(Hyper)-Graphs Inference via Convex Relaxations and Move Making Algorithms: Contributions and Applications in artificial vision

Nikos Komodakis, Pawan Kumar, Nikos Paragios

(Hyper)-Graphs Inference via Convex Relaxations and Move Making Algorithms: Contributions and Applications in artificial vision

Pawan Kumar, Nikos Komodakis, Nikos Paragios

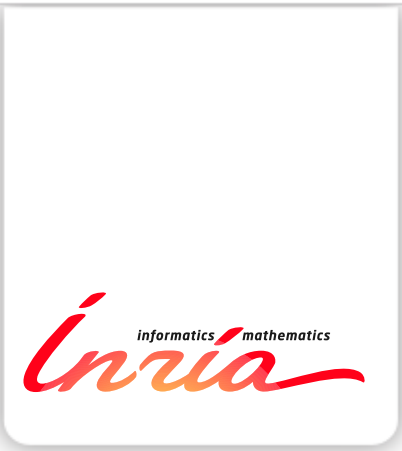
RESEARCH REPORT

N° 8798

October 2015

Project-Teams GALEN

ISSN 0249-6399 ISRN INRIA/RR--8798--FR+ENG



(Hyper)-Graphs Inference via Convex Relaxations and Move Making Algorithms: Contributions and Applications in artificial vision

Pawan Kumar[*†], Nikos Komodakis[*‡], Nikos Paragios[§]

Project-Teams GALEN

Research Report n° 8798 — October 2015 — 62 pages

Abstract: Computational visual perception seeks to reproduce human vision through the combination of visual sensors, artificial intelligence and computing. To this end, computer vision tasks are often reformulated as mathematical inference problems where the objective is to determine the set of parameters corresponding to the lowest potential of a task-specific objective function. Graphical models have been the most popular formulation in the field over the past two decades where the problem is viewed as an discrete assignment labeling one. Modularity, scalability and portability are the main strength of these methods which once combined with efficient inference algorithms they could lead to state of the art results. In this tutorial we focus on the inference component of the problem and in particular we discuss in a systematic manner the most commonly used optimization principles in the context of graphical models. Our study concerns inference over low rank models (interactions between variables are constrained to pairs) as well as as higher order ones (arbitrary set of variables determine hyper-cliques on which constraints are introduced) and seeks a concise, self-contained presentation of prior art as well as the presentation of the current state of the art methods in the field.

Key-words: MRFs, optimization, computer vision, convex programming, linear programming

* Equally contributing authors
† Oxford University, UK.
‡ Universite Paris-Est, École des Ponts ParisTech, France.
§ CentraleSupelec, Inria, University of Paris Saclay, France.

RESEARCH CENTRE
SACLAY – ÎLE-DE-FRANCE

Parc Orsay Université
4 rue Jacques Monod
91893 Orsay Cedex

Inférence dans les (hyper)-graphes à laide de relaxations convexes et dalgorithmes de coups optimisés : contributions et applications en vision par ordinateur

Résumé : La perception numérique visuelle cherche à reproduire la vision humaine grâce à une combinaison de senseurs visuels, dintelligence artificielle et de calcul numérique. Dans ce but, les problèmes de vision numériques sont souvent posés comme des problèmes dinférence mathématiques, dans lesquels lobjectif est de déterminer lensemble de paramètres correspondant au minimum dune énergie adaptée à la tâche visuelle. Les modèles graphiques ont constitué loutil de modélisation le plus populaire du domaine de ces deux dernières décennies ; le problème y est vu comme un problème dassignation de labels discrets. La modularité, lextensibilité et la portabilité sont les atouts majeurs de ces modèles, qui combinées à des méthodes dinférence efficaces peuvent mener à létat de lart en matière de résultats. Dans ce tutoriel nous nous focaliseront sur le problème dinférence ; en particulier, nous discuterons de façon systématique les schémas doptimisation les plus utilisés dans le contexte des modèles graphiques. Notre étude concerne linférence sur des modèles de rang faible (où les interactions entre les variables sont limités aux paires), ainsi que les modèles de rang supérieur (où des sous-ensemble arbitraires de variables déterminent des hyper-cliques sur lesquels des contraintes peuvent être introduites) et vise à présenter un aperçu concis et autonome des méthodes éprouvées et à létat de lart du domaine.

Mots-clés : MRFs, optimization, computer vision, convex programming, linear programming

1 Introduction

Graphical models (Conditional Random Fields (CRFs) or Markov Random Fields (MRFs)) have been introduced in the field of computer vision almost four decades ago. CRFs were introduced in [15] while MRFs were introduced in [18] to address the problem of image restoration. The central idea is to express perception through an inference problem over graph. The process is defined using a set of nodes, a set of labels, and a neighborhood system. The graph nodes often correspond to the parameters to be determined, the labels to a quantized/discrete version of the search space and the connectivity of the graph to the constraints/interactions between variables. Graph-based methods are endowed with numerous advantages as it concerns inference when compared to their alternative that refers to continuous formulations. These methods are in general gradient-free and therefore can easily accommodate changes of the model (graph structure), changes of the objective function (perception task), or changes of the discretization space (precision).

Early approaches to address graph-based optimization in the field of computer vision were primarily based either on annealing like approaches or on local minimum update principles. Simulated annealing was an alternative direction that provides in theory good guarantees as it concerns the optimality properties of the obtained solution. The central idea is to perform a search with a decreasing radius/temperature where at a given iteration the current state is updated to a new state with a tolerance (as it concerns the objective function) that is related to the temperature. Such meta-heuristic methods could lead to a good approximation of the optimal solution if temperature/radius are appropriately defined that in general is not that trivial. Iterated conditional modes or highest confidence first were among the first attempts exploiting local minimum iterative principles. Their underlying principle was to solve the problem progressively through a repetitive local update of the optimal solution towards a new local optimum. These methods were computationally efficient and deterministic in the expense of quite inefficient in terms of capturing the global optimum of the solution and the complete absences of guarantee as it concerns the optimality properties of the obtained solution.

Despite the elegance, modularity and scalability of MRFs/CRFs, their adoption was quite limited (over eighties and nineties) from the image processing/computer vision community and beyond due lack of efficient optimization methods to address their inference. The introduction of efficient inference algorithms inspired from the networks community, like for example the max flow/min cut principle at late nineties that is a special case of the duality theorem for linear programs as well their efficient implementations towards taking advantage of image like graphs [6] or message passing methods [50] that are based on the calculation of the marginal for a given node given the states of the other nodes have re-introduced graphical models in the field of computer vision. During the past two decades we have witnessed a tremendous progress both on their use to address visual perception tasks [63] as well as it concerns their inference. This tutorial aims to provide an overview of the state of the art methods in the field for inference as well as the most recent advances in that direction using move making algorithms and convex relations. The reminder of this paper is organized as follows; Section 2 presents briefly the context and a short review of the the most representative inference methods. Section 3 is dedicated to move making algorithms, while section 4 presents efficient linear programming-inspired principles for graph inference. The last section introduces dual decomposition, a generic, modular and scalable framework to perform (hyper) graph inference.

2 Mathematical background: basic tools for MRF inference

2.1 Markov random fields

A Markov random field (MRF) consists of a set of random variables $\mathbf{X} = \{X_p, p \in \mathbf{V}\}$, where $\mathbf{V}$ contains the set of indices of the n random variables. Each random variable can take one value (or label) from a set of labels $\mathbf{L}$. In this manuscript, we are primarily interested in discrete MRFs, where the label set consists of a finite and discrete set of labels, that is, $\mathbf{L} = \{l_1, \cdots, l_h\}$. In addition, an MRF also defines a neighborhood relationship $\mathbf{E}$ over the random variables, that is, $(p, q) \in \mathbf{E}$ if and only if X_p and X_q are neighboring random variables. Visually, an MRF can be presented by a graph $(\mathbf{V}, \mathbf{E})$ whose vertices correspond to random variables and whose edges connected neighboring random variables.

It would be convenient to introduce the notation $\mathcal{N}(p)$ to denote the set of all neighbors of the random variable X_p. In other words, $q \in \mathcal{N}(p)$ if and only if $(p, q) \in \mathbf{E}$. The neighborhood relationship defines a family of cliques $\mathcal{C}$ that consist of sets $\mathbf{C} \subseteq \mathbf{V}$ such that any two random variables that belong to the same clique are neighbors of each other.

We refer to an assignment of values $\mathbf{x} \in \mathbf{L}^n$ of all the random variables as a labeling. In order to quantitatively distinguish among the h^n possible labelings, we define an energy function which consists of a sum of clique potentials. Formally, the energy $E(\mathbf{x}; \boldsymbol{\theta})$ of a labeling $\mathbf{x}$ is defined as follows:

$$E(\mathbf{x}; \boldsymbol{\theta}) = \sum_{\mathbf{C} \in \mathcal{C}} \theta_{\mathbf{C}}(\mathbf{x}_{\mathbf{C}}). \tag{1}$$

Here $\theta_{\mathbf{C}}(\mathbf{x}_{\mathbf{C}})$ is the clique potential that only depends on the labels $\mathbf{x}_{\mathbf{C}}$ of the subset of variables indexed by $\mathbf{C} \in \mathbf{V}$. The term $\boldsymbol{\theta}$ is referred to as the parameters of the MRF.

The probability of a labeling is said to be proportional to the negative exponential of its energy. In other words, the lower the energy, the more probable the labeling. Thus, in order to estimate the most probable labeling, we need to solve the following *energy minimization* problem:

$$\mathbf{x}^* = \operatorname*{argmin}_{\mathbf{x} \in \mathbf{L}^n} E(\mathbf{x}; \boldsymbol{\theta}). \tag{2}$$

The above problem is also referred to as *maximum a posteriori* inference. As a shorthand, we will simply refer to it as inference. We note here that there also exist other types of inference problems, such as computing the marginal probability of a subset of random variables. However, these problems are out of scope for the current manuscript.

An important special case of MRFs that is widely studied in the literature is that of pairwise MRFs, which consists of two types of clique potentials. First, unary potentials $\theta_p(x_p)$, which depend on the label of one random variable $p \in \mathbf{V}$. Second, pairwise potentials $\theta_{pq}(x_p, x_q)$, which depend on the labels of two neighboring random variables X_p and X_q. In other words, given a pairwise MRF, the energy of a labeling $\mathbf{x}$ is specified as follows:

$$E(\mathbf{x}; \boldsymbol{\theta}) = \sum_{p \in \mathbf{V}} \theta_p(x_p) + \sum_{(p,q) \in \mathbf{E}} \theta_{pq}(x_p, x_q). \tag{3}$$

Note that, even with the restriction imposed by pairwise MRFs on the form of the energy function, inference remains an NP-hard problem. However, there are some special cases that admit more efficient exact algorithms, which we will briefly describe in the remainder of the chapter. These algorithms will form the building blocks for the more complex state of the art inference algorithms for both pairwise and general high-order MRFs in subsequent chapters.

2.2 Reparameterization

Before we delve into the details of some well-known exact inference algorithms for special cases of pairwise MRFs, we first need to define the important concept of reparameterization. Two sets of parmeters $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$ are said to be reparameterizations of each other if they specify the same energy value for all possible labelings, that is,

$$E(\mathbf{x};\boldsymbol{\theta}) = E(\mathbf{x};\boldsymbol{\theta}'), \forall \mathbf{x} \in \mathbf{L}^n. \tag{4}$$

For pairwise MRFs, a sufficient condition for two parameters $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$ to be reparameterizations of each other is that there exist scalars $M_{pq}(l_j)$ and $M_{qp}(l_i)$ for all $(p,q) \in \mathbf{E}$ and $l_i, l_j \in \mathbf{L}$ such that

$$\begin{aligned} \theta'_{pq}(l_i,l_j) &= \theta_{pq}(l_i,l_j) - M_{pq}(l_j) - M_{qp}(l_i), \\ \theta'_{p}(l_i) &= \theta_{p}(l_i) + \sum_{q \in \mathcal{N}(p)} M_{qp}(l_i). \end{aligned} \tag{5}$$

Interestingly, the above equation is also a necessary condition for reparameterization [29]. However, for the purposes of this manuscript, it is sufficient to understand why the above condition is sufficient for reparameterization. Intuitively, the unary potential $\theta'_p(l_i)$ introduces an extra penalty $M_{qp}(l_i)$ for assinging the label l_i to p. However, since the random variable q also needs to be assigned exactly one label, say l_j, this extra penalty is canceled out in the corresponding pairwise potential $\theta'_{pq}(l_i,l_j)$. This ensures that $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$ are reparameterizations of each other. The concept of reparametrization is fundamental to the design of inference algorithms. Specifically, most inference algorithms can be viewed as a series of reparameterizations of the given MRF such that the resulting set of parameters make it easy to minimize the energy over all possible labelings.

2.3 Dynamic programming

We are now ready to describe our first inference algorithm—dynamic programming—that is exact for tree-structured MRFs. In other words, if we visualize an MRF as a graph $(\mathbf{V},\mathbf{E})$, then it has to be singly connected (that is, without any cycles or loops).

It would be helpful to first consider a simple chain MRF defined over n random variables $\mathbf{V} = \{1,\cdots,n\}$ such that $\mathbf{E} = \{(p,p+1), p = 1,\cdots,n-1\}$. In other words, two random variables p and $p+1$ are neighbors of each other, thus forming a graph that resembles a chain. A visual representation of an example chain MRF over four random variables along with the corresponding potential values is provided in Figure 1. Inference on a chain MRF can be specified as the following optimization problem:

$$\begin{aligned} & \min_{\mathbf{x}\in\mathbf{L}^n} \sum_{p\in\mathbf{V}} \theta_p(x_p) + \sum_{(p,q)\in\mathbf{E}} \theta_{pq}(x_p,x_q) \\ = \quad & \min_{\mathbf{x}\in\mathbf{L}^n} \left(\theta_1(x_1) + \theta_{12}(x_1,x_2) + \right. \\ & \left. \sum_{p\in\mathbf{V}\setminus\{1\}} \theta_p(x_p) + \sum_{(p,q)\in\mathbf{E}\setminus\{(1,2)\}} \theta_{pq}(x_p,x_q)\right). \end{aligned} \tag{6}$$

Here, the operator '\' represents the set difference. The above reformulation of inference explicitly considers all the terms that are dependent on the label x_1. In order to solve the above problem, we will obtain a clever reparameterization that reduces it to an equivalent problem defined on a

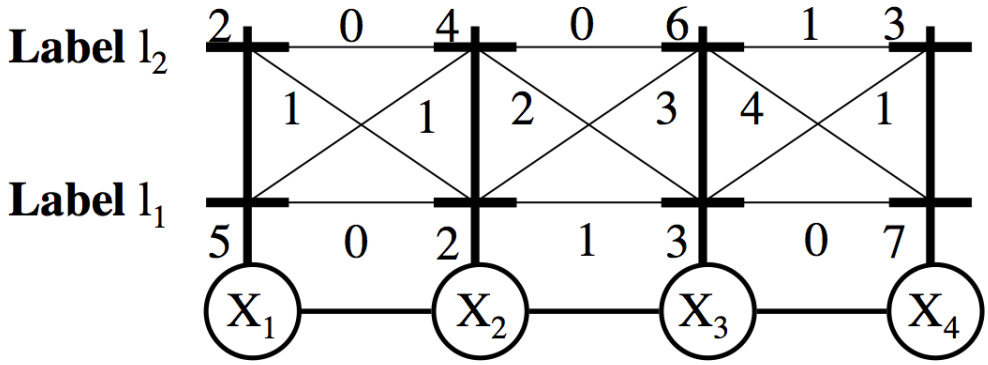


Figure 1: *An example chain MRF over four random variables* $\mathbf{X} = \{X_1, X_2, X_3, X_4\}$*, that is,* $\mathbf{V} = \{1, 2, 3, 4\}$*. Each random variable is depicted as an unfilled circle and can take one label from the set* $\mathbf{L} = \{l_1, l_2\}$*. The labels are shown as branches on the trellises on top of the corresponding random variable. The unary potential value* $\theta_p(l_i)$ *is shown next to the* i*-th branch of the random variable* X_p*. For example,* $\theta_1(l_1) = 5$ *and* $\theta_2(l_2) = 4$*. Similarly, the pairwise potential value* $\theta_{pq}(l_i, l_j)$ *is shown next to the connection between the* i*-th branch of* X_p *and the* j*-th branch of* X_q *respectively. For example,* $\theta_{12}(l_1, l_2) = 1$ *and* $\theta_{34}(l_2, l_1) = 4$*.*

chain MRF with $n-1$ random variables. To this end, we choose the following reparameterization constants:

$$M_{12}(l_j) = \min_{l_i \in \mathbf{L}} \theta_1(l_i) + \theta_{12}(l_i, l_j). \tag{7}$$

Using the above constants, we obtain a reparameterization $\boldsymbol{\theta}'$ of the original parameters $\boldsymbol{\theta}$ as follows:

$$\begin{aligned}
&\theta'_2(l_j) = \theta_2(l_j) + M_{12}(l_j), \forall l_j \in \mathbf{L}, \\
&\theta'_{12}(l_i, l_j) = \theta_{12}(l_i, l_j) - M_{12}(l_j), \forall l_i, l_j \in \mathbf{L}, \\
&\theta'_p(l_i) = \theta_p(l_i), \forall p \in \mathbf{V} \backslash \{2\}, l_i \in \mathbf{L}, \\
&\theta'_{pq}(l_i, l_j) = \theta_{pq}(l_i, l_j), \forall (p, q) \in \mathbf{E} \backslash \{(1, 2)\}, l_i, l_j \in \mathbf{L}.
\end{aligned} \tag{8}$$

In other words, we modify the unary potential of the second random variable by adding the corresponding reparameterization constants and the pairwise potential between the first and the second random variable by subtracting the corresponding reparametrization constants. The remaining potentials remain unchanged. The total time complexity of obtaining the above reparameterization is $O(h^2)$ since we need to compute $O(h)$ constants, each of which takes $O(h)$ time to compute using equation (7). The advantage of the above reparameterization is that for all values of $l_j \in \mathbf{L}$ the following holds true:

$$\min_{l_i} \theta'_1(l_i) + \theta'_{12}(l_i, l_j) = 0. \tag{9}$$

In other words, for any choice of label l_j, we can choose a label l_i such that the contribution of the potentials that depend on l_i is 0. Note that this since we are interested in minimizing the energy over all possible labeling, this choice of l_i will be an optimal one as other choices cannot contribute a negative term to the energy. Using this fact, we can rewrite the inference

problem (6) as follows:

$$\begin{aligned} & \min_{\mathbf{x}\in\mathbf{L}^n} (\theta'_1(x_1) + \theta'_{12}(x_1, x_2) + \\ & \qquad \left. \sum_{p\in\mathbf{V}\setminus\{1\}} \theta'_p(x_p) + \sum_{(p,q)\in\mathbf{E}\setminus\{(1,2)\}} \theta'_{pq}(x_p, x_q) \right) \\ = & \quad \min_{\mathbf{x}\in\mathbf{L}^{n-1}} \sum_{p\in\mathbf{V}\setminus\{1\}} \theta'_p(x_p) + \sum_{(p,q)\in\mathbf{E}\setminus\{(1,2)\}} \theta'_{pq}(x_p, x_q). \end{aligned}$$

The first problem is equivalent to problem (6) since $\boldsymbol{\theta}'$ is a reparameterization of $\boldsymbol{\theta}$. The second problem is equivalent to the first problem since the optimum choice of x_1 for any value of x_2 provides a contribution of 0 to the energy function.

The above argument shows that it is possible to reduce an inference problem on a chain MRF with n random variables to a chain MRF with $n-1$ random variables in $O(h^2)$ time. Taking this argument forward, we can start from one end of the chain and move to the other end in $n-1$ steps. At step p, we can reparameterize the MRF by using constants:

$$M_{pq}(l_j) = \min_{l_i\in\mathbf{L}} \hat{\theta}_p(l_i) + \hat{\theta}_{pq}(l_i, l_j), \tag{10}$$

where $q = p+1$ and $\hat{\boldsymbol{\theta}}$ is the current set of parameters. At the end of step $n-1$, we obtain a reparameterization $\overline{\boldsymbol{\theta}}$ that corresponds to a problem of a chain MRF of size 1. In other words, the energy of the optimum labeling can be computed as $\min_{l\in\mathbf{L}} \overline{\theta}_n(l)$. The total time complexity of the algorithm is $O(nh^2)$, since each step has a complexity of $O(h^2)$ and there are $O(n)$ steps in total.

Note that we can not only compute the minimum energy over all possible labelings, but also an optimal labeling itself. To achieve this, we simply need to keep track of the label x_p that is the optimal for every label x_{p+1} of the random variable $p+1$, that is, the label l_i that is used to compute the reparameterization constant in equation (10). At the end of step $n-1$, we can compute the optimal label of the variable n as $x^*_n = \operatorname{argmin}_{l\in\mathbf{L}} \overline{\theta}_n(l)$, and then backtrack to obtain the optimal label of all the random variables. Figure 2 shows the steps of dynamic programming for the chain MRF depicted in Figure 1.

In summary, dynamic programming for a chain MRF starts from one end of the chain and moves to the other end. At each step, it reparameterizes the current edge (p, q), where the constants are computed using equation (10). This algorithm can be extended to the more general tree-structured MRFs using the same technique of reparameterization. The key observation is that the sequence of reparameterization would proceed from the leaf random variables to the root random variable, where once again the reparameterization constants are computed using equation (10). Once we reach the root random variable X_n, we can compute the minimum energy as $\min_{l\in\mathbf{L}} \overline{\theta}_n(l)$, where $\overline{\boldsymbol{\theta}}$ is the final reparameterization obtained by dynamic programming. Similar to the chain MRF case, we can also obtain the optimal labeling by keeping track of the optimal label to assign to a child random variable for each possible label of its parent random variable. Note that since we have made no assumptions regarding the form of the pairwise potentials, the reparameterization constant for each edge take $O(h^2)$ time to compute. However, for many special cases of pairwise potentials, the computation of reparameterization constants can be significantly speeded up. We refer the interested reader to [14] for details.

2.4 Message passing and belief propagation

The above description of dynamic programming suggests an implementation based on reparameterizing the edges of a given tree-structured MRF. An alternative way to view the above algorithm

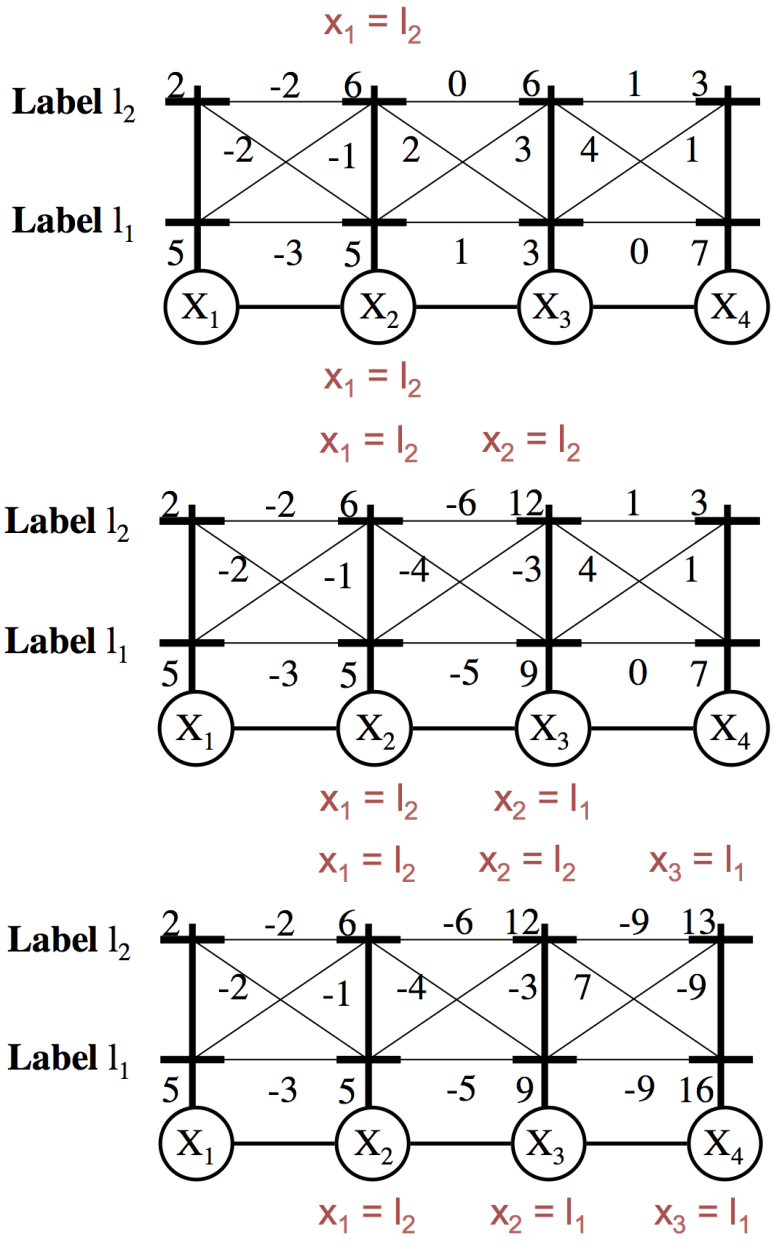


Figure 2: *The three steps of reparameterization required by the dynamic programming algorithm to perform inference on the chain MRF depicted in Figure 1. In the first step (top row), reparameterization affects the unary potentials of X_2 and the pairwise potentials of (X_1, X_2). The reparameterization constants are computed using equation (10). The figure also shows the optimal label of X_1 for the label l_1 (at the top) and the label l_2 (at the bottom) of X_2. This information is used to compute the optimal labeling of the chain MRF. Similarly, the reparameterization corresponding to (X_2, X_3) and (X_3, X_4) is shown in the middle and bottom row respectively. After three iterations, we can determine the optimal label for X_4 as l_2, which implies that the optimal label for X_3, X_2 and X_1 is l_1, l_1 and l_2 respectively. The energy of the optimal labeling is 13.*

is via iterative *message passing* [50]. At each iteration, a random variable p passes a message to each of its neighbors $q \in \mathcal{N}(p)$ (one message per neighbor). The message that p passes to q is a vector $\mathbf{m}_{pq}$ of size $h = |\mathbf{L}|$. In other words, it contains one scalar for each label l_j, which is specified as follows:

$$\mathbf{m}_{pq}(l_j) = \min_{l_i \in \mathbf{L}} \left(\theta_p(l_i) + \theta_{pq}(l_i, l_j) + \sum_{r \in \mathcal{N}(p) \setminus \{q\}} \mathbf{m}_{rp}(l_i) \right) + \eta_1. \tag{11}$$

Here, η_1 is a constant that is used to prevent numerical overflow or underflow. The messages are used to compute the beliefs of a random variable p in a label l_i as follows:

$$bel_p(l_i) = \theta_p(l_i) + \sum_{q \in \mathcal{N}(p)} \mathbf{m}_{qp}(l_i) + \eta_2. \tag{12}$$

Again, η_2 is a constant that prevents numerical instability. Given a tree-structured MRF, we can choose to pass the messages starting from the leaf random variables and moving up to the root random variable. In this case, it can be shown that the above message passing algorithm is equivalent to the reparameterization based dynamic programming algorithm described in the previous section [29]. Interestingly, [50] proved that the following generalization of the algorithm can also be used to obtain the optimal labeling of a tree-structured MRF: (i) pass messages in any arbitrary order; and (ii) terminate the algorithm when the messages do not changes from one iteration to the next. At convergence, the optimal label of each random variable p can be obtained as $x_p^* = \text{argmin}_{l \in \mathbf{L}}\, bel_p(l)$.

The above message passing algorithm, commonly referred to as belief propagation, is guaranteed to be optimal for a tree-structured MRF. However, for a general MRF, belief propagation is not even guaranteed to converge in a finite number of iterations. Nonetheless, it is commonly used as an approximate inference algorithm in practice [45].

