions. The problem of probabilistic inference is to reason about the underlying state of the variables in the model. Given the complex dependencies, it is not surprising that probabilistic inference is often computationally intractable. For example, finding the maximum probability assignment of a graphical model (also known as the MAP assignment) is NP-hard. Finding the MAP assignment remains hard even if the local functions depend on only two variables, as in the protein side-chain example. This has prompted extensive research into approximation algorithms, some of which often work well in practice.

One of the most successful approximation schemes has been to use relaxations of the original MAP problem. Relaxation methods take the original combinatorial problem and pose it as a constrained optimization problem. They then relax some of the constraints in an attempt to factor the problem into more independent subproblems, resulting in a tractable approximation of the original one. Two closely related relaxation schemes are dual decomposition (Johnson, 2008; Komodakis et al., 2011) and linear programming (LP) relaxations (Schlesinger, 1976; Wainwright et al., 2005). Although the approaches use different derivations of the approximation, they result in equivalent optimization problems.

Practical uses of MAP relaxations involve models with thousands of variables and constraints. While the relaxed optimization problems can generally be solved in polynomial time using a variety of methods and off-the-shelf solvers, most of these do not scale well to the problem sizes encountered in practice (e.g., see Yanover et al., 2006, for an evaluation of commercial LP solvers on inference problems). However, often the LP relaxations arising from graphical models have significant structure that can be exploited to design algorithms that scale well with the problem size.

This chapter introduces the dual decomposition approach, also known as Lagrangian relaxation, and focuses on efficient scalable algorithms for solving the relaxed problem. With dual decomposition, the original problem is broken up into smaller subproblems with the help of Lagrange multipliers. The smaller subproblems can be then be solved exactly by using combinatorial algorithms. The decomposition is subsequently optimized with respect to the Lagrange multipliers so as to encourage the subproblems to agree about the variables they share. For example, we could decompose the MAP problem into subproblems corresponding separately to each local function $\theta_f(\boldsymbol{x}_f)$. The Lagrange multipliers introduced in the decomposition would then modify the functions $\theta_f(\boldsymbol{x}_f)$ so that the local maximizing assignments agree across the subproblems. The decomposition may also involve larger

components that nevertheless can be solved efficiently, such as a set of spanning trees that together cover all edges of a pairwise graphical model.

The chapter is organized as follows. We begin in section 8.2 with two example applications that illustrate the types of problems that can be solved using our techniques. We next define the MAP problem formally and introduce the dual decomposition approach in section 8.3. The next two sections, 8.4 and 8.5, describe algorithmic approaches to solving the dual decomposition optimization problem. In section 8.6 we discuss the relation between dual decomposition and LP relaxations of the MAP problem, showing that they are essentially equivalent. Finally, in section 8.7 we discuss how the MAP solution can be approximated from the dual decomposition solutions, and what formal guarantees we can make about it.

8.2 Motivating Applications

We will use two example applications to illustrate the role of local interactions, types of decompositions, and how they can be optimized. The first example, outlined earlier, is the protein side-chain placement problem (Yanover et al., 2008). The goal is to find the minimum energy conformation of amino acid side-chains in a fixed protein backbone structure. Each side-chain orientation is represented by a discrete variable x_i specifying the corresponding rotamer angles. Depending on the discretization and the side-chain (the number of dihedral angles), the variables may take on tens or hundreds of possible values (states). Energy functions for this problem typically separate into individual and pairwise terms; the pairwise terms take into consideration attractive and repulsive forces between side-chains that are near each other in the 3D structure.

The problem is equivalently represented as an inference problem in a pairwise Markov random field (MRF) model. Graphically, the MRF is an undirected graph with nodes $i \in V$ corresponding to variables (side-chain orientations x_i) and edges $ij \in E$ indicating interactions. Each pairwise energy term implies an edge $ij \in E$ in the model and defines a potential function $\theta_{ij}(x_i, x_j)$ over local assignments (x_i, x_j). Single node potential functions $\theta_i(x_i)$ can be included separately or absorbed by the edge potentials. The MAP inference problem is then to find the assignment $\boldsymbol{x} = \{x_i\}_{i \in V}$ that maximizes

$$\sum_{i \in V} \theta_i(x_i) + \sum_{ij \in E} \theta_{ij}(x_i, x_j).$$

Without additional restrictions on the choice of potential functions, or which

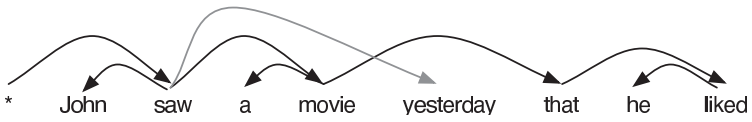

Figure 8.1: Example of dependency parsing for a sentence in English. Every word has one parent (i.e., a valid dependency parse is a directed tree). The red arc demonstrates a non-projective dependency.

edges to include, the problem is known to be NP-hard. Using the dual decomposition approach, we will break the problem into much simpler subproblems involving maximizations of each single node potential $\theta_i(x_i)$ and each edge potential $\theta_{ij}(x_i, x_j)$ independently from the other terms. Although these local maximizing assignments are easy to obtain, they are unlikely to agree with each other without our modifying the potential functions. These modifications are provided by the Lagrange multipliers associated with agreement constraints.

Our second example is dependency parsing, a key problem in natural language processing (McDonald et al., 2005). Given a sentence, we wish to predict the dependency tree that relates the words in the sentence. A dependency tree is a directed tree over the words in the sentence where an arc is drawn from the head word of each phrase to words that modify it. For example, in the sentence shown in Fig. 8.1, the head word of the phrase "John saw a movie" is the verb "saw", and its modifiers are the subject "John" and the object "movie". Moreover, the second phrase "that he liked" modifies "movie". In many languages the dependency tree is *non-projective* in the sense that each word and its descendants in the tree do not necessarily form a contiguous subsequence.

Formally, given a sentence with m words, we have $m(m-1)$ binary arc selection variables $x_{ij} \in \{0, 1\}$. Since the selections must form a directed tree, the binary variables are governed by an overall function $\theta_T(\boldsymbol{x})$ with the idea that $\theta_T(\boldsymbol{x}) = -\infty$ is used to rule out any non-trees. The selections are further biased by weights on individual arcs, through $\theta_{ij}(x_{ij})$, which depend on the given sentence. In a simple arc-factored model, the predicted dependency structure is obtained by maximizing (McDonald et al., 2005)

$$\theta_T(\boldsymbol{x}) + \sum_{ij} \theta_{ij}(x_{ij}),$$

and can be found with directed maximum-weight spanning-tree algorithms.

More realistic dependency parsing models include additional higher-order interactions between the arc selections. For example, we may couple the modifier selections $\boldsymbol{x}_{|i} = \{x_{ij}\}_{j \neq i}$ (all outgoing edges) for a given word

i, expressed by a function $\theta_{i|}(\boldsymbol{x}_{|i})$. Finding the maximizing non-projective parse tree in a model that includes such higher-order couplings, without additional restrictions, is known to be NP-hard (McDonald and Satta, 2007). We consider here models where $\theta_{i|}(\boldsymbol{x}_{|i})$ can be individually maximized by dynamic programming algorithms (e.g., head-automata models), but become challenging as part of the overall dependency tree model:

$$\left(\theta_T(\boldsymbol{x})\right) + \left(\sum_{ij} \theta_{ij}(x_{ij}) + \sum_i \theta_{i|}(\boldsymbol{x}_{|i})\right) = \theta_1(\boldsymbol{x}) + \theta_2(\boldsymbol{x}).$$

The first component $\theta_1(\boldsymbol{x})$ ensures that we obtain a tree, and the second component $\theta_2(\boldsymbol{x})$ incorporates higher-order biases on the modifier selections. A natural dual decomposition in this case will be to break the problem into these two manageable components, which are then forced to agree on the arc selection variables (Koo et al., 2010).

8.3 Dual Decomposition and Lagrangian Relaxation

The previous section described several problems where we wish to maximize a sum over factors, each defined on some subset of the variables. Here we describe this problem in its general form, and introduce a relaxation approach for approximately maximizing it.

Consider a set of n discrete variables $x_1, \ldots, x_n$, and a set F of subsets on these variables (i.e., $f \in F$ is a subset of $V = \{1, \ldots, n\}$), where each subset corresponds to the domain of one of the factors. Also, assume we are given functions $\theta_f(\boldsymbol{x}_f)$ on these factors, as well as functions $\theta_i(x_i)$ on each of the individual variables.[1] The goal of the MAP problem is to find an assignment $\boldsymbol{x} = (x_1, \ldots, x_n)$ for all the variables which maximizes the sum of the factors:

$$\text{MAP}(\boldsymbol{\theta}) = \max_{\boldsymbol{x}} \sum_{i \in V} \theta_i(x_i) + \sum_{f \in F} \theta_f(\boldsymbol{x}_f). \tag{8.1}$$

The maximizing value is denoted by $\text{MAP}(\boldsymbol{\theta})$, and the maximizing assignment is called the MAP assignment. It can be seen that the examples in section 8.2 indeed correspond to this formulation.[2]

1. The singleton factors are not needed for generality, but we keep them for notational convenience and because one often has such factors.

2. Recall the protein side-chain placement problem, where we have an energy function defined on a pairwise MRF. Here, the set of factors F would simply be the edge potentials $\theta_{ij}(x_i, x_j)$ characterizing the couplings between orientations of nearby side-chains. In contrast, in non-projective dependency parsing we have just two factors, $\theta_1(\boldsymbol{x})$ and $\theta_2(\boldsymbol{x})$,

As mentioned earlier, the problem in Eq. 8.1 is generally intractable, and we thus need to resort to approximations. The key difficulty in maximizing Eq. 8.1 is that we need to find an $\boldsymbol{x}$ that maximizes the sum over factors rather than each factor individually. The key assumption we shall make in our approximations is that maximizing each of the factors $\theta_f(\boldsymbol{x}_f)$ can be done efficiently, so that the only complication is their joint maximization.

Our approximation will proceed as follows: we will construct a *dual* function $L(\boldsymbol{\delta})$ with variables $\boldsymbol{\delta}$ such that for all values of $\boldsymbol{\delta}$ it holds that $L(\boldsymbol{\delta}) \geq \text{MAP}(\boldsymbol{\theta})$. In other words, $L(\boldsymbol{\delta})$ will be an upper bound on the value of the MAP assignment. We will then seek a $\boldsymbol{\delta}$ that minimizes $L(\boldsymbol{\delta})$ so as to make this upper bound as tight as possible.

To describe the dual optimization problem, we first specify what the dual variables are. For every choice of $f \in F$, $i \in f$, and x_i, we will have a dual variable denoted by $\delta_{fi}(x_i)$.[3] This variable may be interpreted as the *message* that factor f sends to variable i about its state x_i. The dual function $L(\boldsymbol{\delta})$ and the corresponding optimization problem are then given by

$$\min_{\boldsymbol{\delta}} L(\boldsymbol{\delta}), \tag{8.2}$$

$$L(\boldsymbol{\delta}) = \sum_{i \in V} \max_{x_i} \Big(\theta_i(x_i) + \sum_{f : i \in f} \delta_{fi}(x_i) \Big) + \sum_{f \in F} \max_{\boldsymbol{x}_f} \Big(\theta_f(\boldsymbol{x}_f) - \sum_{i \in f} \delta_{fi}(x_i) \Big).$$

The key property of the function $L(\boldsymbol{\delta})$ is that it involves maximization only over local assignments $\boldsymbol{x}_f$, a task which we assume to be tractable. The dual thus decouples the original problem, resulting in a problem that can be optimized using local operations. Figure 8.2 illustrates this for a simple pairwise model.

We will introduce algorithms that minimize the approximate objective $L(\boldsymbol{\delta})$ using local updates. Each iteration of the algorithms repeatedly finds a maximizing assignment for the subproblems individually, using these to update the dual variables that glue the subproblems together. We describe two classes of algorithms, one based on a *subgradient method* (see section 8.4) and another based on *block coordinate descent* (see section 8.5). These dual algorithms are simple and widely applicable to combinatorial problems in machine learning such as finding MAP assignments of graphical models.

---

both defined on all of the variables, and the single node potentials $\theta_i(x_i)$ are identically zero (not used). The variables here are binary, one for each directed edge, denoting whether a particular dependency exists.

3. We use the notation $\delta_{fi}(x_i)$ to denote a variable indexed by i, f, and x_i. An alternative notation could have been $\delta(x_i, f, i)$, but $\delta_{fi}(x_i)$ is more compact.

