Optimization of Markov Random Fields in Computer Vision

Thalaiyasingam Ajanthan

A thesis submitted for the degree of
Doctor of Philosophy
The Australian National University

November 2017

Declaration

I hereby declare that this submission is my own work (based on publications in collaboration with the co-authors where due acknowledgement is made) and that, to the best of my knowledge, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma at ANU or any other educational institution, except where due acknowledgment has been made.

I also declare that all sources used in this thesis have been fully and properly cited.

[illegible]

Thalaiyasingam Ajanthan
20 November 2017

To my family.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my primary supervisor, Prof. Richard Hartley, for his guidance, motivation, and support throughout my PhD research. It has been an absolute privilege to work with Richard, and this work would not have been possible without his contagious enthusiasm for research. Thank you, professor, for being immensely helpful in many ways both in academic and personal matters. In particular, thank you for inspiring me to work on the exciting field of MRF optimization. Your upcoming book on *Solution of Markov Random Fields* has been instrumental in reducing my learning time.

I am equally grateful to my co-supervisors, As.Prof. Hongdong Li (panel chair) and Dr. Mathieu Salzmann. Thank you for giving me valuable pointers and ideas, shaping my research and carefully reviewing all manuscripts I produced during my PhD. I am indebted to Mathieu for his continuous guidance even after he moved to EPFL, Switzerland. I would also like to thank the other researchers at Data61 and ANU for their feedback on my research during various seminars and reading groups.

I would like to thank Prof. Philip Torr for offering me to visit his group at the University of Oxford for almost 5 months and introducing me to exciting new problems. I am deeply grateful to As.Prof. Pawan Kumar for being a great mentor and working with me during my time in Oxford. I would also like to thank my collaborators, Alban and Rudy and other colleagues at the University of Oxford for warmly welcoming me to their circle.

I am grateful to my colleagues at Data61 and ANU, past and present, who created a wonderful research environment. Especially, I would like to convey my gratitude to Sadeep (for introducing me to Mathieu and guidance on various matters), Kamalaruban (for the countless things he has done for me), Zeeshan, Arash, Saeed, Masoud and Samitha.

I graciously acknowledge and appreciate the financial support from the Australian National University, NICTA (now Data61, CSIRO) and the Australian Government for my PhD research. Their scholarships and generous travel grants have allowed me to focus on my research without having to worry about financial support.

I am indebted to my teachers at the University of Moratuwa, especially, Dr. Ranga Rodrigo for constantly encouraging me to pursue a PhD in computer vision. I am equally indebted to my teachers back in my hometown Jaffna, Sri Lanka.

Most importantly, I would like to express my gratitude to my family and friends. Thank you, my father late Thalaiyasingam, mother Jeyaluxmy, and brother Suganthan, for your unconditional love and support. This thesis is dedicated to you. Finally, my dear friends (in Canberra and other parts of the world), thank you for making my life fun-filled.

Abstract

A large variety of computer vision tasks can be formulated using Markov Random Fields (MRF). Except in certain special cases, optimizing an MRF is intractable, due to a large number of variables and complex dependencies between them. In this thesis, we present new algorithms to perform inference in MRFs, that are either more efficient (in terms of running time and/or memory usage) or more effective (in terms of solution quality), than the state-of-the-art methods.

First, we introduce a memory efficient max-flow algorithm for *multi-label submodular* MRFs. In fact, such MRFs have been shown to be optimally solvable using max-flow based on an encoding of the labels proposed by Ishikawa, in which each variable X_i is represented by ℓ nodes (where ℓ is the number of labels) arranged in a column. However, this method in general requires $2\,\ell^2$ edges for each pair of neighbouring variables. This makes it inapplicable to realistic problems with many variables and labels, due to excessive memory requirement. By contrast, our max-flow algorithm stores $2\,\ell$ values per variable pair, requiring much less storage. Consequently, our algorithm makes it possible to optimally solve multi-label submodular problems involving large numbers of variables and labels on a standard computer.

Next, we present a move-making style algorithm for multi-label MRFs with robust non-convex priors. In particular, our algorithm iteratively approximates the original MRF energy with an appropriately weighted surrogate energy that is easier to minimize. Furthermore, it guarantees that the original energy decreases at each iteration. To this end, we consider the scenario where the weighted surrogate energy is multi-label submodular (*i.e.*, it can be optimally minimized by max-flow), and show that our algorithm then lets us handle of a large variety of non-convex priors.

Finally, we consider the fully connected Conditional Random Field (dense CRF) with Gaussian pairwise potentials that has proven popular and effective for multi-class semantic segmentation. While the energy of a dense CRF can be minimized accurately using a Linear Programming (LP) relaxation, the state-of-the-art algorithm is too slow to be useful in practice. To alleviate this deficiency, we introduce an efficient LP minimization algorithm for dense CRFs. To this end, we develop a proximal minimization framework, where the dual of each proximal problem is optimized via block-coordinate descent. We show that each block of variables can be optimized in a time *linear* in the number of pixels and labels. Consequently, our algorithm enables efficient and effective optimization of dense CRFs with Gaussian pairwise potentials.

We evaluated all our algorithms on standard energy minimization datasets consisting of computer vision problems, such as stereo, inpainting and semantic segmentation. The experiments at the end of each chapter provide compelling evidence that all our approaches are either more efficient or more effective than all existing baselines.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Computer vision aims at building artificial systems that extract useful information (*e.g.*, objects, their boundaries, motion and 3D information, etc.) from images and videos. Unlike humans, this is an extremely difficult task for a computer, because it needs to interpret image data (a bunch of matrices) to descriptions of the world. Due to the inherent uncertainty, many of these problems are formulated as inference in probabilistic graphical models.

Probabilistic Graphical Models (PGM) [Koller and Friedman, 2009] describe multivariate probability distributions which factor according to a graph structure. Specifically, the underlying graph expresses the conditional dependence structure between random variables that hold in the encoded distribution. Two branches of graphical models are commonly used, namely, Bayesian networks and Markov Random Fields (MRF). These two models differ in the graph structure (*i.e.*, in the set of independences they can encode), namely, for Bayesian networks, the underlying graph is directed and acyclic and in the case of MRFs, it is an undirected graph.

In this thesis, we focus on Markov random fields. Inference in an MRF is often expressed as a discrete optimization problem, where the dependencies of the variables are encoded in a graph. Such problems are often referred to as *combinatorial optimization*[1] problems. This combinatorial nature makes it a theoretically interesting research problem and in general, the optimization is NP-hard, *i.e.*, computationally intractable [Nemhauser and Wolsey, 1988]. Furthermore, many computer vision applications of MRF have millions of variables and complex dependencies between them, making the inference more challenging. Therefore, to develop practically efficient algorithms, researchers either restrict themselves to a certain class of MRFs (*e.g.*, restricted graph structures and restricted cost functions) or focus on developing approximate algorithms.

The existing MRF optimization algorithms in computer vision can be categorized into three groups: 1) exact combinatorial algorithms for certain special cases; 2) move-making style algorithms which iteratively minimize surrogate problems; 3) continuous relaxation based methods which make use of convex optimization techniques. In this thesis, we make contributions in all three categories by introducing

[1]Combinatorial optimization is the study of optimization on discrete and combinatorial objects (*e.g.*, graphs).

new algorithms that are either more efficient (in terms of running time and/or memory usage) or more effective (in terms of solution quality), than the state-of-the-art methods.

Before presenting our contributions, we briefly review a few examples of computer vision problems and show how they can be formulated as discrete labelling problems.

1.1 Some Computer Vision Applications

Here we briefly discuss some core computer vision applications that are usually formulated as MRFs and hence a *discrete labelling problem*. We will formally define an MRF and related inference algorithms in Chapter 2. However, for better understanding, we first define the labelling problem and then turn to the examples.

1.1.1 The Labelling Problem

The objective of the labelling problem is to classify a set of nodes $\mathcal{V} = \{1, 2, \ldots, n\}$ by assigning a label to each node from a label set $\mathcal{L}$. In this thesis, we consider the label set $\mathcal{L}$ to be discrete and finite. In particular, the labelling can be represented by a function $x : \mathcal{V} \to \mathcal{L}$, and the label assignment of a node i is usually denoted as $x_i = x(i)$.

Note that there are $|\mathcal{L}|^n$ possible label configurations. In a typical computer vision problem, each label configuration has a cost associated with it (called an *energy*), and the objective is to find the label configuration with the minimum cost (called *energy minimization*). Except in certain special cases, one has to check all $|\mathcal{L}|^n$ possible configurations, making it an intractable problem. It will be seen in the next chapter that a Markov random field can be expressed as a discrete labelling problem.

1.1.2 Stereo Correspondence Estimation

In stereo correspondence estimation [Scharstein and Szeliski, 2002], a pair of *calibrated* images are given, a left image and a right image, and the objective is to find the *disparity* between those two images. Here, by calibrated, we mean that the two images are aligned up to a horizontal displacement only, and disparity means that the horizontal displacement of a pixel in the left image to the corresponding pixel in the right image. Hence, it is a labelling problem where $\mathcal{V}$ is the set of pixels and $\mathcal{L}$ is the set of possible disparities. See Figure 1.1 for an example. In fact, given the disparity, the depth can be determined [Hartley and Zisserman, 2003] and therefore stereo estimation is essential to obtain 3D information of a scene.

1.1.3 Image Denoising and Inpainting

Another low level vision problem is image denoising and inpainting [Szeliski et al., 2008]. Here, given a noisy image with corrupted pixels (due to problems in image

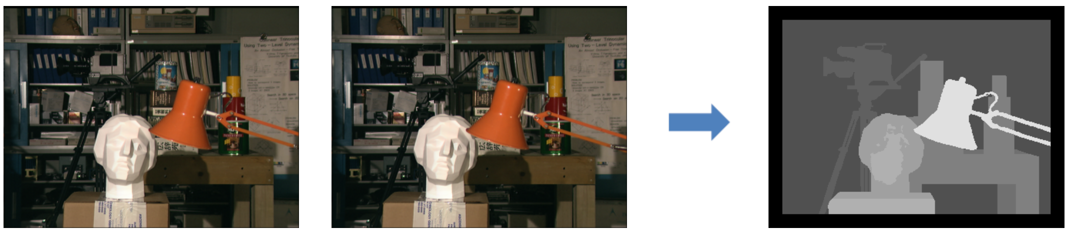

(a) Left image (b) Right image (c) Disparity map

Figure 1.1: *An example of stereo correspondence estimation. Given the left and right images, we need to find the disparity map between them.*

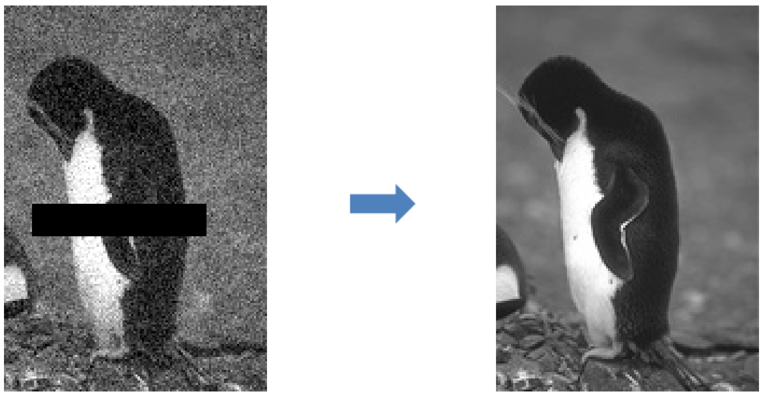

(a) Corrupted image (b) Restored image

Figure 1.2: *An example of image denoising and inpainting. Given the noisy image with corrupted pixels, we need to restore the original image.*

acquisition), the objective is to restore the original content of the image. In this case, one needs to estimate the pixel intensities. Therefore, this is a labelling problem where $\mathcal{V}$ is the set of pixels and $\mathcal{L}$ is the set of intensities ($\{0, \ldots, 255\}$ in case of a gray scale image). See Figure 1.2 for an example.

1.1.4 Semantic Segmentation

Semantic segmentation is a high-level vision task [Everingham et al., 2010] where we need to partition a given an image into regions and assign each region to an object class. In fact, in contrast to putting bounding boxes around objects, we need to segment the images according to object boundaries. Similarly to stereo and inpainting, this is also a labelling problem, where $\mathcal{V}$ is the set of pixels and $\mathcal{L}$ is the possible object classes. A more sophisticated task of semantic segmentation is instance segmentation, where we need to identify and segment different instances of the same object. See Figure 1.3 for examples of object segmentation and instance segmentation. These problems are useful for scene understanding and practical applications, such as autonomous driving.

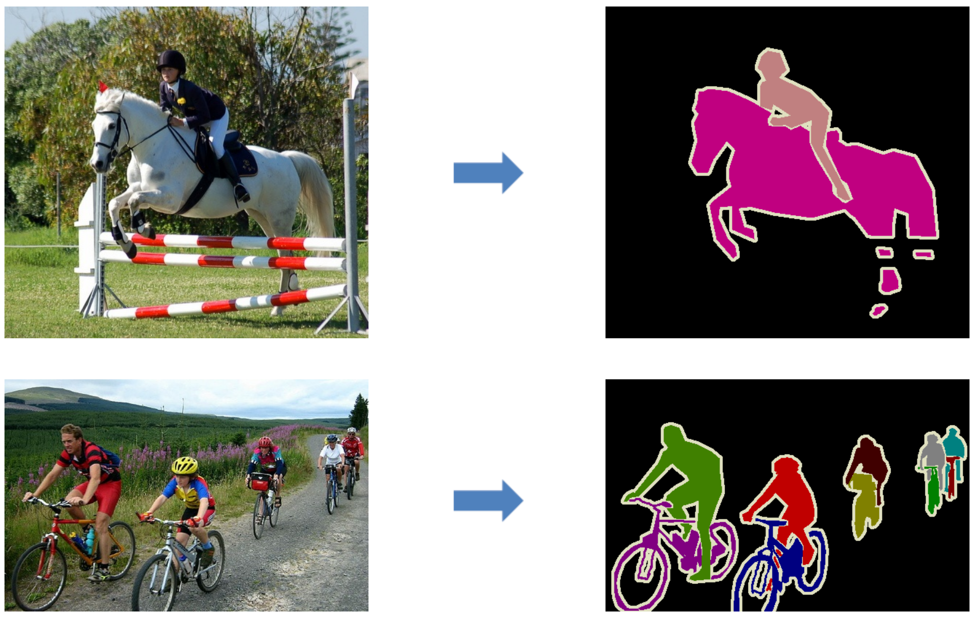

(a) Input image (b) Segmented image

Figure 1.3: *An example of object segmentation (**top**) and instance segmentation (**bottom**). Given the input image, we need to segment the objects (or instances) and label them. The color indicates the class (or instance) label.*

1.2 Limitations of Existing Algorithms

Note that, in most vision applications, there is a variable in the MRF corresponding to each pixel in the image[2]. Therefore, a typical MRF has millions of variables and complex dependencies between them, defined by the underlying graph structure. Usually, in computer vision, a sparse connectivity is assumed (*e.g.*, 4-connected or 8-connected neighbourhood structure). However, as will be seen later, in certain applications, dense connectivity (*i.e.*, each node is connected to every other node) is preferred [Krähenbühl Philipp, 2011]. In fact, the difficulty of optimizing an MRF depends on the underlying graph structure and the form of the cost function used to model the problem.

Over the past decade, various MRF optimization algorithms have been introduced. They can be categorized into three groups: 1) exact algorithms for certain special cases; 2) move-making style algorithms; 3) continuous relaxation based methods. In fact, exact optimization is possible only for certain restricted cases (*e.g.*, submodular energy functions and tree structured MRFs). For example, when the MRF energy is *submodular* (defined later in Chapter 2), it can be optimized using the max-flow algorithm [Kolmogorov and Zabin, 2004]. However, for general MRFs,

[2]Superpixels (polygonal parts in the image resulting from partitioning) based MRFs have also been studied in the literature, but they are usually less accurate in pixelwise labelling tasks [Kappes et al., 2015].

one has to settle for an approximate algorithm. One class of algorithms approximate the original MRF energy by iteratively minimizing a surrogate energy (usually a submodular energy), and guarantee a monotonic decrease in the original MRF energy. At each iteration, such an algorithm moves from the current labelling to a lower energy labelling, and therefore this class of algorithms are referred to as move-making algorithms [Boykov et al., 2001]. On the other hand, the discrete labelling problem can be relaxed to a continuous optimization problem (such as a linear program [Chekuri et al., 2004; Werner, 2007] or a quadratic program [Ravikumar and Lafferty, 2006]) and tackled using convex optimization techniques.

We now point out some limitations of the existing algorithms in each of the above mentioned category, and, in the next section, we briefly discuss how our algorithms overcome these limitations.

1. Ishikawa [Ishikawa, 2003] introduced a max-flow-based method to globally minimize the energy of *multi-label submodular* MRFs (defined later). In the general case, however, this method requires $2\,\ell^2$ directed edges (where ℓ is the number of labels) for each pair of neighbouring variables. For instance, for a 1000×1000, 4-connected image with 256 labels, it would require approximately $1000 \times 1000 \times 2 \times 256^2 \times 2 \times 4 \approx 1000$ GB of memory to store the edges (assuming 4 bytes per edge). Clearly, this is beyond the storage capacity of most computers. Even though the optimal solution can be obtained in polynomial time, due to excessive memory requirement, this algorithm cannot be applied to realistic problems with many variables and labels.

2. As we pointed out previously, in most scenarios one has to rely on an approximate algorithm to optimize a multi-label MRF. Even though move-making algorithms [Boykov et al., 2001] have proven successful for specific problems (*e.g.*, *metric* priors), there does not seem to be a single algorithm that performs well with different non-convex priors. In particular, while widely acknowledged as highly effective in computer vision, optimizing multi-label MRFs with robust non-convex priors, such as the truncated quadratic, the Cauchy function and the corrupted Gaussian still remains challenging.

3. The fully connected Conditional Random Field[3] (dense CRF) with Gaussian pairwise potentials has proven popular and effective for multi-class semantic segmentation [Krähenbühl Philipp, 2011]. For such problems, the Linear Programming (LP) relaxation provides strong theoretical guarantees on the quality of the solution [Kleinberg and Tardos, 2002; Kumar et al., 2009]. Even though there are efficient algorithms for LP relaxation of sparse CRFs [Kolmogorov, 2006], optimizing a dense CRF is extremely challenging and the state-of-the-art algorithm [Desmaison et al., 2016a] is too slow to be useful in practice.

[3]From the optimization point of view, the CRF and the MRF are identical. This will become clear when we formally define them in Chapter 2.

1.3 Contributions

In this thesis, we introduce new algorithms for MRF optimization, targeting computer vision applications. Our algorithms are either more efficient (in terms of running time and/or memory usage) or more effective (in terms of solution quality), than the state-of-the-art methods. Furthermore, we briefly discuss how our algorithms address the limitations of the existing ones.

1. We introduce a memory efficient variant of the max-flow algorithm for multi-label submodular MRFs. Specifically, our max-flow algorithm stores only two ℓ-dimensional vectors (where ℓ is the number of labels) per variable pair instead of the $2\,\ell^2$ edge capacities of the standard max-flow algorithm. Consequently, we reduce the memory requirement of the max-flow algorithm by $\mathcal{O}(\ell)$ for Ishikawa type graphs, while not compromising on optimality. As a result, our approach lets us optimally solve much larger problems on a standard computer.

2. We present a move-making style algorithm for multi-label MRFs with robust non-convex priors. In particular, our algorithm iteratively approximates the original MRF energy with an appropriately weighted surrogate energy that is easier to minimize. We show that, under suitable conditions on the non-convex priors, and as long as the weighted surrogate energy can be decreased, our approach guarantees that the true energy decreases at each iteration. To this end, we consider the scenario where the global minimizer of the weighted surrogate energy can be obtained by a max-flow algorithm (*i.e.*, the surrogate energy is multi-label submodular), and show that our algorithm then lets us handle of a large variety of non-convex priors.

3. We introduce an efficient LP minimization algorithm for dense CRFs with Gaussian pairwise potentials. In particular, we develop a proximal minimization framework, where the dual of each proximal problem is optimized via block-coordinate descent. We show that each block of variables can be optimized in a time *linear* in the number of pixels and labels. To the best of our knowledge, this constitutes the first LP minimization algorithm for dense CRFs that has linear time iterations. Consequently, our algorithm enables efficient and effective optimization of dense CRFs with Gaussian pairwise potentials.

We evaluated all our algorithms on standard energy minimization datasets consisting of computer vision problems, such as stereo, inpainting and semantic segmentation. The experiments at the end of each chapter provide compelling evidence that all our approaches are either more efficient or more effective than all existing baselines.

1.4 Thesis Outline

The remaining chapters of the thesis are summarized below.

Chapter 2. This chapter provides the fundamentals of Markov random fields and reviews relevant state-of-the-art optimization algorithms. The basic theories related to the algorithms are also introduced. In particular, we discuss the max-flow algorithm for submodular MRFs, formalize move-making algorithms and later we discuss the linear programming relaxation of an MRF. The material included in this chapter is needed to understand the rest of the thesis.

Chapter 3. This chapter is based on our work [Ajanthan et al., 2016] with substantial extensions, and the extended version is available in [Ajanthan et al., 2017c]. Here, we introduce our memory efficient max-flow algorithm and show that the memory requirement reduces by $\mathcal{O}(\ell)$ compared to the standard max-flow algorithm. Furthermore, we prove its polynomial time complexity and also discuss its relationship to the min-sum message passing algorithm (reviewed in Chapter 2).

Chapter 4. This chapter is based on our work published in [Ajanthan et al., 2015] with some extensions. Here, we present our iteratively reweighted graph cut algorithm for multi-label MRFs with a certain class of non-convex priors. We show that, by iteratively minimizing a multi-label submodular energy function, we can approximately minimize MRFs with robust non-convex priors. We also discuss the relationship of our algorithm to the majorize-minimize framework.

Chapter 5. Here we present our efficient LP minimization algorithm for fully connected CRFs with Gaussian pairwise potentials. This work is published in [Ajanthan et al., 2017a] and the extended version is available in [Ajanthan et al., 2017b]. In this chapter, we show how the LP relaxation of a dense CRF can be minimized using a block-coordinate descent algorithm that has *linear* time iterations. To this end, we also discuss a modification to the permutohedral lattice based filtering method [Adams et al., 2010], which enables us to perform approximate Gaussian filtering with ordering constraints in linear time. *The work presented in this chapter was conducted under the supervision of Prof. Philip Torr and As.Prof. Pawan Kumar, during my visit at the Torr Vision Group at the University of Oxford, from 4th July 2016 to 4th December 2016.*

Chapter 6. We conclude the thesis by summarizing the works presented in the thesis and suggesting a number of possible future directions to extend them.