2.5 Graph cuts

The previous two sections describe an efficient exact inference algorithm for tree-structured MRFs. Another special case that admits an efficient exact algorithm is when the energy function is submodular. Formally, a pairwise energy function $E(\mathbf{x}; \boldsymbol{\theta})$ is submodular if the pairwise potentials for all $(p, q) \in \mathbf{E}$ satisfy the following inequality:

$$\theta_{pq}(l_i, l_{i+1}) + \theta_{pq}(l_j, l_{j+1}) \leq \theta_{pq}(l_i, l_{j+1}) + \theta_{pq}(l_j, l_{i+1}), \forall i, j \in \{1, \cdots, h-1\}. \tag{13}$$

Note that the submodularity condition places no restrictions on the form of the unary potentials. Specifically, the minimization of a submodular energy function over all possible labelings can be mapped to an equivalent *st*-MINCUT problem for which several efficient algorithms have been proposed in the literature [4]. We briefly describe how a binary submodular energy function (that is, an energy function defined using a label set of size 2) can be mapped to an *st*-MINCUT problem. For the more general setting of $h \geq 2$ labels, we refer the reader to [23, 17] for details. Before describing the mapping, we briefly define the *st*-MINCUT problem for completeness.

The *st*-MINCUT problem. We are given a directed graph $\mathbf{D} = (\mathbf{N}, \mathbf{A})$ where $\mathbf{N}$ is the set of nodes (vertices) and $\mathbf{A}$ is the set of arcs. Associated with each arc $(N_p, N_q) \in \mathbf{A}$ is a non-negative capacity $c(N_p, N_q)$. Given two nodes $s, t \in \mathbf{N}$, known as terminals, an *st*-cut is defined as the outgoing arcs from a subset of nodes $\mathbf{N}_1$ to another subset of nodes $\mathbf{N}_2$ such that the following conditions holds: (i) $s \in \mathbf{N}_1$; (ii) $t \in \mathbf{N}_2$; (iii) $\mathbf{N}_1 \cap \mathbf{N}_2 = \{\}$; and (iv) $\mathbf{N}_1 \cup \mathbf{N}_2 = \mathbf{N}$. In

general, we can have more than one *st*-cut for a given set of terminals. The capacity of an *st*-cut is defined as the sum of the capacities of all the outgoing arcs from $\mathbf{N}_1$ to $\mathbf{N}_2$. Mathematically, we will denote this as follows:

$$\sum_{N_p \in \mathbf{N}_1, N_q \in \mathbf{N}_2} c(N_p, N_q). \tag{14}$$

The *st*-MINCUT problem corresponds to finding the *st*-cut with the minimum capacity.

Binary submodular energy minimization. We are now ready to describe how a binary submodular energy minimization problem can be mapped to an equivalent *st*-MINCUT problem. Note that for a binary energy function, defined using two labels $\mathbf{L} = \{l_1, l_2\}$, the submodularity condition can be simplified to the following:

$$\theta_{pq}(l_1, l_1) + \theta_{pq}(l_2, l_2) \leq \theta_{pq}(l_1, l_2) + \theta_{pq}(l_2, l_1). \tag{15}$$

We define a graph $\mathbf{D} = (\mathbf{N} \cup \{s, t\}, \mathbf{A})$ that contains a node $N_p \in \mathbf{N}$ for each random variable $p \in \mathbf{V}$. For any *st*-cut in $\mathbf{D}$, we define a labeling of the MRF as follows:

$$x_p = \begin{cases} l_1 & \text{if } N_p \in \mathbf{N}_1, \\ l_2 & \text{otherwise.} \end{cases} \tag{16}$$

If we can ensure that the energy of the labeling $\mathbf{x}$ defined above is the same as the capacity of the cut $(\mathbf{N}_1, \mathbf{N}_2)$ up to a constant, then we would have accomplished our goal of mapping energy minimization to *st*-MINCUT. Specifically, by computing the *st*-MINCUT $(\mathbf{N}_1^*, \mathbf{N}_2^*)$ we can obtain an optimal labeling $\mathbf{x}^*$ using the above equation. We now show that it is possible to define non-negative arc capacities such that the energy of the labeling is indeed the same as the capacity of the corresponding cut. To this end, we consider two separate cases: the contribution of the unary potentials to the energy, and the contribution of the pairwise potentials to the energy.

For the unary potentials, consider a random variable $p \in \mathbf{V}$. We model its unary potentials using the arcs defined in Figure 3 (left). Note that when $x_p = l_1$, then $N_p \in \mathbf{N}_1$, which would imply that the arc (N_p, t) would contribute $\theta_p(l_1) + K$ to the capacity of the cut. Similarly, when $x_p = l_2$, then $N_p \in \mathbf{N}_2$, which would imply that the arc (s, N_p) would contribute $\theta_p(l_2) + K$ to the capacity of the cut. For the pairwise potentials, consider two neighboring random variables $(p, q) \in \mathbf{E}$. We model its pairwise potentials using the arcs defined in Figure 3 (right). Once again, the arc capacities have been chosen such that the capacity of the cut is the same as the contribution of the corresponding pairwise potential up to a constant. Given an energy function $E(\cdot; \boldsymbol{\theta})$ defined over n random variables, we can obtain the corresponding directed graph $\mathbf{D}$ using the additivity theorem [32]. Loosely speaking, this involves constructing a directed graph whose arc capacities are the sum of the arc capacities over all individual directed graphs that correspond to each of the unary and pairwise potentials of the energy function. For example, Figure 4 shows the directed graph that can be used to minimize the energy function corresponding to the MRF depicted in Figure 1.

2.6 Linear programming

Often we are faced with a problem that requires a complex MRF, that is, one which is not tree-structured and whose energy function is not submodular. As the general energy minimization problem, even for the special case of pairwise MRFs, is NP-hard, we can only hope to obtain an approximate solution for this problem. One choice for approximate inference is to use belief propagation by passing messages in some arbitrary order. However, this algorithm is not

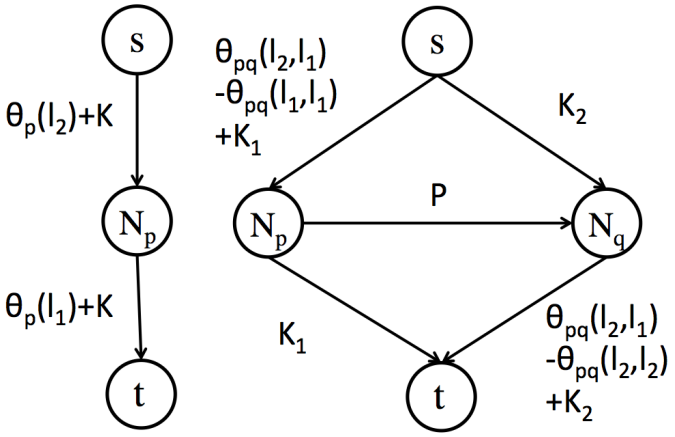


Figure 3: **Left**. *The directed graph corresponding to unary potentials. The constant K is chosen to be sufficiently large to ensure that the arc capacities are non-negative. Note that since the random variable p has to be assigned exactly one label, adding a constant to the arc capacities of (s, N_p) and (N_p, t) simply adds a constant to the capacity of every st-cut, thereby not affecting the st-MINCUT solution.* **Right**. *The directed graph corresponding to the pairwise potentials. The constants K_1 and K_2 ensure that the arc capacities are non-negative. The arc capacity $P = \theta_{pq}(l_1, l_2) + \theta_{pq}(l_2, l_1) - \theta_{pq}(l_1, l_1) - \theta_{pq}(l_2, l_2)$ is guaranteed to be non-negative due to the submodularity of the energy function (see equation (15)).*

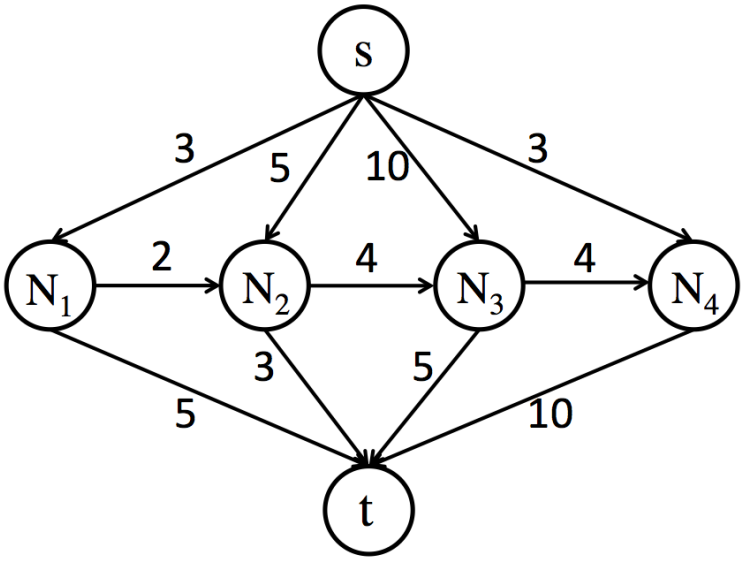


Figure 4: *The directed graph* **D** *corresponding to the submodular energy function of the MRF depicted in Figure 1. Any st-cut of* **D** *has a corresponding labeling for the MRF, defined by equation (16), such that the capacity of the cut is equal to the energy of the labeling plus a constant. Hence, the minimum st-cut of* **D** *provides an optimal labeling of the MRF.*

guaranteed to converge, and even at covergence, it provides weak theoretical guarantees [64]. In contrast, one can obtain strong theoretical guarantees on the quality of the solution by first formulating the inference problem as an integer linear program, and then dropping the integral constraints to obtain a polynomial-time solution linear programming (LP) relaxation.

In order to describe the LP relaxation, we require an overcomplete representation of a labeling. To this end, we will specify the following variables: (i) unary variables $\bar{x}_p(l_i) \in \{0,1\}$ for all $p \in \mathbf{V}$ and $l_i \in \mathbf{L}$ such that $\bar{x}_p(l_i) = 1$ if and only if p is assigned the label l_i; and (ii) pairwise variables $\bar{x}_{pq}(l_i, l_j) \in \{0,1\}$ for all $(p,q) \in \mathbf{E}$ and $l_i, l_j \in \mathbf{L}$ such that $\bar{x}_{pq}(l_i, l_j) = 1$ if and only if p and q are assigned labels l_i and l_j respectively. This allows us to formulate inference as follows:

$$
\begin{aligned}
\min_{\bar{\mathbf{x}}} \quad & \sum_{p \in \mathbf{V}} \sum_{l_i \in \mathbf{L}} \theta_p(l_i) \bar{x}_p(l_i) + \sum_{(p,q) \in \mathbf{E}} \sum_{l_i, l_j \in \mathbf{L}} \theta_{pq}(l_i, l_j) \bar{x}_{pq}(l_i, l_j), \\
\text{s.t.} \quad & \sum_{l_i \in \mathbf{L}} \bar{x}_p(l_i) = 1, \forall p \in \mathbf{V}, \\
& \sum_{l_j \in \mathbf{L}} \bar{x}_{pq}(l_i, l_j) = \bar{x}_p(l_i), \forall (p,q) \in \mathbf{E}, l_i \in \mathbf{L}, \\
& \sum_{l_i \in \mathbf{L}} \bar{x}_{pq}(l_i, l_j) = \bar{x}_q(l_j), \forall (p,q) \in \mathbf{E}, l_j \in \mathbf{L}, \\
& \bar{x}_p(l_i) \in \{0,1\}, \bar{x}_{pq}(l_i, l_j) \in \{0,1\}, \forall p \in \mathbf{V}, (p,q) \in \mathbf{E}, l_i, l_j \in \mathbf{L}.
\end{aligned}
$$

The first set of constraints ensures that each random variables is assigned exactly one label. The second and third sets of constraints ensure that, for binary optimization variables, $\bar{x}_{pq}(l_i, l_j) = \bar{x}_p(l_i)\bar{x}_q(l_j)$. By relaxing the final set of constraints such that the optimization variables can take any value between 0 and 1 inclusive, we obtain a linear program (LP) that is polynomial in the size of the input [8, 54, 62].

The above LP relaxation of the inference problem is guaranteed to be tight for tree-structured MRFs [2, 62] and submodular energy functions [8]. In other words, it provides an optimal solution for the two special cases of inference that admit efficient algorithms. In addition, in order to obtain an approximate solution for NP-hard instances of inference, several randomized rounding procedures have been proposed in the literature, which convert the optimal fractional solution of the LP relaxation to a feasible labeling of the MRF [2, 8, 27]. These rounding procedures, described in more detail in the next chapter, provide provably strong theoretical guarantees on the quality of the approximate solution obtained for several special cases of interest have been established in the literature.

While the LP relaxation can be solved in polynomial time, a general LP solver will typically not be able to solve the above relaxation for large-scale problems encountered in computer vision. However, motivated by the accuracy of the LP relaxation in comparison with other approximate inference algorithms [40, 44], several customized algorithms that have developed to solve the LP relaxation more efficiently. We will cover some such algorithms in chapters 4 and 5.

3 Move-making algorithms

In this chapter, we describe a widely used family of efficient approximate inference algorithms, known as move-making algorithms. Broadly speaking, a move-making algorithm start with an initial labeling $\mathbf{x}^0$ and iteratively moves to a better labeling until a convergence criterion is met. The key to designing a move-making algorithm is an efficient subroutine that moves from one labeling to a better labeling. In this work, we are mostly interested in algorithms that identify a new labeling by searching over a subset of all possible labelings. The subset of labelings is

chosen to satisfy the following two conditions: (i) it contains the current labeling; and (ii) it allows us to identify a new (possibly better) labeling by solving a single *st*-MINCUT problem. Since *st*-MINCUT admits several efficient algorithm, each iteration of a move-making algorithm will also be efficient. Typically, the move-making algorithms also converge in a small number of iterations, thereby making the whole procedure computationally feasible for even large-scale problems encountered in computer vision.

We begin our exploration of move-making algorithms by considering a special case of inference known as metric labeling, which is commonly used to model low-level vision applications. Later on in the chapter we will consider high-order energy functions that are amenable to move-making algorithms. Metric labeling is characterized by a finite, discrete label set and a metric distance function over the labels. The energy function in metric labeling consists of arbitrary unary potentials and pairwise potentials that are proportional to the distance between the labels assigned to them. The problem is known to be NP-hard [60]. Traditionally, in the computer vision community, this problem is solved approximately one of several move-making algorithms [5, 20, 39, 41, 61]. The advantage of move-making algorithms is their computational efficiency. However, most of the early move-making algorithms provide weaker theoretical guarantees on the quality of the solution than the slower LP relaxation discussed in the previous chapter.

At first sight, the difference between move-making algorithms and the LP relaxation appears to be the standard accuracy vs. speed trade-off. However, for some special cases of distance functions, it has been shown that appropriately designed move-making algorithms can match the theoretical guarantees of the LP relaxation [39, 41, 60]. Recently, this result has been extended for a large class of randomized rounding procedures, which we call parallel rounding [38]. In particular it has been shown that for any arbitrary (semi-)metric distance function, there exist move-making algorithms that match the theoretical guarantees provided by parallel rounding. In the following sections, we study such rounding-based move-making algorithms in detail.

3.1 Preliminaries

Metric labeling. A metric labeling problem is a special case of inference that is defined using a metric distance function $d : \mathbf{L} \times \mathbf{L} \rightarrow \mathbb{R}^+$ over the labels. Recall that a metric distance function satisfies the following properties: (i) $d(l_i, l_j) \geq 0$ for all $l_i, l_j \in \mathbf{L}$, and $d(l_i, l_j) = 0$ if and only if $i = j$; and (ii) $d(l_i, l_j) + d(l_j, l_k) \geq d(l_i, l_k)$ for all $l_i, l_j, l_k \in \mathbf{L}$. The energy function consists of arbitrary unary potentials, and pairwise potentials that are proportional to the distance between the two labels. Formally, the energy of a labeling $\mathbf{x}$ is

$$E(\mathbf{x}) = \sum_{p \in \mathbf{V}} \theta_p(x_p) + \sum_{(p,q) \in \mathbf{E}} w_{pq} d(x_p, x_q), \tag{17}$$

where the edge weights w_{pq} are non-negative. Metric labeling requires us to find a labeling with the minimum energy. It is known to be NP-hard, and hence, we have to settle for an approximate solution.

Multiplicative bound. As metric labeling plays a central role in low-level vision, several approximate algorithms have been proposed in the literature. A common theoretical measure of accuracy for an approximate algorithm is the multiplicative bound. In this work, we are interested in the multiplicative bound of an algorithm with respect to a distance function. Formally, given a distance function d, the multiplicative bound of an algorithm is said to be B if the following condition is satisfied for all possible values of unary potentials $\theta_p(\cdot)$ and non-negative edge weights

w_{pq}:

$$\sum_{p \in \mathbf{V}} \theta_p(x'_p) + \sum_{(p,q) \in \mathbf{E}} w_{pq} d(x'_p, x'_q) \leq \sum_{p \in \mathbf{V}} \theta_p(x^*_p) + B \sum_{(p,q) \in \mathbf{E}} w_{pq} d(x^*_p, x^*_q). \tag{18}$$

Here, $\mathbf{x}'$ is the labeling estimated by the algorithm for the given values of unary potentials and edge weights, and $\mathbf{x}^*$ is an optimal labeling. Multiplicative bounds are greater than or equal to 1, and are invariant to reparameterizations that only modify the unary potentials. Note that reparameterizations that modify the pairwise potentials are not considered since the energy function needs to have a specific form for a valid metric labeling problem. A multiplicative bound B is said to be tight if the above inequality holds as an equality for some value of unary potentials and edge weights.

Rounding procedure. In order to prove theoretical guarantees of the LP relaxation, it is common to use a rounding procedure that can covert a feasible fractional solution $\bar{\mathbf{x}}$ of the LP relaxation to a feasible integer solution $\hat{\mathbf{x}}$ of the integer linear program. Several rounding procedures have been proposed in the literature. In this work, we focus on the randomized parallel rounding procedures proposed by chekuri and kleinbergstoc99. These procedures have the property that, given a fractional solution $\bar{\mathbf{x}}$, the probability of assigning a label $l_i \in \mathbf{L}$ to a random variable $p \in \mathbf{V}$ is equal to $\bar{x}_p(l_i)$, that is,

$$\Pr(\hat{x}_p(l_i) = 1) = \bar{x}_p(l_i). \tag{19}$$

We will describe the various rounding procedures in detail in sections 3.2-3.4. For now, we would like to note that our reason for focusing on the parallel rounding of chekuri and kleinbergstoc99 is that they provide the best known polynomial-time theoretical guarantees for metric labeling. Specifically, we are interested in their approximation factor, which is defined next.

Approximation factor. Given a distance function d, the approximation factor for a rounding procedure is said to be F if the following condition is satisfied for all feasible fractional solutions $\bar{\mathbf{x}}$:

$$\mathbb{E}\left(\sum_{l_i, l_j \in \mathbf{L}} d(l_i, l_j) \hat{x}_p(l_i) \hat{x}_q(l_j) \right) \leq F \sum_{l_i, l_j \in \mathbf{L}} d(l_i, l_j) \bar{x}_{ab}(l_i, l_j). \tag{20}$$

Here, $\hat{\mathbf{x}}$ refers to the integer solution, and the expectation is taken with respect to the randomized rounding procedure applied to the feasible solution $\bar{\mathbf{x}}$.

Given a rounding procedure with an approximation factor of F, an optimal fractional solution $\bar{\mathbf{x}}^*$ of the LP relaxation can be rounded to a labeling $\hat{\mathbf{x}}$ that satisfies the following condition:

$$\begin{aligned} & \mathbb{E}\left(\sum_{p \in \mathbf{V}} \sum_{l_i \in \mathbf{L}} \theta_p(l_i) \hat{x}_p(l_i) + \sum_{(p,q) \in \mathbf{E}} \sum_{l_i, l_j \in \mathbf{L}} w_{pq} d(l_i, l_j) \hat{x}_p(l_i) \hat{x}_q(l_j) \right) \\ \leq & \sum_{p \in \mathbf{V}} \sum_{l_i \in \mathbf{L}} \theta_a(l_i) \bar{x}^*_p(l_i) + F \sum_{(p,q) \in \mathbf{E}} \sum_{l_i, l_j \in \mathbf{L}} w_{pq} d(l_i, l_j) \bar{x}^*_{pq}(l_i, l_j). \end{aligned}$$

The above inequality follows directly from properties (19) and (20). Similar to multiplicative bounds, approximation factors are always greater than or equal to 1, and are invariant to reparameterizations that only modify the unary potentials. An approximation factor F is said to be tight if the above inequality holds as an equality for some value of unary potentials and edge weights.

Submodular energy function. We will use the following important fact throughout this chapter. Given an energy function defined using arbitrary unary potentials, non-negative edge weights and a submodular distance function, an optimal labeling can be computed in polynomial time by solving an equivalent minimum *st*-cut problem [17]. Recall that a submodular distance function d' over a label set $\mathbf{L} = \{l_1, l_2, \cdots, l_h\}$ satisfies the following properties: (i) $d'(l_i, l_j) \geq 0$ for all $l_i, l_j \in \mathbf{L}$, and $d'(l_i, l_j) = 0$ if and only if $i = j$; and (ii) $d'(l_i, l_j) + d'(l_{i+1}, l_{j+1}) \leq d'(l_i, l_{j+1}) + d'(l_{i+1}, l_j)$ for all $l_i, l_j \in \mathbf{L} \backslash \{l_h\}$ (where $\backslash$ refers to set difference).

3.2 Complete rounding and complete move

We start with a simple rounding scheme, which we call complete rounding. While complete rounding is not very accurate, it would help illustrate the flavor of rounding-based moves. We will subsequently consider its generalizations, which have been useful in obtaining the best-known approximation factors for various special cases of metric labeling.

The complete rounding procedure consists of a single stage where we use the set of all unary variables to obtain a labeling (as opposed to other rounding procedures discussed subsequently). Algorithm 1 describes its main steps. Intuitively, it treats the value of the unary variable $\bar{x}_p(l_i)$ as the probability of assigning the label $l_i \in \mathbf{L}$ to the random variable $p \in \mathbf{V}$. It obtains a labeling by sampling from all the distributions $\bar{\mathbf{x}}_p = [\bar{x}_p(l_i), \forall l_i \in \mathbf{L}]$ simultaneously using the same random number $r \in [0, 1]$.

It can be shown that using a different random number to sample the distributions $\bar{\mathbf{x}}_p$ and $\bar{\mathbf{x}}_q$ of two neighboring random variables $(p, q) \in \mathbf{E}$ results in an infinite approximation factor. For example, let $\bar{x}_a(l_i) = \bar{x}_q(l_i) = 1/h$ for all $l_i \in \mathbf{L}$, where h is the number of labels. The pairwise variables $\bar{\mathbf{x}}_{pq}$ that minimize the energy function are $\bar{x}_{pq}(l_i, l_i) = 1/h$ and $\bar{x}_{pq}(l_i, l_j) = 0$ when $i \neq j$. For the above feasible solution of the LP relaxation, the RHS of inequality (20) is 0 for any finite F, while the LHS of inequality (20) is strictly greater than 0 if $h > 1$. However, we will shortly show that using the same random number r for all random variables provides a finite approximation factor.

Algorithm 1 *The complete rounding procedure.*

input A feasible solution $\bar{\mathbf{x}}$ of the LP relaxation.

1: Pick a real number r uniformly from $[0, 1]$.
2: **for all** $p \in \mathbf{V}$ **do**
3: Define $\tilde{x}_p(0) = 0$ and $\tilde{x}_p(i) = \sum_{j=1}^{i} \bar{x}_p(l_j)$ for all $l_i \in \mathbf{L}$.
4: Assign the label $l_i \in \mathbf{L}$ to the random variable p if $\tilde{x}_p(i-1) < r \leq \tilde{x}_p(i)$.
5: **end for**

We now turn our attention to designing a move-making algorithm whose multiplicative bound matches the approximation factor of the complete rounding procedure. To this end, we modify the range expansion algorithm proposed by kumarnips08 for truncated convex pairwise potentials to a general (semi-)metric distance function. Our method, which we refer to as the complete move-making algorithm, considers all putative labels of all random variables, and provides an approximate solution in a single iteration. Algorithm 2 describes its two main steps. First, it computes a submodular overestimation of the given distance function by solving the following

optimization problem:

$$\begin{aligned} d^S = \quad & \operatorname*{argmin}_{d'} t \\ \text{s.t.} \quad & d'(l_i, l_j) \leq t d(l_i, l_j), \forall l_i, l_j \in \mathbf{L}, \\ & d'(l_i, l_j) \geq d(l_i, l_j), \forall l_i, l_j \in \mathbf{L}, \\ & d'(l_i, l_j) + d'(l_{i+1}, l_{j+1}) \leq d'(l_i, l_{j+1}) + d'(l_{i+1}, l_j), \forall l_i, l_j \in \mathbf{L} \backslash \{l_h\}. \end{aligned} \tag{21}$$

The above problem minimizes the maximum ratio of the estimated distance to the original distance over all pairs of labels, that is,

$$\max_{i \neq j} \frac{d'(l_i, l_j)}{d(l_i, l_j)}.$$

We will refer to the optimal value of problem (21) as the submodular distortion of the distance function d. Second, it replaces the original distance function by the submodular overestimation and computes an approximate solution to the original metric labeling problem by solving a single minimum st-cut problem. Note that, unlike the range expansion algorithm [41] that uses the readily available submodular overestimation of a truncated convex distance (namely, the corresponding convex distance function), our approach estimates the submodular overestimation via the LP (21). Since the LP (21) can be solved for any arbitrary distance function, it makes complete move-making more generally applicable.

Algorithm 2 *The complete move-making algorithm.*

input Unary potentials $\theta_p(\cdot)$, edge weights w_{pq}, distance function d.

1: Compute a submodular overestimation of d by solving problem (21).

2: Using the approach of flachtr06, solve the following problem via an equivalent minimum st-cut problem:

$$\mathbf{x}' = \operatorname*{argmin}_{\mathbf{x} \in \mathbf{L}^n} \sum_{p \in \mathbf{V}} \theta_p(x_p) + \sum_{(p,q) \in \mathbf{E}} w_{pq} d^S(x_p, x_q).$$

The following theorem establishes the theoretical guarantees of the complete move-making algorithm and the complete rounding procedure. The tight multiplicative bound of the complete move-making algorithm is equal to the submodular distortion of the distance function. Furthermore, the tight approximation factor of the complete rounding procedure is also equal to the submodular distortion of the distance function.

In terms of computational complexities, complete move-making is significantly faster than solving the LP relaxation. Specifically, given an MRF with n random variables (that is, $|\mathbf{V}| = n$) and m edges (that is, $|\mathbf{E}| = m$), and a label set with h labels (that is, $|\mathbf{L}| = h$), the LP relaxation requires at least $O(m^3 h^3 \log(m^2 h^3))$ time, since it consists of $O(mh^2)$ optimization variables and $O(mh)$ constraints. In contrast, complete move-making requires $O(nmh^3 \log(m))$ time, since the graph constructed using the method of flachtr06 consists of $O(nh)$ nodes and $O(mh^2)$ arcs. Note that complete move-making also requires us to solve the linear program (21). However, since problem (21) is independent of the unary potentials and the edge weights, it only needs to be solved once beforehand in order to compute the approximate solution for any metric labeling problem defined using the distance function d.

3.3 Interval rounding and interval moves

Theorem 3.2 implies that the approximation factor of the complete rounding procedure is very large for distance functions that are highly non-submodular. For example, consider the truncated

linear distance function defined as follows over a label set $\mathbf{L} = \{l_1, l_2, \cdots, l_h\}$:

$$d(l_i, l_j) = \min\{|i - j|, M\}.$$

Here, M is a user specified parameter that determines the maximum distance. The tightest submodular overestimation of the above distance function is the linear distance function, that is, $d(l_i, l_j) = |i - j|$. This implies that the submodular distortion of the truncated linear metric is $(h-1)/M$, and therefore, the approximation factor for the complete rounding procedure is also $(h-1)/M$. In order to avoid this large approximation factor, chekuri proposed an interval rounding procedure, which captures the intuition that it is beneficial to assign similar labels to as many random variables as possible.