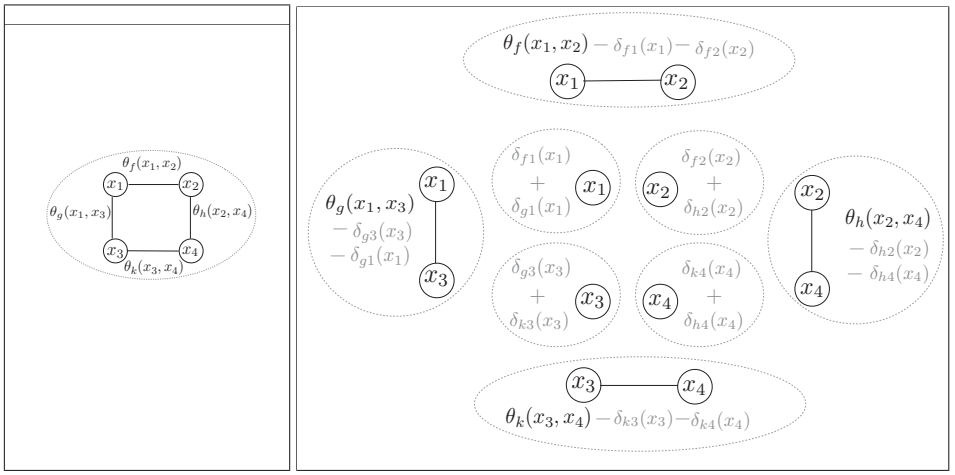

Figure 8.2: Illustration of the the dual decomposition objective. **Left:** The original pairwise model consisting of four factors. **Right:** The maximization problems corresponding to the objective $L(\boldsymbol{\delta})$. Each blue ellipse contains the factor to be maximized over. In all figures the singleton terms $\theta_i(x_i)$ are set to zero for simplicity.

8.3.1 Derivation of the Dual

In what follows we show how the dual optimization in Eq. 8.2 is derived from the original MAP problem in Eq. 8.1. We first slightly reformulate the problem by duplicating the x_i variables, once for each factor, and then enforcing that these are equal. Let x_i^f denote the copy of x_i used by factor f. Also, denote by $\boldsymbol{x}_f^f = \{x_i^f\}_{i \in f}$ the set of variables used by factor f, and, by $\boldsymbol{x}^F = \{\boldsymbol{x}_f^f\}_{f \in F}$, the set of all variable copies. This is illustrated graphically in Fig. 8.3. Then, our reformulated—but equivalent—optimization problem is

$$\begin{array}{ll} \max & \sum_{i \in V} \theta_i(x_i) + \sum_{f \in F} \theta_f(\boldsymbol{x}_f^f) \\ \text{s.t.} & x_i^f = x_i, \quad \forall f, i \in f. \end{array} \tag{8.3}$$

If we did not have the constraints, this maximization would simply decompose into independent maximizations for each factor, each of which we assume can be done efficiently. To remove these complicating constraints, we use the technique of *Lagrangian relaxation* (Geoffrion, 1974; Schlesinger, 1976; Fisher, 1981; Lemaréchal, 2001; Guignard, 2003). First, introduce La-

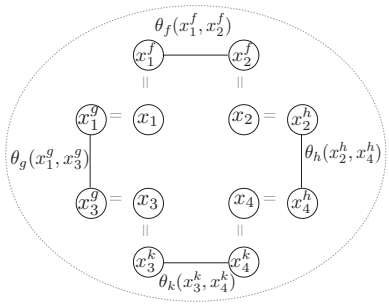

Figure 8.3: Illustration of the the derivation of the dual decomposition objective by creating copies of variables. The graph in Fig. 8.2 is shown here with the corresponding copies of the variables for each of the factors.

grange multipliers $\boldsymbol{\delta} = \{\delta_{fi}(x_i) : f \in F, i \in f, x_i\}$, and define the Lagrangian:

$$\begin{aligned} L(\boldsymbol{\delta}, \boldsymbol{x}, \boldsymbol{x}^F) &= \sum_{i \in V} \theta_i(x_i) + \sum_{f \in F} \theta_f(\boldsymbol{x}_f^f) \\ &\quad + \sum_{f \in F} \sum_{i \in f} \sum_{\hat{x}_i} \delta_{fi}(\hat{x}_i) \Big(1[x_i = \hat{x}_i] - 1[x_i^f = \hat{x}_i] \Big). \end{aligned} \tag{8.4}$$

The following problem is still equivalent to Eq. 8.3 for any value of $\boldsymbol{\delta}$,

$$\begin{aligned} &\max_{\boldsymbol{x}, \boldsymbol{x}^F} \quad L(\boldsymbol{\delta}, \boldsymbol{x}, \boldsymbol{x}^F) \\ &\text{s.t.} \qquad\quad x_i^f = x_i, \quad \forall f, i \in f. \end{aligned} \tag{8.5}$$

This follows because if the constraints in the above hold, then the last term in Eq. 8.4 is zero for any value of $\boldsymbol{\delta}$. In other words, the Lagrange multipliers are unnecessary if we already enforce the constraints.

Solving the maximization in Eq. 8.5 is as hard as the original MAP problem. To obtain a tractable optimization problem, we omit the constraint in Eq. 8.5 and define the function $L(\boldsymbol{\delta})$:

$$\begin{aligned} L(\boldsymbol{\delta}) &= \max_{\boldsymbol{x}, \boldsymbol{x}^F} L(\boldsymbol{\delta}, \boldsymbol{x}, \boldsymbol{x}^F) \\ &= \sum_{i \in V} \max_{x_i} \Big(\theta_i(x_i) + \sum_{f : i \in f} \delta_{fi}(x_i) \Big) + \sum_{f \in F} \max_{\boldsymbol{x}_f^f} \Big(\theta_f(\boldsymbol{x}_f^f) - \sum_{i \in f} \delta_{fi}(x_i^f) \Big). \end{aligned}$$

Since $L(\boldsymbol{\delta})$ maximizes over a larger space ($\boldsymbol{x}$ may not equal $\boldsymbol{x}^F$), we have that $\text{MAP}(\boldsymbol{\theta}) \leq L(\boldsymbol{\delta})$. The *dual problem* is to find the tightest such upper bound by optimizing the Lagrange multipliers: solving $\min_{\boldsymbol{\delta}} L(\boldsymbol{\delta})$.

Note that the maximizations are now unambiguously independent. We obtain Eq. 8.2 by replacing $\boldsymbol{x}_f^f$ with $\boldsymbol{x}_f$.

8.3.2 Reparameterization Interpretation of the Dual

The notion of reparameterization has played a key role in understanding approximate inference in general and MAP approximations in particular (Sontag and Jaakkola, 2009; Wainwright et al., 2003). It can also be used to interpret the optimization problem in Eq. 8.2, as we briefly review here.

Given a set of dual variables $\boldsymbol{\delta}$, define new factors on x_i and $\boldsymbol{x}_f$ given by

$$\begin{aligned} \bar{\theta}_i^{\boldsymbol{\delta}}(x_i) &= \theta_i(x_i) + \sum_{f:i\in f} \delta_{fi}(x_i) \\ \bar{\theta}_f^{\boldsymbol{\delta}}(\boldsymbol{x}_f) &= \theta_f(\boldsymbol{x}_f) - \sum_{i\in f} \delta_{fi}(x_i) \ . \end{aligned} \tag{8.6}$$

It is easy to see that these new factors define essentially the same function as the original factors $\boldsymbol{\theta}$. Namely, for all assignments $\boldsymbol{x}$,

$$\sum_{i\in V} \theta_i(x_i) + \sum_{f\in F} \theta_f(\boldsymbol{x}_f) = \sum_{i\in V} \bar{\theta}_i^{\boldsymbol{\delta}}(x_i) + \sum_{f\in F} \bar{\theta}_f^{\boldsymbol{\delta}}(\boldsymbol{x}_f) \ . \tag{8.7}$$

We call $\bar{\boldsymbol{\theta}}$ a reparameterization of the original parameters $\boldsymbol{\theta}$. Next we observe that the function $L(\boldsymbol{\delta})$ can be written as

$$L(\boldsymbol{\delta}) = \sum_{i\in V} \max_{x_i} \bar{\theta}_i^{\boldsymbol{\delta}}(x_i) + \sum_{f\in F} \max_{\boldsymbol{x}_f} \bar{\theta}_f^{\boldsymbol{\delta}}(\boldsymbol{x}_f). \tag{8.8}$$

To summarize the above, dual decomposition may be viewed as searching over a set of reparameterizations of the original factors $\boldsymbol{\theta}$, where each reparameterization provides an upper bound on the MAP and we are seeking to minimize this bound.

8.3.3 Formal Guarantees

In the previous section we showed that the dual always provides an upper bound on the optimum of our original optimization problem (Eq. 8.1),

$$\max_{\boldsymbol{x}} \sum_{i\in V} \theta_i(x_i) + \sum_{f\in F} \theta_f(\boldsymbol{x}_f) \leq \min_{\boldsymbol{\delta}} L(\boldsymbol{\delta}). \tag{8.9}$$

We do not necessarily have strong duality, that is, equality in the above equation. However, for some functions $\theta(\boldsymbol{x})$ strong duality does hold, as stated in the following theorem[4]

Theorem 8.1. *Suppose that* $\exists \boldsymbol{\delta}^*, \boldsymbol{x}^*$ *where* $x_i^* \in \operatorname{argmax}_{x_i} \bar{\theta}_i^{\boldsymbol{\delta}^*}(x_i)$ *and*

4. Versions of this theorem appear in multiple papers (e.g., see Geoffrion, 1974; Wainwright et al., 2005; Weiss et al., 2007).

$\boldsymbol{x}_f^* \in \operatorname{argmax}_{\boldsymbol{x}_f} \bar{\theta}_f^{\boldsymbol{\delta}^*}(\boldsymbol{x}_f)$. *Then* $\boldsymbol{x}^*$ *is a solution to the maximization problem in Eq. 8.1 and hence* $L(\boldsymbol{\delta}^*) = MAP(\boldsymbol{\theta})$.

Proof. For the given $\boldsymbol{\delta}^*$ and $\boldsymbol{x}^*$ we have that

$$L(\boldsymbol{\delta}^*) = \sum_i \bar{\theta}_i^{\boldsymbol{\delta}^*}(x_i^*) + \sum_{f \in F} \bar{\theta}_f^{\boldsymbol{\delta}^*}(\boldsymbol{x}_f^*) = \sum_i \theta_i(x_i^*) + \sum_{f \in F} \theta_f(\boldsymbol{x}_f^*), \tag{8.10}$$

where the equalities follow from the maximization property of $\boldsymbol{x}^*$ and the reparameterization property of $\bar{\boldsymbol{\theta}}$. On the other hand, from the definition of $\mathrm{MAP}(\boldsymbol{\theta})$ we have that

$$\sum_i \theta_i(x_i^*) + \sum_{f \in F} \theta_f(\boldsymbol{x}_f^*) \leq \mathrm{MAP}(\boldsymbol{\theta}). \tag{8.11}$$

Taking Eq. 8.10 and Eq. 8.11 together with $L(\boldsymbol{\delta}^*) \geq \mathrm{MAP}(\boldsymbol{\theta})$, we have equality in Eq. 8.11, so that $\boldsymbol{x}^*$ attains the MAP value and is therefore the MAP assignment, and $L(\boldsymbol{\delta}^*) = \mathrm{MAP}(\boldsymbol{\theta})$. □

The conditions of theorem 8.1 correspond to the subproblems *agreeing* on a maximizing assignment. Since agreement implies optimality of the dual, it can occur only after our algorithms find the tightest upper bound. Although the agreement is not guaranteed, if we do reach such a state, then theorem 8.1 ensures that we have the *exact* solution to Eq. 8.1. The dual solution $\boldsymbol{\delta}$ is said to provide a *certificate of optimality* in this case. In other words, if we find an assignment whose value matches the dual value, then the assignment has to be the MAP (strong duality).

For both the non-projective dependency parsing and protein side-chain placement problems, exact solutions (with certificates of optimality) are frequently found by using dual decomposition, in spite of the corresponding optimization problems being NP-complete (Koo et al., 2010; Sontag et al., 2008; Yanover et al., 2006).

We show in section 8.6 that equation (8.2) is the dual of an LP relaxation of the original problem. When the conditions of theorem 8.1 are satisfied, it means the LP relaxation is *tight* for this instance.