1.5 Publications

The contributions described in this thesis have previously appeared in the following publications.

1.5.1 Under Review

- **T. Ajanthan**, R. Hartley, and M. Salzmann, Memory Efficient Max Flow for Multi-label Submodular MRFs, *Submitted to Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, February 2017.

1.5.2 Conferences

- **T. Ajanthan**, A. Desmaison, R. Bunel, M. Salzmann, P. H. S. Torr and M. P. Kumar, Efficient Linear Programming for Dense CRFs, *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.
- **T. Ajanthan**, R. Hartley, and M. Salzmann, Memory Efficient Max Flow for Multi-label Submodular MRFs, *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.
- **T. Ajanthan**, R. Hartley, M. Salzmann, and H. Li, Iteratively Reweighted Graph Cut for Multi-label MRFs with Non-convex Priors, *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

Chapter 2

Background and Preliminaries

The role of this chapter is to provide the fundamentals of Markov random fields and review some state-of-the-art optimization algorithms. In Section 2.1, we formally introduce Markov random fields. Section 2.2 discusses binary MRF optimization and introduces the basic concepts on pseudo-boolean optimization and the min-cut/max-flow (graph-cut) algorithm. In Section 2.3, we turn to the multi-label MRFs. In this section, we first review a graph-cut algorithm for *multi-label submodular* MRFs, and then formalize move-making algorithms and discuss some useful special cases. Later, we consider the linear programming relaxation of the discrete MRF problem. In this case, we discuss a tree-based decomposition algorithm where each subproblem (tree MRF) is optimally minimized using the message passing algorithm.

2.1 Markov Random Fields

Let $\mathcal{X} = \{X_1, \ldots, X_n\}$ be a set of random variables taking values in a discrete label set $\mathcal{L}$, with $|\mathcal{L}| = \ell$. Let us define the set of indices $\mathcal{V} = \{1, \ldots, n\}$, and for all $i \in \mathcal{V}$, define the *neighbours* $\mathcal{N}_i \subset \mathcal{V} \setminus \{i\}$, where $j \in \mathcal{N}_i$ implies $i \in \mathcal{N}_j$.

Definition 2.1.1. The set of joint random variables $\mathcal{X} = \{X_1, \ldots, X_n\}$ taking values $\mathbf{x} = \{x_1, \ldots, x_n\}$ with neighbourhood structure $\{\mathcal{N}_i \mid i \in \mathcal{V}\}$ constitutes a Markov Random Field (MRF) if the following conditions are satisfied

1. $P(\mathbf{x}) > 0$ for all $\mathbf{x} \in \mathcal{L}^n$.
2. $P(X_i \mid X_{\mathcal{N}_i}) = P(X_i \mid X_{\mathcal{V}\setminus\{i\}})$.

Here, $P(\mathbf{x})$ denotes the joint probability of the random variables $\mathcal{X}$ taking values $\mathbf{x}$.

The first condition is required to prove an important theorem about MRFs, which allows us to represent the MRF using an energy function (discussed subsequently). The second condition enforces the *Markov property*, which states that the conditional probability distribution of a given random variable X_i depends only on the values of its neighbours.

An MRF is often called an *undirected graphical model* and represented by an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ such that

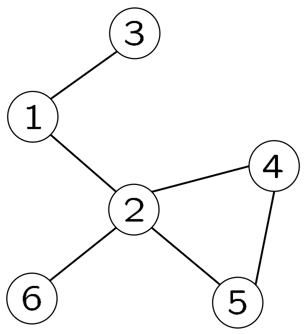


Figure 2.1: *Example of an MRF represented by an undirected graph. Here, we used $i \in \mathcal{V}$ to denote the nodes. Due to their one-to-one correspondence with the random variable X_i, some authors denote them with $X_i\,; i \in \mathcal{V}$. If two nodes are neighbours in the MRF, then they are connected by an edge in this graph. Here the edges correspond to pairwise cliques. The largest clique in this graph is a 3-clique: $(2, 4, 5)$.*

1. The vertices $\mathcal{V}$ (or nodes) are in one-to-one correspondence with the random variables $\mathcal{X}$, denoted using their indices.
2. There is an undirected edge between i and j if and only if $i \in \mathcal{N}_j$.

In this MRF graph, we denote the number of vertices and number of edges with n and m, *i.e.*, $|\mathcal{V}| = n$ and $|\mathcal{E}| = m$. An example of an MRF graph is shown in Figure 2.1. A *clique* in a graph is a subset of its vertices such that every two vertices in the subset are connected by an edge.

We will now state the Hammersley-Clifford theorem [Besag, 1974; Grimmett, 1973] which allows us to represent the MRF using an energy function.

Theorem 2.1.1. *Consider a set of random variables $\mathcal{X} = \{X_1, \ldots, X_n\}$ associated with the vertices of a graph, and let $\mathbf{x} = \{x_1, \ldots, x_n\}$ be a corresponding set of values. For each clique C in the graph, let $F_C(\mathbf{x})$ be a positive function depending only on the values x_i for $i \in C$. Then the joint probability distribution $P(\mathbf{x})$ defines an MRF on the graph if and only if*

$$P(\mathbf{x}) = \frac{1}{Z} \prod_{C \in \mathcal{C}} F_C(\mathbf{x})\,, \tag{2.1}$$

where $\mathcal{C}$ is the set of all cliques and Z is a normalizing constant chosen such that $\sum_{\mathbf{x} \in \mathcal{L}^n} P(\mathbf{x}) = 1$.

Proof. This is a well known theorem. See [Besag, 1974; Grimmett, 1973] for the proof. □

Since F_C is positive for all $C \in \mathcal{C}$, one may define $\theta_C(\mathbf{x}) = -\log(F_C(\mathbf{x}))$, referred to as *clique potentials*, then Eq. (2.1) can be written in the following form

$$P(\mathbf{x}) = \frac{1}{Z} \exp\left(- \sum_{C \in \mathcal{C}} \theta_C(\mathbf{x}) \right) . \tag{2.2}$$

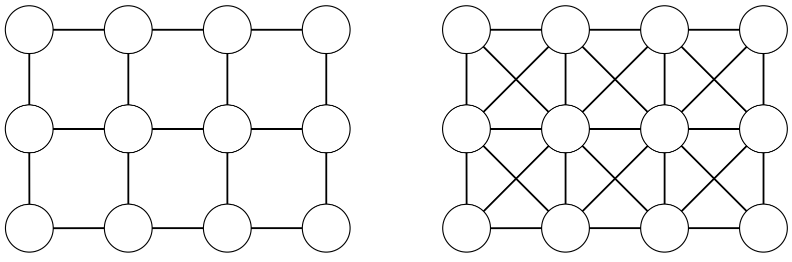

(a) 4-connected (b) 8-connected

Figure 2.2: *Sparse neighbourhood structures often used in computer vision,* ***left:*** *4-connected neighbourhood, often referred to as the grid structure; and* ***right:*** *8-connected neighbourhood structure. Here the circles denote the MRF nodes (usually pixels in the image).*

The function $E(\mathbf{x}) = \sum_{C \in \mathcal{C}} \theta_C(\mathbf{x})$ is often referred to as the *Gibbs energy function* or simply the *energy function* of the MRF. A probability distribution $P(\mathbf{x})$ that is written in the form (2.1) or (2.2) is called a *Gibbs distribution*. Here, the normalizing constant Z is often referred to as the *partition function* and it has the following form

$$Z = \sum_{\mathbf{x} \in \mathcal{L}^n} \exp\left(-E(\mathbf{x})\right) \ . \tag{2.3}$$

In this thesis, we are particularly interested in *pairwise* MRFs, *i.e.*, MRFs with maximum clique size two. Although this seems restrictive, it can be shown that all MRFs with arbitrary size cliques have an equivalent pairwise MRF formulation [Yedidia et al., 2003]. The energy associated with a pairwise MRF takes the following form

$$E(\mathbf{x}) = \sum_{i \in \mathcal{V}} \theta_i(x_i) + \sum_{(i,j) \in \mathcal{E}} \theta_{ij}(x_i, x_j) \ , \tag{2.4}$$

where θ_i and θ_{ij} denote the *unary potentials* (*i.e.*, data costs) and *pairwise potentials* (*i.e.*, interaction costs), respectively. Here, $\mathcal{V}$ is the set of vertices, *e.g.*, corresponding to pixels or *superpixels*[1] in an image, and $\mathcal{E}$ is the set of edges, *e.g.*, encoding a 4-connected or 8-connected grid over the image pixels. See Figure 2.2.

Example 2.1.1. Recall the image inpainting problem discussed in Section 1.1.3. Let us formulate the corresponding energy function in the form of Eq. (2.4). In this case the label set is the set of image intensities. Assuming a gray scale image,

$$\mathcal{L} = \{0, 1, \ldots, 255\} \ . \tag{2.5}$$

The graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ corresponds to the image grid, *i.e.*, $\mathcal{V}$ is the set of pixels and $\mathcal{E}$ has the 4-connected neighbourhood structure (see Figure 2.2). The unary potentials are defined such that the restored intensity at any pixel should be close to

[1]A *superpixel* is a polygonal part of a digital image, larger than a normal pixel, resulting from a process of its partitioning into multiple segments. Usually the pixels in a superpixel have the same (or very similar) intensities.

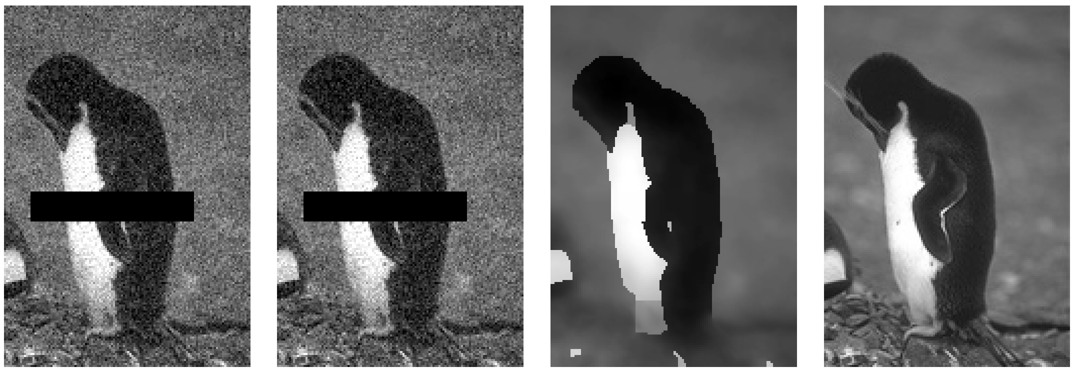

(a) Input (b) Unary only (c) With pairwise (d) Ground truth

Figure 2.3: *An example of image inpainting with (**c**) and without (**b**) pairwise potentials. Note that the pairwise potentials ensure a smooth restoration.*

the observed intensity. An example is [Szeliski et al., 2008],

$$\theta_i(x_i) = |I(i) - x_i|^2 \ , \tag{2.6}$$

where $I(i)$ denotes the observed intensity at pixel i. Now, for this choice of unary term, if there is no pairwise term, then the restored image would be the same as the input (*i.e.*, noisy). Usually the pairwise term is used as a smoothing cost which has the following form [Szeliski et al., 2008],

$$\theta_{ij}(x_i, x_j) = \gamma_{ij}\, \theta(|x_i - x_j|) \ , \tag{2.7}$$

where $\gamma_{ij} \geq 0$ is a weight depending on the pixels i and j and $\theta(\cdot)$ is a non-negative non-decreasing function. The function $\theta(\cdot)$ is often referred to as the *regularizer* (or prior). A regularizer commonly used for this problem is,

$$\theta(|x_i - x_j|) = \min(|x_i - x_j|^2, \kappa) \ , \tag{2.8}$$

where κ is the maximum smoothness penalty. An image inpainting example with the energy formulation discussed above is shown in Figure 2.3.

2.1.1 Conditional Random Fields

Usually in computer vision applications we would like to use energy parameters that are data dependent. A probabilistic model which allows us to use data dependent energy parameters is the Conditional Random Field (CRF) [Lafferty et al., 2001]. A CRF can be defined as follows.

Definition 2.1.2. Consider two finite probability spaces: *data* $\mathcal{D}$, and *labellings* Γ (*e.g.*, Γ may be the set of labellings $\mathcal{L}^n$). A CRF is a joint probability distribution defined on $\Gamma \times \mathcal{D}$, such that, for any event $\mathbf{d} \in \mathcal{D}$, the conditional probability distribution

$P(\mathbf{x} \mid \mathbf{d})$ is an MRF (as a distribution over $\mathbf{x} \in \Gamma$), with the same neighbourhood structure for all $\mathbf{d} \in \mathcal{D}$.

From an optimization point-of-view, the data $\mathbf{d}$ is given, and so the problem of *optimizing* the CRF means the task of optimizing the conditional probability distribution $P(\mathbf{x} \mid \mathbf{d})$, which is an MRF. In fact, apart from defining the probability distribution $P(\mathbf{x} \mid \mathbf{d})$, the data does not play any role in the optimization process. Therefore, from the optimization perspective both MRF and CRF models are identical and hence in this thesis, the term CRF refers to the corresponding MRF.

2.1.2 Optimizing an MRF

In general, optimizing an MRF may imply several meanings, such as finding the most probable assignment, computing the partition function or finding the marginal probabilities. In this thesis, we are interested in finding the most probable assignment or labelling of the MRF, and the term "optimizing an MRF" uniquely refers to this.

In most computer vision applications, we want to find an assignment $\mathbf{x}$ that maximizes the probability. This is often referred to as the Maximum A Posteriori (MAP) estimation in the literature. Formally, it requires us to find the labelling $\mathbf{x}^*$ such that

$$\begin{aligned} \mathbf{x}^* &= \operatorname*{argmax}_{\mathbf{x}\in\mathcal{L}^n} P(\mathbf{x}) \;, \\ &= \operatorname*{argmax}_{\mathbf{x}\in\mathcal{L}^n} \frac{1}{Z} \exp\left(-E(\mathbf{x})\right) \;, \\ &= \operatorname*{argmin}_{\mathbf{x}\in\mathcal{L}^n} E(\mathbf{x}) \;. \end{aligned} \tag{2.9}$$

The above simplification is due to the fact that the partition function is constant, and the exponential function is monotone. Now it becomes clear that, optimizing an MRF is equivalent to an *energy minimization* problem. Note that the minimization is over all possible assignments of $\mathbf{x}$ and the number of assignments is exponentially large. Therefore, this is an intractable (NP-hard) problem in general.

2.1.3 MRF Optimization Algorithms

MRF optimization plays an important role in many computer vision applications including stereo, inpainting and semantic segmentation [Szeliski et al., 2008; Kappes et al., 2015]. In fact, this problem is closely related to several interesting combinatorial optimization problems, such as min-cut [Goemans and Williamson, 1995], multi-way cut [Dahlhaus et al., 1992] and metric labelling [Boykov et al., 2001; Kleinberg and Tardos, 2002]. On the other hand, this discrete optimization can be relaxed to a continuous optimization problem [Chekuri et al., 2004; Kumar et al., 2009], which would enable us to leverage the well studied convex optimization literature [Bertsekas, 1999; Boyd and Vandenberghe, 2009]. Furthermore, on the probability domain, MRF optimization (MAP estimation) has an information theoretic interpretation [Wainwright et al., 2008] and has wide applicability in areas beyond computer vision and machine learning.

Even though, optimizing an MRF is NP-hard in general [Boykov et al., 2001], there are special cases (*i.e.*, restricted classes of pairwise potentials and restricted graph structures) that can be solved optimally in polynomial time. In particular, if a binary MRF energy is *submodular* (defined later in Definition 2.2.2), it is equivalent to a min-cut (sometimes called graph-cut) problem [Kolmogorov and Zabin, 2004] and there are many polynomial time algorithms to solve it [Ford and Fulkerson, 1962; Boykov and Kolmogorov, 2004]. This *submodularity* condition was later extended to multi-label MRFs [Schlesinger and Flach, 2006], hence such MRFs are solvable using a graph-cut algorithm [Ishikawa, 2003]. On the other hand, if the MRF is defined on a *tree* (an acyclic graph), then it can be solved using a message passing algorithm [Pearl, 1988].

Optimal algorithms are available only for certain restricted classes of MRFs and for general MRFs one has to settle with an approximate algorithm. The basic idea of designing an approximate algorithm is to utilize the optimal algorithms in some framework, such as in a move-making algorithm [Boykov et al., 2001] or a decomposition-based algorithm [Wainwright et al., 2005; Komodakis et al., 2011] in the continuous domain. In the former, the idea is to reduce the original MRF into a sequence of binary (or multi-label) submodular problems and solve them using a graph-cut algorithm. In the latter, the MRF is decomposed into optimally solvable subproblems (*e.g.*, tree MRFs) and each subproblem is tackled independently. In most cases, the approximate algorithms provide a bound on their solution relative to the optimal solution.

The remainder of this chapter is dedicated to the state-of-the-art MRF optimization algorithms (both exact and approximate) and their basic theories are introduced in the relevant sections. In the next section, we discuss binary MRFs and their graph solvability. Next, we turn to the multi-label MRFs. In this section, we first review a graph-cut algorithm for *multi-label submodular* MRFs, and then formalize move-making algorithms and discuss useful special cases. Later, we consider the linear programming relaxation of the discrete problem. In this case, we will discuss a tree-based decomposition algorithm where each subproblem (tree MRF) is optimally minimized using the message passing algorithm. Finally, we briefly review other continuous relaxations based methods including the mean-field algorithm.

2.2 Binary MRF Optimization

Let us recall the energy function associated with a pairwise MRF (2.4),

$$E(\mathbf{x}) = \sum_{i \in \mathcal{V}} \theta_i(x_i) + \sum_{(i,j) \in \mathcal{E}} \theta_{ij}(x_i, x_j) \;, \tag{2.10}$$

where $x_i \in \mathcal{L}$ for all $i \in \mathcal{V}$. For a binary MRF, the label set $\mathcal{L} = \{0, 1\}$. Examples of binary MRFs include interactive binary image segmentation [Boykov and Jolly, 2001] and foreground-background segmentation. More than their applications, studying binary MRFs is important to understand the basic concepts of MRF optimization and to design move-making style approximate algorithms.

The idea is to formulate the minimization of the above energy function (2.10) as a min-cut problem and understand the required properties for it to be *graph solvable*, meaning it can be minimized optimally, using a graph-cut (or max-flow) algorithm in polynomial time [Ford and Fulkerson, 1962].

To this end, let us first introduce the concepts on pseuo-boolean optimization and submodular functions.

2.2.1 Pseudo-Boolean Functions

Definition 2.2.1. Define $\mathcal{B} = \{0, 1\}$. A *pseudo-boolean function* is a mapping $f : \mathcal{B}^n \to \mathbb{R}$, where $\mathbb{R}$ is the set of real numbers. A *quadratic pseudo-boolean function* is a pseudo-boolean function with maximum degree two.

Now, we will write the energy function (2.10) as a quadratic pseudo-boolean function. Note that the variables $x_i \in \mathcal{B}$ for all $i \in \mathcal{L}$. We denote the complement of x_i with $\bar{x}_i$, *i.e.*, $\bar{x}_i = 1 - x_i$. This means if $x_i = 1$, then $\bar{x}_i = 0$ and vice versa. With this notation, we can write Eq. (2.10) as

$$E(\mathbf{x}) = \sum_{i \in \mathcal{V}} (\theta_{i:0}\, \bar{x}_i + \theta_{i:1}\, x_i) + \sum_{(i,j) \in \mathcal{E}} \left(\theta_{ij:00}\, \bar{x}_i\, \bar{x}_j + \theta_{ij:01}\, \bar{x}_i\, x_j + \theta_{ij:10}\, x_i\, \bar{x}_j + \theta_{ij:11}\, x_i\, x_j\right) . \tag{2.11}$$

Here, we use the shorthand $\theta_{i:\lambda} = \theta_i(\lambda)$ and $\theta_{ij:\lambda\mu} = \theta_{ij}(\lambda, \mu)$. It is clear that Eq. (2.11) is a quadratic pseudo-boolean function.

In general minimizing a quadratic pseudo-boolean function is an NP-hard problem [Boros and Hammer, 2002], meaning that there does not exist a polynomial time algorithm for finding the minimum (unless $P = NP$). However, we will now see a useful special case, where an efficient polynomial time algorithm can be used to find the minimum.

2.2.2 Submodular Functions

The submodularity condition is usually defined on set functions [Fujishige, 2005]. However, due to the one-to-one correspondence between set functions and pseudo-boolean functions, one can define it as follows.

Definition 2.2.2. A pseudo-boolean function $f : \mathcal{B}^n \to \mathbb{R}$ is *submodular* if

$$f(\mathbf{x}) + f(\mathbf{y}) \geq f(\mathbf{x} \vee \mathbf{y}) + f(\mathbf{x} \wedge \mathbf{y}) , \tag{2.12}$$

for all $\mathbf{x}, \mathbf{y} \in \mathcal{B}^n$. Here $\vee$ and $\wedge$ are componentwise logical OR and AND operations. For a quadratic pseudo-boolean function with $n = 2$, the submodularity definition reduces to

$$f(0, 1) + f(1, 0) \geq f(0, 0) + f(1, 1) . \tag{2.13}$$

Now, it is easy to see that the energy function (2.11) is submodular if the pairwise terms satisfy the following inequality

$$\theta_{ij:01} + \theta_{ij:10} \geq \theta_{ij:00} + \theta_{ij:11} \,, \tag{2.14}$$

for all $(i, j) \in \mathcal{E}$. Note that there is no condition on the unary potentials and hence, they can be arbitrary.

Example 2.2.1. Consider a binary segmentation problem where foreground is denoted with label 1 and background with label 0. Here the unary potentials are based on the image intensities and possibly learned from the dataset. To have a smooth segmentation, the pairwise potentials usually have the following form (see Example 2.1.1),

$$\theta_{ij}(x_i, x_j) = \gamma_{ij}\, \theta(|x_i - x_j|) \,, \tag{2.15}$$

where $\gamma_{ij} \geq 0$ is a weight depending on the pixels i and j and $\theta(\cdot)$ is a non-negative non-decreasing function. For an MRF with binary labels,

$$\begin{aligned} \theta_{ij:00} &= \theta_{ij:11} = \gamma_{ij}\, \theta(0) \,, \\ \theta_{ij:01} &= \theta_{ij:10} = \gamma_{ij}\, \theta(1) \,, \end{aligned} \tag{2.16}$$

where $\theta(0) \leq \theta(1)$. Therefore, from Eq. (2.14), such an MRF is submodular. Furthermore, these pairwise potentials are often referred to as *attractive potentials*.