Algorithm 3 provides a description of interval rounding. The rounding procedure chooses an interval of at most a consecutive labels (step 2). It generates a random number r (step 3), and uses it to attempt to assign labels to previously unlabeled random variables from the selected interval (steps 4-7). It can be shown that the overall procedure converges in a polynomial number of iterations with a probability of 1 [8]. Note that if we fix $a = h$ and $z = 1$, interval rounding becomes equivalent to complete rounding. However, the analyses of chekuri and kleinbergstoc99 shows that other values of a provide better approximation factors for various special cases.

Algorithm 3 *The interval rounding procedure.*

input A feasible solution $\bar{\mathbf{x}}$ of the LP relaxation.

1: **repeat**
2: Pick an integer z uniformly from $[-a+2, h]$. Define an interval of labels $\mathbf{I} = \{l_s, \cdots, l_e\}$, where $s = \max\{z, 1\}$ is the start index and $e = \min\{z + a - 1, h\}$ is the end index.
3: Pick a real number r uniformly from $[0, 1]$.
4: **for all** Unlabeled random variables p **do**
5: Define $\tilde{x}_p(0) = 0$ and $\tilde{x}_p(i) = \sum_{j=s}^{s+i-1} \bar{x}_p(l_j)$ for all $i \in \{1, \cdots, e - s + 1\}$.
6: Assign the label $l_{s+i-1} \in \mathbf{I}$ to the p if $\tilde{x}_p(i-1) < r \leq \tilde{x}_p(i)$.
7: **end for**
8: **until** All random variables have been assigned a label.

Our goal is to design a move-making algorithm whose multiplicative bound matches the approximation factor of interval rounding for any choice of a. To this end, we propose the interval move-making algorithm that generalizes the range expansion algorithm [41], originally proposed for truncated convex distances, to arbitrary distance functions. Algorithm 4 provides its main steps. The central idea of the method is to improve a given labeling $\mathbf{x}'$ by allowing each random variable p to either retain its current label x'_p or to choose a new label from an interval of consecutive labels. In more detail, let $\mathbf{I} = \{l_s, \cdots, l_e\} \subseteq \mathbf{L}$ be an interval of labels of length at most a (step 4). For the sake of simplicity, let us assume that $x'_p \notin \mathbf{I}$ for any random variable p. We define $\mathbf{I}_p = \mathbf{I} \bigcup \{x'_p\}$ (step 5). For each pair of neighboring random variables $(p, q) \in \mathbf{E}$, we compute a submodular distance function $d^S_{x'_p, x'_q} : \mathbf{I}_p \times \mathbf{I}_q \rightarrow \mathbb{R}^+$ by solving the following linear

program (step 6):

$$
\begin{aligned}
d^S_{x'_p,x'_q} = \quad & \operatorname*{argmin}_{d'} t && (22)\\
\text{s.t.} \quad & d'(l_i,l_j) \leq t d(l_i,l_j), \forall l_i \in \mathbf{I}_p, l_j \in \mathbf{I}_q,\\
& d'(l_i,l_j) \geq d(l_i,l_j), \forall l_i \in \mathbf{I}_p, l_j \in \mathbf{I}_q,\\
& d'(l_i,l_j) + d'(l_{i+1},l_{j+1}) \leq d'(l_i,l_{j+1}) + d'(l_{i+1},l_j), \forall l_i, l_j \in \mathbf{I} \backslash \{l_e\},\\
& d'(l_i,l_e) + d'(l_{i+1},x'_q) \leq d'(l_i,x'_q) + d'(l_{i+1},l_e), \forall l_i \in \mathbf{I} \backslash \{l_e\},\\
& d'(l_e,l_j) + d'(x'_p,l_{j+1}) \leq d'(l_e,l_{j+1}) + d'(x'_p,l_j), \forall l_j \in \mathbf{I} \backslash \{l_e\},\\
& d'(l_e,l_e) + d(x'_p,x'_q) \leq d'(l_e,x'_q) + d'(x'_p,l_e).
\end{aligned}
$$

Similar to problem (21), the above problem minimizes the maximum ratio of the estimated distance to the original distance. However, instead of introducing constraints for all pairs of labels, it is only considers pairs of labels l_i and l_j where $l_i \in \mathbf{I}_p$ and $l_j \in \mathbf{I}_q$. Furthermore, it does not modify the distance between the current labels x'_p and x'_q (as can be seen in the last constraint of problem (22)).

Given the submodular distance functions $d^S_{x'_p,x'_q}$, we can compute a new labeling $\mathbf{x}''$ by solving the following optimization problem via minimum *st*-cut using the method of flachtr06 (step 7):

$$
\begin{aligned}
\mathbf{x}'' = \quad & \operatorname*{argmin}_{\mathbf{x}} \sum_{p \in \mathbf{V}} \theta_p(x_p) + \sum_{(p,q) \in \mathbf{E}} w_{pq} d^S_{x'_p,x'_q}(x_p,x_q)\\
\text{s.t.} \quad & x_p \in \mathbf{I}_a, \forall p \in \mathbf{V}. && (23)
\end{aligned}
$$

If the energy of the new labeling $\mathbf{x}''$ is less than that of the current labeling $\mathbf{x}'$, then we update our labeling to $\mathbf{x}''$ (steps 8-10). Otherwise, we retain the current estimate of the labeling and consider another interval. The algorithm converges when the energy does not decrease for any interval of length at most a. Note that, once again, the main difference between interval move-making and the range expansion algorithm is the use of an appropriate optimization problem, namely the LP (22), to obtain a submodular overestimation of the given distance function. This allows us to use interval move-making for the general metric labeling problem, instead of focusing on only truncated convex models.

The following theorem establishes the theoretical guarantees of the interval move-making algorithm and the interval rounding procedure. The tight multiplicative bound of the interval move-making algorithm is equal to the tight approximation factor of the interval rounding procedure. While Algorithms 3 and 4 use intervals of consecutive labels, they can easily be modified to use subsets of (potentially non-consecutive) labels. Our analysis could be extended to show that the multiplicative bound of the resulting subset move-making algorithm matches the approximation factor of the subset rounding procedure. However, our reason for focusing on intervals of consecutive labels is that several special cases of theorem 3.3 have previously been considered separately in the literature [20, 39, 41, 60]. Specifically, the following known results are corollaries of the above theorem. Note that, while the following corollaries have been previously proved in the literature, our work is the first to establish the tightness of the theoretical guarantees. When $a = 1$, the multiplicative bound of the interval move-making algorithm (which is equivalent to the expansion algorithm [5]) for the uniform metric distance is 2. The above corollary follows from the approximation factor of the interval rounding procedure proved by kleinbergstoc99, but it was independently proved by veksler99. When $a = M$, the multiplicative bound of the interval move-making algorithm for the truncated linear distance function is 4. The above corollary follows from the approximation factor of the interval rounding procedure proved by chekuri, but it was independently proved by guptastoc00. When $a = \sqrt{2}M$,

Algorithm 4 *The interval move-making algorithm.*

input Unary potentials $\theta_p(\cdot)$, edge weights w_{pq}, distance function d, initial labeling $\mathbf{x}^0$.

1: Set current labeling to initial labeling, that is, $\mathbf{x}' = \mathbf{x}^0$.
2: **repeat**
3: **for all** $z \in [-a+2, h]$ **do**
4: Define an interval of labels $\mathbf{I} = \{l_s, \cdots, l_e\}$, where $s = \max\{z, 1\}$ is the start index and $e = \min\{z + a - 1, h\}$ is the end index.
5: Define $\mathbf{I}_p = \mathbf{I} \bigcup \{x'_p\}$ for all random variables $p \in \mathbf{V}$.
6: Obtain submodular overestimates $d^S_{x'_p, x'_q}$ for each pair of neighboring random variables $(p, q) \in \mathbf{E}$ by solving problem (22).
7: Obtain a new labeling $\mathbf{x}''$ by solving problem (23).
8: **if** Energy of $\mathbf{x}''$ is less than energy of $\mathbf{x}'$ **then**
9: Update $\mathbf{x}' = \mathbf{x}''$.
10: **end if**
11: **end for**
12: **until** Energy cannot be decreased further.

the multiplicative bound of the interval move-making algorithm for the truncated linear distance function is $2 + \sqrt{2}$. The above corollary follows from the approximation factor of the interval rounding procedure proved by chekuri, but it was independently proved by kumarnips08. Finally, since our analysis does not use the triangular inequality of metric distance functions, it is also applicable to semi-metric labeling. Therefore, we can also state the following corollary for the truncated quadratic distance. When $a = \sqrt{M}$, the multiplicative bound of the interval move-making algorithm for the truncated linear distance function is $O(\sqrt{M})$. The above corollary follows from the approximation factor of the interval rounding procedure proved by chekuri, but it was independently proved by kumarnips08.

An interval move-making algorithm that uses an interval length of a runs for at most $O(h/a)$ iterations. This follows from a simple modification of the result by guptastoc00 (specifically, theorem 3.7). Hence, the total time complexity of interval move-making is $O(nmha^2 \log(m))$, since each iteration solves a minimum st-cut problem of a graph with $O(na)$ nodes and $O(ma^2)$ arcs. In other words, interval move-making is at most as computationally complex as complete move-making, which in turn is significantly less complex than solving the LP relaxation. Note that problem (22), which is required for interval move-making, is independent of the unary potentials and the edge weights. Hence, it only needs to be solved once beforehand for all pairs of labels $(x'_p, x'_q) \in \mathbf{L} \times \mathbf{L}$ in order to obtain a solution for any metric labeling problem defined using the distance function d.

3.4 Hierarchical rounding and hierarchical moves

We now consider the most general form of parallel rounding that has been proposed in the literature, namely the hierarchical rounding procedure [27]. The rounding relies on a hierarchical clustering of the labels. Formally, we denote a hierarchical clustering of m levels for the label set $\mathbf{L}$ by $\mathbf{K} = \{\mathbf{K}(i), i = 1, \cdots, m\}$. At each level i, the clustering $\mathbf{K}(i) = \{\mathbf{K}(i, j) \subseteq \mathbf{L}, j = 1, \cdots, h^i\}$ is mutually exclusive and collectively exhaustive, that is,

$$\bigcup_j \mathbf{K}(i, j) = \mathbf{L}, \mathbf{K}(i, j) \cap \mathbf{K}(i, j') = \emptyset, \forall j \neq j'.$$

Furthermore, for each cluster $\mathbf{K}(i,j)$ at the level $i > 2$, there exists a unique cluster $\mathbf{K}(i-1,j')$ in the level $i-1$ such that $\mathbf{K}(i,j) \subseteq \mathbf{K}(i-1,j')$. We call the cluster $\mathbf{K}(i-1,j')$ the parent of the cluster $\mathbf{K}(i,j)$ and define $a(i,j) = j'$. Similarly, we call $\mathbf{K}(i,j)$ a child of $\mathbf{K}(i-1,j')$. Without loss of generality, we assume that there exists a single cluster at level 1 that contains all the labels, and that each cluster at level m contains a single label.

Algorithm 5 *The hierarchical rounding procedure.*

input A feasible solution $\bar{\mathbf{x}}$ of the LP relaxation.

1: Define $f_p^1 = 1$ for all $p \in \mathbf{V}$.
2: **for all** $i \in \{2, \cdots, m\}$ **do**
3: **for all** $p \in \mathbf{V}$ **do**
4: Define $z_p^i(j)$ for all $j \in \{1, \cdots, h^i\}$ as follows:

$$z_p^i(j) = \begin{cases} \sum_{l_k \in \mathbf{K}(i,j)} \bar{x}_p(l_k) & \text{if } a(i,j) = f_p^{i-1}, \\ 0 & \text{otherwise.} \end{cases}$$

5: Define $\bar{x}_p^i(j)$ for all $j \in \{1, \cdots, h^i\}$ as follows:

$$\bar{x}_p^i(j) = \frac{z_p^i(j)}{\sum_{j'=1}^{h^i} z_p^i(j')}$$

6: **end for**
7: Using a rounding procedure (complete or interval) on $\bar{\mathbf{x}}^i = [x_p^i(j), \forall p \in \mathbf{V}, j \in \{1, \cdots, h^i\}]$, obtain an integer solution $\hat{\mathbf{x}}^i$.
8: **for all** $p \in \mathbf{V}$ **do**
9: Let $k_p \in \{1, \cdots, h^i\}$ such that $\hat{x}^i(k_p) = 1$. Define $f_p^i = k_p$.
10: **end for**
11: **end for**
12: **for all** $p \in \mathbf{V}$ **do**
13: Let l_k be the unique label present in the cluster $\mathbf{K}(m, f_p^m)$. Assign l_k to p.
14: **end for**

Algorithm 5 describes the hierarchical rounding procedure. Given a clustering $\mathbf{K}$, it proceeds in a top-down fashion through the hierarchy while assigning each random variable to a cluster in the current level. Let f_p^i be the index of the cluster assigned to the random variable p in the level i. In the first step, the rounding procedure assigns all the random variables to the unique cluster $\mathbf{K}(1,1)$ (step 1). At each step i, it assigns each random variable to a unique cluster in the level i by computing a conditional probability distribution as follows. The conditional probability $\bar{x}_p^i(j)$ of assigning the random variable p to the cluster $\mathbf{K}(i,j)$ is proportional to $\sum_{l_k \in \mathbf{K}(i,j)} \bar{x}_p(k)$ if $a(i,j) = f_p^{i-1}$ (steps 3-6). The conditional probability $\bar{x}_p^i(j) = 0$ if $a(i,j) \neq f_p^{i-1}$, that is, a random variable cannot be assigned to a cluster $\mathbf{K}(i,j)$ if it wasn't assigned to its parent in the previous step. Using a rounding procedure (complete or interval) for $\bar{\mathbf{x}}^i$, we obtain an assignment of random variables to the clusters at level i (step 7). Once such an assignment is obtained, the values f_p^i are computed for all random variables p (steps 8-10). At the end of step m, hierarchical rounding would have assigned each random variable to a unique cluster in the level m. Since each cluster at level m consists of a single label, this provides us with a labeling of the MRF (steps 12-14).

Our goal is to design a move-making algorithm whose multiplicative bound matches the approximation factor of the hierarchical rounding procedure for any choice of hierarchical clustering

Algorithm 6 *The hierarchical move-making algorithm.*

input Unary potentials $\theta_p(\cdot)$, edge weights w_{pq}, distance function d.

1: **for all** $j \in \{1, \cdots, h\}$ **do**
2: Let l_k be the unique label is the cluster $\mathbf{K}(m, j)$. Define $x_p^{m,j} = l_k$ for all $p \in \mathbf{V}$.
3: **end for**
4: **for all** $i \in \{2, \cdots, m\}$ **do**
5: **for all** $j \in \{1, \cdots, h^{m-i+1}\}$ **do**
6: Define $\mathbf{L}_p^{m-i+1,j} = \{x_p^{m-i+2,j'}, a(m-i+2, j') = j, j' \in \{1, \cdots, h^{m-i+2}\}\}$.
7: Using a move-making algorithm (complete or interval), compute the labeling $\mathbf{x}^{m-i+1,j}$ under the constraint $x_p^{m-i+1,j} \in \mathbf{L}_p^{m-i+1,j}$.
8: **end for**
9: **end for**
10: The final solution is $\mathbf{x}^{1,1}$.

$\mathbf{K}$. To this end, we propose the hierarchical move-making algorithm Algorithm 6 provides its main steps. In contrast to hierarchical rounding, the move-making algorithm traverses the hierarchy in a bottom-up fashion while computing a labeling for each cluster in the current level. Let $\mathbf{x}^{i,j}$ be the labeling corresponding to the cluster $\mathbf{K}(i, j)$. At the first step, when considering the level m of the clustering, all the random variables are assigned the same label. Specifically, $x_p^{m,j}$ is equal to the unique label contained in the cluster $\mathbf{K}(m, j)$ (steps 1-3). At step i, it computes the labeling $\mathbf{x}^{m-i+1,j}$ for each cluster $\mathbf{K}(m-i+1, j)$ by using the labelings computed in the previous step. Specifically, it restricts the label assigned to a random variable p in the labeling $\mathbf{x}^{m-i+1,j}$ to the subset of labels that were assigned to it by the labelings corresponding to the children of $\mathbf{K}(m-i+1, j)$ (step 6). Under this restriction, the labeling $\mathbf{x}^{m-i+1,j}$ is computed by approximately minimizing the energy using a move-making algorithm (step 7). Implicit in our description is the assumption that that we will use a move-making algorithm (complete or interval) in step 7 of Algorithm 6 whose multiplicative bound matches the approximation factor of the rounding procedure (complete or interval) used in step 7 of Algorithm 5.

The following theorem establishes the theoretical guarantees of the hierarchical move-making algorithm and the hierarchical rounding procedure. The tight multiplicative bound of the hierarchical move-making algorithm is equal to the tight approximation factor of the hierarchical rounding procedure.

A special case of the hierarchical move-making algorithm was proposed by kumaruai09, which consider the r-hierarchically well-separted tree (r-HST) metric [3]. As the r-HST metric would play a key role in the rest of the chapter, we provide its formal definition. A rooted tree, as shown in Figure 5, is said to be an r-HST if it satisfy the following properties: (i) all the leaf nodes are the labels; (ii) all edge weights are positive; (iii) the edge lengths from any node to all of its children are the same; and (iv) on any root to leaf path the edge weight decrease by a factor of at least $r > 1$. We can think of a r-HST as a hierarchical clustering of the given label set $\mathbf{L}$. The root node is the cluster at the top level of the hierarchy and contains all the labels. As we go down in the hierarchy, the clusters break down into smaller clusters until we get as many leaf nodes as the number of labels in the given label set. The metric distance function defined on this tree $d^t(.,.)$ is known as the r-HST metric. In other words, the distance $d^t(\cdot,\cdot)$ between any two nodes in the given r-HST is the length of the unique path between these nodes in the tree. Figure 5 shows an example r-HST.

The following result is a corollary of theorem 3.4. The multiplicative bound of the hierarchical move-making algorithm is $O(1)$ for an r-HST metric distance. The above corollary follows from the approximation factor of the hierarchical rounding procedure proved by kleinbergstoc99,

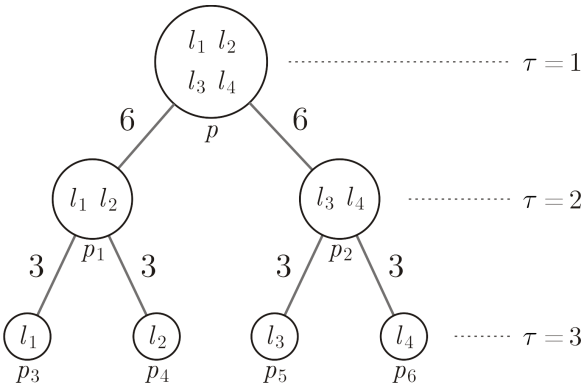


Figure 5: *An example of r-HST for $r = 2$. The cluster associated with root contains all the labels. As we go down, the cluster splits into subclusters and finally we get the singletons, the leaf nodes (labels). The root is at depth of 1 ($\tau = 1$) and leaf nodes at $\tau = 3$. The metric defined over the r-HST is denoted as $d^t(.,.)$, the shortest path between the inputs. For example, $d^t(l_1, l_3) = 18$ and $d^t(l_1, l_2) = 6$.*

but it was independently proved by kumaruai09. It is worth noting that the above result was also used to obtain an approximation factor of $O(\log h)$ for the general metric labeling problem by kleinbergstoc99 and a matching multiplicative bound of $O(\log h)$ by kumaruai09. This result follows from the fact that any general metric distance function $d(\cdot, \cdot)$ can be approximated using a mixture of r-HST distance functions $d_k^t(\cdot, \cdot), k = 1, \cdots, O(h \log h)$ such that

$$1 \leq \frac{\sum_k \rho_k d_k^t(l_i, l_j)}{d(l_i, l_j)} \leq O(\log h), \tag{24}$$

where ρ_k are the mixing coefficients. The algorithm for obtaining the mixture of r-HST distance functions that satisfy the above property was proposed by Fakcharoenphol04TightBound.

Note that hierarchical move-making solves a series of problems defined on a smaller label set. Since the complexity of complete and interval move-making is superlinear in the number of labels, it can be verified that the hierarchical move-making algorithm is at most as computationally complex as the complete move-making algorithm (corresponding to the case when the clustering consists of only one cluster that contains all the labels). Hence, hierarchical move-making is significantly faster than solving the LP relaxation.

3.5 Dense stereo correspondence

Unlike the complete move-making algorithm, interval moves and hierarchical moves provide accurate inference algorithms for the energy functions encountered in real-world problems. We demonstrate their efficacy for dense stereo correspondence.

Data. Given two epipolar rectified images of the same scene, the problem of dense stereo correspondence requires us to obtain a correspondence between the pixels of the images. This problem can be modeled as metric labeling, where the random variables represent the pixels of one of the images, and the labels represent the disparity values. A disparity label $l_i \in \mathbf{L}$ for a random variable $p \in \mathbf{V}$ representing a pixel (u_p, v_p) of an image indicates that its corresponding pixel lies in location $(u_p + i, v_p)$. For the above problem, we use the unary potentials and edge

weights that are specified by Szeliski. We use two types of pairwise potentials: (i) truncated linear with the truncation set at 4; and (ii) truncated quadratic with the truncation set at 16.

Methods. We use three baseline methods. First, the loopy belief propagation (denoted by BP) algorithm described in the previous chapter. Second, the swap algorithm [5] (denoted by SWAP), which is a type of move-making algorithm that does not provide a finite multiplicative bound[1]. Third, the sequential tree-reweighted message passing algorithm [29] (denoted by TRW), which solves the LP relaxation of the inference algorithm described in the previous chapter. We report the results on two types of interval moves. First, the expansion algorithm [5] (denoted by EXP), which fixes the interval length $a = 1$. Second, the optimal interval move (denoted by INT), which uses the interval length that provides the best multiplicative bound. Finally, we also show the results obtained via hierarchical moves (denoted by HIER), where the hierarchices are obtained by approximating the given (semi-)metric as a mixture of r-HST metrics using the method of Fakcharoenphol04TightBound.

Results. Figures 6-11 show the results for various standard pairs of images. Note that TRW is the most accurate in terms of energy, but it is computationally inefficient. The results obtained by BP are not accurate. The standard move-making algorithms, EXP and SWAP, are fast but not as accurate as TRW. Among the rounding-based move-making algorithms INT is slower as it solves an *st*-MINCUT problem on a large graph at each iteration. In contrast, HIER uses an interval length of 1 for each subproblem and is therefore more efficient. The energy obtained by HIER is comparable to TRW.

3.6 Moves for high-order potentials

We now turn our attention to the problem of inference defined on energy functions that contain high-order potentials. In other words, the potentials are not just restricted to be unary and pairwise, but can be defined using an arbitrary sized subset of the random variables. Similar to the previous sections, we focus on a special case of high-order energy functions that are conducive to move-making algorithms that are based on iteratively solving an *st*-MINCUT problem. Specifically, we consider the inference problem of *parsimonious labeling* [12]. In the following subsections, we formally define the parsimonious labeling problem and present efficient and provably accurate move-making algorithms for various useful special cases as well as the general parsimonious labeling problem.

3.6.1 Parsimonious labeling

The parsimonious labeling problem is defined using an energy function that consists of unary potentials and clique potentials defined over cliques of arbitrary sizes. While the parsimonious labeling problem places no restrictions on the unary potentials, the clique potentials are specified using a *diversity* function [7]. Before describing the parsimonious labeling problem in detail, we briefly define the diversity function for the sake of completion.

A diversity is a pair $(\mathbf{L}, \delta)$, where $\mathbf{L}$ is the label set and δ is a non-negative function defined on subsets of $\mathbf{L}$, $\delta : \Gamma \rightarrow \mathbb{R}, \forall \Gamma \subseteq \mathbf{L}$, satisfying

- Non Negativity: $\delta(\Gamma) \geq 0$, and $\delta(\Gamma) = 0$, if and only if, $|\Gamma| \leq 1$.

[1]The swap algorithm iteratively selects two labels l_α and l_β. It allows a variable labeled as l_α to either retain its old label or swap to l_β. Futhermore, it allows a variable labeled as l_β to either retain its old label or swap to l_α. Each swap move is performed by solving a single *st*-MINCUT problem.

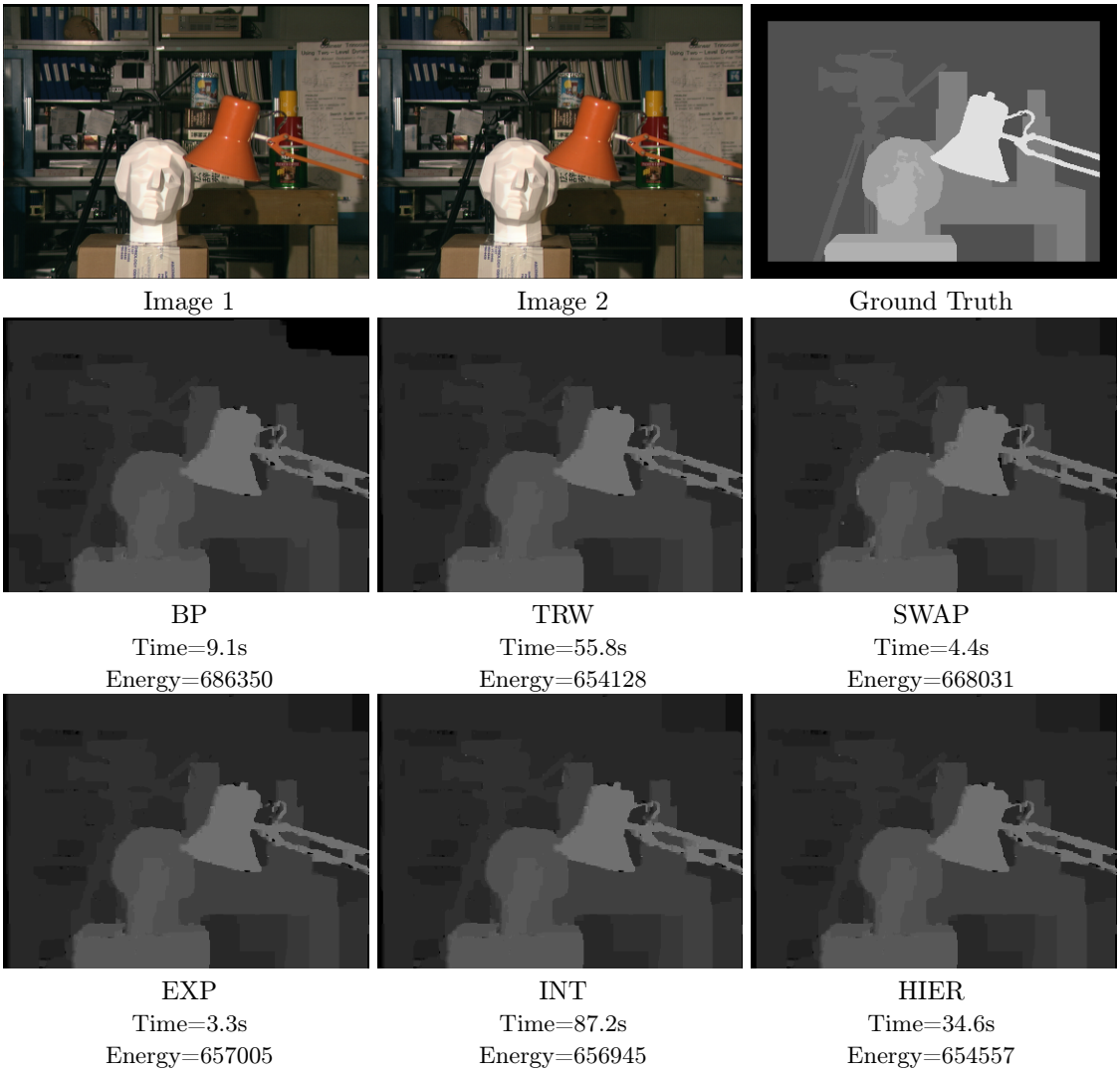


Figure 6: *Results for the 'tsukuba' image pair with truncated linear pairwise potentials.*

- Triangular Inequality: if $\Gamma_2 \neq \emptyset$, $\delta(\Gamma_1 \cup \Gamma_2) + \delta(\Gamma_2 \cup \Gamma_3) \geq \delta(\Gamma_1 \cup \Gamma_3), \forall \Gamma_1, \Gamma_2, \Gamma_3 \subseteq \mathbf{L}$.
- Monotonicity: $\Gamma_1 \subseteq \Gamma_2$ implies $\delta(\Gamma_1) \leq \delta(\Gamma_2)$.