8.4 Subgradient Algorithms

For the remainder of this chapter, we show how to efficiently minimize the upper bound on the MAP assignment provided by the Lagrangian relaxation. Although $L(\boldsymbol{\delta})$ is convex and continuous, it is non-differentiable at all points $\boldsymbol{\delta}$ where $\bar{\theta}_i^{\boldsymbol{\delta}}(x_i)$ or $\bar{\theta}_f^{\boldsymbol{\delta}}(\boldsymbol{x}_f)$ have multiple optima (for some i or f). There are a large number of non-differentiable convex optimization techniques that could be applied in this setting (Fisher, 1981). In this section

we describe the subgradient method, which has been widely applied to solving Lagrangian relaxation problems and is often surprisingly effective, in spite of being such a simple method. The subgradient method is similar to gradient descent, but is applicable to non-differentiable objectives.

A complete treatment of subgradient methods is beyond the scope of this chapter. Our focus will be to introduce the general idea so that we can compare and contrast it with the block coordinate descent algorithms described in the next section. We refer the reader to Komodakis et al. (2011) for a detailed treatment of subgradient methods as they relate to inference problems (see also Held et al. (1974) for an early application of the subgradient method for Lagrangian relaxation, and to Koo et al. (2010) for a recent application to the non-projective dependency parsing problem described earlier). Our dual decomposition differs slightly from these earlier works in that we explicitly included single node factors and enforced that all other factors agree with them. As a result, our optimization problem is unconstrained, regardless of the number of factors. In contrast, Komodakis et al. (2011) have constraints enforcing that some of the dual variables sum to zero, resulting in a projected subgradient method.

A subgradient of a convex function $L(\boldsymbol{\delta})$ at $\boldsymbol{\delta}$ is a vector $g_{\boldsymbol{\delta}}$ such that for all $\boldsymbol{\delta}'$, $L(\boldsymbol{\delta}') \geq L(\boldsymbol{\delta}) + g_{\boldsymbol{\delta}} \cdot (\boldsymbol{\delta}' - \boldsymbol{\delta})$. The subgradient method is very simple to implement, alternating between individually maximizing the subproblems (which provides the subgradient) and updating the dual parameters $\boldsymbol{\delta}$ using the subgradient. More specifically, the subgradient descent strategy is as follows. Assume that the dual variables at iteration t are given by $\boldsymbol{\delta}^t$. Then their value at iteration $t+1$ is given by

$$\delta_{fi}^{t+1}(x_i) = \delta_{fi}^{t}(x_i) - \alpha_t \, g_{fi}^{t}(x_i) \ , \tag{8.12}$$

where $\boldsymbol{g}^t$ is a subgradient of $L(\boldsymbol{\delta})$ at $\boldsymbol{\delta}^t$ (i.e., $\boldsymbol{g} \in \partial L(\boldsymbol{\delta}^t)$) and α_t is a stepsize that may depend on t. We show one way to calculate this subgradient in section 8.4.1.

A well-known theoretical result is that the subgradient method is guaranteed to solve the dual to optimality whenever the stepsizes are chosen such that $\lim_{t\to\infty} \alpha_t = 0$ and $\sum_{t=0}^{\infty} \alpha_t = \infty$ (Anstreicher and Wolsey, 2009). One example of such a stepsize is $\alpha_t = \frac{1}{t}$. However, there are a large number of heuristic choices that can make the subgradient method faster. See Komodakis et al. (2011) for further possibilities of how to choose the step size, and for an empirical evaluation of these choices on inference problems arising from computer vision.

8.4.1 Calculating the Subgradient of $L(\boldsymbol{\delta})$

In this section we show how to calculate the subgradient of $L(\boldsymbol{\delta})$, completing the description of the subgradient algorithm. Given the current dual variables $\boldsymbol{\delta}^t$, we first choose a maximizing assignment for each subproblem. Let x_i^s be a maximizing assignment of $\bar{\theta}_i^{\boldsymbol{\delta}^t}(x_i)$, and let $\boldsymbol{x}_f^f$ be a maximizing assignment of $\bar{\theta}_f^{\boldsymbol{\delta}^t}(\boldsymbol{x}_f)$. The subgradient of $L(\boldsymbol{\delta})$ at $\boldsymbol{\delta}^t$ is then given by the following pseudocode:

$$
\begin{aligned}
& g_{fi}^t(x_i) = 0, \qquad \forall f, i \in f, x_i \\
& \textbf{For } f \in F \text{ and } i \in f\text{:} \\
& \qquad \textbf{If } x_i^f \neq x_i^s\text{:} \\
g_{fi}^t(x_i^s) &= +1 && (8.13) \\
g_{fi}^t(x_i^f) &= -1\ . && (8.14)
\end{aligned}
$$

Thus, each time that $x_i^f \neq x_i^s$, the subgradient update decreases the value of $\bar{\theta}_i^{\boldsymbol{\delta}^t}(x_i^s)$ and increases the value of $\bar{\theta}_i^{\boldsymbol{\delta}^t}(x_i^f)$. Similarly, for all $\boldsymbol{x}_{f\setminus i}$, the subgradient update decreases the value of $\bar{\theta}_f^{\boldsymbol{\delta}^t}(x_i^f, \boldsymbol{x}_{f\setminus i})$ and increases the value of $\bar{\theta}_f^{\boldsymbol{\delta}^t}(x_i^s, \boldsymbol{x}_{f\setminus i})$. Intuitively, as a result of the update, in the next iteration the factors are more likely to agree with one another on the value of x_i in their maximizing assignments.

Typically one runs the subgradient algorithm until either $L(\boldsymbol{\delta})$ stops decreasing significantly or we have reached some maximum number of iterations. If at any iteration t we find that $\boldsymbol{g}^t = 0$, then $x_i^f = x_i^s$ for all $f \in F, i \in f$. Therefore, by theorem 8.1, $\boldsymbol{x}^s$ must be an MAP assignment, and so we have solved the dual to optimality. However, the converse does not always hold: $\boldsymbol{g}^t$ may be non-zero even when $\boldsymbol{\delta}^t$ is dual optimal. We discuss these issues further in section 8.7.

In the *incremental* subgradient method, at each iteration one computes the subgradient using only some of the subproblems, $F' \subset F$, rather than using all factors in F (Bertsekas, 1995). This can significantly decrease the overall running time, and is also more similar to the block coordinate descent methods that we describe next, which make updates with respect to only one factor at a time.

8.4.2 Efficiently Maximizing over the Subproblems

To choose a maximizing assignment for each subproblem, needed for making the subgradient updates, we have to solve the following combinatorial

optimization problem:

$$\max_{\boldsymbol{x}_f} \left[\theta_f(\boldsymbol{x}_f) - \sum_{i \in f} \delta_{fi}(x_i) \right] . \tag{8.15}$$

When the number of possible assignments $\boldsymbol{x}_f$ is small, this maximization can be done simply by enumeration. For example, in pairwise MRFs each factor $f \in F$ consists of just two variables, $|f| = 2$. Suppose each variable takes k states. Then this maximization takes k^2 time.

Often the number of assignments is large but the functions $\theta_f(\boldsymbol{x}_f)$ are *sparse*, allowing the maximization to be performed using dynamic programming or combinatorial algorithms. For example, in the non-projective dependency parsing problem, $\theta_1(\boldsymbol{x}) = -\infty$ if the set of edges specified by $\boldsymbol{x}$ include a cycle, and 0 otherwise. Maximization in Eq. 8.15 can be solved efficiently in this case as follows: We form a directed graph where the weight of the edge corresponding to x_i is $\delta_{1i}(1) - \delta_{1i}(0)$. Then, we solve Eq. 8.15 by finding a directed minimum-weight spanning tree on this graph.

The following are some of the sparse factors that are frequently found in inference problems, all of which can be efficiently maximized over:

- Tree structures (Wainwright et al., 2005; Komodakis et al., 2011)
- Matchings (Lacoste-Julien et al., 2006; Duchi et al., 2007; Yarkony et al., 2010)
- Supermodular functions (Komodakis et al., 2011)
- Cardinality and order constraints (Gupta et al., 2007; Tarlow et al., 2010)
- Functions with small support (Rother et al., 2009)

Consider, for example, a factor which enforces a cardinality constraint over binary variables: $\theta_f(\boldsymbol{x}_f) = 0$ if $\sum_{i \in f} x_i = L$, and $\theta_f(\boldsymbol{x}_f) = -\infty$ otherwise. Let $e_i = \delta_{fi}(1) - \delta_{fi}(0)$. To solve Eq. 8.15, we first sort e_i in ascending order for $i \in f$. Then, the maximizing assignment is obtained by setting the corresponding $x_i = 1$ for the first L values, and $x_i = 0$ for the remainder. Thus, computing the maximum assignment takes only $O(|f| \log |f|)$ time.

8.5 Block Coordinate Descent Algorithms

A different approach to solving the optimization problem in Eq. 8.2 is via coordinate descent. Coordinate-descent algorithms have a long history of being used to optimize Lagrangian relaxations (e.g., see Erlenkotter, 1978; Guignard and Rosenwein, 1989). Such algorithms work by fixing the values

of all dual variables except for a set of variables, and then minimizing the objective as much as possible with respect to that set. The two key design choices to make are which variables to update and how to update them. In all the updates we consider below, the coordinates are updated to their optimal value (i.e., the one that minimizes $L(\boldsymbol{\delta})$), given the fixed coordinates.

There is a significant amount of flexibility with regard to choosing the coordinate blocks, that is, the variables to update. A first attempt at choosing such a block may be to focus only on coordinates for which the subgradient is non-zero. For example, if $x_i^s = \arg\max_{x_i} \bar{\theta}_i^{\boldsymbol{\delta}^t}(x_i)$, we may choose to update only $\delta_{fi}(x_i^s)$, and not $\delta_{fi}(x_i)$, for $x_i \neq x_i^s$. However, this may result in too small a change in $L(\boldsymbol{\delta})$, and the coordinates we have not updated may require update at a later stage. The updates we consider below will update $\delta_{fi}(x_i)$ for all values of x_i regardless of the maximizing assignment. This is very different from the subgradient method, which updates only the dual variables corresponding to the maximizing assignments for the subproblems, $\delta_{fi}(x_i^s)$ and $\delta_{fi}(x_i^f)$.

We shall describe various block coordinate descent algorithms, each algorithm using an increasingly larger block size. The advantage of coordinate descent algorithms is that they are local, parameter free, simple to implement, and often provide faster convergence than subgradient methods. However, as we discuss later, there are cases where coordinate descent algorithms may not reach the dual optimum. In section 8.8 we discuss the issue of choosing between coordinate descent and subgradient based schemes.

For the practitioner interested in an algorithm to apply, in Fig. 8.4 we give pseudocode for one of the block coordinate descent algorithms.

8.5.1 The Max-Sum Diffusion algorithm

Suppose that we fix all of the dual variables $\boldsymbol{\delta}$ except $\delta_{fi}(x_i)$ for a specific f and i. We now wish to find the values of $\delta_{fi}(x_i)$ that minimize the objective $L(\boldsymbol{\delta})$, given the other fixed values. In general there is not a unique solution to this restricted optimization problem, and different update strategies will result in different overall running times.

The max-sum diffusion (MSD) algorithm (Kovalevsky and Koval; Werner, 2007, 2008) performs the following block coordinate descent update (for all x_i simultaneously):

$$\delta_{fi}(x_i) = -\tfrac{1}{2}\delta_i^{-f}(x_i) + \tfrac{1}{2}\max_{\boldsymbol{x}_{f\setminus i}} \left[\theta_f(\boldsymbol{x}_f) - \sum_{\hat{i}\in f\setminus i} \delta_{f\hat{i}}(x_{\hat{i}})\right] , \tag{8.17}$$

where we define $\delta_i^{-f}(x_i) = \theta_i(x_i) + \sum_{\hat{f}\neq f} \delta_{\hat{f}i}(x_i)$. The algorithm iteratively

Inputs:

- A set of factors $\theta_i(x_i), \theta_f(\boldsymbol{x}_f)$

Output:

- An assignment $x_1, \ldots, x_n$ that approximates the MAP

Algorithm:

- Initialize $\delta_{fi}(x_i) = 0, \quad \forall f \in F, i \in f, x_i$
- Iterate until small enough change in $L(\boldsymbol{\delta})$ (see Eq. 8.2):
  For each $f \in F$, perform the updates

$$\delta_{fi}(x_i) = -\delta_i^{-f}(x_i) + \frac{1}{|f|} \max_{\boldsymbol{x}_{f \setminus i}} \left[\theta_f(\boldsymbol{x}_f) + \sum_{\hat{i} \in f} \delta_{\hat{i}}^{-f}(x_{\hat{i}}) \right] \tag{8.16}$$

  simultaneously for all $i \in f$ and x_i. We define $\delta_i^{-f}(x_i) = \theta_i(x_i) + \sum_{\hat{f} \neq f} \delta_{\hat{f}i}(x_i)$
- Return $x_i \in \arg\max_{\hat{x}_i} \bar{\theta}_i^{\boldsymbol{\delta}}(\hat{x}_i)$ (see Eq. 8.6).