The significance of submodularity is that, such an energy function is graph solvable. Let us now discuss the graph representability of quadratic pseudo-boolean functions and then turn to the max-flow algorithm.

2.2.3 Graph Representability

In this section, we discuss how a quadratic pseudo-boolean function can be represented by an *st*-graph.

Definition 2.2.3. An *st*-graph is a weighted directed graph $\hat{\mathcal{G}} = (\hat{\mathcal{V}} \cup \{0, 1\}, \hat{\mathcal{E}}^+, \boldsymbol{\phi})$. Here $\hat{\mathcal{V}}$ is the set of vertices or nodes and $\{0, 1\}$ are special nodes[2], called *source* and *terminal* respectively. Here $\hat{\mathcal{E}}^+$ is the set of directed edges and the undirected edges are denoted with $\hat{\mathcal{E}}$, where $(i, j) \in \hat{\mathcal{E}}$ means $(i, j) \in \hat{\mathcal{E}}^+$ and $(j, i) \in \hat{\mathcal{E}}^+$. The set of edge weights (or capacities) are denoted with $\boldsymbol{\phi}$, which has a real value ϕ_{ij} for each directed edge $(i, j) \in \hat{\mathcal{E}}^+$. Furthermore, we introduce notations $\hat{\mathcal{E}}_e$ and $\hat{\mathcal{E}}_\iota$, where $\hat{\mathcal{E}} = \hat{\mathcal{E}}_e \cup \hat{\mathcal{E}}_\iota$, to denote the *external* edges (*i.e.*, those connecting the source or the terminal) and the *internal* edges (*i.e.*, $\hat{\mathcal{E}}_\iota \subset \hat{\mathcal{V}} \times \hat{\mathcal{V}}$).

Definition 2.2.4. A *partition* or *cut* of an *st*-graph is a division of the vertices $\hat{\mathcal{V}}$ into two disjoint subsets $\hat{\mathcal{V}}_0$ and $\hat{\mathcal{V}}_1$, such that $0 \in \hat{\mathcal{V}}_0$ and $1 \in \hat{\mathcal{V}}_1$. The cost of the partition

[2]Some authors denote these nodes as s and t.

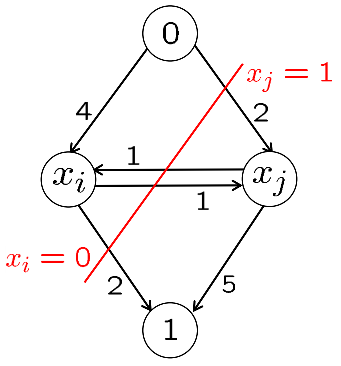


Figure 2.4: *Example of an st-graph. Here, the cut* $(\{x_i\},\{x_j\})$ *corresponds to the labelling* $\mathbf{x} = \{0,1\}$ *and the cost of the cut is* $C_{\hat{\mathcal{G}}}(\{x_i\},\{x_j\}) = 2+1+2 = 5$.

is the sum of weights on the edges passing from $\hat{\mathcal{V}}_0$ to $\hat{\mathcal{V}}_1$,

$$C_{\hat{\mathcal{G}}}(\hat{\mathcal{V}}_0,\hat{\mathcal{V}}_1) = \sum_{i\in\hat{\mathcal{V}}_0, j\in\hat{\mathcal{V}}_1} \phi_{ij} \;. \tag{2.17}$$

Example of an *st*-graph is shown in Figure 2.4. An *st*-graph represents the binary labelling in a different way. In fact, each labelling $\mathbf{x}$ is determined by the partition $(\hat{\mathcal{V}}_0,\hat{\mathcal{V}}_1)$, specifically, $x_i = 1$ if and only if $i \in \hat{\mathcal{V}}_1$. Then the function value for any labelling is equal to the cost of the corresponding partition,

$$f(\mathbf{x}) = C_{\hat{\mathcal{G}}}(\hat{\mathcal{V}}_0,\hat{\mathcal{V}}_1) \;. \tag{2.18}$$

Consider an edge $(i,j) \in \hat{\mathcal{E}}^+$ such that $i \in \hat{\mathcal{V}}_0$ and $j \in \hat{\mathcal{V}}_1$. This edge contributes to the cost of the partition for the labelling $x_i = 0$ and $x_j = 1$. Therefore, an edge (i,j) in the *st*-graph represents the term $\phi_{ij}\,\bar{x}_i\,x_j$. Similarly, the edges from the source (node 0) and the edges to the terminal (node 1) represent the linear terms. In particular, for all $i \in \hat{\mathcal{V}}$,

$$\begin{aligned} \phi_{0i}\,\bar{0}\,x_i &= \phi_{0i}\,x_i && \text{edge } 0 \to i \;, \\ \phi_{i1}\,\bar{x}_i\,1 &= \phi_{i1}\,\bar{x}_i && \text{edge } i \to 1 \;, \\ \phi_{01}\,\bar{0}\,1 &= \phi_{01} && \text{edge } 0 \to 1 \;. \end{aligned} \tag{2.19}$$

Note that the constant term is represented by an edge $0 \to 1$. Now, we can write the quadratic pseudo-boolean function represented by an *st*-graph in the following form,

$$f(\mathbf{x}) = \phi_{01} + \sum_{i\in\hat{\mathcal{V}}} (\phi_{0i}\,x_i + \phi_{i1}\,\bar{x}_i) + \sum_{(i,j)\in\hat{\mathcal{E}}_i^+} \phi_{ij}\,\bar{x}_i\,x_j \;. \tag{2.20}$$

From Eq. (2.18), minimizing $f(\mathbf{x})$ amounts to finding the minimum cut (shortly *min-cut*) solution in the *st*-graph. To obtain the min-cut solution in polynomial time, the edge capacities need to be non-negative [Ford and Fulkerson, 1962]. Note that

Eq. (2.20) can be reparametrized (using the identity $\bar{x}_i = 1 - x_i$) to make the coefficients of the linear terms non-negative. In particular, for all $i \in \hat{\mathcal{V}}$, $\phi_{0i}, \phi_{i1} \geq 0$ and ϕ_{01} may be negative. However, the constant term does not play any role in the minimization and hence it can be removed from the graph. For the quadratic terms, this reparametrization trick do not make any coefficients non-negative and they need to be non-negative to start with. Let us state it as a theorem.

Theorem 2.2.1. *Let $f(\mathbf{x})$ be a quadratic pseudo-boolean function represented by an st-graph as in Eq.* (2.20)*. If $\phi_{ij} \geq 0$ for all $(i,j) \in \hat{\mathcal{E}}_\iota^+$, then $\min_{\mathbf{x}} f(\mathbf{x})$ can be obtained in polynomial time.*

Proof. This is the standard result of graph solvability of quadratic pseudo-boolean functions. See [Ford and Fulkerson, 1962; Boros and Hammer, 2002]. □

2.2.3.1 Representing the Binary MRF Energy in an *st*-graph

We now discuss how one can represent the energy function (2.11) in an *st*-graph. To this end, the set of vertices $\mathcal{V}$ and the set of edges $\mathcal{E}$ of the MRF graph have one-to-one correspondence with the set $\hat{\mathcal{V}}$ and $\hat{\mathcal{E}}_\iota$ of the *st*-graph. Specifically,

$$\begin{aligned} \hat{\mathcal{V}} &= \mathcal{V}\ , \\ \hat{\mathcal{E}}_\iota &= \mathcal{E}\ . \end{aligned} \tag{2.21}$$

Additionally, there are external edges in the *st*-graph connecting the vertices $\hat{\mathcal{V}}$ with the source and the terminal nodes. Now, it remains to find the edge capacities that would represent the energy function exactly for all label assignments.

Let us now rewrite the energy function (2.11) in the form of Eq. (2.20). This can be done using the identity $\bar{x}_i = 1 - x_i$ for all $i \in \mathcal{V}$.

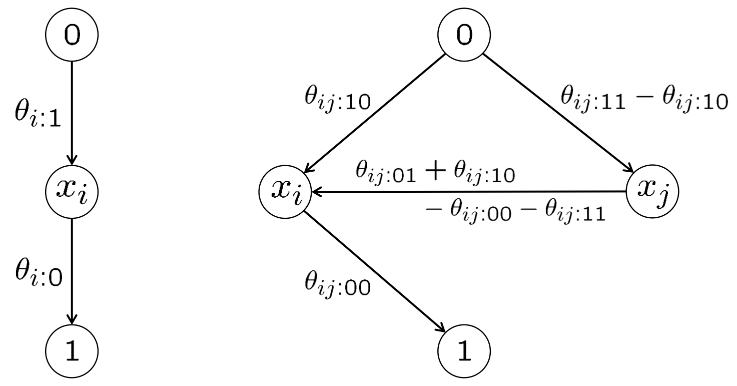
$$\begin{aligned} E(\mathbf{x}) &= \sum_{i\in\mathcal{V}} \left(\theta_{i:0}\,\bar{x}_i + \theta_{i:1}\,x_i\right) + \sum_{(i,j)\in\mathcal{E}} \left(\theta_{ij:00}\,\bar{x}_i\,\bar{x}_j + \theta_{ij:01}\,\bar{x}_i\,x_j + \theta_{ij:10}\,x_i\,\bar{x}_j + \theta_{ij:11}\,x_i\,x_j\right)\ , \\ &= \sum_{i\in\mathcal{V}} \left(\theta_{i:0}\,\bar{x}_i + \theta_{i:1}\,x_i\right) + \sum_{(i,j)\in\mathcal{E}} \left(\theta_{ij:00}\,\bar{x}_i + \theta_{ij:10}\,x_i + (\theta_{ij:11} - \theta_{ij:10})\,x_j\right) \\ &\qquad + \sum_{(i,j)\in\mathcal{E}} \left(\theta_{ij:01} + \theta_{ij:10} - \theta_{ij:00} - \theta_{ij:11}\right)\bar{x}_i\,x_j\ . \end{aligned} \tag{2.22}$$

The edge capacities $\boldsymbol{\phi}$ can be easily derived from the above equation. An *st*-graph representation of a unary term and a pairwise term are shown in Figure 2.5.

From Theorem 2.2.1, to minimize this energy function optimally, the coefficients of the quadratic terms need to be non-negative. Note that, for a submodular quadratic pseudo-boolean function,

$$\phi_{ij} = \theta_{ij:01} + \theta_{ij:10} - \theta_{ij:00} - \theta_{ij:11} \geq 0\ , \tag{2.23}$$

for all $(i,j) \in \mathcal{E}$. Therefore, if the binary MRF energy is submodular, then the mini-



0
$\theta_{i:1}$
x_i
$\theta_{i:0}$
1
0
$\theta_{ij:10}$
$\theta_{ij:11} - \theta_{ij:10}$
$\theta_{ij:01} + \theta_{ij:10}$
x_i
x_j
$- \theta_{ij:00} - \theta_{ij:11}$
$\theta_{ij:00}$
1

Figure 2.5: *The st-graph representation of a binary MRF energy function. Specifically, the unary term (**left**) for the node $i \in \mathcal{V}$ and the pairwise term (**right**) for the edge $(i,j) \in \mathcal{E}$ are represented in the st-graph, for the energy function* (2.22).

mum energy labelling can be obtained in polynomial time using a min-cut algorithm in the corresponding *st*-graph.

2.2.4 Min-Cut and Max-Flow

In this section, we establish the connection between the min-cut and max-flow problems. To this end, let us first define a flow in an *st*-graph as follows.

Definition 2.2.5. Given an *st*-graph $\hat{\mathcal{G}} = (\hat{\mathcal{V}} \cup \{0,1\}, \hat{\mathcal{E}}^+, \boldsymbol{\phi})$, a flow is a mapping $\boldsymbol{\psi} : \hat{\mathcal{E}}^+ \to \mathbb{R}$, denoted by ψ_{ij} for the edge $(i,j) \in \hat{\mathcal{E}}^+$, that satisfies the *anti-symmetry* condition $\psi_{ij} = -\psi_{ji}$ for all $(i,j) \in \hat{\mathcal{E}}$. A flow is called *conservative*[3] if the total flow into a node is zero for all nodes except for the source and the terminal, *i.e.*, for all $i \in \hat{\mathcal{V}}$,

$$\sum_{(j,i) \in \hat{\mathcal{E}}^+} \psi_{ji} = 0 \; . \tag{2.24}$$

Once a flow $\boldsymbol{\psi}$ is passed in an *st*-graph, the edge capacities $\boldsymbol{\phi}$ are updated as,

$$\boldsymbol{\phi}' = \boldsymbol{\phi} - \boldsymbol{\psi} \; , \tag{2.25}$$

where $\boldsymbol{\phi}'$ are called the *residual* capacities.

Definition 2.2.6. A flow $\boldsymbol{\psi}$ is called *permissible* if $\phi_{ij} - \psi_{ij} \geq 0$ for all $(i,j) \in \hat{\mathcal{E}}^+$.

Note that, if a permissible flow is passed in an *st*-graph with non-negative capacities, then the capacities remain non-negative.

As we have seen in the previous section, finding the minimum of a submodular quadratic pseudo-boolean function is the same as finding the min-cut solution of the corresponding *st*-graph. This can be achieved by finding the maximum flow (shortly

[3]A conservative flow is often referred to as a flow in the literature.

max-flow) from the source (0 node) to the terminal (1 node). In fact, min-cut and max-flow are dual LP problems. We now state the well-known min-cut/max-flow theorem [Ford and Fulkerson, 1962] below.

Theorem 2.2.2. *Consider a submodular quadratic pseudo-boolean function $f(\mathbf{x})$ and let $\boldsymbol{\psi}$ be a permissible conservative flow on the corresponding st-graph. Then,*

$$\min_{\mathbf{x}} f(\mathbf{x}) = \max_{\boldsymbol{\psi}} \sum_{(0,i)\in\hat{\mathcal{E}}^+} \psi_{0i} \ . \tag{2.26}$$

Proof. This is a result of the duality between the max-flow and min-cut problems and can be proven by computing the Lagrange dual of one of them. See [Ford and Fulkerson, 1962]. □

Max-Flow Algorithms. Broadly speaking, there are three different kinds of max-flow algorithms: those relying on finding augmenting paths [Ford and Fulkerson, 1962], the push-relabel approach [Goldberg and Tarjan, 1988] and the pseudo-flow techniques [Chandran and Hochbaum, 2009]. The polynomial time guarantee of max-flow was first proven in [Edmonds and Karp, 1972] by modifying the augmenting path algorithm, to always find the *shortest* augmenting path. There are numerous max-flow implementations available for general purpose as well as specific to computer vision applications. Among them, the specialized implementations are significantly faster in practice. In particular, the BK method [Boykov and Kolmogorov, 2004] is arguably the fastest implementation for 2D and sparse 3D graphs arising from computer vision applications. Recently, for dense problems, the EIBFS algorithm [Goldberg et al., 2015] was shown to outperform the BK method.

2.2.5 Non-Submodular MRFs

Non-submodular MRF energy functions can also be represented by an *st*-graph, but the graph would contain negative edges. Therefore the standard max-flow techniques cannot be applied. However, such an energy function can be approximated using a roof-dual technique [Boros and Hammer, 2002]. Such a technique is usually known as Quadratic Pseudo-Boolean Optimization (QPBO) and it can be used to obtain a partially optimal labelling. In other words, the optimal labels can be obtained only for a subset of nodes and the unlabelled nodes are usually assigned based on some heuristics. We refer the interested reader to [Boros and Hammer, 2002] for a detailed treatment on pseudo-boolean optimization.

2.3 Multi-Label MRF Optimization

In this section, we consider a more useful MRF, where the label set is no longer binary. Here, without loss of generality we assume, that the label set $\mathcal{L} = \{0, 1, \ldots, \ell - 1\}$. In contrast to binary MRFs, multi-label MRFs have a variety of applications in computer vision.

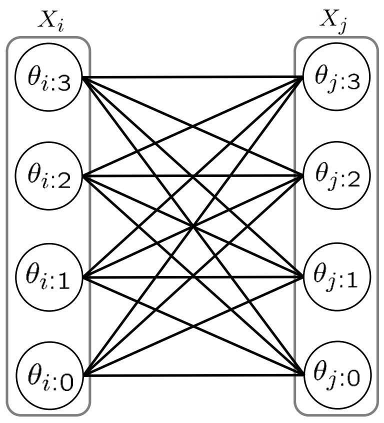


Figure 2.6: *Example of a multi-label graph. Here the nodes represent the unary potentials $\theta_{i:\lambda}$ and the edges represent the pairwise potentials $\theta_{ij:\lambda\mu}$.*

2.3.1 Multi-Label Graph

An alternative way of representing an MRF labelling $\mathbf{x} \in \mathcal{L}^n$ is by defining *indicator variables* $x_{i:\lambda} \in \{0,1\}$, where $x_{i:\lambda} = 1$ if and only if $x_i = \lambda$. For a given i, exactly one of $x_{i:\lambda}\,; \lambda \in \mathcal{L}$ can have value 1. In terms of the indicator variables, the energy function (2.4) may be written as

$$E(\mathbf{x}) = \sum_{i \in \mathcal{V}} \sum_{\lambda \in \mathcal{L}} \theta_{i:\lambda}\, x_{i:\lambda} + \sum_{(i,j) \in \mathcal{E}} \sum_{\lambda,\mu \in \mathcal{L}} \theta_{ij:\lambda\mu}\, x_{i:\lambda}\, x_{j:\mu}\ . \tag{2.27}$$

Here, we use the shorthand $\theta_{i:\lambda} = \theta_i(\lambda)$ and $\theta_{ij:\lambda\mu} = \theta_{ij}(\lambda,\mu)$. One may define a graph, called a *multi-label graph*, with nodes denoted by $X_{i:\lambda}\,;\, i \in \mathcal{V},\ \lambda \in \mathcal{L}$, as shown in Figure 2.6. This graph represents the energy function. Given a labelling $\mathbf{x}$, the value of the energy function is obtained by summing the weights on all nodes with $x_{i:\lambda} = 1$ (in other words $x_i = \lambda$) plus the weights $\theta_{ij:\lambda\mu}$ such that $x_{i:\lambda} = 1$ and $x_{j:\mu} = 1$. This multi-label graph representation of the energy function (2.27) will be useful to understand some properties of the energy as well as the algorithms used to minimize it.

2.3.2 Ishikawa Algorithm

Ishikawa [Ishikawa, 2003] introduced a method to solve (minimize the energy) of multi-label MRFs with convex edge terms. This method can be easily extended [Schlesinger and Flach, 2006] to energy functions satisfying a *multi-label submodularity* condition, analogous to the submodularity condition for MRFs with binary labels.

Multi-Label Submodularity. An energy function can be minimized optimally if it satisfies the multi-label submodularity condition [Schlesinger and Flach, 2006] de-

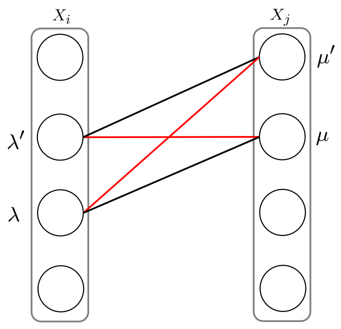


Figure 2.7: *The multi-label submodularity condition illustrated using the multi-label graph. The sum of red edge capacities must be greater than or equal to the sum of black edge capacities.*

fined below.

Definition 2.3.1. The energy function given in Eq. (2.27) is *multi-label submodular* if the pairwise potentials satisfy

$$\theta_{ij:\lambda'\mu} + \theta_{ij:\lambda\mu'} - \theta_{ij:\lambda\mu} - \theta_{ij:\lambda'\mu'} \geq 0\ , \tag{2.28}$$

for all $\lambda < \lambda'$ and $\mu < \mu'$. This condition assumes an ordered label set.

Note that the multi-label submodularity condition (2.28) can be equivalently stated as,

$$\theta_{ij:\lambda+1\mu} + \theta_{ij:\lambda\mu+1} - \theta_{ij:\lambda\mu} - \theta_{ij:\lambda+1\mu+1} \geq 0\ , \tag{2.29}$$

for $\lambda, \mu \in \{0, \dots \ell - 2\}$. This can be easily verified using induction. Figure 2.7 shows this condition graphically, in the multi-label graph.

Example 2.3.1. Consider pairwise potentials of the form,

$$\theta_{ij:\lambda\mu} = \gamma_{ij}\, \theta(|\lambda - \mu|)\ , \tag{2.30}$$

where $\gamma_{ij} \geq 0$ and $\theta(\cdot)$ is a convex function[4]. Substituting this in Eq. (2.29) and neglecting γ_{ij},

$$\theta(|\lambda - \mu + 1|) + \theta(|\lambda - \mu - 1|) - 2\,\theta(|\lambda - \mu|) \geq 0\ . \tag{2.31}$$

This is true for a convex function $\theta(\cdot)$. These pairwise potentials are referred to as *convex priors* and they constitute a multi-label submodular energy function. See Figure 2.8 for some examples of convex priors used in computer vision.

Ishikawa [Ishikawa, 2003] introduced a different way of representing the multi-label energy function. The basic idea behind the Ishikawa construction is to encode the label $X_i = x_i$ of a vertex $i \in \mathcal{V}$ using binary-valued random variables $U_{i:\lambda}$, one

[4]A function $f : \mathcal{X} \to \mathbb{R}$ is *convex* if for all $x, y \in \mathcal{X}$ and $t \in [0,1]$, $f(t\,x + (1-t)\,y) \leq t\,f(x) + (1-t)\,f(y)$ [Boyd and Vandenberghe, 2009]. Furthermore, if f is convex, then $-f$ is concave.