Using a diversity function, we can define a clique potential as follows. We denote by $\Gamma(\mathbf{x}_c)$ the set of unique labels in the labeling of the clique $\mathbf{C}$. Then, $\theta_{\mathbf{C}}(\mathbf{x}_{\mathbf{C}}) = w_{\mathbf{C}}\delta(\Gamma(\mathbf{x}_{\mathbf{C}}))$, where δ is a diversity function and $w_{\mathbf{C}}$ is the non-negative weight corresponding to the clique $\mathbf{C}$. Formally, the parsimonious labeling problem amounts to minimizing the following energy function:

$$E(\mathbf{x}) = \sum_{p \in \mathbf{V}} \theta_p(x_p) + \sum_{\mathbf{C} \in \mathcal{C}} w_{\mathbf{C}}\delta(\Gamma(\mathbf{x}_{\mathbf{C}})) \tag{25}$$

Therefore, given a clique $\mathbf{x}_{\mathbf{C}}$ and the set of unique labels $\Gamma(\mathbf{x}_{\mathbf{C}})$ assigned to the random variables

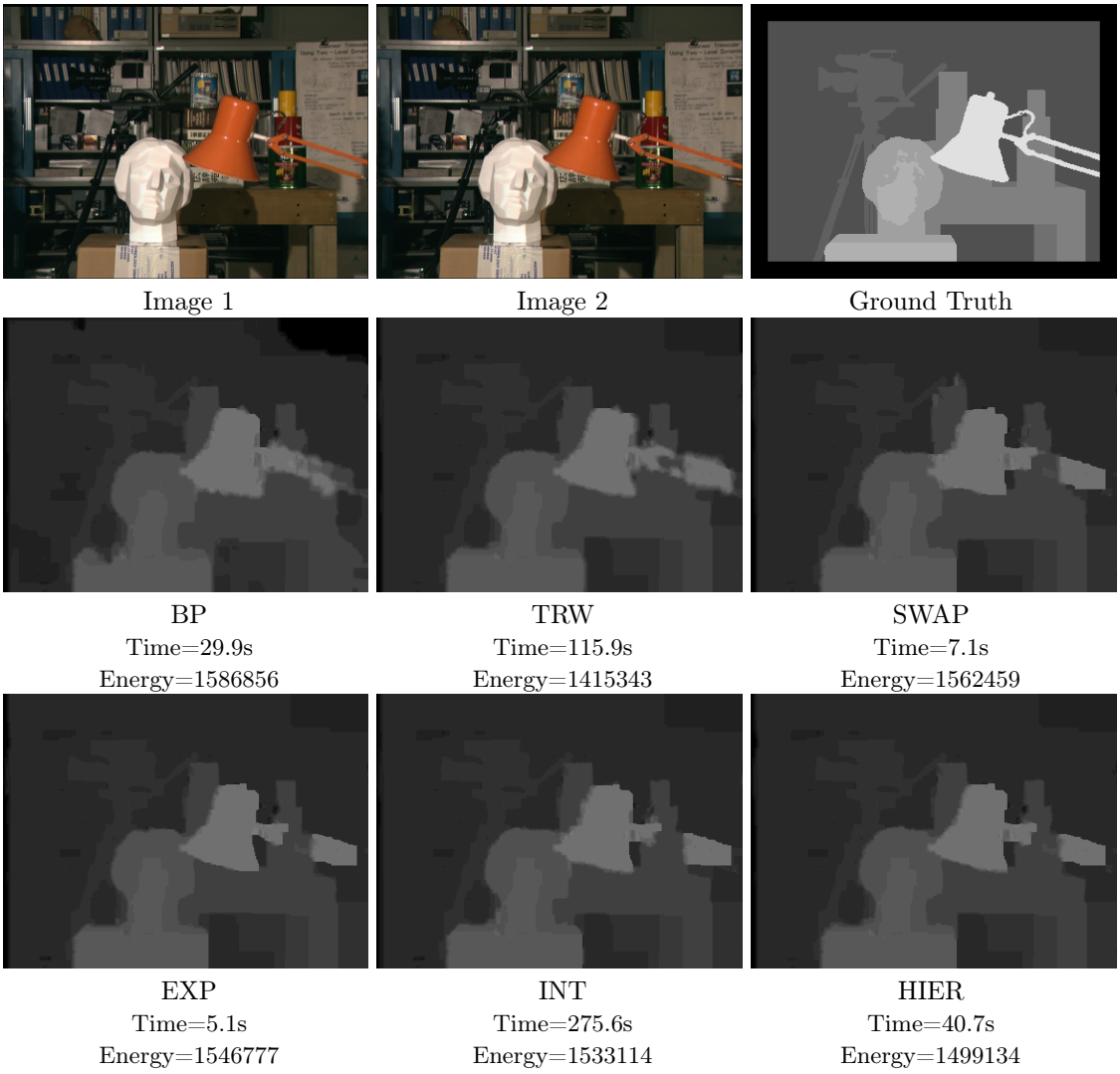


Figure 7: *Results for the 'tsukuba' image pair with truncated quadratic pairwise potentials.*

in the clique, the clique potential function for the parsimonious labeling problem is defined using $\delta(\Gamma(\mathbf{x_C}))$, where $\delta : \Gamma(\mathbf{x_C}) \rightarrow \mathbb{R}$ is a diversity function.

It can be shown that if the size of all the cliques $\mathbf{C} \in \mathcal{C}$ is 2 (that is, they are pairwise cliques), then parsimonious labeling is equivalent to metric labeling. Specifically, we can understand the connection between metrics and diversities using the observation that every diversity induces a metric. In other words, consider $d(l_i, l_i) = \delta(l_i) = 0$ and $d(l_i, l_j) = \delta(\{l_i, l_j\})$. Using the properties of diversities, it can be shown that $d(\cdot, \cdot)$ is a metric distance function. Hence, in case of energy function defined over pairwise cliques, the parsimonious labeling problem reduces to the metric labeling problem.

We are primarily interested in the *diameter diversity* [7], which can be defined as follows. Let $(\mathbf{L}, \delta)$ be a diversity and $(\mathbf{L}, d)$ be the induced metric of $(\mathbf{L}, \delta)$, where $d : \mathbf{L} \times \mathbf{L} \rightarrow \mathbb{R}$ and

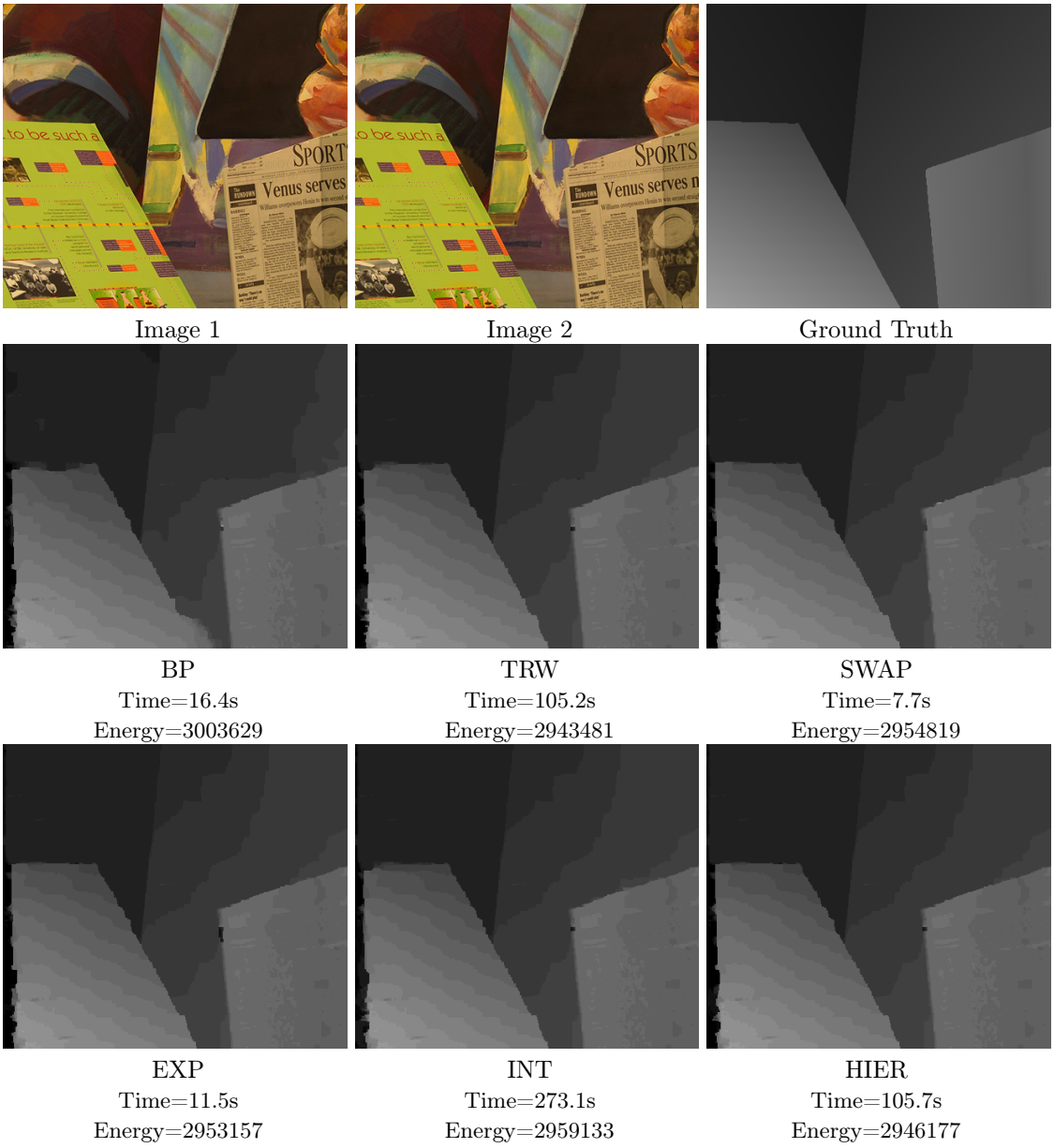


Figure 8: *Results for the 'venus' image pair with truncated linear pairwise potentials.*

$d(l_i, l_j) = \delta(\{l_i, l_j\}), \forall l_i, l_j \in \mathbf{L}$, then for all $\Gamma \subseteq \mathbf{L}$, the diameter diversity is defined as:

$$\delta^{dia}(\Gamma) = \max_{l_i, l_j \in \Gamma} d(l_i, l_j). \tag{26}$$

Clearly, given the induced metric function defined over a set of labels, diameter diversity over any subset of labels gives the measure of the dissimilarity (or diversity) of the labels. More the dissimilarity, based on the induced metric function, higher is the diameter diversity. Therefore,

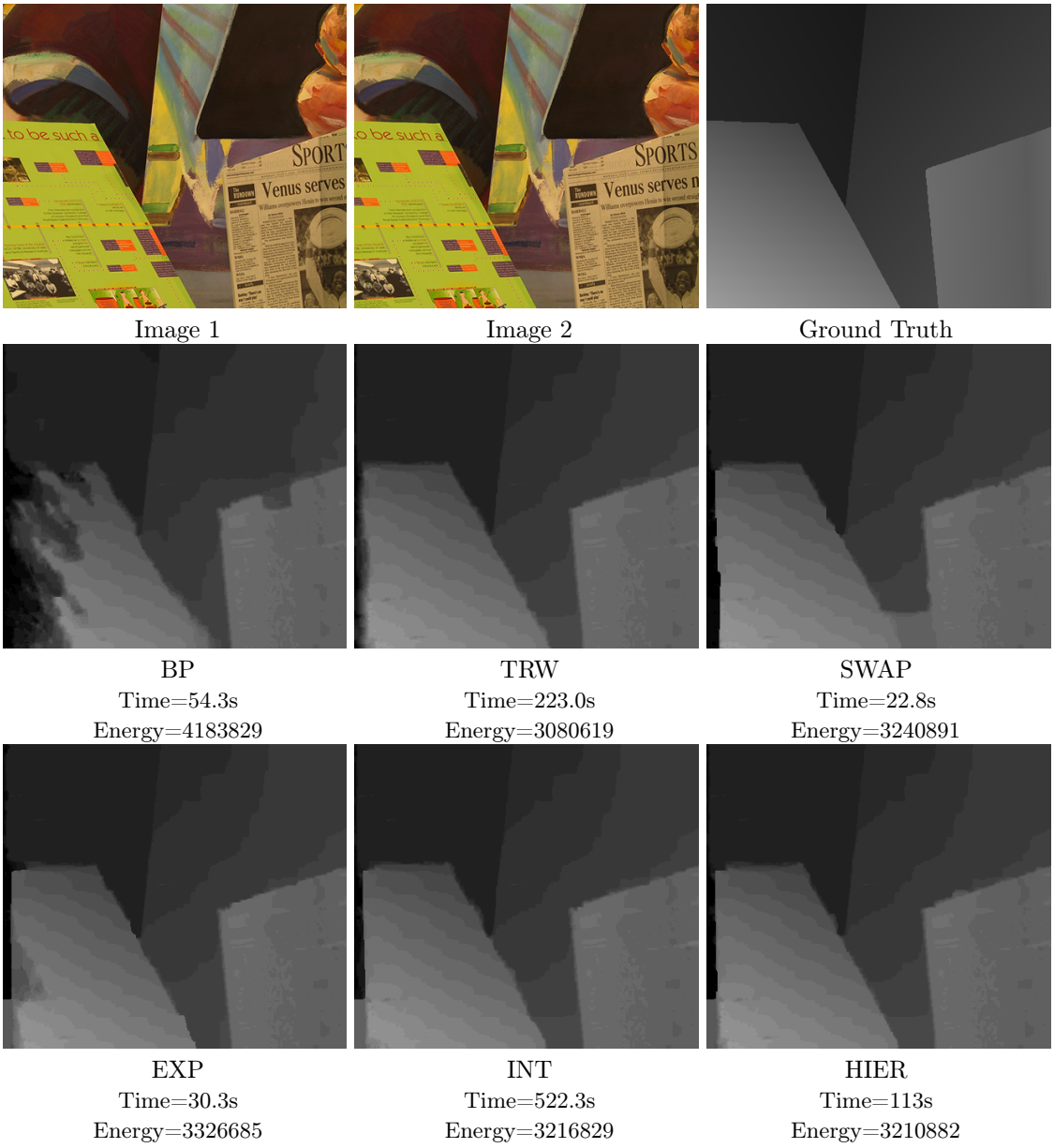


Figure 9: *Results for the 'venus' image pair with truncated quadratic pairwise potentials.*

using diameter diversity as clique potentials enforces the similar labels to be together. Thus, a special case of parsimonious labeling in which the clique potentials are of the form of diameter diversity can be defined as below:

$$E(\mathbf{x}) = \sum_{p \in \mathbf{V}} \theta_p(x_p) + \sum_{\mathbf{C} \in \mathcal{C}} w_{\mathbf{C}} \delta^{dia}(\Gamma(\mathbf{x}_{\mathbf{C}})) \tag{27}$$

In the following two subsections, we consider two special cases of parsimonious labeling defined

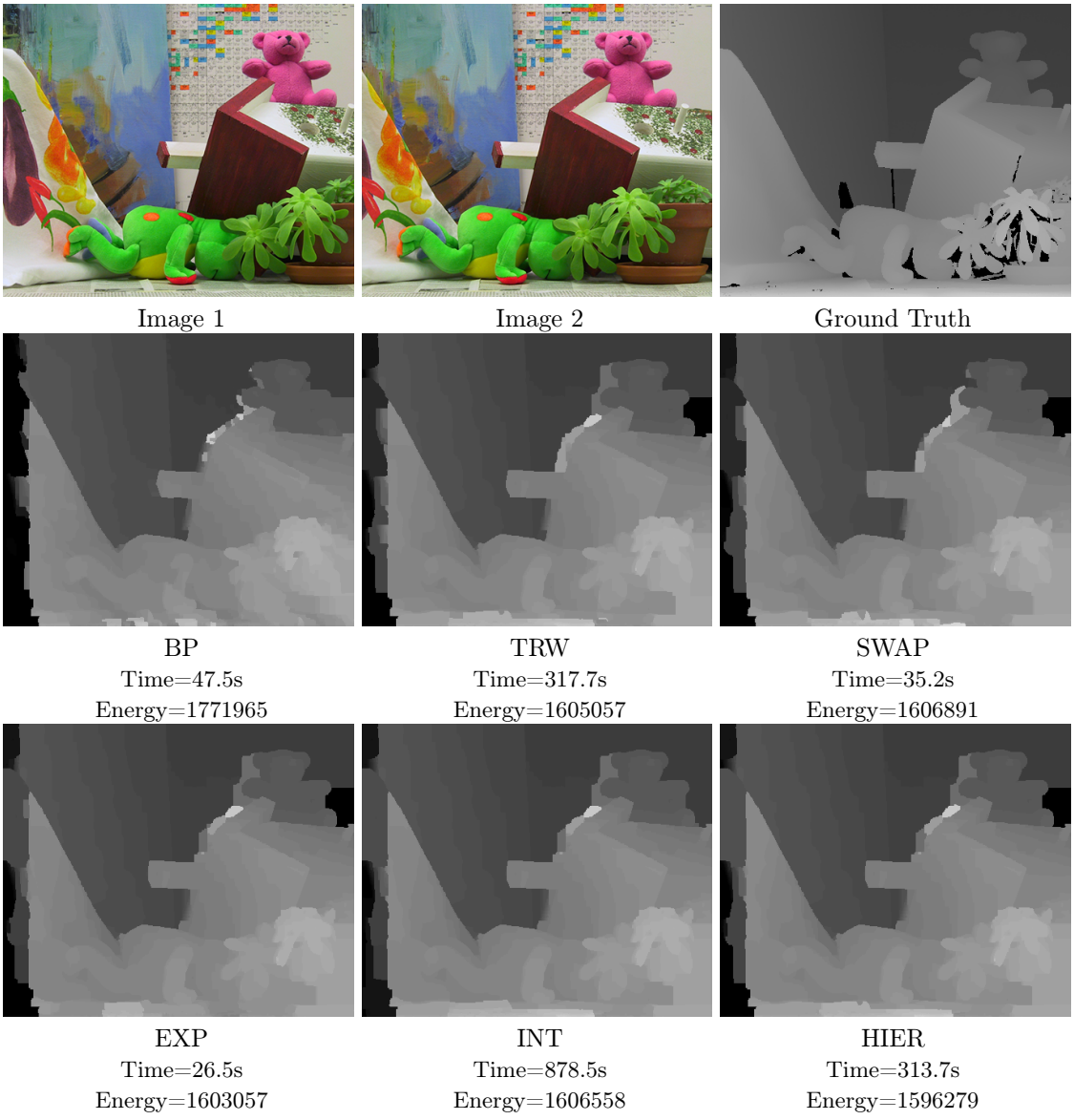


Figure 10: *Results for the 'teddy' image pair with truncated linear pairwise potentials.*

by diameter diversities, which are conducive to move-making algorithms. In subsection 3.6.4, we show how the move-making algorithms designed for the two special cases can be used to solve the general parsimonious labeling problem with strong theoretical guarantees.

3.6.2 P^n Potts model

The P^n Potts model, introduced by kohlipami09, corresponds to the diameter diversity defined by the uniform metric. Formally, the value of the clique potential $\theta_{\mathbf{C}}(\mathbf{x}_C)$ defined by the P^n

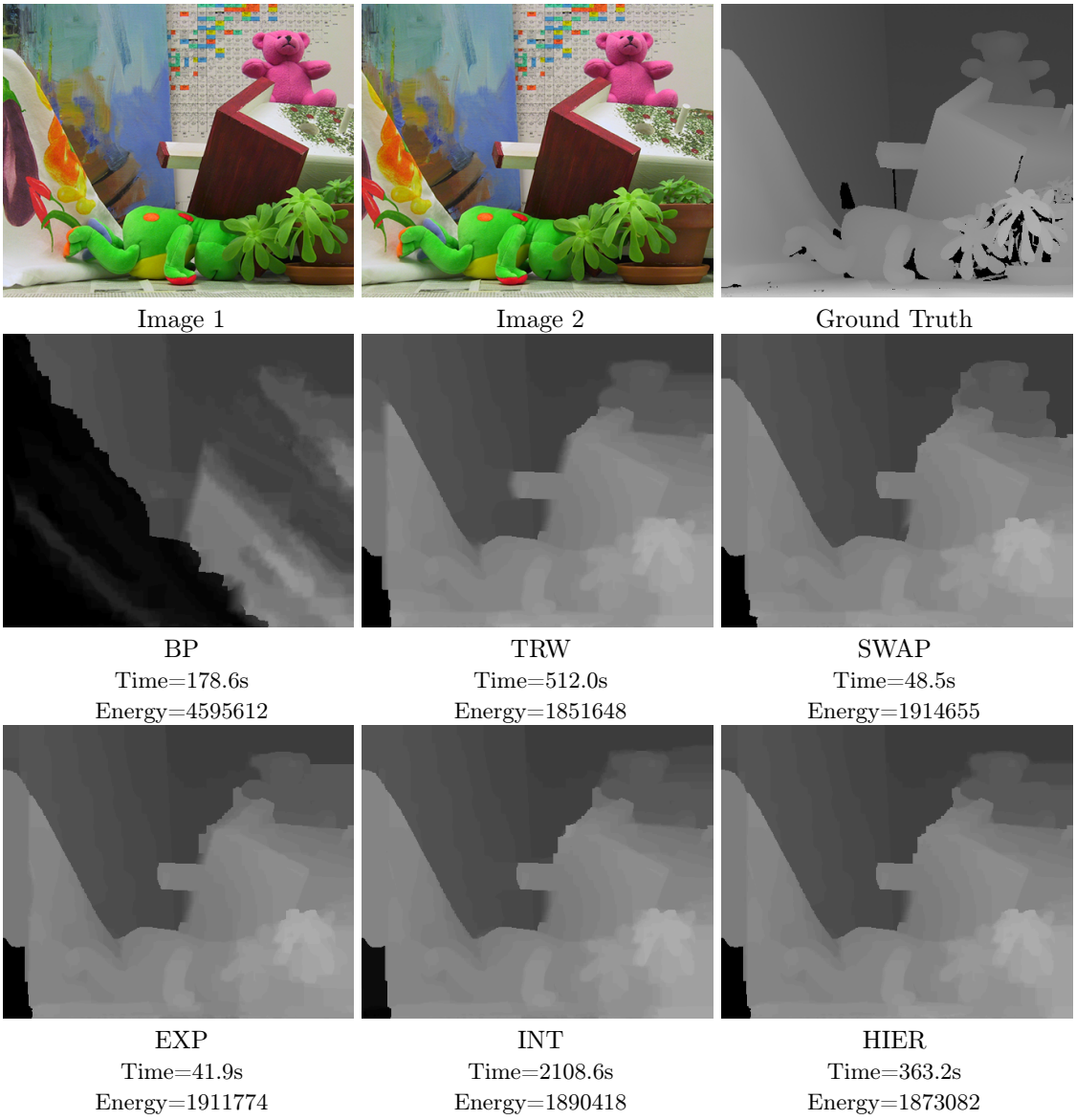


Figure 11: *Results for the 'teddy' image pair with truncated quadratic pairwise potentials.*

Potts model is as follows:

$$\theta_{\mathbf{C}}(\mathbf{x}_{\mathbf{C}}) = \begin{cases} 0, & \text{if there exists } l_k \in \mathbf{L} \text{ such that } \mathbf{x}_{\mathbf{C}} = \{l_k\}^{|\mathbf{C}|}, \\ w_{\mathbf{C}}, & \text{otherwise.} \end{cases} \tag{28}$$

In other words, if all the random variables that constitute the clique $\mathbf{C}$ are assigned the same label, then the value of the clique potential is 0, and otherwise it is equal to the clique weight $w_{\mathbf{C}}$. Since the clique weights are non-negative, the P^n Potts model encourages label consistency over a set of random variables that belong to the same clique. Note that, when $|\mathbf{C}| = 2$, the

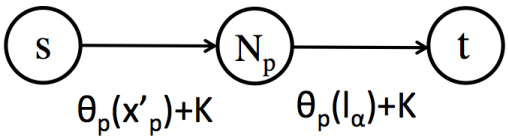


Figure 12: *The arcs that represent the unary potentials of a random variable $p \in \mathbf{V}$ during a single expansion move. The constant K is chosen to be sufficiently large to ensure that the arc capacities are non-negative. Note that since the random variable p has to be assigned exactly one label, adding a constant to the arc capacities of (s, N_p) and (N_p, t) simply adds a constant to the capacity of every st-cut, thereby not affecting the st-MINCUT solution.*

above model reduces to the well-known Potts model (that is, metric labeling defined using the uniform metric) [51].

In order to solve the labeling problem corresponding to the P^n Potts model, kohlipami09 proposed to use the expansion algorithm (that is, the interval move-making algorithm with the length of the interval $a = 1$). Recall that the expansion algorithm starts with an initial labeling $\mathbf{x}'$, for example, by assigning $x'_p = l_1$ for all $p \in \mathbf{V}$. At each iteration, the algorithm moves to a new labeling by searching over a move space. For the expansion algorithm, the move space is defined as the set of labelings where each random variable is either assigned its current label or the label l_α. The key result that makes the expansion algorithm a computationally feasible algorithm for the P^n Potts model is that the minimum energy labeling within a move-space can be obtained using a single *st*-MINCUT operation on a graph that consists of a small number of nodes and arcs.

In more detail, each expansion move is solved by finding the minimum cut on a directed graph $\mathbf{D} = (\mathbf{N} \cup \mathbf{U} \cup \mathbf{W} \cup \{s, t\}, \mathbf{A})$. The nodes s and t are the source and the sink nodes respectively. The nodes $\mathbf{U}$ and $\mathbf{W}$ are auxiliary nodes. As we will see shortly, the auxiliary nodes play a crucial role in representing the P^n Potts model. The arcs $\mathbf{A}$ are of two types. The first type represents the unary potentials, as shown in Figure 12. Note that, according to our convention, if $N_p \in \mathbf{N}_1$ then we assign the random variable $p \in \mathbf{V}$ to its current label x'_p. Otherwise, we assign the random variable p to the new label l_α. It can be verified that the contribution of the arcs in Figure 12 to the corresponding *st*-cut is equal to the unary potential value up to a constant. The second type of arcs represent the high-order potentials, as shown in Figure 13. Again, it can be verified that the contribution of the arcs to the corresponding *st*-cut is equal to the P^n Potts model value up to a constant. The algorithm terminates when the energy cannot be reduced further for any choice of the label l_α.