Figure 8.4: Description of the MPLP block coordinate descent algorithm for minimizing the dual $L(\boldsymbol{\delta})$ (see section 8.5.2). Similar algorithms can be devised for different choices of coordinate blocks. See sections 8.5.1 and 8.5.3. The assignment returned in the final step follows the decoding scheme discussed in section 8.7.

chooses some f and sequentially performs these updates for each $i \in f$. In appendix 8.9.1 we show how to derive this algorithm as block coordinate descent on $L(\boldsymbol{\delta})$. The proof also illustrates the following *equalization* property: after the update, we have $\bar{\theta}_i^{\boldsymbol{\delta}}(x_i) = \max_{\boldsymbol{x}_{f \setminus i}} \bar{\theta}_f^{\boldsymbol{\delta}}(\boldsymbol{x}_f), \forall x_i$. In other words, the reparameterized factors for f and i agree on the utility of state x_i.

8.5.2 The MPLP algorithm

In this section, we show that it is possible to do coordinate descent in closed form over a significantly larger block of coordinates than that of section 8.5.1. The max-product linear programming (MPLP) algorithm was introduced as a coordinate-descent algorithm for LP relaxations of MAP problems (Globerson and Jaakkola, 2008). Here we show that it can also be interpreted as a block coordinate descent algorithm for Eq. 8.2.

Assume we fix all the variables $\boldsymbol{\delta}$ except $\delta_{fi}(x_i)$ for a specific f and *all* i (note that this differs from section 8.5.1, where only one i was not fixed). We now wish to find the values of $\delta_{fi}(x_i)$ that minimize the objective $L(\boldsymbol{\delta})$, given the other fixed values. We claim that the following update achieves

this:

$$\delta_{fi}(x_i) = -\left(1 - \frac{1}{|f|}\right)\delta_i^{-f}(x_i) + \frac{1}{|f|}\max_{\boldsymbol{x}_{f\setminus i}}\left[\theta_f(\boldsymbol{x}_f) + \sum_{\hat{i}\in f\setminus i}\delta_{\hat{i}}^{-f}(x_{\hat{i}})\right] \quad (8.18)$$

The update can be equivalently written as

$$\delta_{fi}(x_i) = -\delta_i^{-f}(x_i) + \frac{1}{|f|}\max_{\boldsymbol{x}_{f\setminus i}}\left[\theta_f(\boldsymbol{x}_f) + \sum_{\hat{i}\in f}\delta_{\hat{i}}^{-f}(x_{\hat{i}})\right] \quad . \quad (8.19)$$

In appendix 8.9.2 we show that this update indeed corresponds to the desired block coordinate descent on $L(\boldsymbol{\delta})$. Note that the update needs to be performed for all $i \in f$ and x_i simultaneously, unlike MSD, which updates for a particular i each time.

One iteration of MPLP (i.e., performing the updates once for every $f \in F$, $i \in f$, and x_i) has exactly the same running time as an iteration of MSD.[5] However, each MPLP update is much more effective, with bigger gains expected as $|f|$ grows. For example, consider a model with single node factors $\theta_i(x_i)$ and only one $\theta_f(\boldsymbol{x}_f)$, that is $|F| = 1$. For this model, MPLP exactly solves the dual in the first iteration, whereas MSD would take several iterations to solve the dual to optimality.

8.5.3 Larger Coordinate Blocks

Ideally we would like to optimize in closed form over as large a coordinate block as possible. It turns out that when the factors are pairwise, we can use considerably larger blocks than those that have been discussed above.

As an illustration of this approach, we now show a *star update* for pairwise models (Globerson and Jaakkola, 2008). Consider a model where factors correspond to pairs of variables, so that the elements of F are of the form $\{i, j\}$. Now, consider a block coordinate descent approach where the free coordinates are $\delta_{\{i,j\}j}(x_j), \delta_{\{i,j\}i}(x_i)$ for a given i and all its neighbors $j \in N(i)$. Although it is non-trivial to derive, we show in appendix 8.9.3 that a closed form update for these coordinates exists. The free coordinates

5. Assuming that we keep $\bar{\theta}_i^{\boldsymbol{\delta}}(x_i)$ in memory and compute $\delta_i^{-f}(x_i) = \bar{\theta}_i^{\boldsymbol{\delta}}(x_i) - \delta_{fi}(x_i)$.

that minimize the objective $L(\boldsymbol{\delta})$ are given by[6]

$$\begin{aligned}\delta_{\{i,j\}i}(x_i) &= -\frac{1}{1+N_i}\gamma_i(x_i) + \gamma_{ji}(x_i) \qquad (8.20)\\ \delta_{\{i,j\}j}(x_j) &= -\tfrac{1}{2}\delta_j^{-i}(x_j) + \tfrac{1}{2}\max_{x_i}\left[\theta_{ij}(x_i,x_j) + \frac{2}{1+N_i}\gamma_i(x_i) - \gamma_{ji}(x_i)\right]\end{aligned}$$

where we define $\delta_j^{-i}(x_j) = \theta_j(x_j) + \sum_{k\in N(j)\setminus i}\delta_{\{k,j\}j}(x_j)$, and

$$\begin{aligned}\gamma_{ji}(x_i) &= \max_{x_j}\left[\theta_{ij}(x_i,x_j) + \delta_j^{-i}(x_j)\right]\\ \gamma_i(x_i) &= \theta_i(x_i) + \sum_{j\in N(i)}\gamma_{ji}(x_i)\end{aligned}$$

where $N_i = |N(i)|$ is the number of neighbors of node i in the pairwise model. For a given node i, the update needs to be performed for all factors $\{i,j\}$, x_i, and x_j simultaneously, where $j \in N(i)$.

Often a closed-form update is not possible, but one can still efficiently compute the updates. For example, Sontag and Jaakkola (2009) give a linear time algorithm for coordinate descent in pairwise models using blocks corresponding to the edges of a tree-structured graph.

8.5.4 Efficiently Computing the Updates

Recall that the subgradient method needs to solve only one maximization per subproblem to perform the subgradient update (see Eq. 8.15). We showed in section 8.4.2 that these can be solved efficiently for a large number of *sparse* factors, even though the factors may involve a large number of variables.

Consider the MPLP update given in Eq. 8.16. To perform the updates, one must compute *max-marginals* for each subproblem, that is, the value of the optimal assignment after fixing each variable (individually) to one of its states. Specifically, for every f, i, and x_i, we must compute

$$\max_{\boldsymbol{x}_{f\setminus i}} h(\boldsymbol{x}_{f\setminus i}, x_i), \qquad h(\boldsymbol{x}_f) = \theta_f(\boldsymbol{x}_f) + \sum_{\hat{i}\in f}\delta_{\hat{i}}^{-f}(x_{\hat{i}}) \ .$$

Often this can be done in the same running time that it takes to find the maximizing assignment, that is, $\max_{\boldsymbol{x}_f} h(\boldsymbol{x}_f)$.[7] For example, if the subproblem is a tree structure, then with just two passes—propagating max-product messages from the leaves to the root, and then from the root back to the leaves—we can compute the max-marginals. Tarlow et al. (2010)

6. The star update that appeared in Globerson and Jaakkola (2008) under the name NMPLP, had an error. This is the correct version.
7. To perform the updates we need only the optimal *value*, not the maximizing assignment.

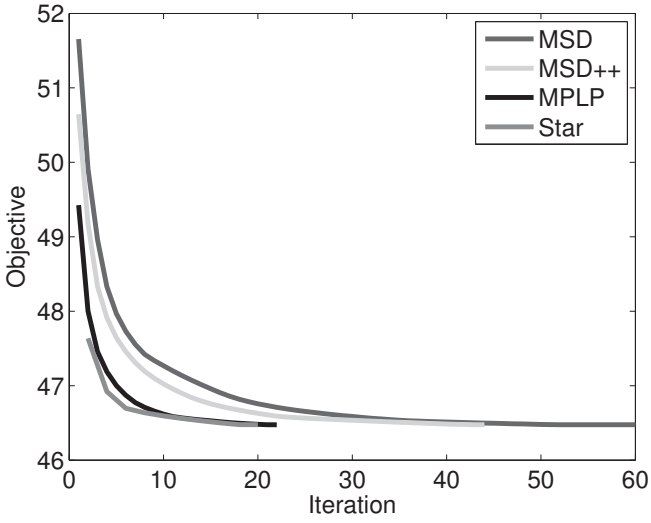

Figure 8.5: Comparison of three coordinate descent algorithms on a 10×10 two dimensional Ising grid. The dual objective $L(\boldsymbol{\delta})$ is shown as a function of iteration number. We multiplied the number of iterations for the star update by two, since each edge variable is updated twice.

show how max-marginals can be efficiently computed for factors enforcing cardinality and order constraints, and Duchi et al. (2007) show how to do this for matchings. In all of these cases, one iteration of MPLP has the same running time as one iteration of the subgradient method.

However, for some problems, computing the max-marginals can take longer. For example, it is not clear how to compute max-marginals efficiently for the factor $\theta_1(\boldsymbol{x})$ which enforced acyclicity in the non-projective dependency problem. The weight for the edge corresponding to x_i is $\delta_i^{-1}(0) - \delta_i^{-1}(1)$. The computational problem is to find, for each edge, the weight of the minimum directed spanning tree (MST) if we force it to include this edge ($x_i = 1$) and if we force it not to include this edge ($x_i = 0$).[8] We could compute the max-marginals by solving $2|V|$ separate directed MST problems, but this would increase the overall running time of each iteration. Thus, it is possible that the subgradient approach is more suitable here.

8.5.5 Empirical Comparison of the Algorithms

All the algorithms described in this section minimize $L(\boldsymbol{\delta})$, but do so via different coordinate blocks. How do these algorithms compare to one another? One criterion is the number of coordinates per block. The more coordinates we optimize in closed form, the faster we would expect our algorithm to make progress. Thus, we would expect the MPLP updates given in section 8.5.2 to minimize $L(\boldsymbol{\delta})$ in fewer iterations than the MSD updates from section 8.5.1, since the former optimizes over more coordinates simultaneously. Comparing the star update from section 8.5.3 to MPLP with edge updates is a bit more delicate since, in the former, the coordinate blocks overlap. Thus, we may be doing redundant computations in the star update. On the other hand, it does provide a closed-form update over larger blocks and thus may result in faster convergence.

To assess the difference between the algorithms, we test them on a pairwise model with binary variables. The graph structure is a two dimensional 10×10 grid and the interactions are Ising (see Globerson and Jaakkola, 2008, for a similar experimental setup). We compare three algorithms:

- MSD – At each iteration, for each edge, it updates the message from the edge to one of its endpoints (i.e., $\delta_{\{i,j\}i}(x_i)$ for all x_i), and then updates the message from the edge to its other endpoint.
- MPLP – At each iteration, for each edge, it updates the messages from the edge to both of its endpoints (i.e., $\delta_{\{i,j\}i}(x_i)$ *and* $\delta_{\{i,j\}j}(x_j)$, for all x_i, x_j).
- Star update – At each iteration, for each node i, it updates the messages from all edges incident on i to both of their endpoints (i.e., $\delta_{\{i,j\}i}(x_i)$ and $\delta_{\{i,j\}j}(x_j)$ for all $j \in N(i), x_i, x_j$).
- MSD++ – See section 8.5.6.

The running times per iteration of MSD and MPLP are identical. We let each iteration of the star update correspond to two iterations of the edge updates to make the running times comparable.

Results for a model with random parameters are shown in Fig. 8.5, and are representative of what we found across a range of model parameters. The results verify the arguments made above: MPLP is considerably faster than MSD, and the star update is somewhat faster than MPLP.

---

8. The corresponding max-marginal is the MST's weight plus the constant $-\sum_{i \in f} \delta_i^{-1}(0)$.

8.5.6 What Makes MPLP Faster Than MSD?

It is clear that MPLP uses a larger block size than MSD. However, the two algorithms also use different equalization strategies, which we will show is just as important. We illustrate this difference by giving a new algorithm, MSD++, which uses the same block size as MPLP, but an equalization method that is more similar to MSD's.