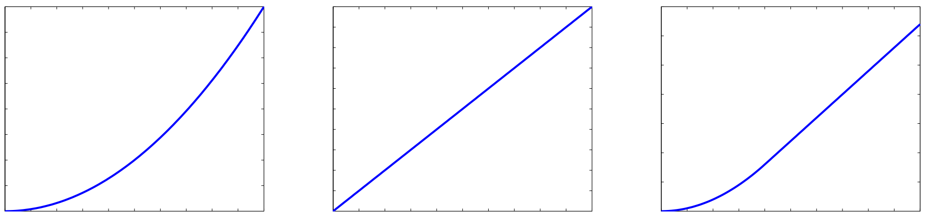

(a) Quadratic (b) Linear (c) Huber

Figure 2.8: *Examples of convex priors:* ***a)*** $\theta(|\delta|)$ *is a quadratic function;* ***b)*** *a linear function; and* ***c)*** *a Huber function (Eq.* (2.43)*).*

for each label $\lambda \in \mathcal{L}$. In particular, the encoding is defined as $u_{i:\lambda} = 1$ if and only if $x_i \geq \lambda$, and 0 otherwise. The Ishikawa graph is then an *st*-graph (see Definition 2.2.3) $\hat{\mathcal{G}} = (\hat{\mathcal{V}} \cup \{0, 1\}, \hat{\mathcal{E}}^+, \boldsymbol{\phi})$, where the set of nodes and the set of edges are defined as follows,

$$\begin{aligned}
\hat{\mathcal{V}} &= \{U_{i:\lambda} \mid i \in \mathcal{V}, \lambda \in \{1, \cdots, \ell - 1\}\} , \\
\hat{\mathcal{E}}^+ &= \hat{\mathcal{E}}^+_v \cup \hat{\mathcal{E}}^+_c , \\
\hat{\mathcal{E}}^+_v &= \{(U_{i:\lambda}, U_{i:\lambda\pm 1}) \mid i \in \mathcal{V}, \lambda \in \{1, \cdots, \ell - 1\}\} , \\
\hat{\mathcal{E}}^+_c &= \{(U_{i:\lambda}, U_{j:\mu}), (U_{j:\mu}, U_{i:\lambda}) \mid (i, j) \in \mathcal{E}, U_{i:\lambda}, U_{j:\mu} \in \hat{\mathcal{V}}\} ,
\end{aligned} \tag{2.32}$$

where $\hat{\mathcal{E}}^+_v$ is the set of vertical edges and $\hat{\mathcal{E}}^+_c$ is the set of cross edges. Note that the directed edges are denoted with $\square^+$. We denote the Ishikawa edges by $e_{ij:\lambda\mu} \in \hat{\mathcal{E}}^+$ and their capacities by $\phi_{ij:\lambda\mu}$. We also denote by $e_{i:\lambda}$ the downward edge $(U_{i:\lambda+1}, U_{i:\lambda})$. An example of an Ishikawa graph is shown in Figure 2.9.

From the definition of $U_{i:\lambda}$ above, we find the basic relation

$$\begin{aligned}
x_{i:\lambda} &= u_{i:\lambda} - u_{i:\lambda+1} , \\
&= \bar{u}_{i:\lambda+1} - \bar{u}_{i:\lambda} , \\
&= \bar{u}_{i:\lambda+1}\, u_{i:\lambda} ,
\end{aligned} \tag{2.33}$$

which also entails the relation $u_{i:\lambda} \geq u_{i:\lambda+1}$. In addition, since $x_i \geq 0$, it follows that $u_{i:0} = 1$. Thus, node $U_{i:0}$ is identified with the vertex 1. Furthermore, since $x_i < \ell$, we may identify $u_{i:\ell} = 0$. In other words, nodes $U_{i:0}$ may be identified with node 1 and $U_{i:\ell}$ with node 0, which is useful in interpreting Eq. (2.33) for all $\lambda \in \mathcal{L}$. Consequently, only the nodes $U_{i:\lambda}$ for $\lambda \in \{1, \cdots, \ell - 1\}$ are included in the set $\hat{\mathcal{V}}$.

In an *st*-graph, a labeling $\mathbf{x}$ is represented by a *cut* (see Definition 2.2.4) in the graph. Therefore, in the Ishikawa graph, if the downward edge $e_{i:\lambda}$ is in the *cut*, then vertex i takes label λ. In MRF energy minimization, each vertex i takes exactly one label x_i, which means that exactly one edge $e_{i:\lambda}$ must be in the min-cut of the Ishikawa graph. This is ensured by having infinite capacity for each upward edge $e_{ii:\lambda\lambda+1}$, *i.e.*, $\phi_{ii:\lambda\lambda+1} = \infty$ for all $i \in \mathcal{V}$ and $\lambda \in \mathcal{L}$. Since the energy of a labelling is represented by a cut in the Ishikawa graph, the capacities $\boldsymbol{\phi}$ and the energy parameters $\boldsymbol{\theta}$ are related

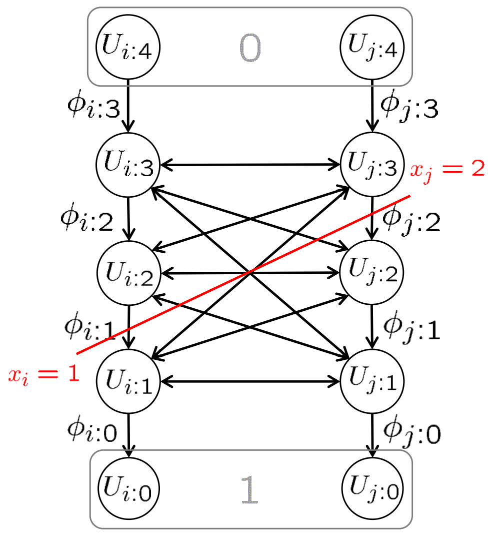


Figure 2.9: *Example of an Ishikawa graph. The graph incorporates edges with infinite capacity from* $U_{i:\lambda}$ *to* $U_{i:\lambda+1}$*, not shown in the graph. Here the cut corresponds to the labeling* $\mathbf{x} = \{1, 2\}$ *where the label set* $\mathcal{L} = \{0, 1, 2, 3\}$*.*

as follows,

$$\begin{aligned} \theta_{i:\lambda} &= \phi_{ii:\lambda+1\lambda} = \phi_{i:\lambda} \,, \\ \theta_{ij:\lambda\mu} &= \sum_{\substack{\lambda'>\lambda \\ \mu'\leq\mu}} \phi_{ij:\lambda'\mu'} + \sum_{\substack{\lambda'\leq\lambda \\ \mu'>\mu}} \phi_{ji:\mu'\lambda'} \,. \end{aligned} \tag{2.34}$$

Now, we need to determine the capacities $\boldsymbol{\phi}$ from the energy parameters $\boldsymbol{\theta}$. To this end, one can work out how an energy function of the form (2.27) may be written in terms of the variables $u_{i:\lambda}$, and hence encoded in an Ishikawa graph. Given the energy function (2.27), one wishes to write it in the form representable by an *st*-graph (see Section 2.2.3),

$$E(\mathbf{x}) = \phi_{01} + \sum_{\substack{i\in\mathcal{V} \\ \lambda\in\mathcal{L}}} \phi_{i:\lambda}\, \bar{u}_{i:\lambda+1}\, u_{i:\lambda} + \sum_{\substack{(i,j)\in\mathcal{E} \\ \lambda,\mu\in\mathcal{L}}} \phi_{ij:\lambda\mu}\, \bar{u}_{i:\lambda}\, u_{j:\mu} \,. \tag{2.35}$$

To this end, we substitute $x_{i:\lambda} = \bar{u}_{i:\lambda+1}\, u_{i:\lambda}$ and

$$\begin{aligned} x_{i:\lambda}\, x_{j:\mu} &= (\bar{u}_{i:\lambda+1} - \bar{u}_{i:\lambda})\,(u_{j:\mu} - u_{j:\mu+1}) \,, \\ &= \bar{u}_{i:\lambda+1}\, u_{j:\mu} + \bar{u}_{i:\lambda}\, u_{j:\mu+1} - \bar{u}_{i:\lambda}\, u_{j:\mu} - \bar{u}_{i:\lambda+1}\, u_{j:\mu+1} \,, \end{aligned} \tag{2.36}$$

which applies for all λ, $\mu \in \mathcal{L}$ provided $u_{i:0}$ is interpreted as 1 and $u_{i:\ell}$ as 0. This leads to the expression,

$$E(\mathbf{x}) = \sum_{\substack{i\in\mathcal{V}\\ \lambda\in\mathcal{L}}} \theta_{i:\lambda}\, \bar{u}_{i:\lambda+1}\, u_{i:\lambda} + \sum_{\substack{(i,j)\in\mathcal{E}\\ \lambda,\mu\in\mathcal{L}}} \theta_{ij:\lambda\mu}\, (\bar{u}_{i:\lambda+1}\, u_{j:\mu} + \bar{u}_{i:\lambda}\, u_{j:\mu+1} - \bar{u}_{i:\lambda}\, u_{j:\mu} - \bar{u}_{i:\lambda+1}\, u_{j:\mu+1})\ , \tag{2.37}$$

which is of the form (2.35) with

$$\begin{aligned} \phi_{i:\lambda} &= \theta_{i:\lambda}\ , \\ \phi_{ij:\lambda\mu} &= \theta_{ij:\lambda\mu-1} + \theta_{ij:\lambda-1\mu} - \theta_{ij:\lambda\mu} - \theta_{ij:\lambda-1\mu-1}\ . \end{aligned} \tag{2.38}$$

In this formula, the value of $u_{i:\ell}$ is to be interpreted as 0, which means that any terms involving $u_{i:\ell}$ is zero. In addition, $u_{j:0} = 1$, so similarly any terms involving $\bar{u}_{j:0}$ is zero. Furthermore, any term involving $u_{j:-1}$ is not defined and hence will be omitted. By omitting the undefined or zero terms, for all $\lambda, \mu \in \{1, \ldots, \ell - 1\}$,

$$\phi_{ij:\lambda 0} = \theta_{ij:\lambda-10} - \theta_{ij:\lambda 0}\ , \tag{2.39a}$$

$$\phi_{ij:\ell\mu} = \theta_{ij:\ell-1\mu} - \theta_{ij:\ell-1\mu-1}\ , \tag{2.39b}$$

$$\phi_{ij:\ell 0} = \theta_{ij:\ell-10}\ , \tag{2.39c}$$

$$\phi_{ij:\lambda\mu} = \theta_{ij:\lambda\mu-1} + \theta_{ij:\lambda-1\mu} - \theta_{ij:\lambda\mu} - \theta_{ij:\lambda-1\mu-1}\ . \tag{2.39d}$$

Note that the equations (2.39a) and (2.39b) correspond to linear terms and Eq. (2.39c) corresponds to the constant term, since,

$$\begin{aligned} \phi_{ij:\lambda 0}\, \bar{u}_{i:\lambda}\, u_{j:0} &= \phi_{ij:\lambda 0}\, \bar{u}_{i:\lambda}\, 1\ , \\ \phi_{ij:\ell\mu}\, \bar{u}_{i:\ell}\, u_{j:\mu} &= \phi_{ij:\ell\mu}\, \bar{0}\, u_{j:\mu}\ , \\ \phi_{ij:\ell 0}\, \bar{u}_{i:\ell}\, u_{j:0} &= \phi_{ij:\ell 0}\, \bar{0}\, 1\ . \end{aligned} \tag{2.40}$$

Therefore, the only quadratic terms are of the form Eq. (2.39d), and they are non-negative for a multi-label submodular energy function. All the linear terms can be made non-negative by reparametrization (see Section. 2.2.3.1). At this point, we have shown that a multi-label submodular energy function can be encoded in an *st*-graph (called an Ishikawa graph) where the edge capacities are all non-negative. Hence, as discussed in Section 2.2.4, the min-cut solution can be obtained in polynomial time using the max-flow algorithm.

Note that, the Ishikawa graph has $|\hat{\mathcal{V}}| = \mathcal{O}(n\,\ell)$ nodes and $|\hat{\mathcal{E}}| = \mathcal{O}(n\,\ell + m\,\ell^2)$ edges, where $|\mathcal{V}| = n$ and $|\mathcal{E}| = m$. For a sparse graph, *i.e.*, $m = \mathcal{O}(n)$, the space complexity of the Ishikawa graph is $\mathcal{O}(n\,\ell^2)$.

Convex Priors. In this section, we are interested in an MRF with convex priors (see Example 2.3.1), *i.e.*,

$$\theta_{ij:\lambda\mu} = \gamma_{ij}\, \theta(|\lambda - \mu|)\ , \tag{2.41}$$

where $\gamma_{ij} \geq 0$ and $\theta(\cdot)$ is a convex function with $\theta(0) = 0$. This is not a restriction as $\theta(\cdot)$ is non-decreasing and a constant can be subtracted from the pairwise potentials to make $\theta(0) = 0$. In this case, the Ishikawa edge capacities take the following form [Ishikawa, 2003],

$$\phi_{i:\lambda} = \theta_{i:\lambda} \; , \tag{2.42}$$
$$\phi_{ij:\lambda\mu} = \begin{cases} 0 & \text{if } \lambda < \mu \\ \frac{\gamma_{ij}}{2}\,\theta''(|\lambda-\mu|) & \text{if } \lambda = \mu \\ \gamma_{ij}\,\theta''(|\lambda-\mu|) & \text{if } \lambda > \mu \end{cases} \; ,$$

where $\theta''(|\delta|) = \theta(|\delta+1|) + \theta(|\delta-1|) - 2\,\theta(|\delta|)$, which is non-negative for a convex function $\theta(\cdot)$. Note that, if $\theta(\cdot)$ is linear, then $\theta''(\delta)$ is zero except for $\delta = 0$. Hence, the space complexity of the Ishikawa graph reduces to $\mathcal{O}(n\,\ell)$. Furthermore, in case of a Huber function [Huber, 1964], *i.e.*,

$$\theta(|\delta|) = \begin{cases} \frac{1}{2}|\delta|^2 & \text{if } |\delta| \le \kappa \\ \kappa\left(|\delta| - \frac{1}{2}\kappa\right) & \text{otherwise} \; , \end{cases} \tag{2.43}$$

where κ is the Huber value, the space complexity is $\mathcal{O}(n\,\kappa\,\ell)$. However, in general, the space complexity of the Ishikawa graph is $\mathcal{O}(n\,\ell^2)$.

This quadratic complexity in the number of labels makes the Ishikawa algorithm inapplicable to realistic problems with many pixels and labels, due to excessive memory requirement. In Chapter 3, we introduce an algorithm, named Memory Efficient Max-Flow (MEMF), that has linear space complexity in ℓ. Consequently, our MEMF algorithm makes it possible to optimally solve multi-label submodular MRFs involving large numbers of pixels and labels on a standard computer.

2.3.3 Move-Making Algorithms

So far, we have discussed submodular MRF energy functions and exact algorithms to minimize them. In this section, we discuss a family of approximate algorithms that make use of exact optimization techniques to efficiently minimize a more general class of MRF energies.

We first characterize the family of move-making algorithms and then turn to useful special cases. A move-making algorithm iteratively updates the current labelling of the multi-label MRF, by minimizing a *surrogate energy* (usually a binary MRF energy). An iteration is called a *move*, and each move decreases the original multi-label energy function.

Let us recall the energy function associated with a multi-label pairwise MRF (2.4),

$$E(\mathbf{x}) = \sum_{i\in\mathcal{V}} \theta_i(x_i) + \sum_{(i,j)\in\mathcal{E}} \theta_{ij}(x_i, x_j) \; , \tag{2.44}$$

where $x_i \in \mathcal{L} = \{0, 1, \ldots, \ell-1\}$. The binary label associated with a move is denoted with $u_i \in \{0,1\}$ for all $i \in \mathcal{V}$. For each i, the value of u_i determines a move in the original MRF, which is defined by a *move-function*. In fact, in a typical move-making algorithm, it is necessary to apply a sequence of different moves. Therefore such move-functions are usually parametrized.

Definition 2.3.2. A set of *move-functions* ρ_ν, parametrized by ν, are defined such that each ρ_ν is a function,

$$x_i^{t+1} = \rho_\nu\left(x_i^t, u_i\right) \ , \tag{2.45}$$

for all $i \in \mathcal{V}$. Here, x_i^t is the assignment of the random variable X_i at iteration t and u_i is the binary label. Furthermore, for some value of u_i, the label x_i must be unchanged, *i.e.*,

$$x_i^{t+1} = \rho_\nu(x_i^t, u_i) = x_i^t \ . \tag{2.46}$$

Usually, for a given parameter ν, the move-function takes the following form,

$$x_i^{t+1} = \rho_\nu\left(x_i^t, u_i\right) = \begin{cases} x_i^0 & \text{if } u_i = 0 \\ x_i^1 & \text{if } u_i = 1 \ , \end{cases} \tag{2.47}$$

where $x_i^0, x_i^1 \in \mathcal{L}$ depend on the current label x_i^t and the parameter ν. One can write this in vector form as

$$\mathbf{x}^{t+1} = \rho_\nu\left(\mathbf{x}^t, \mathbf{u}\right) \ . \tag{2.48}$$

Now, one can write the energy function associated with the move as

$$E^{\rho_\nu}(\mathbf{u}) = E(\mathbf{x}^{t+1}) = E\left(\rho_\nu\left(\mathbf{x}^t, \mathbf{u}\right)\right) \ . \tag{2.49}$$

Note that the above energy function is over the binary variables $\mathbf{u}$. At each iteration t, the optimal move is computed by minimizing the above surrogate energy and the labelling $\mathbf{x}^{t+1}$ is updated. A complete move-making algorithm is given in Algorithm 2.1.

From Algorithm 2.1, it is evident that the surrogate energy must be easy to minimize. The obvious choice would be submodular functions. Even though this choice would restrict the multi-label energy functions that can be minimized using a move-making algorithm, different choices of move-function (or surrogate energy) would let us handle different multi-label energy functions. Furthermore, since $\rho_\nu(x_i^t, u_i) = x_i^t$ for some u_i (see Definition 2.3.2), by finding optimal moves, move-making algorithms guarantee a monotonic decrease of the multi-label energy function.

We will now describe two examples of move-making algorithms that differ only by the choice of the move functions.

2.3.3.1 α-expansion

The α-expansion algorithm was introduced for the metric-labelling problem in [Boykov et al., 2001]. In addition to being fast, it also provides an approximation bound on the solution obtained by it.

Given the current labelling $\mathbf{x}^t$ and a label $\alpha \in \mathcal{L}$, during an iteration of α-expansion, each node i is given a choice to switch to the new label α or retain its

Algorithm 2.1 A Move-Making Algorithm

```
Require: $\rho_v$ for all $v$                                   ▷ The set of move-functions
  $\mathbf{x}^0$                                                 ▷ Initial solution
  repeat
    for each $\rho_v$ from a given set of parameters do
      $\mathbf{u}^* \leftarrow \operatorname{argmin}_{\mathbf{u}} E\left(\rho_v\left(\mathbf{x}^t, \mathbf{u}\right)\right)$      ▷ Minimize the binary energy
      $\mathbf{x}^{t+1} \leftarrow \rho_v\left(\mathbf{x}^t, \mathbf{u}^*\right)$                ▷ Update labelling
    end for
  until $E(\mathbf{x}^{t+1})$ cannot be decreased any further
  return $\mathbf{x}^{t+1}$
```

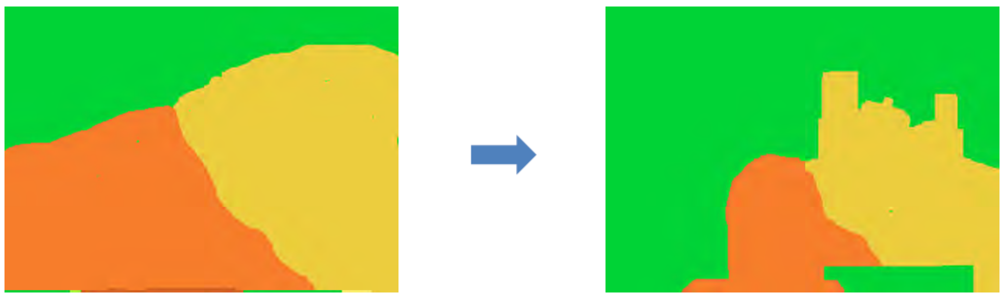

(a) Current labelling (b) After green expansion

Figure 2.10: *Example of an α-expansion iteration. Given the current labelling (**left**), the green label is expanded to obtain a lower energy labelling (**right**).*

current label. See Figure 2.10. Hence, the set of move-functions can be defined as

$$x_i^{t+1} = \rho_\alpha\left(x_i^t, u_i\right) = \begin{cases} x_i^t & \text{if } u_i = 0 \\ \alpha & \text{if } u_i = 1\,. \end{cases} \tag{2.50}$$

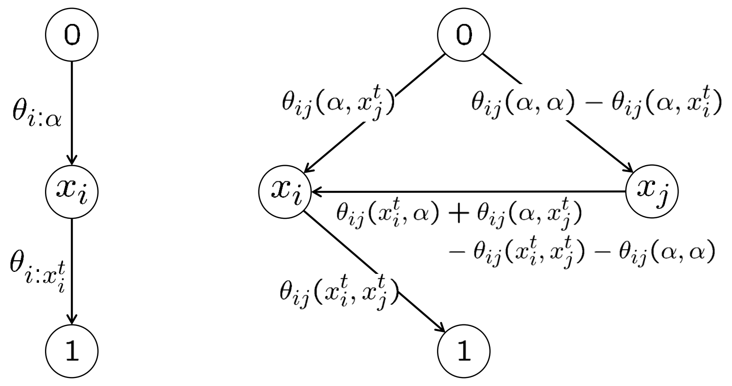
Here, the move-function is parametrized by $\alpha \in \mathcal{L}$. The energy associated with the binary optimization problem can be written as,

$$\begin{aligned} E^\alpha(\mathbf{u}) &= E\left(\rho_\alpha\left(\mathbf{x}^t, \mathbf{u}\right)\right)\ , \\ &= \sum_{i\in\mathcal{V}} \theta_i^\alpha(u_i) + \sum_{(i,j)\in\mathcal{E}} \theta_{ij}^\alpha(u_i, u_j)\ , \end{aligned} \tag{2.51}$$

where the energy parameters $\boldsymbol{\theta}^\alpha$ depend on the current labelling $\mathbf{x}^t$. Let us now derive these energy parameters. Consider the unary term $\theta_i^\alpha(u_i)$. From Eq. (2.50), it is clear that

$$\begin{aligned} \theta_i^\alpha(0) &= \theta_i(x_i^t)\ , \\ \theta_i^\alpha(1) &= \theta_i(\alpha)\ . \end{aligned} \tag{2.52}$$

Let us now turn to the pairwise terms. Similarly, from the move-function, we can



0
$\theta_{i:\alpha}$
x_i
$\theta_{i:x_i^t}$
1
0
$\theta_{ij}(\alpha, x_j^t)$
$\theta_{ij}(\alpha,\alpha) - \theta_{ij}(\alpha, x_i^t)$
x_i
x_j
$\theta_{ij}(x_i^t,\alpha) + \theta_{ij}(\alpha, x_j^t) - \theta_{ij}(x_i^t, x_j^t) - \theta_{ij}(\alpha,\alpha)$
$\theta_{ij}(x_i^t, x_j^t)$
1

Figure 2.11: *The st-graph construction for the α-expansion algorithm. Note that the graph construction is very similar to Figure 2.5, where label* 1 *is replaced with* α *and* 0 *is replaced with the current label* x_i^t.

write

$$\begin{aligned}
\theta^\alpha_{ij}(0,0) &= \theta_{ij}(x_i^t, x_j^t)\ , \qquad (2.53)\\
\theta^\alpha_{ij}(0,1) &= \theta_{ij}(x_i^t, \alpha)\ ,\\
\theta^\alpha_{ij}(1,0) &= \theta_{ij}(\alpha, x_j^t)\ ,\\
\theta^\alpha_{ij}(1,1) &= \theta_{ij}(\alpha, \alpha)\ .
\end{aligned}$$

Note that, for the binary energy function (2.51) to be submodular, the pairwise terms must satisfy

$$\begin{aligned}
\theta^\alpha_{ij}(0,1) + \theta^\alpha_{ij}(1,0) &\ge \theta^\alpha_{ij}(0,0) + \theta^\alpha_{ij}(1,1)\ , \qquad (2.54)\\
\theta_{ij}(x_i^t,\alpha) + \theta_{ij}(\alpha, x_j^t) &\ge \theta_{ij}(x_i^t, x_j^t) + \theta_{ij}(\alpha,\alpha)\ ,
\end{aligned}$$

for all $(i,j) \in \mathcal{E}$ and $\alpha \in \mathcal{L}$. Hence, the optimal α-expansion move can be computed for any multi-label MRF that satisfies the above condition. The corresponding *st*-graph construction for α-expansion is given in Figure 2.11. See Section 2.2 for more detail on binary MRF optimization.