Multiplicative bound. Once again, similar to the metric labeling case, the multiplicative bound of an algorithm is said to be B if the following condition is satisfied for all possible values of the unary potential $\theta_p(\cdot)$ and clique weights $w_{\mathbf{C}}$:

$$\sum_{p \in \mathbf{V}} \theta_p(x'_p) + \sum_{\mathbf{C} \in \mathcal{C}} w_{\mathbf{C}} \delta(\Gamma(\mathbf{x}'_{\mathbf{C}})) \leq \sum_{p \in \mathbf{V}} \theta_p(x^*_p) + B \sum_{\mathbf{C} \in \mathcal{C}} w_{\mathbf{C}} \delta(\Gamma(\mathbf{x}^*_{\mathbf{C}})). \tag{29}$$

Here, $\mathbf{x}'$ is the labeling estimated by the algorithm and $\mathbf{x}^*$ is an optimal labeling. Theorem 3.6.2 gives the multiplicative bound of the expansion algorithm for the P^n Potts model. The expansion algorithm provides a multiplicative bound of $2 \min\{|\mathbf{L}|, \max_{\mathbf{C} \in \mathcal{C}} |\mathbf{C}|\}$ for the P^n Potts model. We refer the reader to [28] for details.

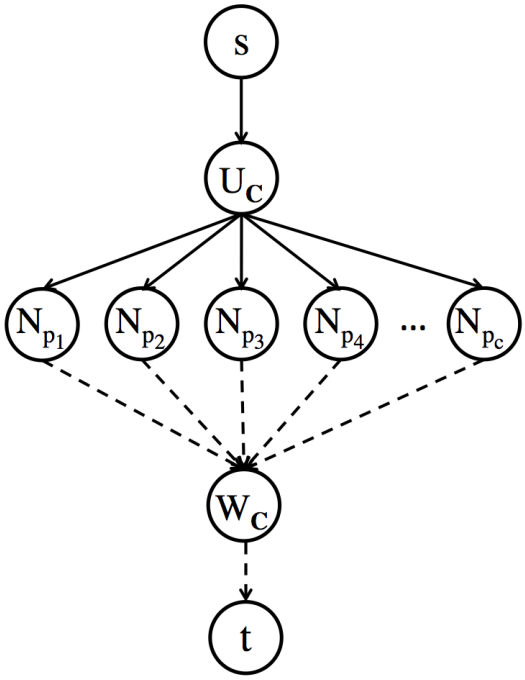


Figure 13: *The arcs that represent the high-order clique potentials of a clique* $\mathbf{C} = \{p_1, \cdots, p_c\}$ *during a single expansion move. The arcs depicted as solid lines have the capacity* $K_{\mathbf{C}} + \theta_{\mathbf{C}}(\mathbf{x}'_{\mathbf{C}})$*, where* $\mathbf{x}'_{\mathbf{C}}$ *is the current labeling of the clique* $\mathbf{C}$*. The arcs depicted as dashed lines have the capacity* $K_{\mathbf{C}} + \theta_{\mathbf{C}}(\{l_\alpha\}^c)$*, where* c *is the size of the clique. By choosing the constant* $K_{\mathbf{C}} = w_{\mathbf{C}} - \theta_{\mathbf{C}}(\mathbf{x}'_{\mathbf{C}}) - \theta_{\mathbf{C}}(\{l_\alpha\}^c)$*, we can ensure that the arc capacities are non-negative. When any of the nodes* $N_{p_i} \in \mathbf{N}_2$*, then the st-MINCUT will include the auxiliary variable* $U_{\mathbf{C}} \in \mathbf{N}_2$*. Similarly, when anyof the nodes* $N_{p_i} \in \mathbf{N}_1$*, then the st-MINCUT will include the auxiliary variable* $W_{\mathbf{C}} \in \mathbf{N}_1$*. It can be verified that this will result in a contribution that is equal to the corresponding value of the clique potential up to a constant* $K_{\mathbf{C}}$*.*

Generalized P^n Potts model. It is worth noting that the above graph construction can be applied to a more general P^n Potts model described by kohlipami09. Specifically, it can be easily modified to compute the optimal expansion move when the value of the clique potential is specified as follows:

$$\theta_{\mathbf{C}}(\mathbf{x}_{\mathbf{C}}) \propto \begin{cases} \gamma_k, & \text{if } x_p = l_k, \forall p \in \mathbf{C}, \\ \gamma_{max}, & \text{otherwise,} \end{cases} \tag{30}$$

where γ_k is the cost of assigning the label l_k to all the random variables $p \in \mathbf{C}$ and $\gamma_{max} > \gamma_k, \forall l_k \in \mathbf{L}$.

3.6.3 Hierarchical P^n Potts model

In the hierarchical P^n Potts model the clique potentials are given by the diameter diversity defined over a given r-HST metric (see section 3.4 for the definition of r-HST metrics). The move making algorithm for the hierarchical P^n Potts model is a divide-and-conquer based approach, similar to the hierarchical move-making algorithm. Instead of solving the actual problem, we divide the problem into smaller subproblems where each subproblem is a P^n Potts model, which can be solved efficiently using the expansion algorithm (as discussed in the previous subsection). More precisely, each node of the given r-HST corresponds to a subproblem. We start with the bottom node of the r-HST, which is a leaf node, and go up in the hierarchy solving each subproblem associated with the nodes encountered.

In more detail, consider a node a of the given r-HST. Recall that any node a in the r-HST is a cluster of labels, which we denote as $\mathbf{L}^a \subseteq \mathbf{L}$ In other words, the leaf nodes of the subtree rooted at a belongs to the subset $\mathbf{L}^a$. Thus, the subproblem defined at node a is to find the labeling $\mathbf{x}^a$ where the label set is restricted to $\mathbf{L}^p$, as defined below.

$$\mathbf{x}^a = \underset{\mathbf{x} \in (\mathbf{L}^a)^N}{\operatorname{argmin}} \Big(\sum_{p \in \mathbf{V}} \theta_p(x_p) + \sum_{\mathbf{C} \in \mathcal{C}} w_{\mathbf{C}} \delta^{dia}(\Gamma(\mathbf{x}_{\mathbf{C}})) \Big) \tag{31}$$

If a is the root node, then the above problem is as difficult as the original labeling problem (since $\mathbf{L}^a = \mathcal{L}$). However, if a is the leaf node then the solution of the problem associated with a is trivial, $x_p^a = l_a$ for all $p \in \mathbf{V}$, where l_a is the label associated with the leaf node a. This insight leads to the design of an approximation algorithm, where we start by solving the simple problems corresponding to the leaf nodes, and use the labelings obtained to address the more difficult problem further up the hierarchy. In what follows, we describe how the labeling of the problem associated with the node a, when a is a non-leaf node, is obtained using the labelings of its chidren node.

Solving the parent labeling problem. Before delving into the details, let us define some notations for the purpose of clarity. Let T be the depth (or the number of levels) of the given r-HST and $\mathcal{N}(\tau)$ be the set of nodes at level τ. The root node is at the top level ($\tau = 1$). Let $\eta(a)$ denote the set of child nodes associated with a non-leaf node a and $\eta(a, k)$ denotes its k^{th} child node. Recall that our approach is bottom up. Therefore, for each child node of a we already have an associated labeling. We denote the labeling associated with the k^{th} child of the node a as $\mathbf{x}^{\eta(a,k)}$. Thus, $x_p^{\eta(a,k)}$ denotes the label assigned to the random variable p by the labeling of the k^{th} child of the node a. We also define an n dimensional vector $\mathbf{t}^a \in \{1, \cdots, |\eta(a)|\}^N$, where $|\eta(a)|$ is the number of child nodes of node a. More precisely, for a given $\mathbf{t}^a$, $t_p^a = k$ denotes that the label for the random variable p comes from the k^{th} child of the node a. Therefore, the labeling problem at node a reduces to finding the optimal $\mathbf{t}^a$. In other words, the labeling

problem at node a amounts to finding the best child index $k \in \{1, \cdots, |\eta(a)|\}$ for each random variable $p \in \mathbf{V}$ so that the label assigned to the random variable comes from the labeling of the k^{th} child.

Using the above notations, associated with a $\mathbf{t}^a$ we define a new energy function as:

$$E(\mathbf{t}^a) = \sum_{p \in \mathbf{V}} \bar{\theta}_p(t_p^a) + \sum_{\mathbf{C} \in \mathcal{C}} w_{\mathbf{C}} \bar{\theta}_{\mathbf{C}}(\mathbf{t}_{\mathbf{C}}^a) \tag{32}$$

where

$$\bar{\theta}_p(t_p^a) = \theta_p(x_p^{\eta(a,k)}) \quad if \quad t_p^a = k. \tag{33}$$

In other words, the unary potential for $t_p^a = k$ is the unary potential associated to the random variable p corresponding to the label $x_p^{\eta(a,k)}$. Similarly, the new clique potential $\bar{\theta}_{\mathbf{C}}(\mathbf{t}_{\mathbf{C}}^a)$ is as defined below:

$$\bar{\theta}_{\mathbf{C}}(\mathbf{t}_{\mathbf{C}}^a) = \begin{cases} \gamma_k^a, & \text{if } t_p^a = k, \forall p \in \mathbf{C} \\ \gamma_{max}^a, & \text{otherwise} \end{cases} \tag{34}$$

where $\gamma_k^a = \delta_{dia}(\Gamma(\mathbf{x}_{\mathbf{C}}^{\eta(a,k)}))$ is the diameter diversity of the set of *unique labels* associated with $\mathbf{x}_{\mathbf{C}}^{\eta(a,k)}$ and $\gamma_{max}^a = \delta_{dia}(\bar{\mathbf{L}}^a)$. The set $\bar{\mathbf{L}}^a = \cup_{k \in \eta(a)} \Gamma(\mathbf{x}_{\mathbf{C}}^{\eta(a,k)})$ is the union of the unique labels associated with the child nodes of a. Recall that, because of the construction of the r-HST, $\mathbf{L}^b \subset \bar{\mathbf{L}}^a \subseteq \mathbf{L}^a$ for all $b \in \eta(a)$. Hence, the monotonicity property of the diameter diversity ensures that $\gamma_{max}^a > \gamma_k^a, \forall k \in \eta(a)$. This is the sufficient criterion to prove that the potential function defined by equation (34) is a (generalized) P^n Potts model. Therefore, the expansion algorithm can be used to obtain the locally optimal $\mathbf{t}^a$ for the energy function (32). Given the locally optimal $\hat{\mathbf{t}}^a$, the labeling $\mathbf{x}^a$ at node a can be obtained as follows: $x_p^a = x_p^{\eta(a,\hat{t}_p^a)}$. In other words, the final label of the random variable p is the one assigned to it by the $(\hat{t}_p^a)^{th}$ child of node a. The making algorithm for the hierarchical P^n Potts model is shown in the Algorithm 7.

Algorithm 7 The move making algorithm for the hierarchical P^n Potts model.

input r-HST metric; $w_{\mathbf{C}}, \forall \mathbf{C} \in \mathcal{C}$; and $\theta_p(x_p), \forall p \in \mathbf{V}$.

1: $\tau = T$, the leaf nodes
2: **repeat**
3: **for** each $a \in \mathcal{N}(\tau)$ **do**
4: **if** $|\eta(a)| = 0$, leaf node **then**
5: $x_p^a = l_a, \forall p \in \mathbf{V}$
6: **else**
7: Fusion Move

$$\hat{\mathbf{t}}^a = \underset{\mathbf{t}^a \in \{1, \cdots, |\eta(a)|\}^n}{\operatorname{argmin}} E(\mathbf{t}^a) \tag{35}$$

8: $x_p^a = x_p^{\eta(a,\hat{t}_p^a)}$.
9: **end if**
10: **end for**
11: $\tau \leftarrow \tau - 1$
12: **until** $\tau = 0$.

Algorithm 8 The move making algorithm for the parsimonious labeling problem.

input Diversity $(\mathbf{L}, \delta)$; $w_{\mathbf{C}}, \forall \mathbf{C} \in \mathbf{C}$; $\theta_p(\cdot), \forall p \in \mathbf{V}$; $\mathbf{L}$; k

1: Approximate the given diversity as the mixture of k hierarchical P^n Potts model using Algorithm 9.
2: **for** each hierarchical P^n Potts model in the mixture **do**
3: Use the hierarchical move making algorithm defined in the Algorithm 7.
4: Compute energy corresponding to the solution obtained.
5: **end for**
6: Choose the solution with the minimum energy.

Algorithm 9 Diversity to mixture of hierarchical P^n Potts model.

input Diversity $(\mathbf{L}, \delta)$, k

1: Compute the induced metric, $d(\cdot, \cdot)$, where $d(l_i, l_j) = \delta(\{l_i, l_j\}), \forall l_i, l_j \in \mathbf{L}$.
2: Approximate $d(.,.)$ into mixture of k r-HST metrics $d^t(.,.)$ using the algorithm proposed in [13].
3: **for** each r-HST metrics $d^t(\cdot, \cdot)$ **do**
4: Obtain the corresponding hierarchical P^n Potts model by defining the diameter diversity over $d^t(\cdot, \cdot)$
5: **end for**

Multiplicative bound. Theorem 3.6.3 gives the multiplicative bound for the move making algorithm for the hierarchical P^n Potts model. The move making algorithm for the hierarchical P^n Potts model, Algorithm 7, gives the multiplicative bound of $\left(\frac{r}{r-1}\right) \min\{|\mathbf{L}|, \max_{\mathbf{C} \in \mathcal{C}} |\mathbf{C}|\}$.

3.6.4 Handling general diversities

In the previous subsection, we proposed a hierarchical move making algorithm for the hierarchical P^n Potts model (Algorithm 7). This restricted us to a small class of clique potentials. In this section we generalize our approach to the much more general *parsimonious labeling problem*.

The move making algorithm for the parsimonious labeling problem is shown in Algorithm 8. Given diversity based clique potentials, non-negative clique weights, and arbitrary unary potentials, Algorithm 8 approximates the diversity into a *mixture of hierarchical P^n Potts models* (using Algorithm 9) and then use the previously defined Algorithm 7 on each of the hierarchical P^n Potts models.

The algorithm for approximating a given diversity into a mixture of hierarchical P^n Potts models is shown in Algorithm 9. The first and the third steps of the Algorithm 9 have already been discussed in the previous sections. The second step, which amounts to finding the mixture of r-HST metrics for a given metric, can be solved using the randomized algorithm proposed by Fakcharoenphol04TightBound.

Multiplicative Bound. Theorem 3.6.4 gives the multiplicative bound for the parsimonious labeling problem, when the clique potentials are any general diversity. The move making algorithm defined in Algorithm 8 gives the multiplicative bound of $\left(\frac{r}{r-1}\right) (|\mathbf{L}|-1) O(\log |\mathbf{L}|) \min\{|\mathbf{L}|, \max_{\mathbf{C} \in \mathcal{C}} |\mathbf{C}|\}$ for the parsimonious labeling problem (equation (25)).

3.6.5 Related works

A robust version of the P^n Potts model was proposed by kohliijcv09, which takes into account the number of random variables that have been assigned an inconsistent label. Similar to the P^n Potts model, its robust version also lends itself to the efficient expansion algorithm, which provides a multiplicative bound on the quality of the solution. While the robust P^n Potts model has been shown to be very useful for semantic segmentation, parsimonious labeling offers a natural extension of the metric labeling problem and is therefore more widely applicable to several low-level vision tasks.

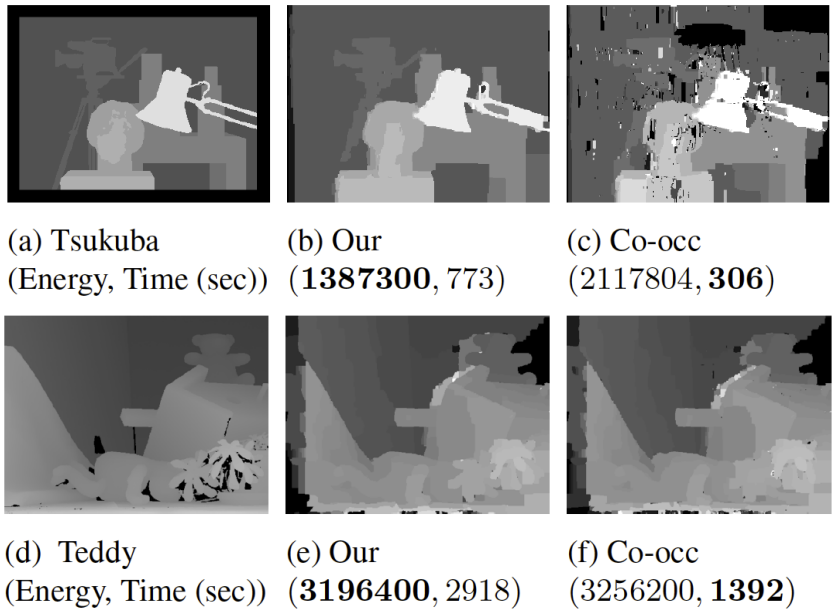
[11] proposed a global clique potential (label cost) that is based on the cost of using a label or a subset of labels in the labeling of the random variables. Similar to the P^n Potts model, the label cost based potential can also be minimized using the expansion algorithm. However, the theoretical guarantee provided by the expansion algorithm is an additive bound, which is not invariant to reparameterization of the energy function. [10] also proposed an extension of their work to hierarchical costs. However, the assumption of a given hierarchy over the label set limits its applications.

ladickyijcv13 proposed a global co-occurrence cost based high order model for a much wider class of energies that encourage the use of a small set of labels in the estimated labeling. Theoretically, the only constraint that [42] enforces in high order clique potential is that it should be monotonic in the label set. In other words, the family of energy functions considered by ladickyijcv13 can be regarded as a generalization of parsimonious labeling. However, they approximately optimize an upperbound on the actual energy function which does not provide any optimality guarantees.

fixiccv11 proposed an algorithm (SoSPD) for high-order random fields with arbitrary clique potentials. Each move of this algorithm requires us to approximately upperbound the clique potential into a submodular function and then optimize it using the submodular max-flow algorithm [31]. However, the parsimonious labeling move making algorithm has a much stronger multiplicative bound and better time complexity compared to [16].

3.6.6 Dense stereo correspondence

Given two rectified stereo pair of images, we wish to find the disparity of each pixel in the reference image. We extend the standard setting of stereo matching to high-order cliques and test our method to the images, 'tsukuba' and 'teddy'. We define the high-order cliques using the superpixels obtained using the mean-shift method [9]. The clique potentials are the diameter diversity of the truncated linear metric. A truncated linear metric enforces smoothness in the pairwise setting, therefore, its diameter diversity will naturally enforce smoothness in the high-order cliques, which is a desired cue for the two applications we are dealing with. We use $w_{\mathbf{C}} = \exp^{-\frac{\rho(\mathbf{x}_{\mathbf{C}})}{\sigma^2}}$, where $\rho(\mathbf{x}_{\mathbf{C}})$ is the variance of the intensities of the pixels in the clique $\mathbf{x}_c$ and σ is a hyperparameter. The unary potentials are the ℓ_1 norm of the difference in the RGB values of the left and the right image pixels. Note that the index for the right image pixel is the index for the left image pixel minus the disparity, where disparity is the label. For 'tsukuba' and 'teddy' we used 16 and 60 labels respectively. In case of 'teddy' the unary potentials are trucated at 16. The weights $w_{\mathbf{C}}$ for the cliques of size 2 (that is, for pairwise potentials) are set to be proportional to the ℓ_1 norm of the gradient of the intensities of the neighboring pixels ∇. In case of 'tsukuba', if $\nabla < 8$, $w_{\mathbf{C}} = 2$, otherwise $w_{\mathbf{C}} = 1$. In case of 'teddy', if $\nabla < 10$, $w_{\mathbf{C}} = 3$, otherwise $w_{\mathbf{C}} = 1$. Figure 14 shows the results obtained. Note that the move making algorithm for parsimonious labeling significantly outperforms [42] in terms of energy and the visual quality for both 'tsukuba' and 'teddy'.



(a) Tsukuba (Energy, Time (sec)) (b) Our (**1387300**, 773) (c) Co-occ (2117804, **306**)

(d) Teddy (Energy, Time (sec)) (e) Our (**3196400**, 2918) (f) Co-occ (3256200, **1392**)

Figure 14: *Stereo Matching Results. Figures (a) and (d) are the ground truth disparity for the 'tsukuba' and 'teddy' respectively. The move-making method for parsimonious labeling significantly outperforms the baseline Co-ooc [42] in both the cases in terms of energy. The results are also more visually appealing. Figures (b) and (e) clearly shows the influence of 'parsimonious labeling' as the regions are smooth and the discontinuity is preserved. Recall that we use super-pixels obtained using the mean-shift as the cliques.*

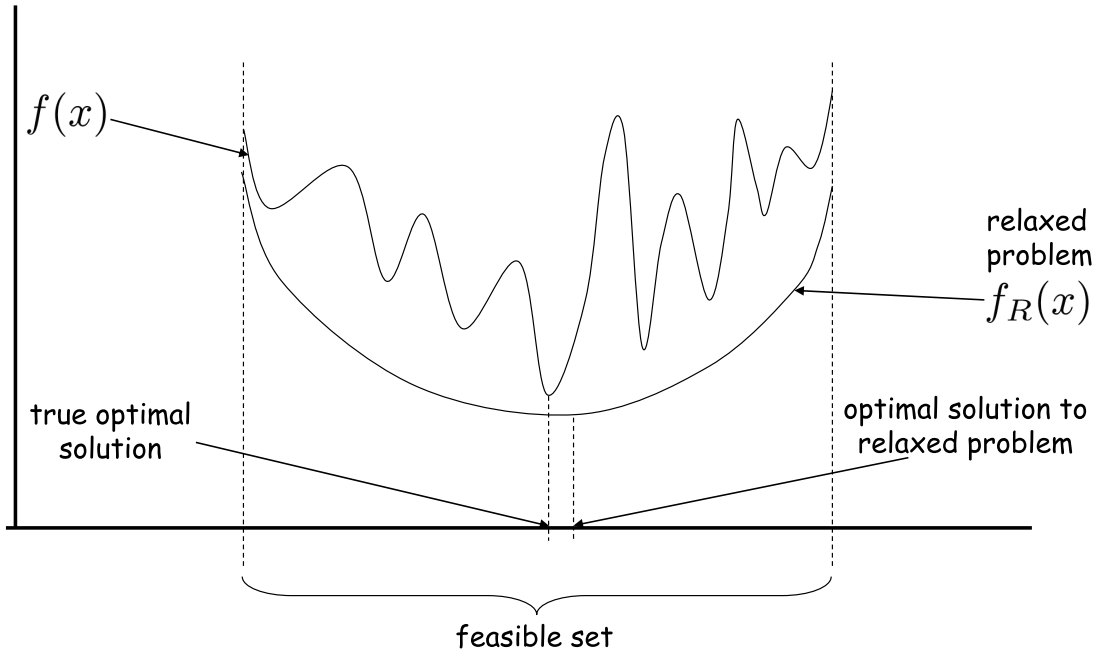


Fig. 15: Toy example to illustrate the idea behind the *relaxation principle*: here the function $f(x)$ is highly non-convex with many local minima, and is thus to difficult to optimize. By approximating it with a smooth convex function $f_R(x)$ and optimizing the resulting relaxed problem (which can be done very easily using, *e.g.*, even gradient descent), we end up with a solution that is close to the optimum of $f(x)$.

4 MRF inference based on the primal-dual schema

As already mentioned in the introduction, many labeling problems in computer vision and image analysis can often be stated as *integer linear programs* (ILPs), which can be expressed under the following form:

$$\text{Primal-ILP:} \quad \begin{array}{ll} \underset{x \in \mathbb{R}^N}{\text{minimize}} & c^\top x \\ \text{s.t.} & Lx \geq b, \quad x \in \mathcal{N} \subset \mathbb{N}^N, \end{array}$$

where $L = (L^{(i,j)})_{1 \leq i \leq K, 1 \leq j \leq N}$ represents a matrix of size $K \times N$, and $b = (b^{(i)})_{1 \leq i \leq K}$, $c = (c^{(j)})_{1 \leq j \leq N}$ are column vectors of size K and N, respectively. Among the problems encountered in practice, many of them lead to a Primal-ILP that is NP-hard to solve. In such cases, a principled approach, as we shall see, for finding an approximate solution is through the use of convex *relaxations*, where the original NP-hard problem is approximated with a surrogate one (the so-called relaxed problem), which is convex and thus much easier to solve (see Fig. 15). The premise is the following: to the extent that the surrogate problem provides a reasonably good approximation to the original optimization task, one can expect to obtain an approximately optimal solution for the latter by essentially making use of or solving the former.

The type of relaxations that are typically preferred in large scale discrete optimization are based on linear programming, involving the minimization of a linear function subject to linear

inequality constraints. These can be naturally obtained by simply relaxing the integrality constraints of Primal-ILP, thus leading to the relaxed primal problem as well as its dual, denoted respectively by Primal-LP and Dual-LP

$$\text{Primal-LP}: \quad \underset{x\in[0,+\infty[^N}{\text{minimize}} \; c^\top x \quad \text{s.t.} \quad Lx \geq b, \tag{36}$$

$$\text{Dual-LP}: \quad \underset{y\in[0,+\infty[^K}{\text{maximize}} \; b^\top y \quad \text{s.t.} \quad L^\top y \leq c. \tag{37}$$

It should be noted that the use of LP-relaxations is often dictated by the need of maintaining a reasonable computational cost. Although more powerful convex relaxations do exist in many cases, these may become intractable as the number of variables grows larger, especially for Semidefinite Programming (SDP) or Second-Order Cone Programming (SOCP) relaxations.

Based on the above observations, in the following we aim to present some very general optimization strategies that can be used in this context, focusing a lot on their underlying principles, which are based on two powerful techniques, the so-called *primal-dual schema* and *dual decomposition*. As we shall see, in order to estimate an approximate solution to Primal-ILP, both approaches make heavy use of the dual of the underlying LP relaxation, i.e., Problem (37). But their strategies for doing so is quite different: the second one essentially aims at solving this dual LP (and then converting the fractional solution into an integral one, trying not to increase the cost too much in the process), whereas the first one simply uses it in the design of the algorithm.

4.1 Relaxations and discrete optimization

Relaxations are very useful for solving approximately discrete optimization problems. Formally, given a problem

$$(\mathcal{P}): \underset{x\in C}{\text{minimize}} \; f(x)$$

where C is a subset of $\mathbb{R}^N$, we say that

$$(\mathcal{P}'): \underset{x\in C'}{\text{minimize}} \; f'(x)$$

with $C' \subset \mathbb{R}^N$ is a relaxation of $(\mathcal{P})$ if and only if (i) $C \subset C'$, and (ii) $(\forall x \in C')\ f(x) \geq f'(x)$.

For instance, let us consider the integer linear program defined by $(\forall x \in \mathbb{R}^N)\ f(x) = c^\top x$ and $C = S \cap \mathbb{Z}^N$, where $c \in \mathbb{R}^N \setminus \{0\}$ and S is a nonempty closed polyhedron defined as

$$S = \left\{ x \in \mathbb{R}^N \;\middle|\; Lx \geq b \right\}$$

with $L \in \mathbb{R}^{K\times N}$ and $b \in \mathbb{R}^K$. One possible linear programming relaxation of $(\mathcal{P})$ is obtained by setting $f' = f$ and $C' = S$, which is typically much easier than $(\mathcal{P})$ (which is generally NP-hard). The quality of $(\mathcal{P}')$ is quantified by its so-called integrality gap defined as $\frac{\inf f(C)}{\inf f'(C')} \geq 1$ (provided that $-\infty < \inf f'(C') \neq 0$).

Hence, for approximation purposes, LP relaxations are not all of equal value. If

$$(\mathcal{P}''): \underset{x\in C''}{\text{minimize}} \; c^\top x$$

is another relaxation of $(\mathcal{P})$ with $C'' \subset C'$, then relaxation $(\mathcal{P}'')$ is tighter. Interestingly, $(\mathcal{P})$ always has a tight LP relaxation (with integrality gap 1) given by $C'' = \text{conv}(S \cap \mathbb{Z}^N)$, where $\text{conv}(C)$ is the convex hull polyhedron of C. Note, however, that if $(\mathcal{P})$ is NP-hard, polyhedron $\text{conv}(S \cap \mathbb{Z}^N)$ will involve exponentially many inequalities.

The relaxations in all of the previous examples involve expanding the original feasible set. But, as mentioned, we can also derive relaxations by modifying the original objective function. For instance, in so-called submodular relaxations [30, 26], one uses as new objective a maximum submodular function that lower bounds the original objective. More generally, convex relaxations allow us to make use of the well-developed duality theory of convex programming for dealing with discrete nonconvex problems.