For a given factor f, assume all $\boldsymbol{\delta}$ variables are fixed except $\delta_{fi}(x_i)$ for all $i \in f$ and x_i (as in MPLP). We would like to minimize $L(\boldsymbol{\delta})$ with respect to the free variables. The part of the objective $L(\boldsymbol{\delta})$ that depends on the free variables is

$$\bar{L}(\boldsymbol{\delta}) = \sum_{i \in f} \max_{x_i} \bar{\theta}_i^{\boldsymbol{\delta}}(x_i) + \max_{\boldsymbol{x}_f} \bar{\theta}_f^{\boldsymbol{\delta}}(\boldsymbol{x}_f) \ ,$$

where $\bar{\theta}$ are the reparameterizations defined in Eq. 8.6.

The MPLP update can be shown to give a $\boldsymbol{\delta}^{t+1}$ such that $\bar{L}(\boldsymbol{\delta}^{t+1}) = \max_{\boldsymbol{x}_f} h^{\boldsymbol{\delta}^t}(\boldsymbol{x}_f)$, where $h^{\boldsymbol{\delta}^t}(\boldsymbol{x}_f) = \sum_{i \in f} \bar{\theta}_i^{\boldsymbol{\delta}^t}(x_i) + \bar{\theta}_f^{\boldsymbol{\delta}^t}(\boldsymbol{x}_f)$. However, there are many possible choices for $\boldsymbol{\delta}^{t+1}$ that would achieve the same decrease in the dual objective. In particular, for any choice of non-negative weights α_f, α_i for $i \in f$ such that $\alpha_f + \sum_{i \in f} \alpha_i = 1$, we can choose $\boldsymbol{\delta}^{t+1}$ so that

$$\begin{aligned} \bar{\theta}_i^{\boldsymbol{\delta}^{t+1}}(x_i) &= \alpha_i \cdot \max_{\boldsymbol{x}_{f \setminus i}} h^{\boldsymbol{\delta}^t}(\boldsymbol{x}_f) \qquad \forall i \in f, \\ \bar{\theta}_f^{\boldsymbol{\delta}^{t+1}}(\boldsymbol{x}_f) &= h^{\boldsymbol{\delta}^t}(\boldsymbol{x}_f) - \sum_{i \in f} \alpha_i \cdot \max_{\hat{\boldsymbol{x}}_{f \setminus i}} h^{\boldsymbol{\delta}^t}(x_i, \hat{\boldsymbol{x}}_{f \setminus i}). \end{aligned}$$

Thus, we have that $\max_{x_i} \bar{\theta}_i^{\boldsymbol{\delta}^{t+1}}(x_i) = \alpha_i \cdot \max_{\boldsymbol{x}_f} h^{\boldsymbol{\delta}^t}(\boldsymbol{x}_f)$ and $\max_{\boldsymbol{x}_f} \bar{\theta}_f^{\boldsymbol{\delta}^{t+1}}(\boldsymbol{x}_f) = \alpha_f \cdot \max_{\boldsymbol{x}_f} h^{\boldsymbol{\delta}^t}(\boldsymbol{x}_f)$. All of these statements can be proved using arguments analogous to those given in appendix 8.9.2.

The MPLP algorithm corresponds to the choice $\alpha_f = 0$, $\alpha_i = \frac{1}{|f|}$ for $i \in f$. The reparameterization results in as much of $h^{\boldsymbol{\delta}^t}(\boldsymbol{x}_f)$ being pushed to the single node terms $\bar{\theta}_i^{\boldsymbol{\delta}^{t+1}}(x_i)$ as possible. Thus, subsequent updates for other factors that share variables with f will be affected by this update as much as possible. Intuitively, this increases the amount of communication between the subproblems. MPLP's dual optimal fixed points will have $\max_{x_i} \bar{\theta}_i^{\boldsymbol{\delta}}(x_i) = \frac{L(\boldsymbol{\delta})}{|V|}$ for all i, and $\max_{\boldsymbol{x}_f} \bar{\theta}_f^{\boldsymbol{\delta}}(\boldsymbol{x}_f) = 0$ for all $f \in F$.

In contrast, each MSD update to $\delta_{fi}(x_i)$ divides the objective equally between the factor term and the node term (see appendix 8.9.1). As a result, MSD's dual optimal fixed points have $\max_{x_i} \bar{\theta}_i^{\boldsymbol{\delta}}(x_i) = \max_{\boldsymbol{x}_f} \bar{\theta}_f^{\boldsymbol{\delta}}(\boldsymbol{x}_f) = \frac{L(\boldsymbol{\delta})}{|V|+|F|}$ for all $i \in V$ and $f \in F$. Consider the alternative block coordinate

descent update given by

$$\delta_{fi}(x_i) = \frac{-|f|}{|f|+1}\delta_i^{-f}(x_i) + \frac{1}{|f|+1}\max_{\boldsymbol{x}_{f\setminus i}}\left[\theta_f(\boldsymbol{x}_f) + \sum_{\hat{i}\in f\setminus i}\delta_{\hat{i}}^{-f}(x_{\hat{i}})\right] \quad (8.21)$$

for *all* $i \in f$ and x_i simultaneously. This update, which we call MSD++, corresponds to the choice $\alpha_i = \alpha_f = \frac{1}{|f|+1}$, and has similar fixed points.

We show in Fig. 8.5 that MSD++ is only slightly faster than MSD, despite using a block size that is the same as that of MPLP. This suggests that it is MPLP's choice of equalization method (i.e., the α_i's) that provides the substantial improvements over MSD, not simply the choice of block size. As $\alpha_f \to 1$, the number of iterations required to solve the dual to optimality increases even further. The extreme case, $\alpha_f = 1$, although still a valid block coordinate descent step, gets stuck after the first iteration.

8.5.7 Convergence of Dual Coordinate Descent

Although coordinate descent algorithms decrease the dual objective at every iteration, they are not generally guaranteed to converge to the dual optimum. The reason is that although the dual objective $L(\boldsymbol{\delta})$ is convex, it is not strictly convex. This implies that the minimizing coordinate value may not be unique, and thus convergence guarantees for coordinate descent algorithms do not hold (Bertsekas, 1995). Interestingly, for pairwise MRFs with binary variables, the fixed points of the coordinate descent algorithms do correspond to global optima (Kolmogorov and Wainwright, 2005; Globerson and Jaakkola, 2008).

One strategy to avoid the above problem is to replace the max function in the objective of Eq. 8.2 with a *soft-max* function (e.g., see Johnson, 2008; Hazan and Shashua, 2010) which is smooth and strictly convex. As a result, coordinate descent converges globally.[9] An alternative approach are the auction algorithms proposed by Bertsekas (1992). However, currently there does not seem to be a coordinate descent approach to Eq. 8.2 that is guaranteed to converge globally for general problems.

8.6 Relations to Linear Programming Relaxations

Linear programming relaxations are a popular approach to approximating combinatorial optimization problems. One of these relaxations is in fact well

9. To solve the original dual, the soft-max needs to be gradually changed to the true max.

known to be equivalent to the dual decomposition approach discussed in this chapter (Schlesinger, 1976; Guignard and Kim, 1987; Komodakis et al., 2011; Wainwright et al., 2005; Werner, 2007). In this section we describe the corresponding LP relaxation.

We obtain a relaxation of the discrete optimization problem given in Eq. 8.1 by replacing it with the following linear program:

$$\max_{\boldsymbol{\mu} \in M_L} \left\{ \sum_f \sum_{\boldsymbol{x}_f} \theta_f(\boldsymbol{x}_f) \mu_f(\boldsymbol{x}_f) + \sum_i \sum_{x_i} \theta_i(x_i) \mu_i(x_i) \right\} \tag{8.22}$$

where the *local marginal polytope* M_L enforces that $\{\mu_i(x_i), \forall x_i\}$ and $\{\mu_f(\boldsymbol{x}_f), \forall \boldsymbol{x}_f\}$ correspond to valid (local) probability distributions and that, for each factor f, $\mu_f(\boldsymbol{x}_f)$, is consistent with $\mu_i(x_i)$ for all $i \in f, x_i$:

$$M_L = \left\{ \boldsymbol{\mu} \geq 0 : \begin{array}{ll} \sum_{\boldsymbol{x}_{f \setminus i}} \mu_f(\boldsymbol{x}_f) = \mu_i(x_i) & \forall f, i \in f, x_i \\ \sum_{x_i} \mu_i(x_i) = 1 & \forall i \end{array} \right\}. \tag{8.23}$$

Standard duality transformations can be used to show that the convex dual of the LP relaxation (Eq. 8.22) is the Lagrangian relaxation (Eq. 8.2). The integral vertices of the local marginal polytope (i.e., $\boldsymbol{\mu}$ such that $\mu_i(x_i) \in \{0, 1\}$) can be shown to correspond 1-to-1 with assignments $\boldsymbol{x}$ to the variables of the graphical model. The local marginal polytope also has *fractional* vertices that do not correspond to any global assignment (Wainwright and Jordan, 2008). Since the relaxation optimizes over this larger space, the value of the LP solution always upper-bounds the value of the MAP assignment. We call the relaxation *tight* for an instance if the optimum of the LP has the same value as the MAP assignment.

The connection to LP relaxations is important for a number of reasons. First, it shows how the algorithms discussed in this chapter apply equally to dual decomposition and linear programming relaxations for the MAP problem. Second, it allows us to understand when two different dual decompositions are equivalent to each other in the sense that, for all instances, the Lagrangian relaxation provides the same upper bound (at optimality). Realizing that two formulations are equivalent allows us to attribute differences in empirical results to the algorithms used to *optimize* the duals, not the tightness of the LP relaxation.

The third reason why the connection to LP relaxations is important is that it allows us to tighten the relaxation by adding valid constraints that are guaranteed not to cut off any integer solutions. The dual decomposition that we introduced in section 8.3 used only single node *intersection sets*, enforcing that the subproblems are consistent with one another on the individual variables. To obtain tighter LP relaxations, we typically have

to use larger intersection sets along with the new constraints.[10] Although further discussion of this is beyond the scope of this chapter, details can be found in Sontag (2010) and Werner (2008).

8.7 Decoding: Finding the MAP Assignment

Thus far, we have discussed how the subgradient method and block coordinate descent can be used to solve the Lagrangian relaxation. Although this provides an upper bound on the *value* of the MAP assignment, for most inference problems we actually want to find the assignment itself. In Fig. 8.4 we suggested one simple way to find an assignment from the dual beliefs, by locally decoding the single node reparameterizations:

$$x_i \leftarrow \operatorname*{argmax}_{\hat{x}_i} \bar{\theta}_i^{\boldsymbol{\delta}}(\hat{x}_i). \tag{8.24}$$

The node terms $\bar{\theta}_i^{\boldsymbol{\delta}}(x_i)$ may not have a unique maximum. However, when they do have a unique maximum, we say that $\boldsymbol{\delta}$ is *locally decodable* to $\boldsymbol{x}$. This section addresses the important question of when this and related approaches will succeed in finding the MAP assignment. Proofs are given in appendix 8.9.4.

Before the dual is solved (close) to optimality, it is typically not possible to give guarantees as to how good an assignment will be found by this local decoding scheme. However, once the algorithms have solved the dual to optimality, if the solution is locally decodable to $\boldsymbol{x}$, then $\boldsymbol{x}$ is both the MAP assignment and the only LP solution, as stated in the next theorem.

Theorem 8.2. *If a dual optimal $\boldsymbol{\delta}^*$ is locally decodable to $\boldsymbol{x}^*$, then the LP relaxation has a unique solution, and it is $\boldsymbol{x}^*$.*

Thus, this result gives a strong *necessary* condition for when locally decoding the dual solution can find the MAP assignments only when the LP relaxation is tight, the MAP assignment is unique, and there are no optimal fractional solutions.[11]

A natural question is whether this assumption that the LP relaxation has a unique solution that is integral, is *sufficient* for a dual solution to be locally decodable. As we illustrate later, in general the answer is no, as not all dual

10. Globerson and Jaakkola (2008) give the MPLP updates for larger intersection sets.
11. An alternative way to find the MAP assignment in this setting would be to directly solve the LP relaxation using a generic LP solver. However, recall that we use dual decomposition for reasons of computational efficiency.

solutions are locally decodable. We are, however, guaranteed that one exists.

Theorem 8.3. *If the LP relaxation has a unique solution and it is integral, there exists a dual optimal $\boldsymbol{\delta}^*$ that is locally decodable.*

Ideally, our algorithms would be guaranteed to find such a locally decodable dual solution. We show in section 8.7.1 that the subgradient method is indeed guaranteed to find a locally decodable solution under this assumption. For coordinate descent methods there do exist cases where a locally decodable solution may not be found. However, in practice this is often not a problem, and there are classes of graphical models where locally decodable solutions are guaranteed to be found (see section 8.7.2).