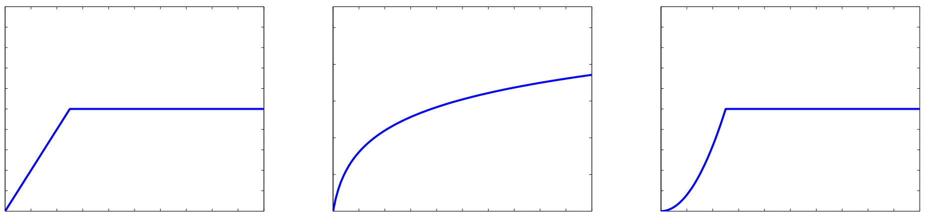
Metric Potentials. We consider pairwise potentials of the form,

$$\theta_{ij}(\lambda,\mu) = \gamma_{ij}\,\theta(\lambda,\mu)\ , \qquad (2.55)$$

where $\gamma_{ij} \ge 0$ and $\theta(\cdot,\cdot)$ is a *metric* distance function.

Definition 2.3.3. A distance function $\theta : \mathcal{L}\times\mathcal{L} \to \mathbb{R}^+$ is called a *metric* if it satisfies the following conditions,

1. $\theta(\lambda,\mu) \ge 0$ for all $\lambda,\mu \in \mathcal{L}$.



(a) Truncated linear (b) Logarithmic (c) Truncated quadratic

Figure 2.12: *Examples of metric and semi-metric potentials:* ***a)*** *$\theta(|\delta|)$ is a truncated linear function;* ***b)*** *a logarithmic function and* ***c)*** *a truncated quadratic function. Here,* ***(a)*** *and* ***(b)*** *are concave and hence metric, and* ***(c)*** *is a semi-metric but not a metric. Note that the plot is given for $\delta \geq 0$.*

2. $\theta(\lambda, \mu) = 0$ if and only if $\lambda = \mu$.
3. $\theta(\lambda, \mu) = \theta(\mu, \lambda)$ for all $\lambda, \mu \in \mathcal{L}$.
4. $\theta(\lambda, \delta) + \theta(\delta, \mu) \geq \theta(\lambda, \mu)$ for all $\lambda, \mu, \delta \in \mathcal{L}$.

The last condition is referred to as the *triangle inequality*. A distance function satisfying all the above properties except the triangle inequality is called a *semi-metric*.

Note that a non-negative function of the form $\theta(|\lambda - \mu|)$ is a metric if $\theta(\cdot)$ is concave. Such functions are often referred to as *concave priors*. See Figure 2.12 for examples of metric and semi-metric distance functions. Note that any metric function is also a semi-metric. Furthermore, in case of metric pairwise potentials, the submodular condition (2.54) reduces to the triangle inequality,

$$\begin{aligned} \theta_{ij}(x_i^t, \alpha) + \theta_{ij}(\alpha, x_j^t) &\geq \theta_{ij}(x_i^t, x_j^t) + \theta_{ij}(\alpha, \alpha) \ , \\ \theta_{ij}(x_i^t, \alpha) + \theta_{ij}(\alpha, x_j^t) &\geq \theta_{ij}(x_i^t, x_j^t) \ . \end{aligned} \tag{2.56}$$

Therefore the optimal α-expansion move can be computed using the max-flow algorithm. Furthermore, for such an MRF, the solution obtained by α-expansion is within a multiplictive bound of $2\,c$, where $c = \frac{\max_{\lambda \neq \mu} \theta(\lambda, \mu)}{\min_{\lambda \neq \mu} \theta(\lambda, \mu)}$.

2.3.3.2 Convex α-expansion

Let us now discuss a different expansion algorithm that makes optimal moves for a multi-label submodular MRFs. Note that, even though Ishikawa algorithm is optimal in this case, this algorithm can be used to quickly approximate the optimal solution, with a fraction of resources (memory and computation power) [Carr and Hartley, 2009]. Since pairwise MRFs with convex priors and ordered label set are mutli-label submodular, this algorithm got its name.

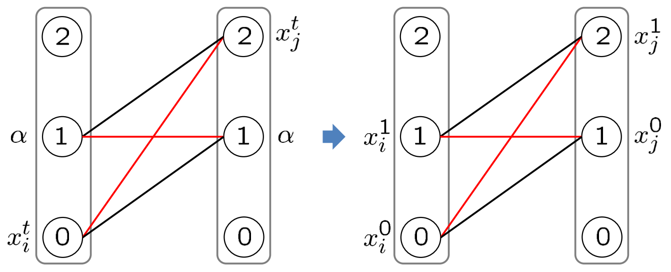


Figure 2.13: *An example binary labelling problem of convex α-expansion, where $x_i^t = 0$ and $x_j^t = 3$ and $\alpha = 2$ (**left**) and corresponding label proposals from Eq.* (2.58) *(**right**). Note that, for a multi-label submodular function, the sum of red edge weights are greater than the sum of black edge weights (see Figure 2.7). Hence, the binary labelling problem is submodular.*

The set of move-functions of convex α-expansion are defined as follows,

$$x_i^{t+1} = \rho_\alpha\left(x_i^t, u_i\right) = \begin{cases} \min(x_i^t, \alpha) & \text{if } u_i = 0 \\ \max(x_i^t, \alpha) & \text{if } u_i = 1 \,. \end{cases} \tag{2.57}$$

Similarly to the original α-expansion, the move-function is parametrized by $\alpha \in \mathcal{L}$. Furthermore, the binary energy associated with this optimization problem has the same form as Eq. (2.51) and one can derive the unary and the pairwise terms in a similar manner. For convenience, let us define $\mathbf{x}^0$ and $\mathbf{x}^1$ below. For all $i \in \mathcal{V}$,

$$\begin{aligned} x_i^0 &= \min(x_i^t, \alpha)\,, \\ x_i^1 &= \max(x_i^t, \alpha)\,. \end{aligned} \tag{2.58}$$

Now, for the binary energy function to be submodular, the pairwise term must satisfy

$$\theta_{ij}(x_i^0, x_j^1) + \theta_{ij}(x_i^1, x_j^0) \ge \theta_{ij}(x_i^0, x_j^0) + \theta_{ij}(x_i^1, x_j^1)\,, \tag{2.59}$$

for all $(i,j) \in \mathcal{E}$. This condition is graphically depicted in Figure 2.13. Note that the binary energy function is submodular, if the original energy function (2.44) is multi-label submodular. Therefore, for a multi-label submodular energy function, the optimal expansion move can be computed.

2.3.3.3 Other Move-Making Algorithms

There are many other move-making algorithms that can handle different pairwise potentials. For example, $\alpha\beta$-swap [Boykov et al., 2001] finds the optimal move for *semi-metric* pairwise potentials. Furthermore, multi-label swap and expansion [Veksler, 2012; Torr and Kumar, 2009] can handle truncated convex priors. On the other hand, FastPD [Komodakis et al., 2008] generalizes α-expansion by formulating it as a primal-dual algorithm, and improves its efficiency by utilizing the dual variables for the optimization. Finally, fusion moves introduced in [Lempitsky et al., 2010] is a more general move-making algorithm to fuse two labellings. In general, the binary optimization problem of fusion moves is not submodular and the QPBO algorithm [Boros and Hammer, 2002] is used to optimize it. Despite this, fusion moves guarantee a monotonic decrease in the energy.

In Chapter 4, we introduce an iterative algorithm, named Iteratively Reweighted Graph-Cut (IRGC), that can approximately minimize MRFs with a certain class of robust non-convex priors. In fact, IRGC is in spirit a move-making algorithm, by which we mean, at each iteration, it solves a graph-cut problem and guarantees monotonic decrease in the multi-label energy.

2.3.4 Linear Programming Relaxation

So far, we have discussed both exact and approximate algorithms that have been based on combinatorial optimization literature. Let us now turn to a more general and powerful approximation algorithm that has been developed using a continuous relaxation, in particular the Linear Programming (LP) relaxation. More specifically, we discuss a tree-based decomposition algorithm to optimize the dual of the LP relaxation, where each subproblem (tree MRF) is optimized using the message passing algorithm. Note that message passing was developed using information theoretic concepts and there have been many variants in the literature [Pearl, 1988; Yedidia et al., 2000; Wainwright et al., 2008]. Here we discuss it in the context of LP relaxation and try to give a simple and unified description.

Recall the multi-label graph representation of the energy function,

$$E(\mathbf{x}) = \sum_{i\in\mathcal{V}} \sum_{\lambda\in\mathcal{L}} \theta_{i:\lambda}\, x_{i:\lambda} + \sum_{(i,j)\in\mathcal{E}} \sum_{\lambda,\mu\in\mathcal{L}} \theta_{ij:\lambda\mu}\, x_{i:\lambda}\, x_{j:\mu}\ , \tag{2.60}$$

$$\text{where}\quad \mathbf{x}\in\Gamma = \left\{ \mathbf{x} \;\middle|\; \begin{array}{l} \sum_{\lambda\in\mathcal{L}} x_{i:\lambda} = 1,\ i\in\mathcal{V} \\ x_{i:\lambda}\in\mathcal{B}=\{0,1\},\ i\in\mathcal{V},\ \lambda\in\mathcal{L} \end{array} \right\}\ .$$

Here $\Gamma \subset \mathcal{B}^{n\ell}$ ($n = |\mathcal{V}|$ and $\ell = |\mathcal{L}|$) denotes the set of all valid integral labellings. Given a labelling $\mathbf{x}\in\Gamma$, the value of the energy function is obtained by summing the weights on all nodes with $x_{i:\lambda} = 1$ (in other words $x_i = \lambda$) plus the weights $\theta_{ij:\lambda\mu}$ such that $x_{i:\lambda} = 1$ and $x_{j:\mu} = 1$. See Figure 2.14. For convenience, we assume that the parameters $\theta_{i:\lambda}$ and $\theta_{ij:\lambda\mu}$ are non-negative. This can be ensured by adding a constant value to any $\theta_{i:\lambda}\,;\ \lambda\in\mathcal{L}$ or $\theta_{ij:\lambda\mu}\,;\ \lambda,\mu\in\mathcal{L}$.

Let us define the *parameter vector*

$$\boldsymbol{\theta} = [\dots,\theta_{i:\lambda},\dots,\theta_{ij:\lambda\mu},\dots]\ , \tag{2.61}$$

where the elements $\theta_{i:\lambda}\,;\ i\in\mathcal{V},\ \lambda\in\mathcal{L}$ and $\theta_{ij:\lambda\mu}\,;\ (i,j)\in\mathcal{E},\ \lambda,\mu\in\mathcal{L}$ are listed in some particular order. The dimension of the parameter vector is $N = n\,\ell + m\,\ell^2$, where $n = |\mathcal{V}|$ and $m = |\mathcal{E}|$. Similarly, let us introduce additional variables $x_{ij:\lambda\mu}$ for the product $x_{i:\lambda}\,x_{j:\mu}$ and define the vector

$$\tilde{\mathbf{x}} = [\dots,x_{i:\lambda},\dots,x_{ij:\lambda\mu},\dots]\ . \tag{2.62}$$

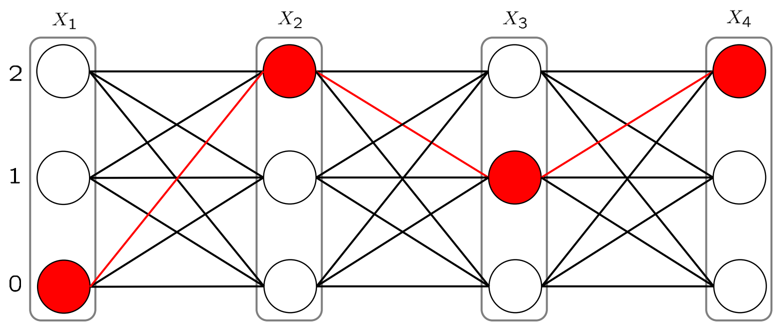


Figure 2.14: *An example of a multi-label graph corresponding to a linear MRF. Here the highlighted path corresponds to the labelling* $\mathbf{x} = \{(1,0,0),(0,0,1),(0,1,0),(0,0,1)\}$*. The cost of this labelling is the sum of the weights of the highlighted nodes and edges.*

Here, the elements are listed in the same order as the parameter vector $\boldsymbol{\theta}$. With this notation, the energy function (2.60) can be written as,

$$E_{\boldsymbol{\theta}}(\mathbf{x}) = \langle \boldsymbol{\theta}, \tilde{\mathbf{x}} \rangle \ , \tag{2.63}$$

where $\langle \cdot, \cdot \rangle$ is the standard inner product in $\mathbb{R}^N$.

2.3.4.1 Marginal Polytope

Note that the vector $\mathbf{x} \in \Gamma$ and $\tilde{\mathbf{x}} \in \mathcal{B}^N$, where $\mathcal{B}^N$ is a discrete subset of $\mathbb{R}^N$. Let us denote the set of all points $\tilde{\mathbf{x}}$ for $\mathbf{x} \in \Gamma$ as $\mathcal{T}$. Writing $\mathcal{T}$ explicitly,

$$\mathcal{T} = \left\{ \tilde{\mathbf{x}} \;\middle|\; \begin{array}{l} \sum_{\lambda \in \mathcal{L}} x_{i:\lambda} = 1,\, i \in \mathcal{V} \\ x_{i:\lambda} \in \{0,1\},\, i \in \mathcal{V},\, \lambda \in \mathcal{L} \\ x_{ij:\lambda\mu} = x_{i:\lambda}\, x_{j:\mu},\, (i,j) \in \mathcal{E},\, \lambda, \mu \in \mathcal{L} \end{array} \right\} . \tag{2.64}$$

Here, $\mathcal{T} \subset \mathcal{B}^N$ is a discrete set of ℓ^n points, where ℓ is the number of labels. Now, from Eq. (2.63), we can write the optimization problem of minimizing the multi-label energy function as an Integer Linear Program (ILP),

$$\min_{\mathbf{x} \in \Gamma} E_{\boldsymbol{\theta}}(\mathbf{x}) = \min_{\boldsymbol{\tau} \in \mathcal{T}} \langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle \ . \tag{2.65}$$

The idea is to relax the integer constraints in the optimization problem. To this end, let us define the *marginal polytope* as follows.

Definition 2.3.4. Let $\mathcal{T}$ be the set of all points $\tilde{\mathbf{x}}$ for $\mathbf{x} \in \Gamma$. The *marginal polytope* Ω is defined as the *convex hull*[5] of $\mathcal{T}$, *i.e.*,

$$\Omega = \mathrm{conv}(\mathcal{T}) \ . \tag{2.66}$$

[5]The *convex hull* of a set C is the set of all convex combinations of points in C [Boyd and Vandenberghe, 2009].

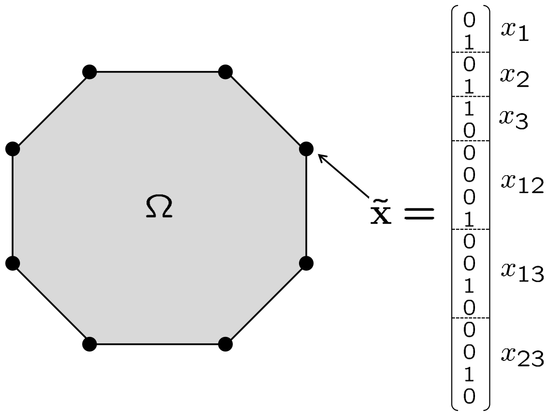


Figure 2.15: *The marginal polytope of an example binary MRF with* 3 *nodes. The vertices are in one-to-one correspondence with the integral labellings of the MRF.*

The marginal polytope Ω is a *polytope*[6] in $\mathbb{R}^N$. The vertices of the marginal polytope and the set $\mathcal{T}$ are related according to the following Lemma.

Lemma 2.3.1. *The set of vertices of the marginal polytope Ω is in fact the set $\mathcal{T}$.*

Proof. Let $\mathcal{V}_\Omega$ denote the vertices of Ω. Since $\Omega = \text{conv}(\mathcal{T})$, the set of vertices of Ω is a subset of $\mathcal{T}$, *i.e.*, $\mathcal{V}_\Omega \subset \mathcal{T}$. Also, since $\mathcal{T} \subset \Omega$, we have,

$$\min_{\boldsymbol{\tau}\in\Omega} \langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle \le \min_{\boldsymbol{\tau}\in\mathcal{T}} \langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle \ . \tag{2.67}$$

Furthermore, the minimum of a linear program over a convex polytope is attained at a vertex of the polytope [Dantzig, 2016]. This means, the minimum point $\boldsymbol{\tau}^* \in \mathcal{V}_\Omega \subset \mathcal{T}$. Therefore the above inequality is tight. Since one has the flexibility of choosing any parameter vector $\boldsymbol{\theta}$, any $\boldsymbol{\tau} \in \mathcal{T}$ can be the minimum. Hence, $\mathcal{V}_\Omega = \mathcal{T}$. □

An example of a marginal polytope is illustrated in Figure 2.15. Let us now consider the following relaxed optimization problem,

$$\min_{\boldsymbol{\tau}\in\Omega} \langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle \ . \tag{2.68}$$

From the proof of Lemma 2.3.1, it is clear that, even though the integrality constraints are relaxed, the optimum of the ILP (2.65) can be obtained. Therefore, we have,

$$\min_{\mathbf{x}\in\Gamma} E_{\boldsymbol{\theta}}(\mathbf{x}) = \min_{\boldsymbol{\tau}\in\mathcal{T}} \langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle = \min_{\boldsymbol{\tau}\in\Omega} \langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle \ . \tag{2.69}$$

An interesting point to note here is that finding the optimal labelling of an MRF is equivalent to a linear program. This might be contradictory, as MRF optimization is

[6]A *polytope* is the intersection of a finite set of half-spaces. A *half-space* is the set of points satisfying a linear inequality $\mathbf{a}^T\mathbf{x} + \mathbf{b} \le 0$ [Boyd and Vandenberghe, 2009].

an NP-hard problem in general. However the explanation is that the LP in Eq. (2.68) has exponentially many constraints, making it an NP-hard problem.

2.3.4.2 Local Polytope

As mentioned in the previous section, minimizing an arbitrary multi-label MRF energy can be written as a linear program over the marginal polytope Ω. However, in general, Ω has exponentially many faces and vertices, hence making it an intractable problem. In this section, we will approximate the marginal polytope with a simpler polytope that has a polynomial number of faces.

To this end, one may think of a different relaxation for the set of points $\mathcal{T}$. Before that, let us first relax the integral constraints on the labelling $\mathbf{x} \in \Gamma$.

Definition 2.3.5. Let us define the set of all valid real (continuous) labellings as

$$\mathcal{S} = \left\{ \mathbf{y} \;\middle|\; \begin{array}{l} \sum_{\lambda \in \mathcal{L}} y_{i:\lambda} = 1,\, i \in \mathcal{V} \\ y_{i:\lambda} \geq 0,\, i \in \mathcal{V},\, \lambda \in \mathcal{L} \end{array} \right\} . \tag{2.70}$$

The set of real labellings and the integral labellings are related as follows.

Lemma 2.3.2. *The set of real labellings $\mathcal{S}$ is the convex hull of the set of valid integral labellings* Γ, i.e., $\mathcal{S} = \text{conv}(\Gamma)$.

Proof. It can be easily proven that any point $\mathbf{y} \in \mathcal{S}$ is a convex combination of points in Γ. Now, since $\mathcal{S}$ is a convex set, $\mathcal{S} = \text{conv}(\Gamma)$. □

Similar to Eq. (2.62), by introducing additional variables $y_{ij:\lambda\mu}$, we define the vector $\tilde{\mathbf{y}}$ as

$$\tilde{\mathbf{y}} = [\ldots, y_{i:\lambda}, \ldots, y_{ij:\lambda\mu}, \ldots] , \tag{2.71}$$

with the same ordering of elements as in Eq. (2.62). Note that the vector $\tilde{\mathbf{y}} \in \mathbb{R}^N$. Now we are ready to define a convex polytope containing $\mathcal{T}$ that has a polynomial number of faces.

Definition 2.3.6. Let the vector $\tilde{\mathbf{y}} \in \mathbb{R}^N$ be as defined in Eq. (2.71). The *local polytope* Λ is defined as

$$\Lambda = \left\{ \tilde{\mathbf{y}} \;\middle|\; \begin{array}{l} \sum_{\lambda \in \mathcal{L}} y_{i:\lambda} = 1,\, i \in \mathcal{V} \\ y_{i:\lambda} \geq 0,\, i \in \mathcal{V},\, \lambda \in \mathcal{L} \\ \sum_{\lambda \in \mathcal{L}} y_{ij:\lambda\mu} = y_{j:\mu},\, (i,j) \in \mathcal{E},\, \mu \in \mathcal{L} \\ \sum_{\mu \in \mathcal{L}} y_{ij:\lambda\mu} = y_{i:\lambda},\, (i,j) \in \mathcal{E},\, \lambda \in \mathcal{L} \\ y_{ij:\lambda\mu} \geq 0,\, (i,j) \in \mathcal{E},\, \lambda, \mu \in \mathcal{L} \end{array} \right\} . \tag{2.72}$$

The relationship between Ω and Λ can be stated as follows [Wainwright et al., 2005].