4.2 The primal-dual schema for integer linear programming

The primal-dual schema is a well-known technique in the combinatorial optimization community that has its origins in LP duality theory. It is worth noting that it started as an exact method for solving linear programs. As such, it had initially been used in deriving exact polynomial-time algorithms for many cornerstone problems in combinatorial optimization that have a tight LP relaxation. Its first use probably goes back to Edmond's famous Blossom algorithm for constructing maximum matchings on graphs, but it had been also applied to many other combinatorial problems including max-flow (e.g., Ford and Fulkerson's augmenting path-based techniques for max-flow can essentially be understood in terms of this schema), shortest path, minimum branching, and minimum spanning tree [49]. In all of these cases, the primal-dual schema is driven by the fact that optimal LP solutions should satisfy the *complementary slackness conditions* (see equations (38) and (39) below). More specifically, If $\widehat{x} = (\widehat{x}^{(j)})_{1 \leq j \leq N}$ is a solution to Primal-LP, a solution $\widehat{y} = (\widehat{y}^{(i)})_{1 \leq i \leq K}$ to Dual-LP can be obtained by the *primal complementary slackness condition*:

$$(\forall j \in \{1, \ldots, N\}) \quad \text{such that} \quad \widehat{x}^{(j)} > 0, \qquad \sum_{i=1}^{K} L^{(i,j)}\, \widehat{y}^{(i)} = c^{(j)}, \tag{38}$$

whereas, if $\widehat{y}$ is a solution to Dual-LP, a solution $\widehat{x}$ to Primal-LP can be obtained by the *dual complementary slackness condition*:

$$(\forall i \in \{1, \ldots, K\}) \quad \text{such that} \quad \widehat{y}^{(i)} > 0, \qquad \sum_{j=1}^{N} L^{(i,j)}\, \widehat{x}^{(j)} = b^{(i)}. \tag{39}$$

Starting with an initial primal-dual pair of feasible solutions, the primal-dual schema therefore iteratively steers them towards satisfying the above complementary slackness conditions (by trying at each step to minimize their total violation). Once this is achieved, both solutions (the primal and the dual) are guaranteed to be optimal. Moreover, since the primal is always chosen to be updated integrally during the iterations, it is ensured that an integral optimal solution is obtained at the end. A notable feature of the primal-dual method is that it often reduces the original LP, which is a weighted optimization problem, to a series of purely combinatorial unweighted ones (related to minimizing the violation of complementary slackness conditions at each step).

Interestingly, today the primal-dual schema is no longer used for providing exact algorithms. Instead, its main use concerns deriving approximation algorithms to NP-hard discrete problems that admit an ILP formulation, for which it has proved to be a very powerful and widely applicable tool. As such, it has been applied to many NP-hard combinatorial problems up to now, including set-cover, Steiner-network, scheduling, Steiner tree, feedback vertex set, facility location, to mention only a few [59, 22]. With regard to problems from the domains of computer vision and image analysis, the primal-dual schema has been introduced recently in [36, 37], and has been used for modeling a broad class of tasks from these fields.

It should be noted that for NP-hard ILPs an integral solution is no longer guaranteed to satisfy the complementary slackness conditions (since the LP-relaxation is not exact). How could

it then be possible to apply this schema to such problems? It turns out that the answer to this question consists of using an appropriate relaxation of the above conditions. To understand exactly how we need to proceed in this case, let us consider the problem Primal-ILP above. As already explained, the goal is to compute an optimal solution to it, but, due to the integrality constraints $x \in \mathcal{N}$, this is assumed to be NP-hard, and so we can only estimate an approximate solution. To achieve that, we will first need to relax the integrality constraints, thus giving rise to the relaxed primal problem in (36) as well as its dual (37). A primal-dual algorithm attempts to compute an approximate solution to Primal-ILP by relying on the following principle (see also the framebox "PRIMAL-DUAL PRINCIPLE IN THE DISCRETE CASE" for an explanation):

Primal-dual principle in the discrete case: *Let $x \in \mathbb{R}^N$ and $y \in \mathbb{R}^K$ be integral-primal and dual feasible solutions (i.e. $x \in \mathcal{N}$ and $Lx \geq b$, and $y \in [0, +\infty[^K$ and $L^\top y \leq c$). Assume that there exists $\nu \in [1, +\infty[$ such that*

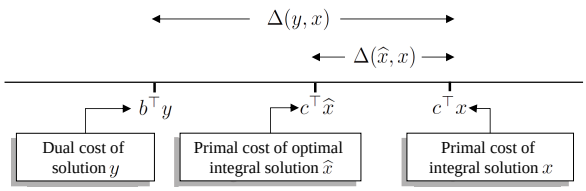
$$c^\top x \leq \nu\, b^\top y. \tag{40}$$

Then, x can be shown to be a ν-approximation to an unknown optimal integral solution $\widehat{x}$, i.e.

$$c^\top \widehat{x} \leq c^\top x \leq \nu\, c^\top \widehat{x}. \tag{41}$$

PRIMAL-DUAL PRINCIPLE IN THE DISCRETE CASE

Essentially, the proof of this principle relies on the fact that the sequence of optimal costs of problems Dual-LP, Primal-LP, and Primal-ILP is increasing.



Specifically, by weak LP duality, the optimal cost of Dual-LP is known to not exceed the optimal cost of Primal-LP. As a result of this fact, the cost $c^\top \widehat{x}$ (of an unknown optimal integral solution $\widehat{x}$) is guaranteed to be at least as large as the cost $b^\top y$ of any dual feasible solution y. On the other hand, by definition, $c^\top \widehat{x}$ cannot exceed the cost $c^\top x$ of an integral-primal feasible solution x. Therefore, if the gap $\Delta(y, x)$ between the costs of y and x is small (e.g., it holds $c^\top x \leq \nu\, b^\top y$), the same will be true for the gap $\Delta(\widehat{x}, x)$ between the costs of $\widehat{x}$ and x (i.e., $c^\top x \leq \nu\, c^\top \widehat{x}$), thus proving that x is a ν-approximation to optimal solution $\widehat{x}$.

Although the above principle lies at the heart of many primal-dual techniques (i.e., in one way or another, primal-dual methods often try to fulfill the assumptions imposed by this principle), it does not directly specify how to estimate a primal-dual pair of solutions (x, y) that satisfies these assumptions. This is where the so-called *relaxed complementary slackness conditions* come into play, as they typically provide an alternative and more convenient (from an algorithmic viewpoint) way for generating such a pair of solutions. These conditions generalize the complementary slackness conditions (38) and (39) associated with an arbitrary pair of primal-dual linear programs. The latter conditions apply only in cases when there is no duality gap, like between Primal-LP and Dual-LP, but they are not applicable to cases like Primal-ILP and Dual-LP, when a duality gap exists as a result of the integrality constraint imposed on variable x. As in the exact case, two types of relaxed complementary slackness conditions exist, depending on whether the primal or dual variables are checked for being zero.

Relaxed Primal Complementary Slackness Conditions with relaxation factor $\nu_{\text{primal}} \leq 1$. For a given $x = (x^{(j)})_{1\leq j\leq N} \in \mathbb{R}^N$, $y = (y^{(i)})_{1\leq i\leq K} \in \mathbb{R}^K$, the following conditions are assumed to hold:

$$(\forall j \in J_x) \qquad \nu_{\text{primal}}\, c^{(j)} \leq \sum_{i=1}^{K} L^{(i,j)} y^{(i)} \leq c^{(j)} \tag{42}$$

where $J_x = \big\{j \in \{1,\dots,N\} \;\big|\; x^{(j)} > 0\big\}$.

Relaxed Dual Complementary Slackness Conditions with relaxation factor $\nu_{\text{dual}} \geq 1$. For a given $y = (y^{(i)})_{1\leq i\leq K} \in \mathbb{R}^K$, $x = (x^{(j)})_{1\leq j\leq N} \in \mathbb{R}^N$, the following conditions are assumed to hold:

$$(\forall i \in I_y) \qquad b^{(i)} \leq \sum_{j=1}^{N} L^{(i,j)} x^{(j)} \leq \nu_{\text{dual}}\, b^{(i)} \tag{43}$$

where $I_y = \big\{i \in \{1,\dots,K\} \;\big|\; y^{(i)} > 0\big\}$.

When both $\nu_{\text{primal}} = 1$ and $\nu_{\text{dual}} = 1$, we recover the exact complementary slackness conditions in (38) and (39). The use of the above conditions in the context of a primal-dual approximation algorithm becomes clear by the following result:

Theorem .1 *If $x = (x^{(j)})_{1\leq j\leq N}$ and $y = (y^{(i)})_{1\leq i\leq K}$ are feasible with respect to* Primal-ILP *and* Dual-LP *respectively, and satisfy the relaxed complementary slackness conditions (42) and (43), then the pair (x, y) satisfies the primal-dual principle in the discrete case with $\nu = \frac{\nu_{\text{dual}}}{\nu_{\text{primal}}}$. Therefore, x is a ν-approximate solution to* Primal-ILP.

This result simply follows from the inequalities

$$\begin{aligned} c^\top x = \sum_{j=1}^{N} c^{(j)} x^{(j)} &\overset{(42)}{\leq} \sum_{j=1}^{N} \Big(\frac{1}{\nu_{\text{primal}}} \sum_{i=1}^{K} L^{(i,j)} y^{(i)}\Big) x^{(j)} \\ &= \frac{1}{\nu_{\text{primal}}} \sum_{i=1}^{K} \Big(\sum_{j=1}^{N} L^{(i,j)} x^{(j)}\Big) y^{(i)} \\ &\overset{(43)}{\leq} \frac{\nu_{\text{dual}}}{\nu_{\text{primal}}} \sum_{i=1}^{K} b^{(i)} y^{(i)} = \frac{\nu_{\text{dual}}}{\nu_{\text{primal}}} b^\top y. \end{aligned} \tag{44}$$

Based on the above result, iterative schemes can be devised yielding a primal-dual ν-approximation algorithm. For example, we can employ the following algorithm:

Note that, in this scheme, primal solutions are always updated integrally. Also, note that, when applying the primal-dual schema, different implementation strategies are possible. The strategy described in Algorithm 10 is to maintain feasible primal-dual solutions (x_n, y_n) at iteration n, and iteratively improve how tightly the (primal or dual) complementary slackness conditions get satisfied. This is performed through the introduction of slackness variables $(q^{(i)})_{i\in I_{y_n}}$ and $(r^{(j)})_{j\in J_{x_n}}$ the sums of which measure the degrees of violation of each relaxed slackness condition and have thus to be minimized. Alternatively, for example, we can opt to maintain solutions (x_n, y_n) that satisfy the relaxed complementary slackness conditions, but may be infeasible, and iteratively improve the feasibility of the generated solutions. For instance, if we start with a feasible dual solution but with an infeasible primal solution, such a scheme would result into improving the feasibility of the primal solution, as well as the optimality of the dual solution

Algorithm 10 Primal-dual schema

Generate a sequence $(x_n, y_n)_{n\in\mathbb{N}}$ *of elements of* $\mathbb{R}^N \times \mathbb{R}^K$ *as follows:*

Set $\nu_{\text{primal}} \leq 1$ and $\nu_{\text{dual}} \geq 1$

Set $y_0 \in [0, +\infty[^K$ such that $L^\top y_0 \leq c$

For $n = 0, 1, \ldots$

$$\left\lfloor \begin{array}{l} \text{Find } x_n \in \big\{x \in \mathcal{N} \;\big|\; Lx \geq b\big\} \text{ minimizing} \\ \quad \textstyle\sum_{i\in I_{y_n}} q^{(i)} \;\text{ s.t. }\; (\forall i \in I_{y_n}) \quad \sum_{j=1}^N L^{(i,j)} x^{(j)} \leq \nu_{\text{dual}}\, b^{(i)} + q^{(i)},\; q^{(i)} \geq 0 \\ \text{Find } y_{n+1} \in \big\{y \in [0, +\infty[^K \;\big|\; L^\top y \leq c\big\} \text{ minimizing} \\ \quad \textstyle\sum_{j\in J_{x_n}} r^{(j)} \;\text{ s.t. }\; (\forall j \in J_{x_n}) \quad \sum_{i=1}^K L^{(i,j)} y^{(i)} + r^{(j)} \geq \nu_{\text{primal}}\, c^{(j)},\; r^{(j)} \geq 0. \end{array} \right. \tag{45}$$

at each iteration, ensuring that a feasible primal solution is obtained at the end. No matter which one of the above two strategies we choose to follow, the end result will be to gradually bring the primal and dual costs $c^\top x_n$ and $b^\top y_n$ closer and closer together so that asymptotically the primal-dual principle gets satisfied with the desired approximation factor. Essentially, at each iteration, through the coupling by the complementary slackness conditions the current primal solution is used to improve the dual, and vice versa.

Three remarks are worth making at this point: the first one relates to the fact that the two processes, i.e. the primal and the dual, make only local improvements to each other. Yet, in the end they manage to yield a result that is almost globally optimal. The second point to emphasize is that, for computing this approximately optimal result, the algorithm requires no solution to the Primal-LP or Dual-LP to be computed, which are replaced by simpler optimization problems. This is an important advantage from a computational standpoint since, for large scale problems, solving these relaxations can often be quite costly. In fact, in most cases where we apply the primal-dual schema, purely combinatorial algorithms can be obtained that contain no sign of linear programming in the end. A last point to be noticed is that these algorithms require appropriate choices of the relaxation factors ν_{primal} and ν_{dual}, which are often application-guided.

4.2.1 Application to the set cover problem

For a simple illustration of the primal-dual schema, let us consider the problem of set-cover, which is known to be NP-hard. In this problem, we are given as input a finite set $\mathcal{V}$ of K elements $(v^{(i)})_{1\leq i\leq K}$, a collection of (non disjoint) subsets $\mathcal{S} = \{S_j\}_{1\leq j\leq N}$ where, for every $j \in \{1, \ldots, N\}$, $S_j \subset \mathcal{V}$, and $\bigcup_{j=1}^N S_j = \mathcal{V}$. Let $\varphi\colon \mathcal{S} \to \mathbb{R}$ be a function that assigns a cost $c_j = \varphi(S_j)$ for each subset S_j. The goal is to find a set cover (i.e. a subcollection of $\mathcal{S}$ that covers all elements of $\mathcal{V}$) that has minimum cost (see Fig. 16).

The above problem can be expressed as the following ILP:

$$\underset{x=(x^{(j)})_{1\leq j\leq N}}{\text{minimize}} \quad \sum_{j=1}^N \varphi(S_j)\, x^{(j)} \tag{46}$$

$$\text{s.t. } (\forall i \in \{1, \ldots, K\}) \quad \sum_{\substack{j\in\{1,\ldots,N\} \\ v^{(i)}\in S_j}} x^{(j)} \geq 1, \quad x \in \{0,1\}^N, \tag{47}$$

where indicator variables $(x^{(j)})_{1\leq j\leq N}$ are used for determining if a set in $\mathcal{S}$ has been included in

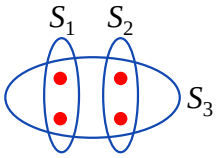


Figure 16: A toy set-cover instance with $K = 4$ and $N = 3$, where $\varphi(S_1) = \frac{1}{2}$, $\varphi(S_2) = 1$, $\varphi(S_3) = 2$. In this case, the optimal set-cover is $\{S_1, S_2\}$ and has a cost of $\frac{3}{2}$.

the set cover or not, and (47) ensures that each one of the elements of $\mathcal{V}$ is contained in at least one of the sets that were chosen for participating to the set cover.

An LP-relaxation for this problem is obtained by simply replacing the Boolean constraint with the constraint $x \in [0, +\infty[^N$. The dual of this LP relaxation is given by the following linear program:

$$\underset{y=(y^{(i)})_{1\leq i\leq K}\in[0,+\infty[^K}{\text{maximize}} \quad \sum_{i=1}^{K} y^{(i)} \tag{48}$$

$$\text{s.t.} \quad (\forall j \in \{1,\ldots,N\}) \quad \sum_{\substack{i\in\{1,\ldots,K\}\\ v^{(i)}\in S_j}} y^{(i)} \leq \varphi(S_j). \tag{49}$$

Let us denote by $F_{\max}$ the maximum frequency of an element in $\mathcal{V}$, where by the term *frequency* we mean the number of sets this element belongs to. In this case, we will use the primal-dual schema to derive an $F_{\max}$-approximation algorithm by choosing $\nu_{\text{primal}} = 1$, $\nu_{\text{dual}} = F_{\max}$. This results into the following complementary slackness conditions, which we will need to satisfy:

Primal Complementary Slackness Conditions

$$(\forall j \in \{1,\ldots,N\}) \text{ if } x^{(j)} > 0 \text{ then } \sum_{\substack{i\in\{1,\ldots,K\}\\ v^{(i)}\in S_j}} y^{(i)} = \varphi(S_j) \tag{50}$$

Relaxed Dual Complementary Slackness Conditions (with relaxation factor $F_{\max}$)

$$(\forall i \in \{1,\ldots,K\}) \text{ if } y^{(i)} > 0 \text{ then } \sum_{\substack{j\in\{1,\ldots,N\}\\ v^{(i)}\in S_j}} x^{(j)} \leq F_{\max}. \tag{51}$$

A set S_j with $j \in \{1,\ldots,N\}$ for which $\sum_{\substack{i\in\{1,\ldots,K\}\\ v^{(i)}\in S_j}} y^{(i)} = \varphi(S_j)$ will be called *packed*. Based on this definition, and given that the primal variables $(x^{(j)})_{1\leq j\leq N}$ are always kept integral (i.e., either 0 or 1) during the primal-dual schema, Conditions (50) basically say that only packed sets can be included in the set cover (note that overpacked sets are already forbidden by feasibility constraints (49)). Similarly, Conditions (51) require that an element $v^{(i)}$ with $i \in \{1,\ldots,K\}$ associated with a nonzero dual variable $y^{(i)}$ should not be covered more than $F_{\max}$ times, which is, of course, trivially satisfied given that $F_{\max}$ represents the maximum frequency of any element in $\mathcal{V}$.

Based on the above observations, the iterative method whose pseudocode is shown in Algorithm 11 emerges naturally as a simple variant of Algorithm 10. Upon its termination, both x and y will be feasible given that there will be no uncovered element and no set that violates (49).

Algorithm 11 Primal-dual schema for set-cover.

Set $x_0 \leftarrow 0, y_0 \leftarrow 0$

Declare all elements in $\mathcal{V}$ as uncovered

While $\mathcal{V}$ contains uncovered elements

1. Select an uncovered element $v^{(i)}$ with $i \in \{1, \ldots, K\}$ and increase $y^{(i)}$ until some set becomes packed
2. For every packed set S_j with $j \in \{1, \ldots, N\}$, set $x^{(j)} \leftarrow 1$ (include all the sets that are packed in the cover)
3. Declare all the elements belonging to at least one set S_j with $x^{(j)} = 1$ as covered.

Furthermore, given that the final pair (x, y) satisfies the relaxed complementary slackness conditions with $\nu_{\text{primal}} = 1$, $\nu_{\text{dual}} = F_{\max}$, the set cover defined by x will provide an $F_{\max}$-approximate solution.

4.3 Primal-dual schema for discrete MRF optimization

Based on the above discussion, to apply the primal-dual schema to MRF optimization, we must first complete the following 3 tasks: express MRF optimization as a linear integer program, form the dual LP, and finally choose the relaxed complementary slackness conditions. Concerning the first of these tasks, we will use the following integer LP formulation of MRF optimization, which was already introduced in section 2.6:

$$\min_{\bar{\mathbf{x}}} \quad \sum_{p \in \mathbf{V}} \sum_{l_i \in \mathbf{L}} \theta_p(l_i) \bar{x}_p(l_i) + \sum_{(p,q) \in \mathbf{E}} \sum_{l_i, l_j \in \mathbf{L}} \theta_{pq}(l_i, l_j) \bar{x}_{pq}(l_i, l_j), \tag{52}$$

$$\text{s.t.} \quad \sum_{l_i \in \mathbf{L}} \bar{x}_p(l_i) = 1, \forall p \in \mathbf{V}, \tag{53}$$

$$\sum_{l_j \in \mathbf{L}} \bar{x}_{pq}(l_i, l_j) = \bar{x}_p(l_i), \forall (p,q) \in \mathbf{E}, l_i \in \mathbf{L}, \tag{54}$$

$$\sum_{l_i \in \mathbf{L}} \bar{x}_{pq}(l_i, l_j) = \bar{x}_q(l_j), \forall (p,q) \in \mathbf{E}, l_j \in \mathbf{L}, \tag{55}$$

$$\bar{x}_p(l_i) \in \{0,1\}, \bar{x}_{pq}(l_i, l_j) \in \{0,1\}, \forall p \in \mathbf{V}, (p,q) \in \mathbf{E}, l_i, l_j \in \mathbf{L}.$$

We recall that the above binary variables $\{\bar{x}_p(\cdot)\}$ (resp. $\{\bar{x}_{pq}(\cdot,\cdot)\}$) are indicators for the labels assigned to each node p (resp. pair of nodes p, q) (*i.e.*, it holds $\bar{x}_p(a) = 1 \Leftrightarrow p$ takes label a, and $\bar{x}_{pq}(a,b) = 1 \Leftrightarrow p, q$ take labels a, b). Constraints (53) simply encode the fact that each node can be assigned only one label, while constraints (54), (55) enforce consistency between unary variables $\{\bar{x}_p(\cdot)\}$, $\{\bar{x}_q(\cdot)\}$ and pairwise variables $\{\bar{x}_{pq}(\cdot,\cdot)\}$ by ensuring that $\bar{x}_{pq}(a,b) = 1$ holds whenever $\bar{x}_p(a) = \bar{x}_q(b) = 1$ holds.

If one relaxes the integrality constraints in (52) to $\bar{x}_p(\cdot), \bar{x}_p(\cdot,\cdot) \geq 0$, the dual linear program program[2] to the resulting LP takes the following form, which, as can be seen, contains one dual

[2]To recognize that the dual (56) is an LP, one needs to introduce auxiliary variables $\{z_p\}_{p \in \mathcal{V}}$ that satisfy the

variable $y_{e,p}(x_p)$ for each edge $e \in \mathbf{E}$, $p \in e$ and label $x_p \in \mathbf{L}$:

$$\max_{\mathbf{y}} \sum_p \min_{x_p \in \mathbf{L}} h_p(x_p) \tag{56}$$

$$\text{s.t.} \sum_{p \in e} y_{e,p}(x_p) \leq \theta_e(x_e) \ , \quad \forall e \in \mathbf{E}, \ x_e = (x_p)_{p \in e} \in \mathbf{L} \times \mathbf{L} \tag{57}$$

$$h_p(x_p) = \theta_p(x_p) + \sum\nolimits_{e:p \in e} y_{e,p}(x_p) \ , \ \forall p \in \mathbf{V}, x_p \in \mathbf{L} \ . \tag{58}$$

Intuitively, one can think of the dual variables as redistributing the pairwise cost $\theta_e(x_e)$ to the unary terms, thus forming a MRF that has no pairwise costs but only the unary terms $h_p(x_p)$. The minimum energy of this MRF is given by $\sum_p \min_{x_p} h_p(x_p)$, and the goal of the above dual LP is to maximize the value of this energy by adjusting the dual variables.

In this case, the primal complementary slackness conditions consist of the following 2 type of constraints, corresponding to unary and pairwise primal variables, where $\mathbf{x}'$ denotes the labeling defined by an integral primal solution to (52) and $\mathbf{y}$ denotes a dual solution:

$$\textbf{unary complementary slackness:} \quad h_p(x'_p) = \min_{x_p \in \mathbf{L}} h_p(x_p) \tag{59}$$

$$\textbf{pairwise complementary slackness:} \quad \sum_{p \in e} y_{e,p}(x'_p) = \theta_e(x'_e), \tag{60}$$

where we denote $x'_e = (x'_p)_{p \in e}$ (*e.g.*, if $e = (p, q)$ then $x'_e = (x'_p, x'_q)$).

In general, given that MRF optimization is an NP-hard problem, the above complementary slackness conditions cannot be satisfied exactly. To derive a primal-dual based approximation scheme, we choose in this case a relaxed version of these conditions using $f_{\text{primal}} = 2\frac{\min f_e(x_e)}{\max f_e(x_e)}$ and $f_{\text{dual}} = 1$. This will lead (by theorem .1) to an algorithm with a worst case approximation factor of $2\frac{\max f_e(x_e)}{\min f_e(x_e)}$ [36, 37]. For instance, this gives an approximation factor of 2 for the commonly used Potts model, where pairwise costs $\theta_e(x_e)$ are assumed to take a positive value w_e if $x_e = (x_p, x_q)$ satisfies $x_p \neq x_q$ and are 0 otherwise.

By looking at conditions (59), (60), it becomes apparent that the adjustment of the primal and dual variables with the goal of satisfying a relaxed version of these conditions during the primal-dual schema is far from trivial. For instance, the unary slackness conditions (59) require that the label x'_p assigned to a vertex p should have minimum height[3], but if we simply decrease one of the dual variables $y_{e,p}(x'_p)$ in order to decrease the height $h_p(x'_p)$, we might then violate the pairwise slackness conditions (60) (or even a relaxed version of them). Instead, during each iteration of the primal-dual schema, it is necessary to change the assigned labels for multiple vertices and at the same time to also adjust multiple dual variables. It turns out that a systematic way for doing this adjustment is through solving a series of max-flow/min-cut problems (one problem per iteration of the primal-dual schema), where the flows determine how to update the dual variables $\mathbf{y}$ and the corresponding min-cut solution specifies how to change the primal-integral variables, i.e., essentially the labels assigned to vertices (the relationship between graph-cuts and MRF optimization observed here should not come as a surprise given that these 2 problems are actually equivalent in certain cases, as already explained in section 2). After solving each max-flow/min-cut problem, both primal optimality and dual feasibility are improved. In the end, the dual solution is guaranteed to be feasible, which together with the fulfillment of the relaxed slackness conditions guarantee that an approximately optimal labeling is obtained. The resulting primal-dual algorithm is known as FastPD, and is described in detail in [37, 36].

linear constraints $z_p \leq h_p(\cdot)$, *i.e.*, $z_p = \min_a h_p(a)$.

[3]The term $h_p(x_p)$ is called the *height* of label x_p at node p.

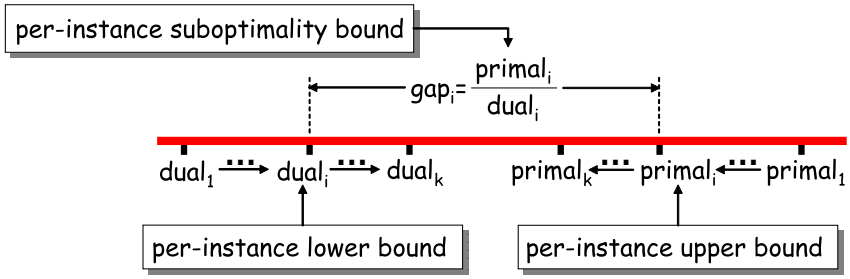
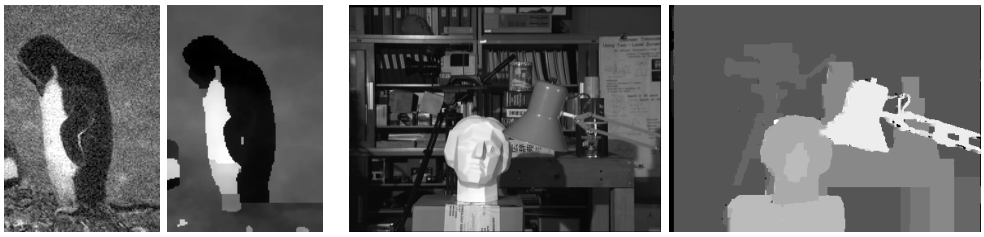
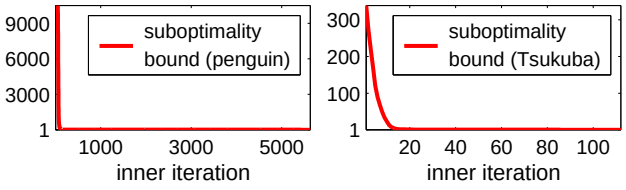


Fig. 17: During FastPD the ratios of the computed primal and dual costs provide per-instance approximation factors, which become tighter over time.