Rather than locally decoding an assignment, we could try searching for a global assignment $\boldsymbol{x}^*$ which maximizes each of the local subproblems. We showed in theorem 8.1 that such an *agreeing* assignment would be an MAP assignment, and its existence would imply that the LP relaxation is tight. Thus, the LP relaxation being tight is a necessary requirement for this strategy to succeed. For pairwise models with binary variables, an agreeing assignment can be found in linear time when one exists (Johnson, 2008). However, this does not extend to larger factors or non-binary variables.

Theorem 8.4. *Finding an agreeing assignment is NP-complete even when the LP relaxation is tight and the MAP assignment is unique.*

The construction in the above theorem uses a model where the optimum of the LP relaxation is not unique; although there may be a unique MAP assignment, there are also fractional vertices that have the same optimal value. When the LP has a unique and integral optimum, the decoding problem is no longer hard (e.g., it can be solved by optimizing an LP). Thus, asking that the LP relaxation have a unique solution that is integral seems to be a very reasonable assumption for a decoding method to succeed. When the LP relaxation does not have a unique solution, a possible remedy is to perturb the objective function by adding a small (in magnitude) random vector. If the LP relaxation is not tight, we can attempt to tighten the relaxation, for example by using the approaches discussed in Sontag et al. (2008).

8.7.1 Subgradient Method

The problem of recovering a primal solution (i.e., a solution to the LP relaxation given in Eq. 8.22) when solving a Lagrangian relaxation using subgradient methods has been well studied (e.g., see Anstreicher and Wolsey, 2009; Nedić and Ozdaglar, 2009; Shor, 1985). We next describe one such

approach (the simplest) and discuss its implications for finding the MAP assignment from the dual solution.

Given the current dual variables $\boldsymbol{\delta}^t$, define the following indicator functions for a maximizing assignment of the subproblems: $\mu_f^t(\boldsymbol{x}_f) = 1[\boldsymbol{x}_f = \text{argmax}_{\hat{\boldsymbol{x}}_f} \bar{\theta}_f^{\boldsymbol{\delta}^t}(\hat{\boldsymbol{x}}_f)]$, and $\mu_i^t(x_i) = 1[x_i = \text{argmax}_{\hat{x}_i} \bar{\theta}_i^{\boldsymbol{\delta}^t}(\hat{x}_i)]$. When the maximum is not unique, choose any one of the maximizing assignments.

Next, consider the average of these indicator functions across all subgradient iterations:

$$\begin{aligned} \overline{\mu}_f(\boldsymbol{x}_f) &= \frac{1}{T}\sum_{t=1}^{T} \mu_f^t(\boldsymbol{x}_f) \\ \overline{\mu}_i(x_i) &= \frac{1}{T}\sum_{t=1}^{T} \mu_i^t(x_i). \end{aligned}$$

For many common choices of stepsizes, the estimate $\overline{\boldsymbol{\mu}}$ can be shown to converge to a solution of the LP relaxation given in Eq. 8.22, as $T \to \infty$.

When the LP relaxation has a unique solution and it is integral, then $\overline{\boldsymbol{\mu}}$ is guaranteed to converge to the unique MAP assignment. In particular, this implies that there must exist a subgradient iteration when $\boldsymbol{\mu}^t$ corresponds to the MAP assignment. At this iteration the subgradient is zero, and we obtain a certificate of optimality by theorem 8.1. When the LP relaxation is not tight, recovering a primal solution may be helpful for finding additional constraints to use in tightening the relaxation.

8.7.2 Coordinate Descent

As mentioned in section 8.5, the coordinate descent algorithms, while always providing a monotonic improvement in the dual objective, are not in general guaranteed to solve the Lagrangian relaxation to optimality. However, for some graphical models, such as pairwise models with binary variables, fixed points of the coordinate descent algorithms can be used to construct a solution to the LP relaxation (Kolmogorov and Wainwright, 2005; Globerson and Jaakkola, 2008). Thus, for these graphical models, all fixed points of the algorithms are dual optimal.

We additionally show that when the LP relaxation has a unique solution that is integral, then the fixed point must be locally decodable to the MAP assignment. On the other hand, for more general graphical models, there do exist degenerate cases when a fixed point of the coordinate descent algorithm is *not* locally decodable, even if it corresponds to a dual solution.

Theorem 8.5. *Suppose the LP relaxation has a unique solution and it is*

integral.[12] *Then the following hold:*

1. *For binary pairwise MRFs, fixed points of the coordinate descent algorithms are locally decodable to* $\boldsymbol{x}^*$.

2. *For pairwise tree-structured MRFs, fixed points of the coordinate descent algorithms are locally decodable to* $\boldsymbol{x}^*$.

3. *There exist non-binary pairwise MRFs with cycles and dual optimal fixed points of the coordinate descent algorithms that are not locally decodable.*

The second result can be generalized to graphs with a single cycle. Despite the theoretical difficulty hinted at in the third result, empirically we have found that when the LP relaxation has a unique solution that is integral, local decoding nearly always finds the MAP assignment, when used with the coordinate descent algorithms (Sontag et al., 2008).

8.8 Discussion

The dual decomposition formulation that we introduced in this chapter is applicable whenever one can break an optimization problem into smaller subproblems that can be solved exactly by using combinatorial algorithms. The dual objective is particularly simple, consisting of a sum of maximizations over the individual subproblems, and provides an upper bound on the value of the MAP assignment. Minimizing the Lagrangian relaxation makes this bound tighter by pushing the subproblems to agree with one another on their maximizing assignment. If we succeed in finding an assignment that maximizes each of the individual subproblems, then this assignment is the MAP assignment and the dual provides a certificate of optimality. Even when not exact, the upper bound provided by the dual can be extremely valuable when used with a branch-and-bound procedure to find the MAP assignment (Geoffrion, 1974).

We described both subgradient methods and block coordinate algorithms for minimizing the Lagrangian relaxation. Both are notable for their simplicity and their effectiveness at solving real-world inference problems. A natural question is how to choose between the two for a specific problem. This question has been the subject of continual debate since the early work on Lagrangian relaxations (see Guignard, 2003, for a review).

Each approach has its advantages and disadvantages. The subgradient method is guaranteed to converge, but does not monotonically decrease

12. This implies that the MAP assignment $\boldsymbol{x}^*$ is unique.

the dual and requires a delicate choice of stepsize. The coordinate descent algorithms are typically much faster at minimizing the dual, especially in the initial iterations, and are also monotonic. However, to perform each update, one must compute max-marginals, and this may be impractical for some applications. We presented several coordinate update schemes. These vary both in the coordinates that they choose to minimize in each update, and in the way this minimization is performed. MPLP appears to be the best general choice.

It may be advantageous to use the two approaches together. For example, one could first use coordinate descent and then, when the algorithm is close to convergence, use the subgradient method to ensure global optimality. Another possibility is to alternate between the updates, as in the spacer step approach in optimization (see Bertsekas, 1995).

There are often many different ways to decompose a problem when applying dual decomposition. For example, in this chapter we suggested using a decomposition for pairwise models that uses one subproblem per edge. An alternative decomposition is to use a set of spanning trees that together cover all of the edge potentials (Wainwright et al., 2005). Still another approach is, before constructing the decomposition, to split some of the variables and introduce new potentials to enforce equality among all copies of a variable (Guignard, 2003). Using this technique, Yarkony et al. (2010) give a decomposition that uses a single tree spanning *all* of the original edge potentials. Although in all of these cases the decompositions correspond to the same LP relaxation, often a different decomposition can result in tighter or looser bounds being provided by the Lagrangian relaxation.

Dual decomposition methods have a wide range of potential applications in machine learning and, more broadly, engineering and the sciences. We have already observed some empirical successes for natural language processing, computational biology, and machine vision. We expect that as graphical models become more complex, techniques like the ones discussed in this chapter will become essential for performing fast and accurate inference.

Acknowledgments We thank Michael Collins, Ce Liu, and the editors for their very useful feedback, and also M. Collins for Fig. 8.1. This work was partly supported by BSF grant 2008303.

Appendix: Technical Details

8.9.1 Derivation of the Max-Sum Diffusion Updates

For a given factor f and variable i, assume all $\boldsymbol{\delta}$ variables are fixed except $\delta_{fi}(x_i)$. We would like to minimize $L(\boldsymbol{\delta})$ in Eq. 8.2 w.r.t. the free variables. Here we prove that the choice of $\delta_{fi}(x_i)$ in Eq. 8.17 corresponds to this minimization (other updates that achieve the same objective are possible since $L(\boldsymbol{\delta})$ is not strictly convex).

The part of the objective $L(\boldsymbol{\delta})$ that depends on the free parameters is

$$\begin{aligned}
&\max_{x_i}\Big(\theta_i(x_i)+\sum_{f:i\in f}\delta_{fi}(x_i)\Big)+\max_{\boldsymbol{x}_f}\Big(\theta_f(\boldsymbol{x}_f)-\sum_{\hat{i}\in f}\delta_{f\hat{i}}(x_{\hat{i}})\Big)\\
=\;&\max_{x_i}\Big(\delta_{fi}(x_i)+\delta_i^{-f}(x_i)\Big)+\max_{x_i}\Big(-\delta_{fi}(x_i)+\max_{\boldsymbol{x}_{f\setminus i}}\Big[\theta_f(\boldsymbol{x}_f)-\sum_{\hat{i}\in f\setminus i}\delta_{f\hat{i}}(x_{\hat{i}})\Big]\Big)\\
\geq\;&\max_{x_i}\Big(\delta_i^{-f}(x_i)+\max_{\boldsymbol{x}_{f\setminus i}}\Big[\theta_f(\boldsymbol{x}_f)-\sum_{\hat{i}\in f\setminus i}\delta_{f\hat{i}}(x_{\hat{i}})\Big]\Big)=\max_{x_i}g(x_i).
\end{aligned}$$

This lower bound can be achieved by choosing a $\delta_{fi}(x_i)$ such that

$$\begin{aligned}
\delta_{fi}(x_i)+\delta_i^{-f}(x_i) &= \tfrac{1}{2}g(x_i),\\
-\delta_{fi}(x_i)+\max_{\boldsymbol{x}_{f\setminus i}}\Big[\theta_f(\boldsymbol{x}_f)-\sum_{\hat{i}\in f\setminus i}\delta_{f\hat{i}}(x_{\hat{i}})\Big] &= \tfrac{1}{2}g(x_i).
\end{aligned}$$

Indeed, the $\delta_{fi}(x_i)$ given by the update in Eq. 8.17 satisfies the above.

8.9.2 Derivation of the MPLP Updates

For a given factor f assume all $\boldsymbol{\delta}$ variables are fixed except $\delta_{fi}(x_i)$ for all $i\in f$. We would like to minimize $L(\boldsymbol{\delta})$ in Eq. 8.2 w.r.t. the free variables. Here we prove that Eq. 8.18 does this optimally.