Lemma 2.3.3. *The local polytope Λ is a convex set containing the marginal polytope Ω,* i.e., $\Omega \subset \Lambda$, *and its vertex set includes all the vertices of the marginal polytope.*

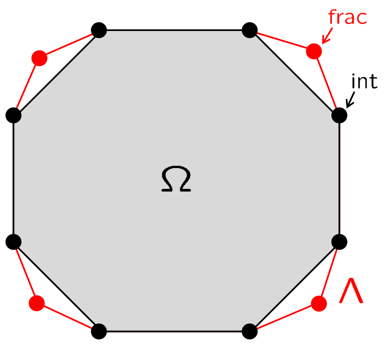


Figure 2.16: *The local polytope Λ (**red**) is a convex outer bound of the marginal polytope Ω. The additional vertices of the local polytope are often referred to as fractional vertices.*

Proof. Since Λ is defined using linear constraints, it is convex. Note that any point in $\mathcal{T}$ satisfies the constraints defined in Eq. (2.72) and by definition $\Omega = \text{conv}(\mathcal{T})$. Since Λ is a convex set containing $\mathcal{T}$, it must also contain the convex hull of $\mathcal{T}$. Hence, $\Omega \subset \Lambda$.

From Lemma 2.3.1, the vertices of the marginal polytope is the set $\mathcal{T}$. Let us now prove that any point in $\mathcal{T}$ is a vertex of Λ. We prove this by showing that there is no line segment in Λ such that $\tilde{\mathbf{x}} \in \mathcal{T}$ is an *interior point*[7]. Consider $\tilde{\mathbf{x}} \in \mathcal{T}$ and $\tilde{\mathbf{y}} \in \Lambda$, and let

$$\boldsymbol{\tau} = (1-\alpha)\,\tilde{\mathbf{x}} + \alpha\,\tilde{\mathbf{y}}\;, \tag{2.73}$$

for some $\alpha \in [-\epsilon, \epsilon]$. Note that, when $\alpha < 0$, $\tilde{\mathbf{x}}$ is an interior point of the line segment connecting $\tilde{\mathbf{y}}$ and $\boldsymbol{\tau}$. We show that $\boldsymbol{\tau} \notin \Lambda$ if $\alpha < 0$. Choose $i \in \mathcal{V}$ and $\lambda \in \mathcal{L}$ such that $x_{i:\lambda} = 0$ and $y_{i:\lambda} > 0$. This is possible unless $\tilde{\mathbf{y}} = \tilde{\mathbf{x}}$. Let the element in $\boldsymbol{\tau}$ corresponding to indices $\{i, \lambda\}$ be $\tau_{i:\lambda}$. Then,

$$\tau_{i:\lambda} = (1-\alpha)\,x_{i:\lambda} + \alpha\,y_{i:\lambda} = \alpha\,y_{i:\lambda}\;. \tag{2.74}$$

Here, if $\alpha < 0$, then $\tau_{i:\lambda} < 0$. This means $\boldsymbol{\tau} \notin \Lambda$. Hence, any point $\tilde{\mathbf{x}} \in \mathcal{T}$ must be a vertex of Λ. □

See Figure 2.16 for an illustration of this Lemma. Now, it is clear that the local polytope has an exponential number of vertices (because $\mathcal{T}$ has an exponential number of elements) and a polynomial number of faces (because Λ is defined using a polynomial number of inequalities). However, the local polytope may contain additional vertices, which are often referred to as *fractional vertices*. See Figure 2.17 for an example of a fractional vertex of the local polytope. Since the local polytope is defined using a polynomial number of constraints, the minimization over it can be done in polynomial time[8] [Karmarkar, 1984].

[7]If C is a subset of a Euclidean space, then x is an *interior point* of C, if there exists an *open ball* centered at x which is completely contained in C [Boyd and Vandenberghe, 2009]. An *open ball* centered at x is defined as, $B(x,r) = \{y \mid \|y - x\| \le r\}$, for some $r > 0$, where $\|\cdot\|$ is some norm.

[8]The worst case time complexity of an LP is polynomial in the number of variables and constraints.

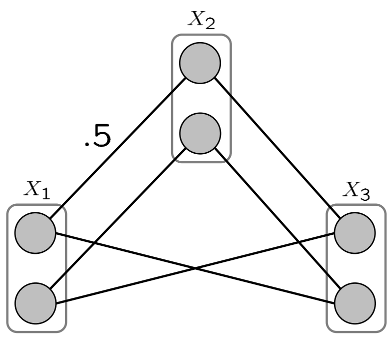


Figure 2.17: *An MRF example where a fractional labelling is a vertex of the local polytope. Here, we consider a binary MRF with* 3 *nodes, and the labelling is denoted with shaded nodes and edges, where, for all the nodes,* $y_{i:\lambda} = 0.5$ *and, for the edges shown in the figure,* $y_{ij:\lambda\mu} = 0.5$*. It can be verified that the labelling shown is a vertex of* Λ *by following the same argument as in the proof of Lemma 2.3.3.*

In general, Λ is strictly larger than Ω. Hence, the minimization over the local polytope provides a *lower bound* to the optimal energy,

$$\min_{\boldsymbol{\tau}\in\Lambda}\langle\boldsymbol{\theta},\boldsymbol{\tau}\rangle \le \min_{\boldsymbol{\tau}\in\Omega}\langle\boldsymbol{\theta},\boldsymbol{\tau}\rangle = \min_{\mathbf{x}\in\Gamma} E_{\boldsymbol{\theta}}(\mathbf{x}) \ . \tag{2.75}$$

This lower bound is tight for multi-label submodular MRFs [Werner, 2007] and tree structured MRFs [Wainwright et al., 2005]. Specifically, for such MRFs, the optimal labelling can be obtained by minimizing the LP relaxation over the local polytope. This means the minimum obtained by optimizing over Λ is a vertex of the marginal polytope. In fact, the marginal polytope and local polytope depend on the MRF graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and not on the energy parameters $\boldsymbol{\theta}$. Therefore, for tree structured MRFs, the relationship between Ω and Λ can be stated as follows.

Theorem 2.3.1. *For a tree structured MRF,* $\Omega = \Lambda$.

Proof. The idea is that, for tree structured MRFs, the optimal labelling can be obtained by minimizing over the local polytope [Wainwright et al., 2008], *i.e.*,

$$\min_{\boldsymbol{\tau}\in\Lambda}\langle\boldsymbol{\theta},\boldsymbol{\tau}\rangle = \min_{\boldsymbol{\tau}\in\Omega}\langle\boldsymbol{\theta},\boldsymbol{\tau}\rangle = \min_{\mathbf{x}\in\Gamma} E_{\boldsymbol{\theta}}(\mathbf{x}) \ , \tag{2.76}$$

and let $\boldsymbol{\tau}^*$ be the point where the minimum is attained. Now, if $\Omega \subset \Lambda$ and $\Omega \neq \Lambda$, then, one can find a $\boldsymbol{\theta}$ such that $\boldsymbol{\tau}^* \in \Lambda$ and $\boldsymbol{\tau}^* \notin \Omega$, which is a contradiction. Therefore, $\Omega = \Lambda$. See [Wainwright et al., 2008] for more detail. □

Note that, for multi-label submodular MRFs defined on general graphs, Λ is a strict outer bound of Ω. However, due to the restriction on the energy parameters $\boldsymbol{\theta}$, the exact minimum is obtained by optimizing over Λ. In more general cases,

However, a strongly polynomial time algorithm has not yet been discovered.

optimization over Λ yields fractional labellings. Such a fractional labelling can be rounded to obtain an integral labelling using simple *argmax* rounding. Specifically, let $\mathbf{y}^*$ be the optimal fractional labelling, then the integral labelling $\mathbf{x}^*$ is given by

$$x_i^* = \operatorname*{argmax}_{\lambda \in \mathcal{L}} y_{i:\lambda}^* \quad \forall\, i \in \mathcal{V} \,. \tag{2.77}$$

In general, this rounding procedure is not optimal. Nevertheless, more sophisticated rounding schemes that provide theoretical bounds on the rounded labelling have also been introduced [Kleinberg and Tardos, 2002; Ravikumar et al., 2008].

2.3.4.3 Message Passing and Reparametrization

In this section, we discuss an algorithm that can be used to minimize the LP relaxation of a multi-label MRF over the local polytope. In the remainder of this section, we use the term *LP relaxation* to denote the LP over the local polytope,

$$\min_{\boldsymbol{\tau} \in \Lambda} \langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle \,. \tag{2.78}$$

To this end, let us first briefly describe the well-known *min-sum message passing* (or *min-sum belief propagation*) [Pearl, 1988].

Min-Sum Message Passing. Min-sum message passing is an approximate algorithm for minimizing the multi-label energy function (2.60). This algorithm maintains a *message vector* $m_{ij:\mu}$ *;* $\mu \in \mathcal{L}$ for each directed edge[9] $(i,j) \in \mathcal{E}^+$. The notation $m_{ij:\mu}$ denotes the message from $i \to j$ indexed by μ. The basic operation of this algorithm is passing a message from $i \to j$ for each directed edge $(i,j) \in \mathcal{E}^+$. This operation can be written as

$$m_{ij:\mu} \leftarrow \min_{\lambda \in \mathcal{L}} \left(\theta_{ij:\lambda\mu} + \theta_{i:\lambda} + \sum_{\substack{(k,i) \in \mathcal{E}^+ \\ k \neq j}} m_{ki:\lambda} \right) \,, \tag{2.79}$$

for all $\mu \in \mathcal{L}$. A message from $i \to j$ is *valid* if this update does not change the message vector $\mathbf{m}_{ij}$. The idea of this algorithm is to keep passing messages of this form, in some particular order, until convergence, *i.e.*, until all messages are valid. Note that the time complexity of one message passing step is $\mathcal{O}(\ell^2)$, where ℓ is the number of labels. Upon convergence, the algorithm provides *beliefs* (approximate *min-marginals*) for all random variables,

$$\theta'_{i:\lambda} = \theta_{i:\lambda} + \sum_{(k,i) \in \mathcal{E}^+} m_{ki:\lambda} \,, \tag{2.80}$$

[9] $\mathcal{E}^+$ denotes the set of directed edges between the vertices in the MRF, *i.e.*, if $(i,j) \in \mathcal{E}$ then, $(i,j) \in \mathcal{E}^+$ and $(j,i) \in \mathcal{E}^+$.

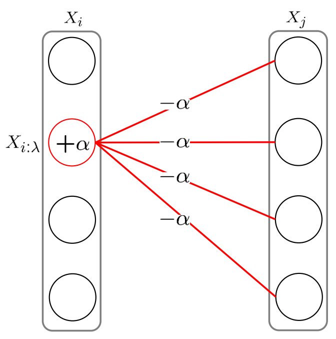


Figure 2.18: *An elementary reparametrization. Subtract α from all incoming edges of node $X_{i:\lambda}$ and add it to the node potential. This corresponds to the message $m_{ji:\lambda} = \alpha$. The reverse operation is also a valid reparametrization.*

for all $i \in \mathcal{V}$ and $\lambda \in \mathcal{L}$. Then, the approximate labelling can be obtained from the beliefs,

$$x'_{i:\lambda} = \operatorname*{argmin}_{\lambda \in \mathcal{L}} \theta'_{i:\lambda} \ . \tag{2.81}$$

It will be seen later, that min-sum message passing is optimal on tree structured MRFs. However, for a graph with loops, the convergence of this algorithm is not guaranteed. Despite of this shortcoming, the idea of min-sum message passing will be useful in developing algorithms to minimize the LP relaxation.

Reparametrization. A given multi-label energy function can be written in different ways as a sum of unary and pairwise terms. In particular, there may exist a different set of parameters $\boldsymbol{\theta}'$ such that $E_{\boldsymbol{\theta}}(\mathbf{x}) = E_{\boldsymbol{\theta}'}(\mathbf{x})$ for all $\mathbf{x}$, denoted as $E_{\boldsymbol{\theta}} \equiv E_{\boldsymbol{\theta}'}$.

Lemma 2.3.4. *Two energy functions $E_{\boldsymbol{\theta}}$ and $E_{\boldsymbol{\theta}'}$ are equivalent if and only if there exist values $m_{ji:\lambda}$ and $m_{ij:\mu}$ for $(i,j) \in \mathcal{E}$ and $\lambda, \mu \in \mathcal{L}$ such that*

$$\begin{aligned} \theta'_{ij:\lambda\mu} &= \theta_{ij:\lambda\mu} - m_{ji:\lambda} - m_{ij:\mu} \ , \\ \theta'_{i:\lambda} &= \theta_{i:\lambda} + \sum_{(k,i)\in\mathcal{E}^+} m_{ki:\lambda} \ . \end{aligned} \tag{2.82}$$

Proof. This result is well known, and we refer the interested reader to [Kolmogorov, 2006; Werner, 2007]. □

The values of $m_{ij:\mu}$ constitute a *message* $\mathbf{m}_{ij}$ passed from the edge (i,j) to the node j[10]; it may be thought of as a message vector (indexed by μ). A message $\mathbf{m}_{ij}$ causes values $m_{ij:\mu}$ to be *swept out* of all the edges $\theta_{ij:\lambda\mu}$ and added to the nodes $\theta_{j:\mu}$. Messages are passed in both directions from an edge (i,j). An elementary reparametrization is illustrated in Figure 2.18.

[10]This is the same as the *message vector* passed from $i \to j$ in the min-sum message passing algorithm. However, thinking it as a message from the edge (i,j) to node j may be more intuitive.

In fact, reparametrization provides an alternative way of implementing the min-sum message passing algorithm, where the parameters $\boldsymbol{\theta}$ are updated directly based on the messages. Similarly to Eq. (2.81), the labelling can be obtained as

$$x'_{i:\lambda} = \operatorname*{argmin}_{\lambda \in \mathcal{L}} \theta'_{i:\lambda} \;, \tag{2.83}$$

where $E_{\boldsymbol{\theta}'} \equiv E_{\boldsymbol{\theta}}$. This suggests a lower bound of $E_{\boldsymbol{\theta}}(\mathbf{x})$, *i.e.*,

$$\sum_{i \in \mathcal{V}} \min_{\lambda \in \mathcal{L}} \theta'_{i:\lambda} \le E_{\boldsymbol{\theta}}(\mathbf{x}) \;. \tag{2.84}$$

Now, one can try to maximize this lower bound over all possible reparametrizations, while keeping the parameters non-negative (the parameters $\boldsymbol{\theta}$ are non-negative to start with). This leads to the following optimization problem,

$$\begin{aligned} \max_{\boldsymbol{\theta}'} & \sum_{i \in \mathcal{V}} \min_{\lambda \in \mathcal{L}} \theta'_{i:\lambda} \;, \\ \text{s.t.} \quad & E_{\boldsymbol{\theta}'} \equiv E_{\boldsymbol{\theta}} \;, \\ & \theta'_{ij:\lambda\mu} \ge 0 \quad \forall\, (i,j) \in \mathcal{E},\, \lambda, \mu \in \mathcal{L} \;. \end{aligned} \tag{2.85}$$

We refer to this problem as the *optimal message passing* problem. The interesting relationship between the above problem and the LP relaxation is characterized by the following theorem [Werner, 2010].

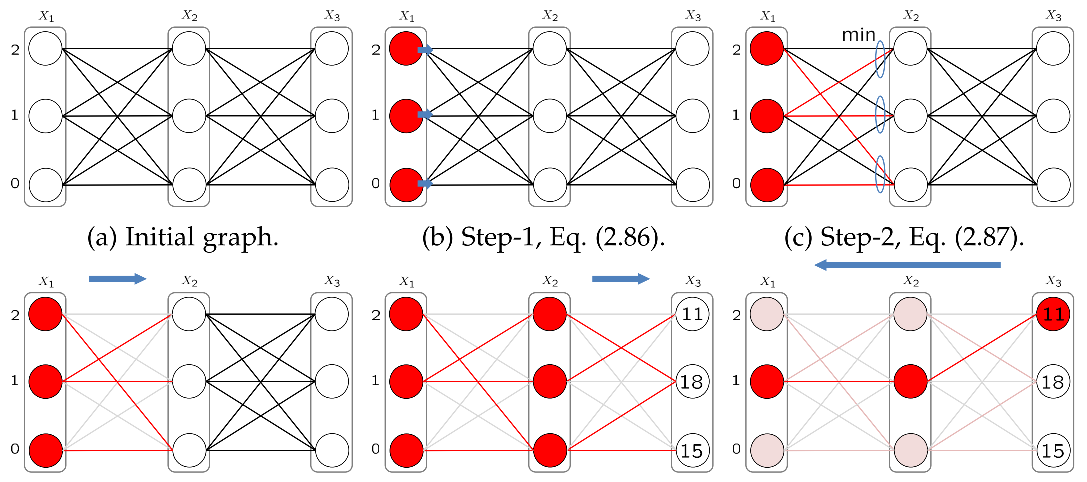
Theorem 2.3.2. *The LP relaxation over the local polytope* (2.78) *and the optimal message passing* (2.85) *are dual LP problems.*

Proof. The proof of this theorem is a direct application of Lagrange duality to the LP relaxation over the local polytope. See [Werner, 2010] for details. □

This means that the lower bound computed by optimizing either of these problems yields the same value. Therefore, reparametrization (or message passing) is a good candidate to optimize the LP relaxation. However, the order in which the messages should be passed is still not clear. In particular, an algorithm based on message passing that would optimize the LP relaxation would be ideal. To this end, let us first give an algorithm for MRFs defined on trees and then turn to the more general algorithm.

2.3.4.4 Optimality of Message Passing on Tree Structured MRFs

On tree structured MRFs, there is a natural order (from leaves to root) to pass the messages and in two passes (called *forward pass* and *backward pass*) the algorithm converges and yields the optimal labelling [Pearl, 1988]. In particular, for an arbitrarily chosen *root* node $r \in \mathcal{V}$, in the forward pass, the messages are passed (reparametrization) from *leaf* nodes to the root node. In the backward pass, the messages are passed from root to leaf nodes. This message passing algorithm can be better explained using the multi-label graph. See Figure 2.19.



(a) Initial graph. (b) Step-1, Eq. (2.86). (c) Step-2, Eq. (2.87).

(d) Message passing $X_1 \to X_2$. (e) Forward pass completed. (f) Backward pass completed.

Figure 2.19: *Message passing on a linear MRF with 3 nodes. Here, red indicates that the node or edge has zero weight. At the end of the forward pass, except for the root node (X_3), the weights on all other nodes are zero. In the backward pass, the optimal labelling* $\mathbf{x}^* = \{(0,1,0),(0,1,0),(0,0,1)\}$ *is recovered by backtracking through zero edges. Here, the cost of the optimal labelling* $E(\mathbf{x}^*) = 11$.

At any stage of the forward pass, consider a directed edge $(i,j) \in \mathcal{E}^+$ in the forward path. Let $\boldsymbol{\theta}'$ be the current parameters (reparametrized according to the messages passed until now). Note that the messages $\mathbf{m}_{ji}$ and $\mathbf{m}_{ij}$ are only used locally and are initialized to zero. There are two steps of reparametrization at each message passing step. In the first step, $\boldsymbol{\theta}'$ is reparametrized as follows,

$$\begin{aligned} m_{ji:\lambda} &\leftarrow -\theta'_{i:\lambda} && \forall\, \lambda \in \mathcal{L}\,, \\ \theta'_{ij:\lambda\mu} &\leftarrow \theta'_{ij:\lambda\mu} - m_{ji:\lambda} = \theta'_{ij:\lambda\mu} + \theta'_{i:\lambda} && \forall\, \lambda,\mu \in \mathcal{L}\,, \\ \theta'_{i:\lambda} &\leftarrow \theta'_{i:\lambda} + m_{ji:\lambda} = 0 && \forall\, \lambda \in \mathcal{L}\,. \end{aligned} \tag{2.86}$$

The second step of reparametrization is,

$$\begin{aligned} m_{ij:\mu} &\leftarrow \min_{\lambda \in \mathcal{L}} \theta'_{ij:\lambda\mu} && \forall\, \mu \in \mathcal{L}\,, \\ \theta'_{ij:\lambda\mu} &\leftarrow \theta'_{ij:\lambda\mu} - m_{ij:\mu} && \forall\, \lambda,\mu \in \mathcal{L}\,, \\ \theta'_{j:\mu} &\leftarrow \theta'_{j:\lambda} + m_{ij:\mu} && \forall\, \mu \in \mathcal{L}\,. \end{aligned} \tag{2.87}$$

Note that, after the first step, the unary parameters $\theta'_{i:\lambda} = 0$ for all $\lambda \in \mathcal{L}$. Furthermore, during the second step, the minimum edge weight is subtracted from all the edges $\theta'_{ij:\lambda\mu}$. This means that, after the second step, at least one of $\theta'_{ij:\lambda\mu}$ is zero for all $\mu \in \mathcal{L}$. Therefore, at the end of the forward pass, the resulting parameters satisfy the following conditions,

1. The weights on nodes $\theta'_{i:\lambda} = 0$, for all $i \in \mathcal{V} \setminus \{r\}$, where r is the root node and for all $\lambda \in \mathcal{L}$.

2. There exists $\lambda' \in \mathcal{L}$ such that $\theta'_{ij:\lambda'\mu} = 0$, for all $(i,j) \in \mathcal{E}$ and $\mu \in \mathcal{L}$. Here, λ' may be different for each edge (i,j) and label μ.

Now, it is evident that the total cost of the optimal labelling $\mathbf{x}^*$ can be computed as,

$$E_{\boldsymbol{\theta}}(\mathbf{x}^*) = E_{\boldsymbol{\theta}'}(\mathbf{x}^*) = \min_{\lambda \in \mathcal{L}} \theta'_{r:\lambda} \; . \tag{2.88}$$

Furthermore, the optimal labelling $\mathbf{x}^*$ is computed by back-tracking through the zero weight edges and nodes (*backward pass*). This two pass procedure is optimal for tree structured MRFs and, for a graph with loops, it is not guaranteed to find the optimum. In fact, it has been shown that, for multi-label submodular MRFs, one can find an ordering of optimal message passing that would guarantee optimality [Werner, 2007, 2010]. However, in general, the algorithm may not even converge. Despite that, the optimality on tree MRFs made it an ideal choice to tackle the LP relaxation, in a dual-decomposition framework [Wainwright et al., 2005; Komodakis et al., 2011].

2.3.4.5 Tree Reweighted Message Passing

Tree Reweighted Message Passing (TRW) [Wainwright et al., 2005] can be thought of as a dual-decomposition [Bertsekas, 1999] algorithm for minimizing the LP relaxation over the local polytope (2.78), where the dual subproblems are defined over tree structured MRFs.

Recall that the MRF graph is denoted by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Consider a collection of subtrees of $\mathcal{G}$, denoted by $\mathcal{T}_{\mathcal{G}}$. Let us denote the parameter vector of the original graph MRF with $\boldsymbol{\theta}_{\mathcal{G}}$ and the parameter vector for each tree $T \in \mathcal{T}_{\mathcal{G}}$ with $\boldsymbol{\theta}_T$.