(a) Denoising of 'penguin' image

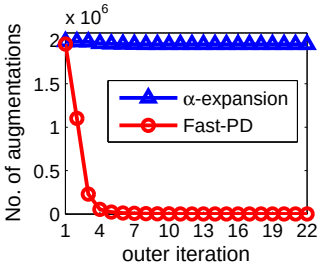
(b) 'Tsukuba' left image and estimated disparity



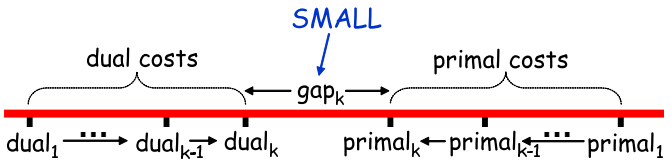
(c) Approximation factors per iteration

Fig. 18: FastPD results on image denoising and stereo matching along with corresponding approximation factors per iteration. As can be seen, these factors drop very close to 1, meaning that the generated solutions are almost optimal.

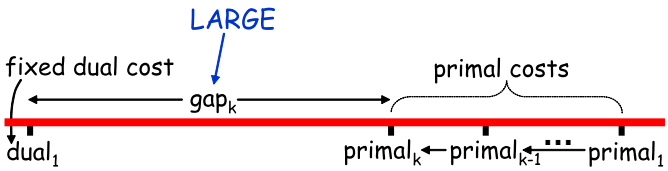
It is important to note that FastPD, due to being a primal-dual method, can provide not only worst-case (*i.e.*, theoretical) approximation factors (like those that have been mentioned earlier), but it can also provide *per-instance* approximation factors, *i.e.*, approximation guarantees that are specific to the current discrete MRF that is being optimized. And it can do that at no extra cost as these per-instance approximation factors can be computed directly based on the ratio of the primal and dual costs computed during the course of the algorithm (see Fig. 17). Obviously, these factors are continuously refined as the algorithm runs, and actually prove to be much tighter in practice than their theoretical counterparts. *E.g.*, Fig. 18 shows how these ratios vary per iteration for two standard benchmark problems from the middlebury dataset, a stereo matching example ('tsukuba') and a denoising example ('penguin'). As one can notice, they finally drop very close to 1, meaning that an almost optimal solution is estimated at the end (despite the problems being NP-hard).



(a)



(b) High-level view of FastPD



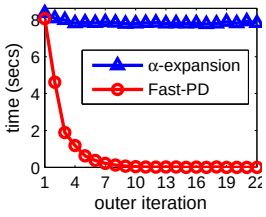
(c) High-level view of α-expansion

Fig. 19: **(a)** Number of augmenting paths per outer iteration for the 'penguin' example. Notice the dramatic decrease over time in the case of FastPD. **(b)** FastPD generates pairs of primal-dual solutions iteratively, with the goal of reducing the gap between the primal and dual costs. Since this gap provides a rough approximation to the number of augmenting paths (see Thm. .2), the latter is forced to reduce over time. **(c)** On the contrary, α-expansion works only in the primal domain (*i.e.*, it is as if a fixed dual cost is used at the start of each iteration), and thus the primal-dual gap never becomes small enough. Therefore, no significant reduction in the number of augmenting paths takes place over time.

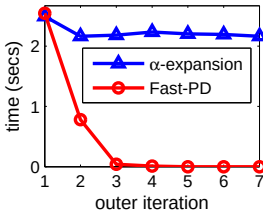
4.4 Computational efficiency of primal-dual MRF optimization

Besides maintaining strong optimality properties, another very important advantage of primal-dual algorithms over other graph-cut based methods such as α-expansion, is that they prove to be more efficient in practice. In fact, the computational efficiency of all such methods is largely determined from the time taken by each max-flow problem, which, in turn, depends on the number of augmenting paths that need to be computed per max-flow. Fortunately, for the case of FastPD, this number decreases dramatically over time. This is illustrated in Fig. 19(a) for the case of the 'penguin' image-denoising problem, where a plot of the corresponding number of augmenting paths per outer-iteration (*i.e.*, per group of $|\mathbf{L}|$ iterations) is shown. Notice that whereas this number remains very high (*i.e.*, almost $2 \cdot 10^6$) throughout α-expansion, it drops towards zero very quickly in the case of FastPD, *e.g.*, only 4905 and 7 augmenting paths had to be found during the 8^{th} and last outer-iteration, respectively. In fact, this behavior is typical of FastPD, where, after only a few iterations, a very small number of augmenting paths need to be computed per max-flow, which, of course, boosts the algorithm's performance.

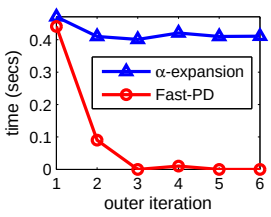
This property can be explained by the fact that FastPD makes full use of both primal and dual information throughout its execution. This is more clearly illustrated in Fig. 19(b). Intuitively, what happens is that FastPD ultimately aims to close the gap between the primal and the dual cost (recall the "Primal-dual principle in the discrete case"), and, for this, it iteratively generates primal-dual pairs, with the goal of continuously decreasing the size of this gap. However, the gap's size can be thought of as a rough approximation to the number of augmenting paths per iteration (see Thm. .2 below). Therefore, as FastPD keeps reducing this gap throughout its execution, the number of augmenting paths is forced to decrease over time as well, which, in turn, results in a significant improvement in the efficiency of the max-flow algorithm (recall that a path augmenting algorithm for max-flow essentially proceeds by keep finding augmenting paths).



(a) 'penguin'



(b) 'Tsukuba'



(c) 'SRI tree'

total time (secs)	Fast-PD	α-expansion
penguin	17.44	173.1
tsukuba	3.37	15.63
SRI tree	0.54	2.56

(d) Total running times

Fig. 20: **(a)**, **(b)** and **(c)** show the running times per outer iteration for 3 standard benchmark vision problems, while **(d)** lists the corresponding total running times (all experiments were measured on a 1.6GHz CPU).

On the contrary, a method like α-expansion, that works only in the primal domain, ignores dual solutions completely. It is, roughly speaking, as if α-expansion is resetting the dual solution to zero at the start of each iteration, thus effectively forgetting that solution thereafter (see Fig. 19(c)). For this reason, it fails to substantially reduce the primal-dual gap and thus also fails to achieve a reduction in path augmentations over time, *i.e.*, across iterations. This, of course, results in more time to be needed per iteration.

The above mentioned relationship between primal-dual gap and number of augmenting paths is formally described in the next theorem:

Theorem .2 ([37]) *During FastPD, the primal-dual gap at the current iteration forms an upper bound for the number of augmenting paths at each iteration thereafter.*

As a result, the time per iteration of FastPD decreases dramatically over time. This has been verified experimentally with virtually all problems FastPD was applied to. For instance, Fig. 20 shows total running times, as well as running times per outer-iteration, for some standard benchmark vision problems. Notice how much faster an outer-iteration of FastPD becomes over time. *E.g.*, for the 'SRI-tree' stereo matching example in Fig. 20, the last outer-iteration of FastPD took less than 1 msec (as only 4 augmenting paths had to be computed), and thus it was more than 400 times faster than the corresponding iteration of α-expansion. Similarly, for the 'tsukuba' example in Fig. 20, the last iteration of FastPD was more than 2000 times faster than the last iteration of α-expansion.

5 MRF inference based on dual-decomposition

In the previous chapter we saw how to make use of a convex relaxation (in particular, a LP relaxation) to optimize a MRF based on the primal-dual schema. In that case, the LP relaxation was used only in the design of the algorithm but a solution to that relaxation was never actually computed. In this chapter we will follow a totally different strategy: we will optimize a MRF by first actually trying to solve the same convex relaxation and then simply converting the resulting fractional solution into a integral MRF solution. In this case, the main computational challenge/bottleneck comes from the first step, *i.e.*, from computing an optimal solution to the convex relaxation. To efficiently accomplish this task, we will resort to the use of a powerful technique called *dual-decomposition*, which is widely used in optimization. As we shall see, such a technique will allow us to derive efficient and flexible MAP estimation algorithms for MRF optimization problems.

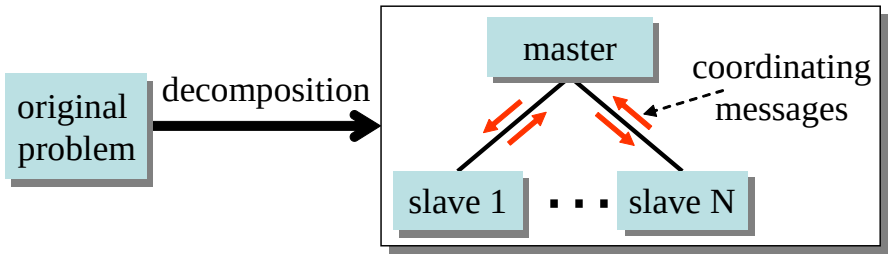


Fig. 21: The dual decomposition principle: the original (possibly difficult) optimization problem is decomposed into easier subproblems (called the *slaves*) that are coordinated by a *master* process by exchanging messages with each other.

5.1 The principle of dual decomposition

The core idea behind dual-decomposition essentially follows a divide and conquer strategy: that is, given a difficult or high-dimensional optimization problem, we decompose it into smaller easy-to-handle subproblems and then extract an overall solution by cleverly combining the solutions from these subproblems.

Although simple as a concept, decomposition is an extremely general and powerful technique that has been succesfully used many times for deriving efficient algorithms to difficult and large scale optimization problems. To apply such a technique, one first has to carefully choose the set of subproblems that will be used (which will be referred to as *slaves* hereafter). Subsequently, all these slave problems are iteratively adjusted by a so-called *master* process that is used for combining the local solutions of the slaves into a global solution of the original problem, and which acts as the coordinator of all slaves (see Fig. 21).

To illustrate how this technique works, we will provide a simple example. To that end, let us consider the following problem (where $\boldsymbol{x}$ denotes a vector of variables and $\mathcal{C}$ is a closed[4] set):

$$\begin{array}{ll} \min\limits_{\boldsymbol{x}} & \sum\limits_{m=1}^{M} f^m(\boldsymbol{x}) \\ \text{s.t.} & \boldsymbol{x} \in \mathcal{C} \end{array} \tag{61}$$

We assume that separately minimizing each $f^m(\cdot)$ over vector $\boldsymbol{x} \in \mathcal{C}$ is easy, but minimizing the sum $\sum_m f^m(\cdot)$ is hard. Using auxiliary variables $\{\boldsymbol{x}^m\}_{m=1}^M$, we thus transform our problem into:

$$\begin{array}{ll} \min_{\{\boldsymbol{x}^m\},\boldsymbol{x}} & \sum\limits_{m=1}^{M} f^m(\boldsymbol{x}^m) \\ \text{s.t.} & \boldsymbol{x}^m \in \mathcal{C},\ \boldsymbol{x}^m = \boldsymbol{x} \end{array}$$

Obviously this is equivalent to our original problem (61). Furthermore, if the coupling constraints $\boldsymbol{x}^m = \boldsymbol{x}$ were absent, the problem would decouple. We therefore relax them via introducing multipliers $\{\boldsymbol{\lambda}^m\}_{m=1}^M$ and form the following Lagrangian dual function:

$$\begin{aligned} g(\{\boldsymbol{\lambda}^m\}) &= \min_{\{\boldsymbol{x}^m \in \mathcal{C}\},\boldsymbol{x}} \textstyle\sum_m f^m(\boldsymbol{x}^m) + \sum_m \boldsymbol{\lambda}^m \cdot (\boldsymbol{x}^m - \boldsymbol{x}) \\ &= \min_{\{\boldsymbol{x}^m \in \mathcal{C}\},\boldsymbol{x}} \textstyle\sum_m [f^m(\boldsymbol{x}^m) + \boldsymbol{\lambda}^m \cdot \boldsymbol{x}^m] - (\sum_m \boldsymbol{\lambda}^m)\boldsymbol{x} \end{aligned}$$

We next eliminate $\boldsymbol{x}$ from $g(\{\boldsymbol{\lambda}^m\})$ by minimizing over that variable. This simply results in having $\{\boldsymbol{\lambda}^m\} \in \Lambda = \{\{\boldsymbol{\lambda}^m\} | \sum_m \boldsymbol{\lambda}^m = 0\}$ (since if $\{\boldsymbol{\lambda}^m\} \notin \Lambda$ then it is easy to check that $g(\{\boldsymbol{\lambda}^m\}) =$

[4] In case the set $\mathcal{C}$ is not closed, min has to be replaced with inf in the derivations that follow.

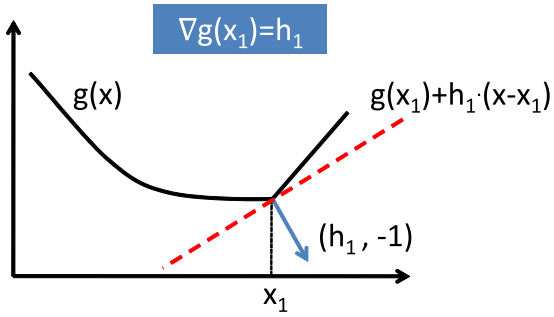


Fig. 22: The vector h_1 is a subgradient of function $g(\cdot)$ at x_1 if and only if $(h_1, -1)$ specifies a supporting hyperplane to the epigraph of $g(\cdot)$ at $(x_1, g(x_1))$.

$-\infty$). Therefore, the resulting dual function becomes equal to:

$$g(\{\boldsymbol{\lambda}^m\}) = \min_{\{\boldsymbol{x}^m \in \mathcal{C}\}} \sum\nolimits_m [f^m(\boldsymbol{x}^m) + \boldsymbol{\lambda}^m \cdot \boldsymbol{x}^m]$$

We can now setup a Lagrangian dual problem, *i.e.*maximize $g(\{\boldsymbol{\lambda}^m\})$ over the feasible set Λ, or

$$\max\nolimits_{\{\boldsymbol{\lambda}^m\} \in \Lambda} \; g(\{\boldsymbol{\lambda}^m\}) = \sum\nolimits_m g^m(\boldsymbol{\lambda}^m), \tag{62}$$

where this dual problem (also called the master) has now decoupled into the following slave problems (one per m):

$$\begin{aligned} g^m(\boldsymbol{\lambda}^m) = \; & \min_{\boldsymbol{x}^m} f^m(\boldsymbol{x}^m) + \boldsymbol{\lambda}^m \cdot \boldsymbol{x}^m \\ & \text{s.t. } \; \boldsymbol{x}^m \in \mathcal{C} \end{aligned} \tag{63}$$

Problem (62) is always convex[5] and so it can be solved using, *e.g.*, a projected subgradient method (due to that $g(\cdot)$ is typically not differentiable). According to that method, at each iteration the dual variables $\{\boldsymbol{\lambda}^m\}$ are updated as follows:

$$\boldsymbol{\lambda}^m \leftarrow [\boldsymbol{\lambda}^m + \alpha_t \nabla g^m(\boldsymbol{\lambda}^m)]_\Lambda \, .$$

In the above update, $\{\alpha_t\}$ denotes a sequence of positive step-sizes (where a_t is the step-size used during the t-th iteration), $[\,\cdot\,]_\Lambda$ denotes projection onto the set Λ, while $\nabla g^m(\boldsymbol{\lambda}^m)$ denotes a subgradient[6] of function $g^m(\cdot)$ at $\boldsymbol{\lambda}^m$. We recall that the subgradient of a convex function is a generalization of the notion of gradient for non-differentiable functions, and its estimation essentially corresponds to specifying a supporting hyperplane to a function's epigraph (see Fig. 22). It thus only remains to see how subgradient $\nabla g^m(\boldsymbol{\lambda}^m)$ can be computed. To do that, we can rely on the following well-known lemma:

Lemma 1 *Let function $g(\boldsymbol{\lambda})$ be defined as $g(\boldsymbol{\lambda}) = \min_{x \in \mathcal{C}}\{a(\boldsymbol{x}) + \boldsymbol{\lambda} \cdot b(\boldsymbol{x})\}$, where $a(\cdot), b(\cdot)$ represent functions over a compact set $\mathcal{C}$. Let also vector $\hat{\boldsymbol{x}}$ be an optimal solution to problem $\min_{x \in \mathcal{C}}\{a(\boldsymbol{x}) + \boldsymbol{\lambda} \cdot b(\boldsymbol{x})\}$,* i.e., *$g(\boldsymbol{\lambda}) = a(\hat{\boldsymbol{x}}) + \boldsymbol{\lambda} \cdot b(\hat{\boldsymbol{x}})$. Then $b(\hat{\boldsymbol{x}})$ is a subgradient of $g(\cdot)$ at λ.*

[5] Note that the convexity of problem (62) holds regardless of whether or not the objective function of problem (61) is convex.

[6] Throughout the paper, by abuse of notation, we use $\nabla g(x)$ to denote a subgradient of function $g(\cdot)$ at point x, *i.e.*, a vector that belongs in the subdifferential $\partial g(x)$. Note that this notation is non-conventional, since, in the literature $\nabla g(x)$ is used only to refer to the gradient of a differentiable function.

The theorem follows from

$$g(\lambda') \leq a(\hat{\boldsymbol{x}}) + \boldsymbol{\lambda}' \cdot b(\hat{\boldsymbol{x}}) = (a(\hat{\boldsymbol{x}}) + \boldsymbol{\lambda} \cdot b(\hat{\boldsymbol{x}})) + (\boldsymbol{\lambda}' - \boldsymbol{\lambda}) \cdot b(\hat{\boldsymbol{x}}) = g(\boldsymbol{\lambda}) + (\boldsymbol{\lambda}' - \boldsymbol{\lambda}) \cdot b(\hat{\boldsymbol{x}})$$

From the above lemma it follows directly that it holds:

$$\nabla g^m(\boldsymbol{\lambda}^m) = \hat{\boldsymbol{x}}^m(\boldsymbol{\lambda}^m) \ ,$$

where $\hat{\boldsymbol{x}}^m(\boldsymbol{\lambda}^m)$ denotes any optimal solution to the m-th slave problem (63). By putting all of the above elements together, we get an iterative algorithm that relies on a communication between the master and the slaves that proceeds as follows:

1. The master sends the current $\{\boldsymbol{\lambda}^m\}$ to the slaves and asks them to optimize their problems.
2. The slaves respond to the master by solving their (easy) problems and sending back to him the resulting minimizers $\{\hat{\boldsymbol{x}}^m(\boldsymbol{\lambda}^m)\}$.
3. The master then collects the minimizers and updates each $\boldsymbol{\lambda}^m$ by setting $\boldsymbol{\lambda}^m \leftarrow [\boldsymbol{\lambda}^m + \alpha_t \hat{\boldsymbol{x}}^m(\boldsymbol{\lambda}^m)]_\Lambda$
4. The above three steps are repeated until convergence.

In essence, a solution to the dual is obtained by operating at two levels: at the global level, the master process coordinates the slaves simply by updating $\{\boldsymbol{\lambda}^m\}$ based on the currently extracted optimal solutions $\{\hat{\boldsymbol{x}}^m(\boldsymbol{\lambda}^m)\}$. Subsequently, at the local level, based on the updated $\{\boldsymbol{\lambda}^m\}$ each of the decoupled slave problems (63) is again solved independently to generate a new set of minimizers $\{\hat{\boldsymbol{x}}^m(\boldsymbol{\lambda}^m)\}$ for the next iteration.

5.2 Dual decomposition for discrete MRF optimization

In this section, we show how to apply the dual decomposition method to the MRF optimization problem by following essentially a reasoning similar to that in the previous example.

Let us thus consider the integer programming formulation of the energy minimization problem for a MRF defined on a general graph $G = (\mathbf{V}, \mathbf{E})$

$$\underset{x \in \mathcal{X}_G}{\text{minimize}} \ f(x;\theta) = \sum_{p \in \mathbf{V},\, l_p \in \mathbf{L}} \theta_p(l_p)\, x_p(l_p) + \sum_{e \in \mathbf{E},\, l_e \in \mathbf{L}^2} \theta_e(l_e)\, x_e(l_e), \tag{64}$$

where set $\mathcal{X}_G$ is defined as

$$\mathcal{X}^{\mathcal{G}} = \left\{ x \;\middle|\; \begin{array}{ll} \sum_{l \in \mathbf{L}} x_p(l) = 1, & \forall\ p \in \mathbf{V} \\ \sum_{l' \in \mathbf{L}} x_e(l,l') = x_p(l), & \forall\ e = (p,q) \in \mathbf{E},\ \forall l \in \mathbf{L} \\ \sum_{l \in \mathbf{L}} x_e(l,l') = x_q(l'), & \forall\ e = (p,q) \in \mathbf{E},\ \forall l' \in \mathbf{L} \\ x_p(\cdot) \in \{0,1\}, & \forall\ p \in \mathbf{V} \\ x_e(\cdot,\cdot) \in \{0,1\}, & \forall\ e \in \mathbf{E} \end{array} \right\} \ .$$

Our goal will be to decompose this problem (64) into easier slave subproblems, which, in this case, involve optimizing MRFs defined on subgraphs of G. More specifically, let $\{G_m = (\mathbf{V}_m, \mathbf{E}_m)\}_{1 \leq m \leq M}$ be a set of subgraphs that form a decomposition of $G = (\mathbf{V}, \mathbf{E})$, i.e.,

$$\cup_{m=1}^{M} \mathbf{V}_m = \mathbf{V}, \quad \cup_{m=1}^{M} \mathbf{E}_m = \mathbf{E} \ .$$

On each of these subgraphs, we define a local MRF with corresponding (unary and pairwise) potentials $\theta^m = \left\{ \{\theta_p^m\}_{p \in \mathbf{V}_m}, \{\theta_e^m\}_{e \in \mathbf{E}_m} \right\}$, whose cost function $f^m(x;\theta^m)$ is thus given by

$$f^m(x;\theta^m) = \sum_{p \in \mathbf{V}_m,\, l_p \in \mathbf{L}} \theta_p^m(l_p)\, x_p(l_p) + \sum_{e \in \mathbf{E}_m,\, l_e \in \mathbf{L}^2} \theta_e^m(l_e)\, x_e(l_e). \tag{65}$$

Moreover, the potential functions θ^m are initialized in such a manner such that their sum (over m) reproduces the potentials θ of the original MRF on G, i.e.,[7]

$$(\forall p \in \mathbf{V})(\forall e \in \mathbf{E}) \quad \sum_{m:\, p \in \mathbf{V}_m} \theta_p^m = \theta_p, \qquad \sum_{m:\, e \in \mathbf{E}_m} \theta_e^m = \theta_e. \tag{66}$$

This guarantees that $f = \sum_{m=1}^{M} f^m$, thus allowing us to re-express problem (64) as follows

$$\underset{x \in \mathcal{X}_G}{\text{minimize}} \quad \sum_{m=1}^{M} f^m(x; \theta^m). \tag{67}$$

To take advantage of the above decomposition of the MRF energy, we introduce, for every $m \in \{1, \ldots, M\}$, an *auxiliary copy* $x^m \in \mathcal{X}_{G_m}$ for the variables of the local MRF defined on G_m, which are thus constrained to coincide with the corresponding variables in vector x, i.e., it holds

$$x^m = x_{|G_m} \ ,$$

where $x_{|G_m}$ is used to denote the subvector of x containing only those variables associated with vertices and edges of subgraph G_m. In this way, Problem (67) can be transformed into

$$\underset{x \in \mathcal{X}_G, \{x^m \in \mathcal{X}_{G_m}\}_{1 \leq m \leq M}}{\text{minimize}} \quad \sum_{m=1}^{M} f^m(x^m; \theta^m) \tag{68}$$

$$\text{s.t.} \quad \big(\forall m \in \{1, \ldots, M\}\big) \qquad x^m = x_{|G_m}. \tag{69}$$

It is clear that without the constraints $x^m = x_{|G_m}$, this problem would decouple into a series of local MRF subproblems (one per subgraph G_m). Therefore, it is natural to relax these coupling constraints (by introducing Lagrange multipliers $\lambda^m = \big\{\{\lambda_p^m\}_{p \in \mathbf{V}_m}, \{\lambda_e^m\}_{e \in \mathbf{E}_m}\big\}$) and form the Lagrangian dual function as:

$$\begin{aligned} g(\{\lambda^m\}) &= \min_{\{x^m \in \mathcal{X}_{G_m}\}, x} \sum_{m=1}^{M} f^m(x^m; \theta^m) + \sum_{m=1}^{M} \lambda^m \cdot (x^m - x_{|G_m}) \\ &= \min_{\{x^m \in \mathcal{X}_{G_m}\}, x} \sum_{m=1}^{M} f^m(x^m; \theta^m + \lambda^m) - \sum_{m=1}^{M} \lambda^m \cdot x_{|G_m} \end{aligned}$$

Vector x can be eliminated from $g(\{\lambda^m\})$ by directly minimizing over this variable, which simply imposes the constraint $\{\lambda^m\} \in \Lambda$,[8] where the feasible set Λ is now defined as:

$$\Lambda = \left\{ \{\lambda^m\}_{m=1}^{M} \ \Big| \ \sum_{m: p \in \mathbf{V}_m} \lambda_p^m = 0, \ \sum_{m: e \in \mathbf{E}_m} \lambda_e^m = 0, \ \ \forall p \in \mathbf{V}, e \in \mathbf{E} \right\},$$

while the resulting Lagrangian dual function simplifies to:

$$g(\{\lambda^m\}) = \min_{\{x^m \in \mathcal{X}_{G_m}\}} \sum_{m=1}^{M} f^m(x^m; \theta^m + \lambda^m) \ .$$

[7] For instance, to ensure (66) we can simply set: $(\forall m \in \{1, \ldots, M\})$ $\theta_p^m = \frac{\theta_p}{|\{|m'| p \in \mathbf{V}_{m'}\}|}$ and $\theta_e^m = \frac{\theta_e}{|\{m'| e \in \mathbf{E}_{m'}\}|}$.

[8] It is easy to see that if $\{\lambda^m\} \notin \Lambda$, then $g(\{\lambda^m\}) = -\infty$.