The part of the objective $L(\boldsymbol{\delta})$ that depends on the free parameters is

$$\bar{L}(\boldsymbol{\delta})=\sum_{i\in f}\max_{x_i}\Big(\theta_i(x_i)+\sum_{\hat{f}:i\in\hat{f}}\delta_{\hat{f}i}(x_i)\Big)+\max_{\boldsymbol{x}_f}\Big(\theta_f(\boldsymbol{x}_f)-\sum_{i\in f}\delta_{fi}(x_i)\Big). \quad (8.25)$$

Denote $A_i(\boldsymbol{\delta})=\max_{x_i}\Big(\theta_i(x_i)+\sum_{\hat{f}:i\in\hat{f}}\delta_{\hat{f}i}(x_i)\Big)$ and $A_f(\boldsymbol{\delta})=\max_{\boldsymbol{x}_f}\Big(\theta_f(\boldsymbol{x}_f)-\sum_{i\in f}\delta_{fi}(x_i)\Big)$. Then it follows that

$$\bar{L}(\boldsymbol{\delta})=\sum_{i\in f}A_i(\boldsymbol{\delta})+A_f(\boldsymbol{\delta})\geq\max_{\boldsymbol{x}_f}\Big(\theta_f(\boldsymbol{x}_f)+\sum_{i\in f}\delta_i^{-f}(x_i)\Big)=B.$$

This gives a lower bound B on the minimum of $\bar{L}(\boldsymbol{\delta})$. We next show that this lower bound is achieved by the MPLP update in Eq. 8.18. For each of the terms $A_i(\boldsymbol{\delta})$ we have (after the update)

$$\begin{aligned} A_i(\boldsymbol{\delta}) &= \max_{x_i} \frac{1}{|f|}\delta_i^{-f}(x_i) + \frac{1}{|f|}\max_{\boldsymbol{x}_{f\setminus i}}\left[\theta_f(\boldsymbol{x}_f) + \sum_{\hat{i}\in f\setminus i}\delta_{\hat{i}}^{-f}(x_{\hat{i}})\right] \\ &= \frac{1}{|f|}\max_{\boldsymbol{x}_f}\Big(\theta_f(\boldsymbol{x}_f) + \sum_{i\in f}\delta_i^{-f}(x_i)\Big) = \frac{B}{|f|}. \end{aligned} \tag{8.26}$$

The value of $A_f(\boldsymbol{\delta})$ after the MPLP update is

$$\begin{aligned} &\max_{\boldsymbol{x}_f}\Big(\theta_f(\boldsymbol{x}_f) + \sum_{i\in f}\Big\{\frac{|f|-1}{|f|}\delta_i^{-f}(x_i) - \frac{1}{|f|}\max_{\hat{\boldsymbol{x}}_{f\setminus i}}\Big[\theta_f(x_i,\hat{\boldsymbol{x}}_f) + \sum_{\hat{i}\in f\setminus i}\delta_{\hat{i}}^{-f}(\hat{x}_{\hat{i}})\Big]\Big\}\Big) \\ &= \frac{1}{|f|}\max_{\boldsymbol{x}_f}\Big(\sum_{i\in f}\Big\{\theta_f(\boldsymbol{x}_f) + \sum_{\hat{i}\in f\setminus i}\delta_{\hat{i}}^{-f}(x_{\hat{i}}) - \max_{\hat{\boldsymbol{x}}_{f\setminus i}}\Big[\theta_f(x_i,\hat{\boldsymbol{x}}_f) + \sum_{\hat{i}\in f\setminus i}\delta_{\hat{i}}^{-f}(\hat{x}_{\hat{i}})\Big]\Big\}\Big) \\ &\le \frac{1}{|f|}\sum_{i\in f}\max_{\boldsymbol{x}_f}\Big(\theta_f(\boldsymbol{x}_f) + \sum_{\hat{i}\in f\setminus i}\delta_{\hat{i}}^{-f}(x_{\hat{i}}) - \max_{\hat{\boldsymbol{x}}_{f\setminus i}}\Big[\theta_f(x_i,\hat{\boldsymbol{x}}_f) + \sum_{\hat{i}\in f\setminus i}\delta_{\hat{i}}^{-f}(\hat{x}_{\hat{i}})\Big]\Big) \\ &= \frac{1}{|f|}\sum_{i\in f}\max_{x_i}\Big(\max_{\boldsymbol{x}_{f\setminus i}}\Big[\theta_f(\boldsymbol{x}_f) + \sum_{\hat{i}\in f\setminus i}\delta_{\hat{i}}^{-f}(x_{\hat{i}})\Big] - \max_{\boldsymbol{x}_{f\setminus i}}\Big[\theta_f(\boldsymbol{x}_f) + \sum_{\hat{i}\in f\setminus i}\delta_{\hat{i}}^{-f}(x_{\hat{i}})\Big]\Big) \\ &= 0, \end{aligned}$$

where the first equality used $\sum_{i\in f}(|f|-1)\delta_i^{-f}(x_i) = \sum_{i\in f}\sum_{\hat{i}\in f\setminus i}\delta_{\hat{i}}^{-f}(x_{\hat{i}})$. From the above and Eq. 8.26 it follows that $\bar{L}(\boldsymbol{\delta}) \le B$. However, we know that B is a lower bound on $\bar{L}(\boldsymbol{\delta})$, so we must have $\bar{L}(\boldsymbol{\delta}) = B$, implying the optimality of the MPLP update.

8.9.3 Derivation of the Star Update

The MPLP update gives the closed form solution for block coordinate descent on coordinates $\delta_{\{i,j\}j}(x_j)$ and $\delta_{\{i,j\}i}(x_i)$ for a particular $\{i,j\} \in F$ (i.e., an edge update). We now wish to find the update where the free coordinates are $\delta_{\{i,j\}j}(x_j), \delta_{\{i,j\}i}(x_i)$ for a fixed i and all $j \in N(i)$. One way to find the optimal coordinates is by iterating the MPLP update for all the edges in the star until convergence. Notice that $\delta_j^{-i}(x_j)$, $\gamma_{ji}(x_i)$, and $\gamma_i(x_i)$ do not change after each edge update. We solve for $\delta_{\{i,j\}j}(x_j)$ and $\delta_{\{i,j\}i}(x_i)$ in terms of these quantities, using fixed point equations for MPLP.

The MPLP edge update for $f = \{i,j\}$ is given by

$$\delta_{\{i,j\}i}(x_i) = -\tfrac{1}{2}\delta_i^{-j}(x_i) + \tfrac{1}{2}\gamma_{ji}(x_i). \tag{8.27}$$

Multiplying by 2, and then subtracting $\delta_{\{i,j\}i}(x_i)$ from both sides, we obtain

$$\delta_{\{i,j\}i}(x_i) = -\delta_i(x_i) + \gamma_{ji}(x_i), \tag{8.28}$$

where we define $\delta_i(x_i) = \theta_i(x_i) + \sum_{j \in N(i)} \delta_{\{i,j\}i}(x_i)$. Summing this over all neighbors of i, and adding $\theta_i(x_i)$ to both sides, we obtain

$$\begin{aligned} \delta_i(x_i) &= -N_i \delta_i(x_i) + \gamma_i(x_i) \\ \delta_i(x_i) &= \frac{1}{1+N_i} \gamma_i(x_i). \end{aligned} \tag{8.29}$$

Applying Eq. 8.28 and Eq. 8.29, we obtain the following for $\delta_i^{-j}(x_i)$:

$$\delta_i^{-j}(x_i) = \delta_i(x_i) - \delta_{\{i,j\}i}(x_i) = 2\delta_i(x_i) - \gamma_{ji}(x_i) = \frac{2\gamma_i(x_i)}{1+N_i} - \gamma_{ji}(x_i) \ .$$

Substituting this back into the MPLP update (Eq. 8.27) yields the update for $\delta_{\{i,j\}i}(x_i)$ given in Eq. 8.20. The update for $\delta_{\{i,j\}j}(x_j)$ is obtained by taking the MPLP update and substituting the above expression for $\delta_i^{-j}(x_i)$.

The dual variables given by the star update in Eq. 8.20 can be seen to be a fixed point of MPLP for all edges in the star. Since any fixed point of MPLP on a tree is dual optimal (see section 8.7.2), these updates provide the optimum for these coordinates.

8.9.4 Proofs of the Decoding Results

Duality in linear programming specifies *complementary slackness* conditions that every primal and dual solution must satisfy. In particular, it can be shown that for any optimal $\boldsymbol{\mu}^*$ for the local LP relaxation given in Eq. 8.22 and any optimal $\boldsymbol{\delta}^*$ for the Lagrangian relaxation $\min_{\boldsymbol{\delta}} L(\boldsymbol{\delta})$:

$$\mu_i^*(x_i) > 0 \quad \Rightarrow \quad \bar{\theta}_i^{\boldsymbol{\delta}^*}(x_i) = \max_{\hat{x}_i} \bar{\theta}_i^{\boldsymbol{\delta}^*}(\hat{x}_i), \tag{8.30}$$

$$\mu_f^*(\boldsymbol{x}_f) > 0 \quad \Rightarrow \quad \bar{\theta}_f^{\boldsymbol{\delta}^*}(\boldsymbol{x}_f) = \max_{\hat{\boldsymbol{x}}_f} \bar{\theta}_f^{\boldsymbol{\delta}^*}(\hat{\boldsymbol{x}}_f). \tag{8.31}$$

Proof of theorem 8.2. Consider any $x_i \neq x_i^*$. Since $\bar{\theta}_i^{\boldsymbol{\delta}^*}(x_i) < \bar{\theta}_i^{\boldsymbol{\delta}^*}(x_i^*)$, complementary slackness (8.30) implies that $\mu_i^*(x_i) = 0$ for all optimal $\boldsymbol{\mu}^*$. Thus, the only solution to the LP relaxation corresponds to $\boldsymbol{x}^*$. □

Proof of theorem 8.3. This follows from strict complementary slackness (Vanderbei, 2007), which guarantees that a primal-dual pair $(\boldsymbol{\mu}^*, \boldsymbol{\delta}^*)$ exists that satisfies the implication in (8.30) both ways. Since the LP relaxation has only one solution, and it corresponds to the MAP assignment, strict complementary slackness guarantees that such a $\boldsymbol{\delta}^*$ is locally decodable. □

Proof of theorem 8.4. We reduce from 3SAT. First, we encode 3SAT as an

optimization problem of the form given in Eq. 8.1. The variables $x_i \in \{0, 1\}$ are the same as in the 3SAT formula. We have one factor $\theta_f(\boldsymbol{x}_f)$ for each clause f in the formula, defined on the corresponding three variables. $\theta_f(\boldsymbol{x}_f)$ is 0 if the clause is satisfied by $\boldsymbol{x}_f$, and $-\infty$ otherwise. $\theta_i(x_i) = 0$ for all i, x_i.

Suppose we could efficiently find an agreeing assignment when one exists. By theorem 8.1, this would be an MAP assignment (and the LP relaxation is tight), which in this case corresponds to a satisfying assignment of the 3SAT formula. Thus, if we could efficiently find an MAP assignment whenever the LP relaxation is tight, then we could efficiently solve 3SAT.

Finding the MAP assignment is hard even if it is unique, because the problem of finding a satisfying assignment when we are guaranteed that a formula has at most one satisfying assignment, called Unique-SAT, is also NP-hard, under randomized reductions (Valiant and Vazirani, 1985). □

For 3SAT, the LP relaxation always has a fractional solution $\boldsymbol{\mu} \in M_L$ with objective value 0. Thus, by theorem 8.2 the dual solutions will never be locally decodable. Let $\mu_i(0) = \mu_i(1) = .5$ for all i. For each clause f, if it is satisfied by both $\boldsymbol{x}_f^1 = (0, 0, 0)$ and $\boldsymbol{x}_f^2 = (1, 1, 1)$, then let $\mu_f(\boldsymbol{x}_f^1) = \mu_f(\boldsymbol{x}_f^2) = .5$ and $\mu_f(\boldsymbol{x}_f) = 0$ for $\boldsymbol{x}_f \neq \boldsymbol{x}_f^1, \boldsymbol{x}_f^2$. Otherwise, f must be satisfied by $\boldsymbol{x}_f^1 = (0, 1, 1)$ and $\boldsymbol{x}_f^2 = (1, 0, 0)$, so we set $\mu_f(\boldsymbol{x}_f^1) = \mu_f(\boldsymbol{x}_f^2) = .5$.

Proof of theorem 8.5, part 1. The claim follows from (Globerson and Jaakkola, 2008, proposition 4). This result shows how to construct a fractional primal solution whenever at least one of the nodes i has $\bar{\theta}_i^{\boldsymbol{\delta}}(0) = \bar{\theta}_i^{\boldsymbol{\delta}}(1)$, that is, when $\boldsymbol{\delta}$ is not locally decodable. However, this would contradict our assumption that the LP relaxation has a unique solution and it is integral. Thus, $\boldsymbol{\delta}$ must be locally decodable. □

All fixed points of the coordinate descent algorithms can be shown to satisfy *max-consistency*. Let A_i consist of all states $\hat{x}_i$ that maximize $\bar{\theta}_i^{\boldsymbol{\delta}}(x_i)$. By max-consistency, we mean that for all $f \in F, i \in f$, and $x_i \in A_i$, $\max_{\boldsymbol{x}_{f \setminus i}} \bar{\theta}_f^{\boldsymbol{\delta}}(\boldsymbol{x}_f) = \max_{\hat{\boldsymbol{x}}_f} \bar{\theta}_f^{\boldsymbol{\delta}}(\hat{\boldsymbol{x}}_f)$. This is trivial to see for MSD, since at a fixed point, $\bar{\theta}_i^{\boldsymbol{\delta}}(x_i) = \max_{\boldsymbol{x}_{f \setminus i}} \bar{\theta}_f^{\boldsymbol{\delta}}(\boldsymbol{x}_f)\ \forall f, i \in f$ and x_i. Thus, for $x_i \in A_i$, $\bar{\theta}_i^{\boldsymbol{\delta}}(x_i) = \max_{\hat{\boldsymbol{x}}_f} \bar{\theta}_f^{\boldsymbol{\delta}}(\hat{\boldsymbol{x}}_f)$. Putting these together shows max-consistency.