Definition 2.3.7. A tree decomposition $T \in \mathcal{T}_{\mathcal{G}}$ is *valid* if the parameter vectors $\boldsymbol{\theta}_T$ satisfy

$$\sum_{T \in \mathcal{T}_{\mathcal{G}}} \boldsymbol{\theta}_T = \boldsymbol{\theta}_{\mathcal{G}} \; . \tag{2.89}$$

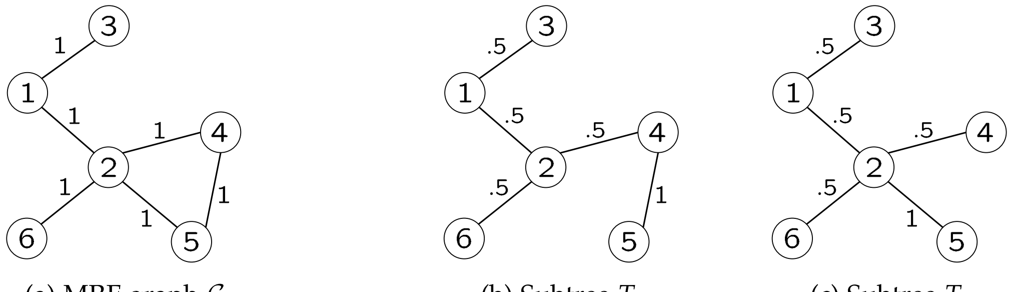
An example of a valid tree decomposition is shown in Figure 2.20. For a valid tree decomposition,

$$\langle \boldsymbol{\tau}, \boldsymbol{\theta}_{\mathcal{G}} \rangle = \sum_{T \in \mathcal{T}_{\mathcal{G}}} \langle \boldsymbol{\tau}, \boldsymbol{\theta}_T \rangle \; , \tag{2.90}$$

for all $\boldsymbol{\tau} \in \Omega$, where Ω is the marginal polytope (see Definition 2.3.4) of the original MRF. Now, the tree-based lower bound can be written as

$$\min_{\mathbf{x} \in \Gamma} E_{\boldsymbol{\theta}_{\mathcal{G}}}(\mathbf{x}) = \min_{\boldsymbol{\tau} \in \Omega} \langle \boldsymbol{\tau}, \boldsymbol{\theta}_{\mathcal{G}} \rangle \geq \sum_{T \in \mathcal{T}_{\mathcal{G}}} \min_{\boldsymbol{\tau} \in \Omega} \langle \boldsymbol{\tau}, \boldsymbol{\theta}_T \rangle \; . \tag{2.91}$$

The inequality is due to the fact, that each tree $T \in \mathcal{T}_{\mathcal{G}}$ is minimized independently and therefore the minimization is less constrained. Note that, once the minimization is completed, if the labelling (given $\boldsymbol{\tau}^*$, the corresponding labelling $\mathbf{x}^*$ can be



(a) MRF graph $\mathcal{G}$ (b) Subtree T_1 (c) Subtree T_2

Figure 2.20: *A valid tree decomposition, where each subtree is chosen to be a spanning tree. Here, the edge parameters (or weights) are shown next to the edges. Note that, all the nodes appear in both the subtrees, and the node parameters are divided equally in both the trees (not shown). For the edges, except the edges* $(2,5)$ *and* $(4,5)$, *all other edges appear in both the trees and each of them has* 0.5 *weight. The edges* $(2,5)$ *and* $(4,5)$ *retain their original weights in their respective trees.*

obtained) is consistent across all the trees, then the optimum is found. This situation is referred to as the *tree agreement* in [Wainwright et al., 2005]. However, for general MRFs, this is not the case, and therefore one tries to maximize this lower bound. This is the basic idea behind the TRW algorithm, and the corresponding optimization problem can be written as

$$\begin{aligned} \max_{\boldsymbol{\theta}_T} \sum_{T\in\mathcal{T}_\mathcal{G}} \min_{\boldsymbol{\tau}\in\Omega} \langle \boldsymbol{\tau}, \boldsymbol{\theta}_T \rangle \ , \\ \text{s.t.} \quad \sum_{T\in\mathcal{T}_\mathcal{G}} \boldsymbol{\theta}_T = \boldsymbol{\theta}_\mathcal{G} \ . \end{aligned} \tag{2.92}$$

Since subproblems $\min_{\boldsymbol{\tau}\in\Omega}\langle \boldsymbol{\tau}, \boldsymbol{\theta}_T \rangle$ are defined over trees, their optimal labellings can be obtained using the message passing algorithm discussed above. Once the subproblems are optimized, the tree parameters $\boldsymbol{\theta}_T$ are updated in the direction that maximizes the lower bound. Theorem 2.3.2 stated that the optimal message passing and the LP relaxation are dual problems. Similarly, the relationship of the TRW problem (2.92) and LP relaxation (2.78) can be stated as follows [Wainwright et al., 2005],

Theorem 2.3.3. *The tree reweighted message passing (TRW) problem* (2.92) *and the LP relaxation over the local polytope* (2.78) *are dual LP problems.*

Proof. This is also an application of the Lagrange dual of the LP relaxation (2.78). See [Wainwright et al., 2005] for a detailed proof. □

The term dual-decomposition is clear now, as the TRW problem (2.92) is the dual of the LP relaxation, which is decomposed into tree-based subproblems. An important consequence of this theorem is that any valid tree decomposition (see Definition 2.3.7) will yield the same bound on the optimal MRF energy. Furthermore, note that the maximum values of the TRW problem (2.92) and the optimal message passing problem (2.85) are the same since they are duals of the same LP problem.

After the introduction of the TRW algorithm in [Wainwright et al., 2005], many algorithms following the same idea have been introduced [Kolmogorov, 2006; Globerson and Jaakkola, 2008; Werner, 2010; Komodakis et al., 2011]. Among them, the sequential version of TRW, referred to as TRWS [Kolmogorov, 2006] is practically efficient and used widely in computer vision applications. Furthermore, Komadakis [Komodakis et al., 2011] identified this idea as a dual-decomposition technique, and proposed a more general algorithm, where the subproblems are not only restricted to tree MRFs. Since these algorithms optimize a subset of variables (subproblem) at a time, this family of algorithms are often referred to as *block-coordinate descent* methods in the literature.

Note that these methods efficiently optimize the LP relaxation by exploiting the sparse structure of the MRF. By contrast, in Chapter 5, we introduce a block-coordinate descent method for fully connected MRFs (or CRFs), for efficient optimization of the LP relaxation. In fact, the time complexity of one iteration of our algorithm is linear in the number of pixels and labels, which is the first LP minimization algorithm for dense CRFs to have linear time iterations.

2.3.5 Other Continuous Relaxations

In addition to the LP relaxation, there are many other continuous relaxations have been studied in the literature. This includes Quadratic Programming (QP) [Ravikumar and Lafferty, 2006], Second Order Cone Programming (SOCP) [Muramatsu and Suzuki, 2003] and Semi-Definite Programming (SDP) [Torr, 2003] relaxations. Among them, the LP relaxation was shown to provide a better approximation than a large class of QP and SOCP relaxations (see [Kumar et al., 2009]) and the SDP relaxation was shown to scale poorly [Olsson et al., 2007]. Furthermore, researchers have studied approaches to tighten the LP relaxation [Sontag et al., 2008; Komodakis and Paragios, 2008].

On the other hand, the Mean-Field (MF) method [Wainwright et al., 2008] can also be used to perform approximate inference on an MRF. In particular, the mean-field algorithm[11] approximates the true probability distribution $P(\mathbf{x})$ of the MRF (defined in Eq. (2.1)) by a *fully-factorized distribution*[12] $Q(\mathbf{x})$, by minimizing the KL-divergence [Kullback and Leibler, 1951] between them (denoted by $D_{KL}(Q\|P)$). Once the distribution Q is obtained, finding the most probable configuration is straight forward, since Q is a product of independent distributions defined over each pixel. In fact, the mean-field algorithm can also be thought of as a continuous relaxation method, which optimizes a linear objective function over a non-convex subset of the marginal polytope (Definition 2.3.4). We refer the interested reader to [Ravikumar and Lafferty, 2006; Wainwright et al., 2008] for more detail.

[11]This algorithm is referred to as naïve mean-field in [Wainwright et al., 2008].

[12]A distribution is *fully-factorized* if it is a product of independent distributions. In this case $Q(\mathbf{x}) = \prod_{i\in\mathcal{V}} Q_i(\mathbf{x}_i)$.

2.4 Summary

In this chapter, we have studied the important theories on Markov random fields and some state-of-the-art optimization algorithms. These algorithms can be categorized into three groups: 1) max-flow for submodular MRFs; 2) move-making algorithms; and 3) continuous relaxations. In the subsequent chapters, we describe our contributions in these three categories. First, in the next chapter, we discuss a memory efficient variant of the max-flow algorithm for multi-label submodular MRFs. This algorithm constitutes the method of choice for Ishikawa type graphs when the complete graph cannot be stored in memory. Next, in Chapter 4, we explain a move-making style algorithm for multi-label MRFs with a certain class of robust non-convex priors. This approach is effective in obtaining significantly lower energy solutions than other move-making algorithms for MRFs with such robust non-convex priors. Later, in Chapter 5, we describe a block-coordinate descent algorithm for the LP relaxation of fully connected CRFs. This constitutes the first LP relaxation algorithm for dense CRFs that has linear time iterations.

Chapter 3

Memory Efficient Max-Flow for Multi-Label Submodular MRFs

In this chapter, we introduce our memory efficient max-flow algorithm and show that the memory requirement reduces by $\mathcal{O}(\ell)$ (where ℓ is the number of labels) compared to the standard max-flow algorithm described in Chapter 2. Furthermore, we prove its polynomial time complexity and also discuss its relationship to the min-sum message passing algorithm (reviewed in Section 2.3.4.3). This chapter is based on our work [Ajanthan et al., 2016] with substantial extensions available in [Ajanthan et al., 2017c].

3.1 Introduction

As discussed in Section 2.3.2, Ishikawa [Ishikawa, 2003] introduced a max-flow-based method to globally minimize the energy of multi-label MRFs with convex edge terms. In [Schlesinger and Flach, 2006], this method was extended to energy functions satisfying the *multi-label submodularity* condition, analogous to the submodularity condition for MRFs with binary labels. In the general case, however, this method requires $2\,\ell^2$ directed edges for each pair of neighbouring variables. For instance, for a 1000×1000, 4-connected image with 256 labels, it would require approximately $1000 \times 1000 \times 2 \times 256^2 \times 2 \times 4 \approx 1000$ GB of memory to store the edges (assuming 4 bytes per edge). Clearly, this is beyond the storage capacity of most computers.

In this chapter, we introduce a variant of the max-flow algorithm that requires storing only two ℓ-dimensional vectors per variable pair instead of the $2\,\ell^2$ edge capacities of the standard max-flow algorithm. In the example discussed above, our algorithm would therefore use only 4 GB of memory for the edges. As a result, our approach lets us optimally solve much larger problems.

More specifically, in contrast to the usual augmenting path algorithm [Ford and Fulkerson, 1962], we do not store the residual edge capacities at each iteration. Instead, our algorithm records two ℓ-dimensional flow-related quantities for every pair of neighbouring variables. We show that, at any stage of the algorithm, the residual edge capacities can be computed from these flow-related quantities and the initial edge capacities. This, of course, assumes that the initial capacities can be computed by some memory-efficient routine, which is almost always the case in computer vision.

The optimality of Ishikawa's formalism made it a method of choice as a subroutine in many approximate energy minimization algorithms, such as multi-label moves [Torr and Kumar, 2009; Veksler, 2012]. Since our approach can simply replace the standard max-flow algorithm [Boykov and Kolmogorov, 2004] in Ishikawa-type graphs, it also allows us to minimize the energy of much larger non-submodular MRFs in such approximate techniques. Furthermore, due to the similarity to standard max-flow, our algorithm can easily be extended to handle dynamic MRFs [Kohli and Torr, 2005] and also be accelerated using the parallel max-flow technique [Strandmark and Kahl, 2010].

We demonstrate the effectiveness of our algorithm on the problems of stereo correspondence estimation and image inpainting. Our experimental evaluation shows that our method can solve much larger problems than standard max-flow on a standard computer and is an order of magnitude faster than state-of-the-art message-passing algorithms [Kolmogorov, 2006; Komodakis et al., 2011; Savchynskyy et al., 2012]. Our code is available at https://github.com/tajanthan/memf.

3.2 Preliminaries

Let us recall the energy function associated with a pairwise MRF (2.4),

$$E(\mathbf{x}) = \sum_{i\in\mathcal{V}} \theta_i(x_i) + \sum_{(i,j)\in\mathcal{E}} \theta_{ij}(x_i, x_j) \;, \tag{3.1}$$

where $x_i \in \mathcal{L}$ for all $i \in \mathcal{V}$. Here, $\mathcal{V}$ is the set of vertices, *e.g.*, corresponding to pixels or superpixels in an image, and $\mathcal{E}$ is the set of edges, *e.g.*, encoding a 4-connected or 8-connected grid over the image pixels.

In this chapter, we consider a pairwise MRF with an ordered label set $\mathcal{L} = \{0, 1, \cdots, \ell - 1\}$, and we assume that the pairwise terms satisfy the *multi-label submodularity* condition (Definition 2.3.1):

$$\theta_{ij}(\lambda', \mu) + \theta_{ij}(\lambda, \mu') - \theta_{ij}(\lambda, \mu) - \theta_{ij}(\lambda', \mu') \geq 0 \;, \tag{3.2}$$

for all $\lambda, \lambda', \mu, \mu' \in \mathcal{L}$, where $\lambda < \lambda'$ and $\mu < \mu'$. Furthermore, we assume that the pairwise potentials can be computed either by some routine or can be stored in an efficient manner. In other words, we assume that we do not need to store each individual pairwise term. Note that, in computer vision, this comes at virtually no loss of generality.

3.2.1 The Ishikawa Graph

We briefly revisit the Ishikawa algorithm and more details can be found in Section 2.3.2. Ishikawa [Ishikawa, 2003] introduced a method to represent the multi-label energy function (3.1) in a graph. The basic idea behind the Ishikawa construction is

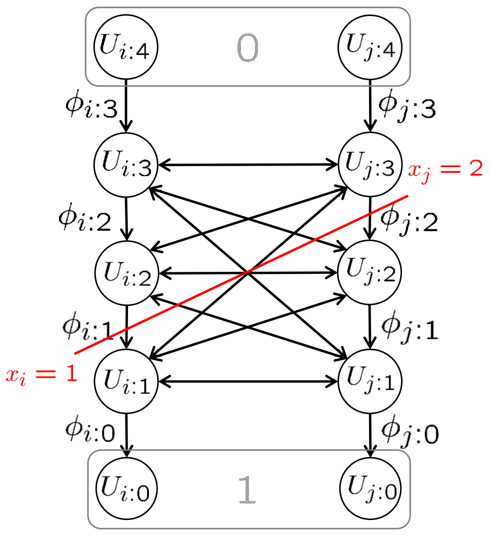


Figure 3.1: *Example of an Ishikawa graph. The graph incorporates edges with infinite capacity from $U_{i:\lambda}$ to $U_{i:\lambda+1}$, not shown in the graph. Here the cut corresponds to the labeling* $\mathbf{x} = \{1, 2\}$ *where the label set* $\mathcal{L} = \{0, 1, 2, 3\}$.

to encode the label $X_i = x_i$ of a vertex $i \in \mathcal{V}$ using binary-valued random variables $U_{i:\lambda}$, one for each label $\lambda \in \mathcal{L}$. In particular, the encoding is defined as $u_{i:\lambda} = 1$ if and only if $x_i \geq \lambda$, and 0 otherwise. The Ishikawa graph is then an *st*-graph $\hat{\mathcal{G}} = (\hat{\mathcal{V}} \cup \{0, 1\}, \hat{\mathcal{E}}^+, \boldsymbol{\phi})$, where the set of nodes and the set of edges are defined as follows,

$$\begin{aligned} \hat{\mathcal{V}} &= \{U_{i:\lambda} \mid i \in \mathcal{V}, \lambda \in \{1, \cdots, \ell-1\}\} , \\ \hat{\mathcal{E}}^+ &= \hat{\mathcal{E}}^+_v \cup \hat{\mathcal{E}}^+_c , \\ \hat{\mathcal{E}}^+_v &= \{(U_{i:\lambda}, U_{i:\lambda\pm1}) \mid i \in \mathcal{V}, \lambda \in \{1, \cdots, \ell-1\}\} , \\ \hat{\mathcal{E}}^+_c &= \{(U_{i:\lambda}, U_{j:\mu}), (U_{j:\mu}, U_{i:\lambda}) \mid (i,j) \in \mathcal{E}, U_{i:\lambda}, U_{j:\mu} \in \hat{\mathcal{V}}\} , \end{aligned} \tag{3.3}$$

where $\hat{\mathcal{E}}^+_v$ is the set of vertical edges and $\hat{\mathcal{E}}^+_c$ is the set of cross edges. Note that the directed edges are denoted with $\square^+$. Furthermore, the nodes $U_{i:\ell}$ and $U_{i:0}$ are identified as node 0 and node 1, respectively. We denote the Ishikawa edges by $e_{ij:\lambda\mu} \in \hat{\mathcal{E}}^+$ and their capacities by $\phi_{ij:\lambda\mu}$. We also denote by $e_{i:\lambda}$ the downward edge $(U_{i:\lambda+1}, U_{i:\lambda})$. An example of an Ishikawa graph is shown in Figure 3.1.

Note that, by construction of the Ishikawa graph, the capacities $\boldsymbol{\phi}$ and the energy parameters $\boldsymbol{\theta}$ are related according to the following formula:

$$\begin{aligned} \theta_i(\lambda) &= \phi_{ii:\lambda+1\lambda} = \phi_{i:\lambda} , \\ \theta_{ij}(\lambda, \mu) &= \sum_{\substack{\lambda' > \lambda \\ \mu' \leq \mu}} \phi_{ij:\lambda'\mu'} + \sum_{\substack{\lambda' \leq \lambda \\ \mu' > \mu}} \phi_{ji:\mu'\lambda'} . \end{aligned} \tag{3.4}$$

Finding the minimum energy labeling is a min-cut problem, which can be solved optimally using the max-flow algorithm [Ford and Fulkerson, 1962] when the edge capacities are non-negative. A multi-label submodular energy function can be represented by an Ishikawa graph with non-negative edge capacities $\boldsymbol{\phi}$ and can therefore be minimized optimally by max-flow.

3.2.2 Max-Flow

As discussed in Section 2.2.4, the most popular max-flow algorithm in computer vision [Boykov and Kolmogorov, 2004] is an augmenting path algorithm that finds a path from node 0 to node 1 through positive edges (called an *augmenting path*) and then pushes the maximum flow without exceeding the edge capacities (called *augmentation*). The augmentation operation changes the edge capacities in the graph, and therefore, the residual graph needs to be stored. That is, when applied to the Ishikawa graph, the max-flow algorithm stores $2\,\ell^2$ values per pair of neighbouring variables. For large number of labels and of variables, the memory requirement is high and, in many practical problems, exceeds the capacity of most computers.

3.2.3 Our Idea

Let us assume that the max-flow algorithm is applied to the Ishikawa graph. As the algorithm proceeds, the capacities on the edges in the graph change in response to the flow. Here, instead of storing the residual graph, we propose recording the flow that has been applied to the graph.

However, since storing the flow would also require $2\,\ell^2$ values per variable pair, we propose recording two ℓ-dimensional quantities related to the flow between pair of variables. More precisely, for each directed edge[1] $(i,j) \in \mathcal{E}^+$, we record the sum of outgoing flows from each node $U_{i:\lambda}$ to the nodes $U_{j:\mu}$ for all $\mu \in \{1,\cdots,\ell-1\}$. We call this quantity an *exit-flow*, denoted by $\Sigma_{ij:\lambda}$ (defined below in Eq. (3.6)). We show that these exit-flows allow us to reconstruct a *permissible* flow, which in turn lets us compute the residual edge capacities from the initial ones. Importantly, while flow reconstruction is not unique, we show that all such reconstructions are equivalent up to a *null* flow (Definition 3.3.5), which does not affect the energy function. Note that this idea can be applied to any augmenting path algorithm, as long as the residual graph can be rapidly constructed.

For increased efficiency, we then show how finding an augmenting path can be achieved in a simplified Ishikawa graph that amalgamates the nodes in each column into blocks. We then perform augmentation, which translates to updating our exit-flows, in this simplified graph. As a side effect, since an augmenting path in our simplified graph corresponds to a collection of augmenting paths in the Ishikawa graph, our algorithm converges in fewer iterations than the standard max-flow implementation of [Boykov and Kolmogorov, 2004].

[1] $\mathcal{E}^+$ denotes the set of directed edges between the vertices in the MRF, *i.e.*, if $(i,j) \in \mathcal{E}$ then, $(i,j) \in \mathcal{E}^+$ and $(j,i) \in \mathcal{E}^+$.

3.3 Memory Efficient Flow Encoding

Before we introduce our memory efficient max flow algorithm, let us describe how the cumulative flow can be stored in a memory efficient manner. This technique can be used in any augmenting path flow algorithm, by reconstructing the residual edge capacities whenever needed.

To this end, let us first recall the definition of a flow and then turn to our memory efficient flow encoding technique.

Definition 3.3.1. A flow is a mapping $\boldsymbol{\psi} : \hat{\mathcal{E}} \to \mathbb{R}$, denoted by $\psi_{ij:\lambda\mu}$ for the edges $e_{ij:\lambda\mu}$, that satisfies the *anti-symmetry* condition $\psi_{ij:\lambda\mu} = -\psi_{ji:\mu\lambda}$ for all $e_{ij:\lambda\mu} \in \hat{\mathcal{E}}$.

A flow is called *conservative* if the total flow into a node is zero for all nodes, except for the source and the terminal, *i.e.*,

$$\sum_{j,\mu | e_{ji:\mu\lambda} \in \hat{\mathcal{E}}^+} \psi_{ji:\mu\lambda} = 0 \quad \forall\, U_{i:\lambda} \in \hat{\mathcal{V}}\ . \tag{3.5}$$

Given $\boldsymbol{\psi}$, the residual capacities of the Ishikawa graph are updated as $\boldsymbol{\phi} = \boldsymbol{\phi}^0 - \boldsymbol{\psi}$, where $\boldsymbol{\phi}^0$ represents the initial edge capacities. Furthermore, we call the flow restricted to each column *column-flows*, which we denote by $\psi_{i:\lambda}\,;\, i \in \mathcal{V}, \lambda \in \mathcal{L}$.

At first sight, it might seem that, to apply the max-flow algorithm, it is necessary to keep track of all the values $\psi_{ij:\lambda\mu}$, which would require the same order of storage as recording all the edge capacities. Below, however, we show that it is necessary to store only $\mathcal{O}(\ell)$ values for each $(i,j) \in \mathcal{E}$, instead of $\mathcal{O}(\ell^2)$.