We can now setup a dual problem, which is the maximization of the above dual function $g(\{\lambda^m\})$ over its feasible set Λ, or

$$\max_{\{\lambda^m\}\in\Lambda} g(\{\lambda^m\}) = \sum_{m=1}^{M} g^m(\lambda^m), \tag{70}$$

where each function $g^m(\cdot)$ is defined as:

$$\begin{array}{cc} g^m(\lambda^m) = \min\limits_{x^m} & f^m(x^m;\theta^m+\lambda^m) \\ \text{s.t.} & x^m \in \mathcal{X}_{G_m} \ . \end{array} \tag{71}$$

5.3 Applying projected subgradient to the MRF dual

The above dual problem provides a relaxation to the original MRF problem (64). Furthermore, note that this relaxation leads to a convex optimization problem, although the original one is not. As such, it can always be solved in an optimal manner. A possible way of doing this consists of using a projected subgradient method. As explained in section 5.1, at each iteration of this method the dual variables λ^m must first be updated as follows

$$\lambda^m \leftarrow \lambda^m + \alpha_t \nabla g^m(\boldsymbol{\lambda}^m) \ , \tag{72}$$

where $\nabla g^m(\lambda^m)$ denotes a subgradient of $g^m(\cdot)$ at λ^m, and then be projected back onto the feasible set Λ. Based on lemma 1, a subgradient of $g^m(\cdot)$ at λ^m is given by

$$\nabla g^m(\lambda^m) = \hat{x}^m \ ,$$

where $\hat{x}^m$ represents any optimal solution to slave MRF (71). Therefore, the above update (72) reduces to setting

$$\lambda^m \leftarrow \lambda^m + \alpha_t \hat{x}^m \ . \tag{73}$$

It then only remains projecting the resulting $\{\lambda^m\}_{m=1}^M$ onto the feasible set Λ. Due to the definition of Λ, this projection reduces to subtracting the average vector $\frac{\sum_{m:p\in\mathbf{V}_m}\lambda_p^m}{|_{m:p\in\mathbf{V}_m}|}$ from each λ_p^m (so that $\sum_{m:p\in\mathbf{V}_m}\lambda_p^m = 0$), as well as subtracting the average vector $\frac{\sum_{m:e\in\mathbf{E}_m}\lambda_e^m}{|_{m:e\in\mathbf{E}_m}|}$ from each λ_e^m (so that $\sum_{m:e\in\mathbf{E}_m}\lambda_e^m = 0$). By aggregating update (73) with the above projection operations, the overall projected subgradient update is easily seen to reduce to $\lambda_p^m \mathrel{+}= \Delta\lambda_p^m$, $\lambda_e^m \mathrel{+}= \Delta\lambda_e^m$, where:

$$\Delta\lambda_p^m = \alpha_t \cdot \left(\hat{x}_p^m - \frac{\sum_{m':p\in\mathbf{V}_{m'}} \hat{x}_p^{m'}}{|_{m':p\in\mathbf{V}_{m'}}|} \right) \tag{74}$$

$$\Delta\lambda_e^m = \alpha_t \cdot \left(\hat{x}_e^m - \frac{\sum_{m':e\in\mathbf{E}_{m'}} \hat{x}_e^{m'}}{|_{m':e\in\mathbf{E}_{m'}}|} \right) \tag{75}$$

In addition, we observe that the dual variables λ^m participate in the objective function only for defining the MRF-parameters $\theta^m + \lambda^m$ of the slave MRF in (71). Therefore, if we make the change of variables

$$(\forall m \in \{1,\ldots,M\}) \qquad \theta^m = \theta^m + \lambda^m, \tag{76}$$

this eliminates the need of updating and storing the dual variables, thus allowing us to directly update the MRF-parameters $\{\theta^m\}_{m=1}^M$ as follows

$$\theta_p^m \mathrel{+}= \Delta\lambda_p^m, \ \theta_e^m \mathrel{+}= \Delta\lambda_e^m \ .$$

Algorithm 12 Dual decomposition for MRF optimization.

Choose a decomposition $\{G_m = (\mathbf{V}_m, \mathbf{E}_m)\}_{1 \leq m \leq M}$ of G
Initialize potentials of slave MRFs:
$(\forall m \in \{1, \ldots, m\})(\forall p \in \mathbf{V}_m)\ \theta_p^m = \frac{\theta_p}{|\{m' | p \in \mathbf{V}_{m'}\}|}, (\forall e \in \mathbf{E}_m)\ \theta_e^m = \frac{\theta_e}{|\{m' | e \in \mathbf{E}_{m'}\}|}$
for $t = 0, \ldots$

Compute minimizers of slave MRF problems:
$(\forall m \in \{1, \ldots, M\})\ \widehat{x}^m \in \underset{x^m \in \mathcal{X}_{G_m}}{\operatorname{argmin}}\ f^m(x^m; \theta^m)$
Update potentials of slave MRFs:
$(\forall m \in \{1, \ldots, m\})(\forall p \in \mathbf{V}_m)\ \theta_p^m = \theta_p^m + \alpha_t \left(\hat{x}_p^m - \frac{\sum_{m' : p \in \mathbf{V}_m} \hat{x}_p^{m'}}{|\{m' | p \in \mathbf{V}_{m'}\}|} \right)$
$(\forall m \in \{1, \ldots, m\})(\forall e \in \mathbf{E}_m)\ \theta_e^m = \theta_e^m + \alpha_t \left(\hat{x}_e^m - \frac{\sum_{m' : e \in \mathbf{E}_{m'}} \hat{x}_e^{m'}}{|\{m' | e \in \mathbf{E}_{m'}\}|} \right)$

This is actually how we end up with the pseudocode shown in algorithm 12, which describes one basic update of the resulting subgradient method.

Note that this algorithm requires only *solutions to local subproblems* to be computed, which is, of course, a task much easier that furthermore can be executed in a *parallel manner*. Moreover, as the following theorem certifies, this algorithm is guaranteed to solve the relaxation of problem 64 that corresponds to the chosen decomposition (see proposition 2.2 in [46] for a proof of a generalized version of this theorem):

Theorem .1 *The optimization problem corresponding to any decomposition* $\{G_m = (\mathbf{V}_m, \mathbf{E}_m)\}_{1 \leq m \leq M}$ *is a relaxation to the original MRF optimization problem 64 and the above algorithm computes the optimum of that relaxation if the sequence of multipliers* $\{\alpha_t\}$ *satisfies the following conditions*

$$\alpha_t \geq 0,\ \lim_{t \to \infty} \alpha_t = 0,\ \sum_{t=0}^{\infty} \alpha_t = \infty\ . \tag{77}$$

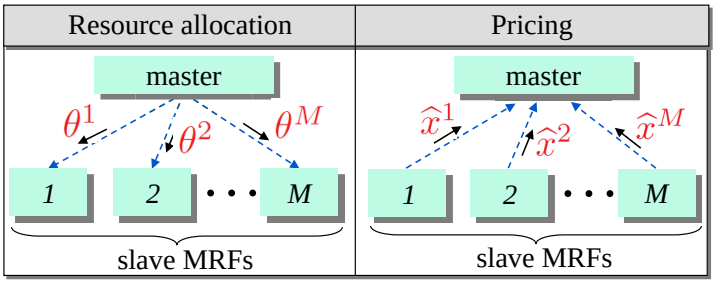
After convergence of the algorithm, the solution to the master MRF can be filled in from local solutions $\left\{ \{\hat{x}_p^m\}_{p \in \mathbf{V}_m}, \{\hat{x}_e^m\}_{e \in \mathbf{E}_m} \right\}_{1 \leq m \leq M}$. We should note at this point that the recovery of primal solutions based on dual subgradients is a subject that has attracted significant interest in the optimization literature for subgradient methods. For instance, a popular way of generating such solutions, that has been studied in a number of existing works, is via utilizing ergodic sequences (*i.e.*sequences of weighted averages) of dual subgradients (note that the use of an ergodic sequence forms a common technique for inducing convergence properties that an original sequence lacks). An early example of such an averaging scheme based on dual subgradient information is the method of Shor [57] for linear optimization problems. That work has been extended by Sherali and Choi [56] to allow for more general choices for the weights used during averaging. Furthermore, recently, Larsson *et al.* [43] have generalized these results to convex constrained optimization problems. The method proposed by Larsson *et al.* utilizes ergodic sequences either of the form

$$\hat{x}^k = \frac{\sum_{t=1}^{k} a_t s^t}{\sum_{t=1}^{k} a_t}\ ,\ k = 1, 2, \ldots \tag{78}$$

or of the form

$$\hat{x}^k = \frac{\sum_{t=1}^{k} s^t}{k}\ ,\ k = 1, 2, \ldots \tag{79}$$

MASTER-SLAVE COMMUNICATION
During dual decomposition a communication between a master process and the slaves (local subproblems) can be thought of as taking place, which can also be interpreted as a resource allocation/pricing stage.



Resource allocation: At each iteration, the master assigns new MRF potentials (i.e., resources) $(\theta^m)_{1\leq m\leq M}$ to the slaves based on the current local solutions $(\hat{x}^m)_{1\leq m\leq M}$.
Pricing: The slaves respond by adjusting their local solutions $(\hat{x}^m)_{1\leq m\leq M}$ (i.e., the prices) so as to maximize their welfares based on the newly assigned resources $(\theta^m)_{1\leq m\leq M}$.

Figure 23: MRF dual decomposition as a resource-allocation and pricing scheme.

In the above formulas, s^t represents the dual subgradient at the t-th iteration, while a_t denotes the stepsize used at the t-th iteration. As shown in [43] the resulting sequence $\{\hat{x}^k\}$ is guaranteed to converge to an optimal primal solution. In general, convergence happens only in the limit. However, more recent work [47] also provides convergence rates estimates, including per iteration estimates for the amount of feasibility violation, as well as upper and lower bounds for the primal objective function. Note that, given that in our case each subgradient s^t is composed of local MRF solutions, the above ergodic schemes essentially reduces to taking averages of local solutions estimated across different iterations. Intuitively, this can be thought of as some sort of voting scheme, where one counts the number of times a label is assinged to a MRF node (taking into account all local solutions up to the current iteration).

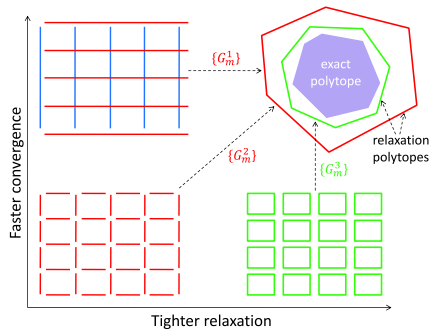
For a better intuition for the updates of dual variables $\left\{\{\theta_p^m\}_{p\in\mathbf{V}_m}, \{\theta_e^m\}_{e\in\mathbf{E}_m}\right\}_{1\leq m\leq M}$ in Algorithm 12, we should note that their aim is essentially to bring a consensus among the solutions of the local subproblems (an easy way to see that is by noticing that these updates are based on taking the *average* of local solutions). In other words, they try to adjust the potentials of the slave MRFs so that in the end the corresponding local solutions are consistent with each other, i.e., all variables corresponding to a common vertex or edge are assigned the same value by the different subproblems. If this condition is satisfied (i.e., there is a full consensus) then the overall solution that results from combining the consistent local solutions is guaranteed to be optimal. In general, though, this might not always be true given that the above procedure is solving only a *relaxation* of the original NP-hard problem. Another intuitive way to interpret the above algorithm is also in terms of a resource allocation and pricing scheme (see Fig. 23 for more details).

5.4 Choice of MRF decompositions

There is great flexibility with respect to the choice of decompositions that can be used in the above framework. Interestingly, if we choose to use a decomposition consisting only of subgraphs

DECOMPOSITIONS AND RELAXATIONS

Different decompositions can lead to different relaxations and/or can affect the speed of convergence. We show below, for instance, 3 possible decompositions for an MRF assumed to be defined on a 5×5 image grid.



Decompositions $\{G_m^1\}, \{G_m^2\}, \{G_m^3\}$ consist respectively of one suproblem per row and column, one subproblem per edge, and one subproblem per 2×2 subgrid of the original 5×5 grid. Both $\{G_m^1\}$ and $\{G_m^2\}$ (due to using solely subgraphs that are trees) lead to the same LP relaxation of (64), whereas $\{G_m^3\}$ leads to a relaxation that is tighter (due to containing loopy subgraphs).
On the other hand, decomposition $\{G_m^1\}$ leads to faster convergence compared with $\{G_m^2\}$ due to using larger subgraphs that allow a faster propagation of information during message-passing.

Figure 24: Decompositions and relaxations

that are trees, then the resulting relaxation can be shown to actually coincide with the standard LP-relaxation of linear integer program (64) (obtained by replacing the integrality constraints with non-negativity constraints on the variables). This is certified in the following theorem

Theorem .2 *If each subgraph G_m is a tree, then the MRF relaxation corresponding to decomposition $\{G_m = (\mathbf{V}_m, \mathbf{E}_m)\}_{1 \leq m \leq M}$ is equivalent to the standard LP relaxation of the linear integer programming formulation (64).*

This also means that when this LP-relaxation is tight (as, *e.g.*, in the case of submodular MRFs) an optimal MRF solution is computed (it should be noted, though, that for submodular problems much faster graph-cut based optimization techniques exist, as already explained in section 2.5). This directly leads to the following result:

Theorem .3 *Tree-based dual decomposition approaches can estimate a globally optimal solution for binary submodular MRFs.*

Furthermore, when using subgraphs that are trees, a minimizer to each slave problem can be computed efficiently by applying the Belief Propagation algorithm [50], which is a message-passing method. Therefore, in this case, Algorithm 12 essentially reduces to and can also be implemented as a continuous exchange of messages between the nodes of graph G. Such an algorithm relates to or generalizes various other message-passing approaches [62, 29, 65, 19, 66, 21].

In general, besides tree-structured subgraphs, other types of decompositions or subproblems can be used as well, which can serve different purposes, such as

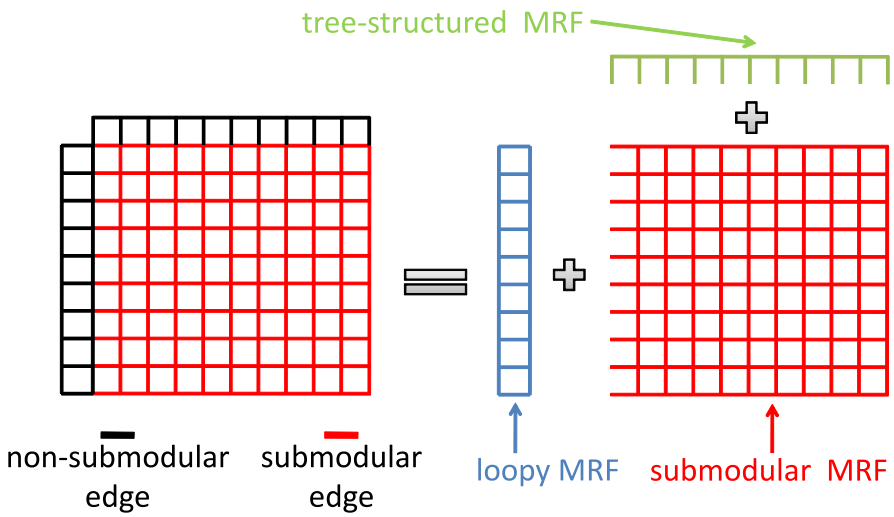


Fig. 25: A decomposition of a MRF into three types of subproblems: a loopy MRF with small tree width, a submodular MRF and a tree-structured MRF. Dual decomposition provides a principled way of combining the subproblems' solutions for optimizing the MRF on the full grid (note that a global minimizer can be computed efficiently for all three types of subproblems using respectively graph-cut based techniques, the junction tree algorithm and belief propagation).

- decompositions for high-order MRFs [34]
- decompositions that lead to tighter MRF relaxations [35, 33, 58], which are based, *e.g.*, on utilizing loopy subgraphs of small tree-width, for which slave MRFs can still be efficiently optimized (see Fig. 24)
- planar decompositions consisting of binary slave problems that are defined over loopy planar graphs (of arbitrary tree-width) [55, 67]
- and submodular decompositions [48, 24].

More generally, when applying dual decomposition to MRF optimization, we can make use of and combine different types of slave subproblems. An illustration of this is provided in Fig. 25, where a MRF is shown that has been decomposed into a submodular subproblem, a loopy MRF with small tree width, and a tree-structured MRF. Note that a global minimizer can be computed efficiently for all 3 types of subproblems using respectively graph-cut based techniques, the junction tree algorithm and belief propagation.

5.5 ADMM

Besides the projected subgradient method, one can alternatively choose to apply an ADMM-based scheme for solving the dual relaxation (70). The main difference, in this case, is that the optimization of a slave MRF problem is performed by solving a (usually simple) local quadratic problem, which can again be solved efficiently for an appropriate choice of the decomposition. This method again penalizes disagreements among slaves, but it does so even more aggressively than the subgradient method since there is no longer a requirement for step-sizes $(\alpha_t)_{t\in\mathbb{N}}$ converging to zero. [1] show how to efficiently optimize the resulting local quadratic subproblems

for several types of MRFs. Furthermore, besides projected subgradient and ADMM, alternative smoothed accelerated schemes exist and can be applied as well [25, 52, 53].

References

[1] M. V. Afonso, J. M. Bioucas-Dias, and M. A. T. Figueiredo. An augmented Lagrangian approach to the constrained optimization formulation of imaging inverse problems. 20(3):681–695, Mar. 2011.

[2] A. Archer, J. Fakcharoenphol, C. Harrelson, R. Krauthgamer, K. Talvar, and E. Tardos. Approximate classification via earthmover metrics. In *SODA*, 2004.

[3] Y. Bartal. On approximating arbitrary metrics by tree metrics. In *STOC*, 1998.

[4] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *PAMI*, 2004.

[5] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. 23(11):1222–1239, Nov. 2001.

[6] Yuri Boykov, Olga Veksler, and Ramin Zabih. A variable window approach to early vision. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(12):1283–1294, 1998.

[7] D. Bryant and P. F. Tupper. Hyperconvexity and tight-span theory for diversities. In *Advances in Mathematics*, 2012.

[8] C. Chekuri, S. Khanna, J. Naor, and L. Zosin. Approximation algorithms for the metric labeling problem via a new linear programming formulation. In *12^{th} Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 109–118, Washington D.C., USA, 7-9 Jan. 2001.

[9] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. In *PAMI*, 2002.

[10] A. Delong, L. Gorelick, O. Veksler, and Y. Boykov. Minimizing energies with hierarchical costs. In *IJCV*, 2012.

[11] A. Delong, A. Osokin, H. Isack, and Y. Boykov. Fast approximate energy minimization with label costs. In *CVPR*, 2010.

[12] P. Dokania and M. Pawan Kumar. Parsimonious labeing. In *ICCV*, 2015.

[13] J. Fakcharoenphol, S. Rao, and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. In *STOC*, 2003.

[14] P. Felzenszwalb and D. Huttenlocher. Efficient belief propagation for early vision. In *CVPR*, 2004.

[15] M. A. Fischler and R. A. Elschlager. The representation and matching of pictorial structures. *IEEE Trans. Comput.*, 22(1):67–92, January 1973.

[16] A. Fix, A. Gruber, E. Boros, and R. Zabih. A graph cut algorithm for high-order Markov random fields. In *ICCV*, 2011.

[17] B. Flach and D. Schlesinger. Transforming an arbitrary minsum problem into a binary one. Technical Report TUD-F106-01, Dresden University of Technology, 2006.

[18] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 6(6):721–741, November 1984.

[19] A. Globerson and T. Jaakkola. Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations. pages 553–560, Vancouver and Whistler, Canada, 3-6 Dec. 2007.

[20] A. Gupta and E. Tardos. A constant factor approximation algorithm for a class of classification problems. In *STOC*, 2000.

[21] T. Hazan and A. Shashua. Norm-product belief propagation: Primal-dual message-passing for approximate inference. 56(12):6294–6316, Dec. 2010.

[22] D. S. Hochbaum, editor. *Approximation Algorithms for NP-hard Problems*. PWS Publishing Co., Boston, MA, USA, 1997.

[23] H. Ishikawa. Exact optimization for Markov random fields with convex priors. *PAMI*, 25(10):1333–1336, October 2003.

[24] S. Jegelka, F. Bach, and S. Sra. Reflection methods for user-friendly submodular optimization. pages 1313–1321, Lake Tahoe, NV, USA, 5-10 Dec. 2013.

[25] V. Jojic, S. Gould, and D. Koller. Fast and smooth: Accelerated dual decomposition for MAP inference. pages 503–510, Haifa, Israel, 21-24 June 2010.

[26] F. Kahl and P. Strandmark. Generalized roof duality. *Discrete Appl. Math.*, 160(16-17):2419–2434, 2012.

[27] J. Kleinberg and E. Tardos. Approximation algorithms for classification problems with pairwise relationships: Metric labeling and Markov random fields. In *STOC*, 1999.

[28] P. Kohli, M. Pawan Kumar, and P. Torr. p^3 and beyond: Move-making algorithms for solving higher order functions. *PAMI*, 2009.

[29] V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. 28(10):1568–1583, Aug. 2006.

[30] V. Kolmogorov. Generalized roof duality and bisubmodular functions. pages 1144–1152, Vancouver, Canada., 6-9 Dec. 2010.

[31] V. Kolmogorov. Minimizing a sum of submodular functions. In *Discrete Applied Mathematics*, 2012.

[32] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? 26(2):147–159, Feb. 2004.

[33] N. Komodakis and N. Paragios. Beyond loose LP-relaxations: Optimizing MRFs by repairing cycles. pages 806–820, Marseille, France, 12-18 Oct. 2008.

[34] N. Komodakis and N. Paragios. Beyond pairwise energies: Efficient optimization for higher-order MRFs. pages 2985–2992, Miami, FL, USA, 20-25 June 2009.

[35] N. Komodakis, N. Paragios, and G. Tziritas. MRF energy minimization and beyond via dual decomposition. 33(3):531–552, 2011.

[36] N. Komodakis and G. Tziritas. Approximate labeling via graph-cuts based on linear programming. 29(8):1436–1453, Aug. 2007.

[37] N. Komodakis, G. Tziritas, and N. Paragios. Performance vs computational efficiency for optimizing single and dynamic MRFs: Setting the state of the art with primal-dual strategies. 112:14–29, 2008.

[38] M. P. Kumar. Rounding-based moves for metric labeling. In *NIPS*, 2014.

[39] M. P. Kumar and D. Koller. MAP estimation of semi-metric MRFs via hierarchical graph cuts. In *UAI*, 2009.

[40] M. P. Kumar, V. Kolmogorov, and P. Torr. An analysis of convex relaxations for MAP estimation. In *NIPS*, 2007.

[41] M. P. Kumar and P. Torr. Improved moves for truncated convex models. In *NIPS*, 2008.

[42] L. Ladicky, C. Russell, P. Kohli, and P. Torr. Inference methods for CRFs with co-occurence statistics. *IJCV*, 2013.

[43] T. Larsson, M. Patriksson, and A. Stromberg. Ergodic primal convergence in dual subgradient schemes for convex programming. *Mathematical Programming*, 86:283–312, 1999.

[44] R. Manokaran, J. Naor, P. Raghavendra, and R. Schwartz. SDP gaps and UGC hardness for multiway cut, 0-extension and metric labeling. In *STOC*, 2008.

[45] K. Murphy, Y. Weiss, and M. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *UAI*, pages 467–476, 1999.

[46] A. Nedic and D. P. Bertsekas. Incremental subgradient methods for nondifferentiable optimization. *SIAM Journal on Optimization*, 12:109–138, 2001.

[47] Angelia Nedic and Asuman Ozdaglar. Approximate primal solutions and rate analysis for dual subgradient methods. *SIAM Journal on Optimization*, 19:1757–1780, 2009.

[48] A. Osokin, D. Vetrov, and V. Kolmogorov. Submodular decomposition framework for inference in associative markov networks with global constraints. pages 1889–1896, Colorado Springs, USA, 21-23 June 2011.

[49] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, Englewoods Cliffs, N.J., 1982.

[50] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, San Francisco, CA, USA, 1988.

[51] R. B. Potts. Some generalized order-disorder transformations. In *Cambridge Philosophical Society*, 1952.

[52] B. Savchynskyy, J. H. Kappes, S. Schmidt, and C. Schnörr. A study of Nesterov's scheme for Lagrangian decomposition and MAP labeling. pages 1817–1823, Colorado Springs, USA, 21-23 June 2011.

[53] B. Savchynskyy, S. Schmidt, J. H. Kappes, and C. Schnörr. Efficient MRF energy minimization via adaptive diminishing smoothing. pages 746–755, Catalina Island, USA, 15-17 Aug. 2012.

[54] M. Schlesinger. Syntactic analysis of two-dimensional visual signals in noisy conditions. *Kibernetika*, 1976.

[55] N. N. Schraudolph. Polynomial-time exact inference in NP-hard binary MRFs via reweighted perfect matching. In *13-th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 717–724, Chia Laguna Resort, Sardinia, Italy, 13-15 May 2010.

[56] H.D. Sherali and G. Choi. Recovery of primal solutions when using subgradient optimization methods to solve lagrangian duals of linear programs. *Operations Research Letters*, 19:105–113, 1996.

[57] N.Z. Shor. *Minimization methods for nondifferentiable functions*. Springer, Berlin, 1985.

[58] D. Sontag, T. Meltzer, A. Globerson, Y. Weiss, and T. Jaakkola. Tightening LP relaxations for MAP using message passing. pages 656–664, Helsinki, Finland, 9-12 Jul. 2008.

[59] V. V. Vazirani. *Approximation Algorithms*. Springer-Verlag, New York, NY, USA, 2001.

[60] O. Veksler. *Efficient graph-based energy minimization methods in computer vision*. PhD thesis, Cornell University, 1999.

[61] O. Veksler. Graph cut based optimization for MRFs with truncated convex priors. In *CVPR*, 2007.

[62] M. Wainwright, T. Jaakkola, and A. Willsky. MAP estimation via agreement on trees: message-passing and linear programming. 51(11):3697–3717, Nov. 2005.

[63] Chaohui Wang, Nikos Komodakis, and Nikos Paragios. Markov random field modeling, inference & learning in computer vision & image understanding: A survey. *Comput. Vis. Image Underst.*, 117(11):1610–1627, November 2013.

[64] Y. Weiss and W. Freeman. On the optimality of solutions of the max-product belief propagation algorithm in arbitrary graphs. *IEEE Trans. on Information Theory*, 47(2):723–735, 2001.

[65] T. Werner. A linear programming approach to max-sum problem: A review. 29(7):1165–1179, Jul. 2007.

[66] C. Yanover, T. Talya Meltzer, and Y. Weiss. Linear programming relaxations and belief propagation – an empirical study. 7:1887–1907, Sep. 2006.

[67] J. Yarkony, R. Morshed, A. T. Ihler, and C. Fowlkes. Tightening MRF relaxations with planar subproblems. pages 770–777, Barcelona, Spain, 14-17 Jul. 2011.

Contents

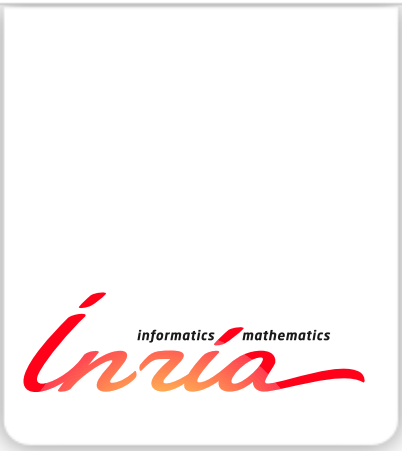
1 Introduction **3**

2 Mathematical background: basic tools for MRF inference **4**
2.1 Markov random fields . . . 4
2.2 Reparameterization . . . 5
2.3 Dynamic programming . . . 5
2.4 Message passing and belief propagation . . . 7
2.5 Graph cuts . . . 9
2.6 Linear programming . . . 10

3 Move-making algorithms **12**
3.1 Preliminaries . . . 13
3.2 Complete rounding and complete move . . . 15
3.3 Interval rounding and interval moves . . . 16
3.4 Hierarchical rounding and hierarchical moves . . . 19
3.5 Dense stereo correspondence . . . 22
3.6 Moves for high-order potentials . . . 23
3.6.1 Parsimonious labeling . . . 23
3.6.2 P^n Potts model . . . 28
3.6.3 Hierarchical P^n Potts model . . . 32
3.6.4 Handling general diversities . . . 34
3.6.5 Related works . . . 35
3.6.6 Dense stereo correspondence . . . 35

4 MRF inference based on the primal-dual schema **37**
4.1 Relaxations and discrete optimization . . . 38
4.2 The primal-dual schema for integer linear programming . . . 39
4.2.1 Application to the set cover problem . . . 42
4.3 Primal-dual schema for discrete MRF optimization . . . 44
4.4 Computational efficiency of primal-dual MRF optimization . . . 47

5 MRF inference based on dual-decomposition **48**
5.1 The principle of dual decomposition . . . 49
5.2 Dual decomposition for discrete MRF optimization . . . 51
5.3 Applying projected subgradient to the MRF dual . . . 53
5.4 Choice of MRF decompositions . . . 55
5.5 ADMM . . . 57



RESEARCH CENTRE
SACLAY – ÎLE-DE-FRANCE

Parc Orsay Université
4 rue Jacques Monod
91893 Orsay Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399