Proof of theorem 8.5, part 2. First, construct a reduced pairwise MRF with potentials $\boldsymbol{\theta}'$ where for each variable i we consider only those states x_i that maximize $\bar{\theta}_i^{\boldsymbol{\delta}}(x_i)$ (hereafter ignoring the other states). We let $\theta'_{ij}(\hat{x}_i, \hat{x}_j) = 0$ if $\hat{x}_i, \hat{x}_j$ maximize $\bar{\theta}_{ij}^{\boldsymbol{\delta}}(x_i, x_j)$, and $-\infty$ otherwise. By complementary slackness, all solutions of the LP relaxation for $\boldsymbol{\theta}'$ are also optimal for $\boldsymbol{\theta}$. By max-consistency, for every state x_i, $\exists x_j$ such that $\theta'(x_i, x_j) = 0$.

Suppose that there exist a vertex i and a state x'_i where $x'_i \neq x_i^*$ (that is, $\boldsymbol{\delta}$ is not locally decodable to $\boldsymbol{x}^*$). Then, by max-consistency we can construct

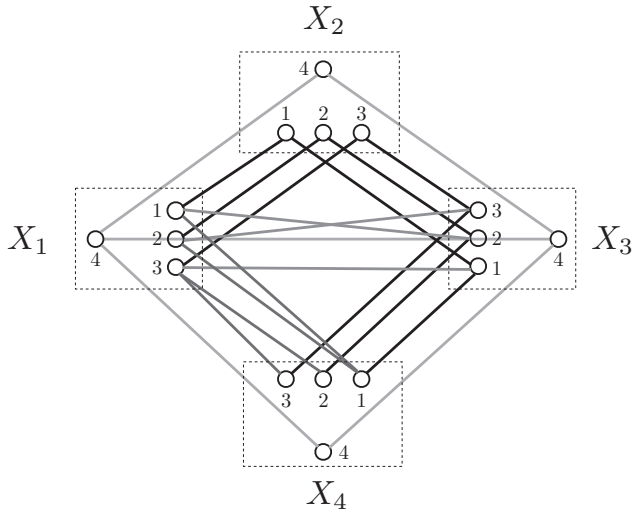

Figure 8.6: Illustration of the parameters of the pairwise MRF that we use in the proof of theorem 8.5, part 3. Each node represents a state $x_i \in \{1, 2, 3, 4\}$. An edge between x_i and x_j signifies that $\theta_{ij}(x_i, x_j) = 0$, whereas no edge between x_i and x_j signifies that $\theta_{ij}(x_i, x_j) = -1$.

an assignment (going one edge at a time in the tree rooted at i) $\boldsymbol{x}' \neq \boldsymbol{x}^*$ such that $\theta'(\boldsymbol{x}') = 0$. This shows that $\boldsymbol{x}'$ is an MAP assignment for $\boldsymbol{\theta}$. However, this contradicts the uniqueness of the MAP assignment. □

Proof of theorem 8.5, part 3. Consider a pairwise MRF on four variables with edges $E = \{(1, 2), (2, 3), (3, 4), (1, 4), (1, 3)\}$ and where each variable has four states. Let the parameters $\boldsymbol{\theta}$ be as specified in Fig. 8.6, and let the dual variables $\boldsymbol{\delta}$ be identically zero. Since the MAP assignment ($x_i = 4$ for all i) has value 0 and $L(\boldsymbol{\delta}) = 0$, $\boldsymbol{\delta}$ is dual optimal and the LP relaxation is tight. $\boldsymbol{\delta}$ is also a fixed point of the MPLP update given in Eq. 8.18.

What remains is to show that the LP relaxation has a *unique* solution (i.e., $x_i = 4$ for all i). First, note that for any primal optimal $\boldsymbol{\mu}^*$, $\mu^*_{ij}(x_i, x_j)$ must be 0 whenever $\theta_{ij}(x_i, x_j) = -1$. Subject to these constraints, we next show that $\mu^*_i(x_i)$ must be 0 for all i and $x_i \in \{1, 2, 3\}$. For $x \in \{1, 2, 3\}$, the local consistency constraints (see Eq. 8.23) along edges $(1, 2)$, $(2, 3)$ and $(3, 4)$ imply that $\mu^*_1(x) = \mu^*_{12}(x, x) = \mu^*_2(x) = \mu^*_{2,3}(x, x) = \mu^*_3(x) = \mu^*_{3,4}(x) = \mu^*_4(x)$. Similarly, the local consistency constraints along edge $(1, 3)$ imply that $\mu^*_1(1) = \mu^*_{1,3}(1, 2) = \mu^*_3(2)$ and $\mu^*_1(2) = \mu^*_{1,3}(2, 3) = \mu^*_3(3)$. Together, these imply that $\mu^*_i(x_i) = a$ for all i and $x_i \in \{1, 2, 3\}$. Now consider the local consistency constraints for edge $(1, 4)$. One of the constraints is that $\mu^*_1(3) = \mu^*_{1,4}(3, 3) + \mu^*_{1,4}(3, 2) = \mu^*_4(3) + \mu^*_4(2)$. Thus, we must have that $a = 2a$, which implies that $a = 0$. □

Note that there do exist locally decodable solutions for the example given

in Fig. 8.6. In particular, consider $\boldsymbol{\delta}$ defined as $\delta_{\{i,j\}i}(4) = \delta_{\{i,j\}j}(4) = 0$ and $\delta_{\{i,j\}i}(x) = \delta_{\{i,j\}j}(x) = \epsilon$ for $x \in \{1, 2, 3\}$. When $-.5 \leq \epsilon < 0$, $\boldsymbol{\delta}$ is dual optimal, locally decodable, and a fixed point of MPLP.

8.10 References

K. M. Anstreicher and L. A. Wolsey. Two "well-known" properties of subgradient optimization. *Mathematical Programming*, 120(1):213–220, 2009.

D. P. Bertsekas. Auction algorithms for network flow problems: A tutorial introduction. *Computational Optimization and Applications*, 1:7–66, 1992.

D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA, 1995.

J. Duchi, D. Tarlow, G. Elidan, and D. Koller. Using combinatorial optimization within max-product belief propagation. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 369–376. MIT Press, 2007.

D. Erlenkotter. A dual-based procedure for uncapacitated facility location. *Operations Research*, 26(6):992–1009, 1978.

M. L. Fisher. The lagrangian relaxation method for solving integer programming problems. *Management Science*, 27(1):1–18, 1981.

A. M. Geoffrion. Lagrangean relaxation for integer programming. *Mathematical Programming Study*, 2:82–114, 1974.

A. Globerson and T. Jaakkola. Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 553–560. MIT Press, Cambridge, MA, 2008.

M. Guignard. Lagrangean relaxation. *TOP: An Official Journal of the Spanish Society of Statistics and Operations Research*, 11(2):151–200, 2003.

M. Guignard and S. Kim. Lagrangean decomposition: A model yielding stronger Lagrangean bounds. *Mathematical Programming*, 39(2):215–228, 1987.

M. Guignard and M. Rosenwein. An application-oriented guide for designing lagrangean dual ascent algorithms. *European Journal of Operational Research*, 43(2):197–205, 1989.

R. Gupta, A. Diwan, and S. Sarawagi. Efficient inference with cardinality-based clique potentials. In *Proceedings of the 24th International Conference on Machine Learning*, pages 329–336. ACM Press, New York, 2007.

T. Hazan and A. Shashua. Norm-product belief propagation: Primal-dual message-passing for approximate inference. *IEEE Transactions on Information Theory*, 56(12):6294–6316, 2010.

M. Held, P. Wolfe, and H. Crowder. Validation of subgradient optimization. *Mathematical Programming*, 6(1):62–88, 1974.

J. Johnson. *Convex Relaxation Methods for Graphical Models: Lagrangian and Maximum Entropy Approaches*. PhD thesis, Department of Electrical Engineering and Computer Science, MIT, 2008.

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

V. Kolmogorov and M. Wainwright. On the optimality of tree-reweighted max-product message-passing. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*, pages 316–323. AUAI Press, Arlington, VA, 2005.

N. Komodakis, N. Paragios, and G. Tziritas. MRF energy minimization and beyond via dual decomposition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(3):531–552, 2011.

T. Koo, A. M. Rush, M. Collins, T. Jaakkola, and D. Sontag. Dual decomposition for parsing with non-projective head automata. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1288–1298, 2010.

V. A. Kovalevsky and V. K. Koval. A diffusion algorithm for decreasing the energy of the max-sum labeling problem. Unpublished, Glushkov Institute of Cybernetics, Kiev, USSR, circa 1975. Personally communicated to T. Werner by M. I. Schlesinger.

S. Lacoste-Julien, B. Taskar, D. Klein, and M. I. Jordan. Word alignment via quadratic assignment. In R. C. Moore, J. A. Bilmes, J. Chu-Carroll, and M. Sanderson, editors, *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 112–119. The Association for Computational Linguistics, New York, 2006.

C. Lemaréchal. Lagrangian relaxation. In *Computational Combinatorial Optimization*, pages 112–156. Berlin, Springer, 2001.

R. McDonald and G. Satta. On the complexity of non-projective data-driven dependency parsing. In *Proceedings of the 10th International Conference on Parsing Technologies*, pages 121–132. Association for Computational Linguistics, Morristown, NJ, 2007.

R. McDonald, F. Pereira, K. Ribarov, and J. Hajic. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 523–530, 2005.

A. Nedić and A. Ozdaglar. Approximate primal solutions and rate analysis for dual subgradient methods. *SIAM Journal on Optimization*, 19(4):1757–1780, 2009.

C. Rother, P. Kohli, W. Feng, and J. Jia. Minimizing sparse higher order energy functions of discrete variables. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1382–1389, 2009.

M. I. Schlesinger. Syntactic analysis of two-dimensional visual signals in noisy conditions. *Kibernetika*, 4:113–130, 1976. in Russian.

N. Z. Shor. *Minimization Methods for Non-Differentiable Functions*. Springer-Verlag, New York, NY, USA, 1985.

D. Sontag. *Approximate Inference in Graphical Models using LP Relaxations*. PhD thesis, Department of Electrical Engineering and Computer Science, MIT, 2010.

D. Sontag and T. Jaakkola. Tree block coordinate descent for MAP in graphical models. In *Proceedings of the 12th International Workshop on Artificial Intelligence and Statistics*, volume 9, pages 544–551. JMLR: W&CP, 2009.

D. Sontag, T. Meltzer, A. Globerson, T. Jaakkola, and Y. Weiss. Tightening LP relaxations for MAP using message passing. In *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence*, pages 503–510. AUAI Press, Arlington, VA, 2008.

D. Tarlow, I. Givoni, and R. Zemel. HOP-MAP: Efficient message passing with high order potentials. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, volume 9, pages 812–819. JMLR: W&CP, 2010.

L. G. Valiant and V. V. Vazirani. NP is as easy as detecting unique solutions. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing*, pages 458–463, New York, 1985. ACM Press.

R. Vanderbei. *Linear Programming: Foundations and Extensions*. Springer, 3rd edition, 2007.

M. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008.

M. Wainwright, T. Jaakkola, and A. Willsky. Tree-based reparameterization framework for analysis of sum-product and related algorithms. *IEEE Transactions on Information Theory*, 49(5):1120–1146, 2003.

M. Wainwright, T. Jaakkola, and A. Willsky. MAP estimation via agreement on trees: message-passing and linear programming. *IEEE Transactions on Information Theory*, 51(11):3697–3717, 2005.

Y. Weiss, C. Yanover, and T. Meltzer. MAP estimation, linear programming and belief propagation with convex free energies. In *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence*, pages 416–425. AUAI Press, Arlington, VA, 2007.

T. Werner. A linear programming approach to max-sum problem: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(7):1165–1179, 2007.

T. Werner. High-arity interactions, polyhedral relaxations, and cutting plane algorithm for soft constraint optimisation (MAP-MRF). In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

C. Yanover, T. Meltzer, and Y. Weiss. Linear programming relaxations and belief propagation – an empirical study. *Journal of Machine Learning Research*, 7: 1887–1907, 2006.

C. Yanover, O. Schueler-Furman, and Y. Weiss. Minimizing and learning energy functions for side-chain prediction. *Journal of Computational Biology*, 15(7): 899–911, 2008.

J. Yarkony, C. Fowlkes, and A. Ihler. Covering trees and lower-bounds on quadratic assignment. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2010.