To this end, the flow values that we store in our algorithm, namely *source-flows* and *exit-flows* are defined below.

Definition 3.3.2. For each $i \in \mathcal{V}$, the flow out from the source node $\psi_{i:\ell-1}$ is called a *source-flow*.

Definition 3.3.3. For each $(i,j) \in \mathcal{E}^+$ and $\lambda \in \{1, \cdots, \ell-1\}$, we define an *exit-flow* as

$$\Sigma_{ij:\lambda} = \sum_{\mu} \psi_{ij:\lambda\mu}\ . \tag{3.6}$$

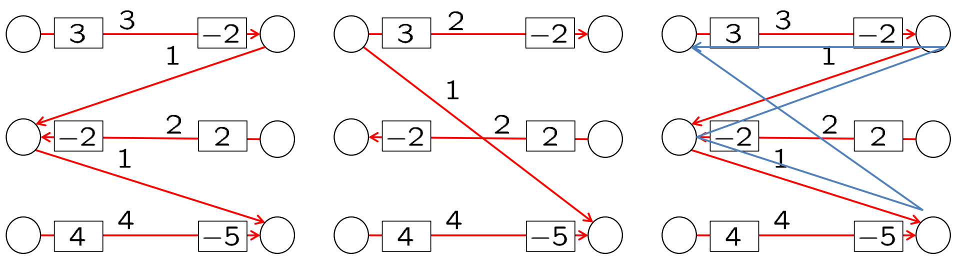
We will show that these source-flows and exit-flows permit the flow $\boldsymbol{\psi}$ to be reconstructed up to equivalence.

Now, let us define some additional properties of flow, which will be useful in our exposition.

Definition 3.3.4. A flow ψ is called *permissible* if $\phi^0_{ij:\lambda\mu} - \psi_{ij:\lambda\mu} \geq 0$ for all $e_{ij:\lambda\mu} \in \hat{\mathcal{E}}^+$.

Definition 3.3.5. A flow ψ is called *null* if the total flow into a node is zero for all nodes including the source and the terminal, *i.e.*, satisfies Eq. (3.5) for all $U_{i:\lambda} \in \hat{\mathcal{V}} \cup \{0, 1\}$.

Note that a null flow does not change the energy function represented by the *st*-graph and it is identical to passing flow around loops. Also, if $\boldsymbol{\psi}$ is a null flow then so is $-\boldsymbol{\psi}$.



(a) $\boldsymbol{\psi}$ (b) $\boldsymbol{\psi}'$ (c) $\boldsymbol{\psi} \equiv \boldsymbol{\psi}'$

Figure 3.2: *An example of two equivalent flow representations with the same exit-flows. Note that each red arrow represents the value $\psi_{ij:\lambda\mu}$ and the opposite arrows $\psi_{ji:\mu\lambda}$ are not shown. Furthermore, the exit-flows Σ are shown next to the nodes and the initial edges $\boldsymbol{\phi}^0$ are not shown. In* **(c)***, the flow $\boldsymbol{\psi}'$ is obtained from $\boldsymbol{\psi}$ by passing flow around a loop.*

Furthermore, and as discussed in Section 2.2.3 the energy function encoded by an *st*-graph is a quadratic pseudo-boolean function [Boros and Hammer, 2002], and a *reparametrization* of such a function is identical to a null flow in the corresponding *st*-graph.

Lemma 3.3.1. *Two sets of capacities $\boldsymbol{\phi}$ and $\boldsymbol{\phi}'$ represent the same energy function exactly (not up to a constant), written as $E_{\boldsymbol{\phi}} \equiv E_{\boldsymbol{\phi}'}$, if and only if $\boldsymbol{\phi}' - \boldsymbol{\phi}$ is a null flow.*

Proof. A null flow can be shown to be equivalent to reparametrizing the pseudo-boolean function using the identity $\bar{y}_i = 1 - y_i$ (where $y_i, \bar{y}_i \in \{0, 1\}$ and $\bar{y}_i$ is the complement of y_i). A brief overview of pseudo-boolean optimization is given in Section 2.2. We refer the interested reader to [Boros and Hammer, 2002] for more detail.

In fact, this lemma is a restatement of the reparametrization lemma (Lemma 2.3.4) in the context of *st*-graphs. □

Let $\boldsymbol{\phi}$ and $\boldsymbol{\phi}'$ be two sets of residual capacities obtained from an initial set of capacities $\boldsymbol{\phi}^0$ by passing two flows $\boldsymbol{\psi}$ and $\boldsymbol{\psi}'$, *i.e.*, $\boldsymbol{\phi} = \boldsymbol{\phi}^0 - \boldsymbol{\psi}$ and $\boldsymbol{\phi}' = \boldsymbol{\phi}^0 - \boldsymbol{\psi}'$. If $\boldsymbol{\phi}$ and $\boldsymbol{\phi}'$ are equivalent, then, by Lemma 3.3.1, $(\boldsymbol{\phi}^0 - \boldsymbol{\psi}) - (\boldsymbol{\phi}^0 - \boldsymbol{\psi}') = \boldsymbol{\psi}' - \boldsymbol{\psi}$ is a null flow. Hence $\boldsymbol{\psi}'$ can be obtained from $\boldsymbol{\psi}$ by passing flow around loops in the graph. See Figure 3.2.

We can now state our main theorem.

Theorem 3.3.1. *Let $\boldsymbol{\phi}^0$ be the initial capacities of an Ishikawa graph, and let Σ be a set of exit-flows. Suppose that $\boldsymbol{\psi}$ and $\boldsymbol{\psi}'$ are two conservative flows with identical source-flows, then $E_{\boldsymbol{\phi}^0 - \boldsymbol{\psi}} \equiv E_{\boldsymbol{\phi}^0 - \boldsymbol{\psi}'}$. Furthermore, if both the flows are compatible with Σ, meaning that (3.6) holds for both $\boldsymbol{\psi}$ and $\boldsymbol{\psi}'$, then $\boldsymbol{\psi}$ and $\boldsymbol{\psi}'$ have identical column-flows.*

The idea is then as follows. If a permissible conservative flow $\boldsymbol{\psi}$ is obtained during an augmenting path flow algorithm, but only the exit-flows $\Sigma_{ij:\lambda}$ are retained for each $(i, j) \in \mathcal{E}^+$ and label λ, then one wishes, when required, to reconstruct the

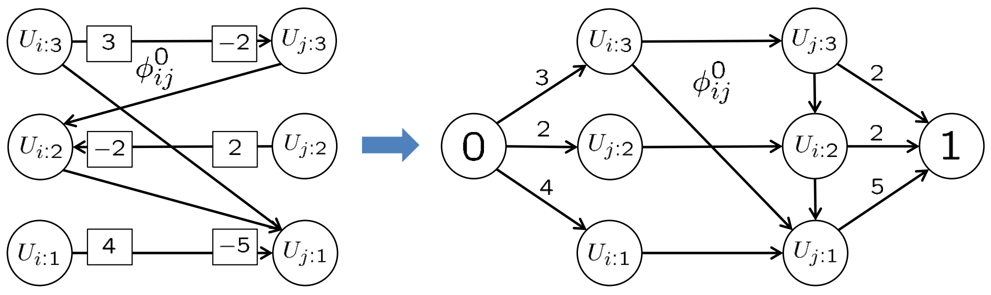


Figure 3.3: *Given $\boldsymbol{\phi}^0$ and Σ (**left**), flow reconstruction is formulated as a max-flow problem (**right**). Here the nodes with positive exit-flows are connected to the source (0) and those with negative exit-flows are connected to the terminal (1).*

flow $\boldsymbol{\psi}$ on a given edge $(i,j) \in \mathcal{E}$. Although the reconstructed flow $\boldsymbol{\psi}'$ may not be identical with the flow $\boldsymbol{\psi}$, the two will result in equivalent energy functions (not just equal up to a constant, but exactly equal for all assignments). In the augmenting path algorithm, the current flow values are only needed temporarily, one edge at a time, to find a new augmenting path, and hence do not need to be stored, as long as they can be rapidly computed.

Now we prove Theorem 3.3.1.

Proof. First we prove the equivalence. Note that, if two conservative flows $\boldsymbol{\psi}$ and $\boldsymbol{\psi}'$ have identical source-flows, then from Definition 3.3.5, $\boldsymbol{\psi}' - \boldsymbol{\psi} = (\boldsymbol{\phi}^0 - \boldsymbol{\psi}) - (\boldsymbol{\phi}^0 - \boldsymbol{\psi}')$ is a null-flow. Therefore, from Lemma 3.3.1, $E_{\boldsymbol{\phi}^0 - \boldsymbol{\psi}} \equiv E_{\boldsymbol{\phi}^0 - \boldsymbol{\psi}'}$.

Now we prove that $\boldsymbol{\psi}$ and $\boldsymbol{\psi}'$ have identical column-flows. For a conservative flow

$$\psi_{i:\lambda} - \left(\psi_{i:\lambda-1} + \sum_{(i,j)\in\mathcal{E}^+} \Sigma_{ij:\lambda}\right) = 0\,, \tag{3.7}$$

for all $i \in \mathcal{V}$ and $\lambda \in \{1, \ldots, \ell - 1\}$. Since $\boldsymbol{\psi}$ and $\boldsymbol{\psi}'$ are compatible with Σ and have identical source-flows, $\psi_{i:\lambda} = \psi'_{i:\lambda}$ for all $i \in \mathcal{V}$ and $\lambda = \mathcal{L}$. Hence they have identical column-flows. □

3.3.1 Flow Reconstruction

Note that, from Eq. (3.7) it is clear that given the source-flows $\psi_{i:\ell-1}\,;\, i \in \mathcal{V}$, the column-flows $\psi_{i:\lambda}\,;\, i \in \mathcal{V}, \lambda \in \mathcal{L}$ can be computed in a top-down fashion. Now we turn to the problem of finding the flows along the cross-edges $e_{ij:\lambda\mu}$.

Given the set of exit-flows Σ, the objective is to find a permissible flow $\boldsymbol{\psi}'$ satisfying Eq. (3.6). Note that there exists a permissible conservative flow $\boldsymbol{\psi}$ compatible with Σ and hence we find $\boldsymbol{\psi}'$ such that $\boldsymbol{\psi}' - \boldsymbol{\psi}$ is a null flow. We do this by considering one edge $(i,j) \in \mathcal{E}$ at a time and reconstruct the flow by formulating a small max-flow problem.

Considering all the nodes $U_{i:\lambda}$ and $U_{j:\mu}$ for a given pair (i, j), we join them with edges with initial capacities $\phi^0_{ij:\lambda\mu}$. Nodes with positive exit-flow $\Sigma_{ij:\lambda}$ are joined to the source with edges of capacities $|\Sigma_{ij:\lambda}|$. Similarly, those with negative exit-flow are joined to the terminal. See Figure 3.3.

Note that, in this network, the edges from the source can be thought of as "supply" and the edges to the terminal can be thought of as "demand". Since the total supply equals the total demand in this network and there exists a permissible flow $\boldsymbol{\psi}_{ij}$ compatible with Σ (*i.e.*, satisfying the supply-demand equality), the maximum flow solution of this network $\boldsymbol{\psi}'_{ij}$ is compatible with Σ, *i.e.*, satisfies Eq. (3.6). In fact we are interested in non-negative residual capacities $\boldsymbol{\phi}'_{ij} = \boldsymbol{\phi}^0_{ij} - \boldsymbol{\psi}'_{ij}$ which are readily available in this network.

This problem can be solved using a greedy augmenting path algorithm. While this graph has $\mathcal{O}(\ell)$ nodes and $\mathcal{O}(\ell^2)$ edges, this remains perfectly tractable, since we only consider one edge (i, j) at a time. Therefore, ultimately, flow reconstruction can be done efficiently.

At this point, given the initial capacities $\boldsymbol{\phi}^0$, the source-flows $\psi_{i:\ell-1}\,; i \in \mathcal{V}$ and the set of exit-flows Σ, we have shown how to reconstruct the non-negative residual edge capacities $\boldsymbol{\phi}'$. This requires $\mathcal{O}(n + m\,\ell)$ values to be stored, where $n = |\mathcal{V}|$, $m = |\mathcal{E}|$ and $\ell = |\mathcal{L}|$.

3.4 Polynomial Time Memory Efficient Max-Flow

We now introduce our polynomial time memory efficient max flow algorithm, which minimizes multi-label submodular MRF energies with pairwise interactions. Our algorithm follows a similar procedure as the standard Edmonds-Karp algorithm [Edmonds and Karp, 1972], in that it iteratively finds the shortest augmenting path and then pushes the maximum flow through it without exceeding the edge capacities. However, instead of storing the residual graph, we store exit-flows as proposed in Section 3.3, which, at any stage of the algorithm, would allow us to compute the residual graph. Below, we discuss how one can find an augmenting path and update the exit-flows, *i.e.*, perform augmentation, without storing the full Ishikawa graph.

3.4.1 Finding an Augmenting Path

Our algorithm finds an augmenting path in a subgraph of the Ishikawa graph, called *lower-graph*. In particular, the lower-graph contains only a subset of Ishikawa edges which satisfy the *lowest-cross-edge* property.

Definition 3.4.1. Consider a directed edge $(i, j) \in \mathcal{E}^+$. For each node $U_{i:\lambda}$, the *lowest-cross-edge* is defined as, the edge $e_{ij:\lambda\mu}$ where μ is the smallest value such that $\phi_{ij:\lambda\mu} > 0$.

More specifically, in addition to the vertical edges $\hat{\mathcal{E}}^+_v$, the lower-graph contains the lowest-cross-edges. Therefore, we only store $\mathcal{O}(\ell)$ edges per variable pair (i, j).

Now, the relationship between augmenting paths in the original Ishikawa graph and the lower-graph can be characterized by the following theorem.

Theorem 3.4.1. *Given the Ishikawa graph, there is an augmenting path in the lower-graph if and only if there exists an augmenting path in the Ishikawa graph.*

Proof. Since the lower-graph is a subgraph of the Ishikawa graph, if there is an augmenting path in the lower-graph, then there exists an augmenting path in the Ishikawa graph.

We will now prove the converse. Consider a directed edge $(i, j) \in \mathcal{E}^+$. Let $e_{ij:\lambda\mu}$ and $e_{ij:\lambda\mu'}$ be two positive capacity edges from $U_{i:\lambda}$ and $e_{ij:\lambda\mu'}$ be the lowest-cross-edge. Then, due to the upward infinite capacity edges from $U_{j:\mu'} \rightsquigarrow U_{j:\mu}$, there is a positive capacity path from $U_{i:\lambda} \rightsquigarrow U_{j:\mu}$ through the lowest-cross-edge $e_{ij:\lambda\mu'}$. This proves the theorem. $\square$

This enables us to find all the augmenting paths in the Ishikawa graph by searching in a smaller graph that has $\mathcal{O}(\ell)$ edges per variable pair (i, j).

Note that, as mentioned earlier, we find the *shortest* augmenting path in this lower-graph. However, in contrast to the Edmonds-Karp algorithm, the path distance is computed considering *zero* distance for the infinite capacity edges and unit distance for other edges, instead of unit distance for all the edges. The intuition for this modification is that the infinite capacity edges will never become saturated (or eliminated from the graph) for the entire course of the algorithm. Note that, with this definition of path distance, the augmenting paths in both lower-graph and Ishikawa graph have the same length. This will enable us to prove the polynomial time bound of our algorithm in a similar manner as the standard Edmonds-Karp algorithm. Note that, even in this case, the shortest augmenting path can be found using a Breadth First Search (BFS) scheme.

3.4.2 Augmentation

Now, given an augmenting path p, we want to push the maximum permissible flow through it. The edges in the augmenting path p are updated in the same manner as in the usual max-flow algorithm. In addition to that, for each cross edge $e_{ij:\lambda\mu} \in \hat{\mathcal{E}}_c^+$ that is in the augmenting path, the exit-flows are updated as follows,

$$\begin{aligned} \Sigma_{ij:\lambda} &= \Sigma_{ij:\lambda} + \alpha \ , \\ \Sigma_{ji:\mu} &= \Sigma_{ji:\mu} - \alpha \ , \end{aligned} \tag{3.8}$$

where α is the maximum possible flow along the path p.

After the flow augmentation, the lower-graph needs to be updated to maintain the lowest-cross-edge property. Note that the lowest-cross-edge property may be violated due to the following reasons:

1. A new lowest-cross-edge $e_{ij:\lambda\mu}$ is created due to a flow along the edge $e_{ji:\mu\lambda}$.

Algorithm 3.1 Memory Efficient Max-Flow (MEMF) - Polynomial Time Version

```
Require: $\boldsymbol{\phi}^0$                                   ▷ Initial Ishikawa capacities
  $\Sigma \leftarrow 0$                                          ▷ Initialize exit-flows
  $\bar{\boldsymbol{\phi}} \leftarrow$ lower-graph($\boldsymbol{\phi}^0$)        ▷ Store the lowest-cross-edges
  repeat
    $p \leftarrow$ shortest-augmenting-path($\bar{\boldsymbol{\phi}}$)            ▷ Section 3.4.1
    $(\bar{\boldsymbol{\phi}}, \Sigma) \leftarrow$ augment($p, \bar{\boldsymbol{\phi}}$)          ▷ Section 3.4.2
    for each edge $e_{ij:\lambda\mu}$ becomes saturated do
      $\boldsymbol{\phi}_{ij} \leftarrow$ compute-edges($\boldsymbol{\phi}^0, \Sigma, i, j$)      ▷ Section 3.3.1
      $\bar{\boldsymbol{\phi}}_{ij} \leftarrow$ lower-graph($\boldsymbol{\phi}_{ij}, i, j$)          ▷ Section 3.4.1
    end for
  until no augmenting paths possible
  return get-labelling($\bar{\boldsymbol{\phi}}$)                 ▷ Find the cut using BFS
```

2. A new lowest-cross-edge $e_{ij:\lambda\mu}$ is created due to a saturating flow along $e_{ij:\lambda\mu'}$ for some $\mu' < \mu$, *i.e.*, the edge $e_{ij:\lambda\mu'}$ disappears from the Ishikawa graph.

Note that, during an augmentation, if a new lowest-cross-edge is created due to a flow in the opposite direction (case-1 above), then the new lowest-cross-edge is known and the lower-graph can be updated directly, *i.e.*, the new lowest-cross-edge can be stored.

On the other hand, if a cross edge becomes saturated (case-2), then we need to run the flow reconstruction algorithm to find the new lowest-cross-edge and update the lower-graph. This can be done in a memory efficient manner, since it only involves one edge $(i, j) \in \mathcal{E}$ at a time.

3.4.3 Summary

Our memory efficient max flow is summarized in Algorithm 3.1. Let us briefly explain the subroutines below.

lower-graph. Given the initial Ishikawa edge capacities $\boldsymbol{\phi}^0$, this subroutine constructs the lower-graph (with edge capacities $\bar{\boldsymbol{\phi}}$) by retaining the lowest-cross-edges from each node $U_{i:\lambda} \in \hat{\mathcal{V}}$, for each directed edge $(i, j) \in \mathcal{E}^+$ (see Section 3.4.1). If the input to this subroutine is the Ishikawa capacities $\boldsymbol{\phi}_{ij}$ corresponding to the edge $(i, j) \in \mathcal{E}$, then it retains the lowest-cross-edges $\bar{\boldsymbol{\phi}}_{ij}$.

shortest-augmenting-path. Given the lower-graph parameters $\bar{\boldsymbol{\phi}}$, this subroutine finds the shortest augmenting p using BFS, as discussed in Section 3.4.1.

augment. Given the path p, this subroutine finds the maximum possible flow through the path and updates the lower-graph and the set of exit-flows, as discussed in Section 3.4.2. In addition, if a new lowest-cross-edge is created due to a flow in the opposite direction (case-1 in Section 3.4.2), then it also updates the lower-graph capacities $\bar{\boldsymbol{\phi}}$.

compute-edges. Given the initial Ishikawa edge capacities $\boldsymbol{\phi}^0$ and the set of exit-flows Σ, this subroutine computes the non-negative residual Ishikawa capacities $\boldsymbol{\phi}_{ij}$ corresponding to the given edge (i, j). This is accomplished by solving a small max-flow problem (see Section 3.3.1).

get-labelling. This subroutines finds the partition of the lower-graph by running BFS.

As discussed above, the exit-flows Σ require $\mathcal{O}(\ell)$ storage for each edge $(i, j) \in \mathcal{E}$. In addition, the lower-graph can have at most $\mathcal{O}(n\,\ell)$ nodes and $\mathcal{O}(m\,\ell)$ edges. Furthermore, recall that we assume that the initial Ishikawa edge capacities $\boldsymbol{\phi}^0$ can be stored efficiently. Therefore, ultimately, the space complexity of our algorithm is $\mathcal{O}((n + m)\,\ell) = \mathcal{O}(m\,\ell)$. Let us now prove the polynomial time bound of our algorithm.

3.4.4 Time Complexity Analysis

We follow the time complexity analysis given in [Cormen et al., 2001] for the standard Edmonds-Karp algorithm to derive a polynomial time bound on our algorithm. In particular, the analysis first proves that the shortest path distance from the source (node 0) to any node is monotonically increasing with each flow augmentation. Then, it derives a bound on the number of augmentations. In fact, the number of augmentations of our MEMF algorithm also has the same bound as the Edmonds-Karp algorithm.

Let us denote the Ishikawa graph by $\hat{\mathcal{G}} = (\hat{\mathcal{V}} \cup \{0, 1\}, \hat{\mathcal{E}})$ and the lower-graph by $\hat{\mathcal{G}}^s = (\hat{\mathcal{V}} \cup \{0, 1\}, \hat{\mathcal{E}}^s)$[2]. In this section, we denote the nodes with u, v, etc. The notation $u_1 \leq u$ means that the node u_1 and u are in the same column where u_1 is below u. Let $\hat{\mathcal{G}}^s_f$ denote the residual graph of the lower-graph after the flow f and similarly $\hat{\mathcal{E}}^s_f$ denotes the set of non-zero residual edges. Let $d_f(u, v)$ denote the shortest path distance from u to v calculated by MEMF (Algorithm 3.1).

Lemma 3.4.1. *If the MEMF algorithm (Algorithm 3.1) is run on the Ishikawa graph $\hat{\mathcal{G}} = (\hat{\mathcal{V}} \cup \{0, 1\}, \hat{\mathcal{E}})$ with source 0 and terminal 1, then for any node $v \in \hat{\mathcal{V}}$, the shortest path distance $d_f(0, v)$ in the residual lower-graph $\hat{\mathcal{G}}^s_f$ increases monotonically with each flow augmentation.*

Proof. We suppose that, for some node $v \in \hat{\mathcal{V}}$, there is a flow augmentation that causes the shortest path distance from 0 to v to decrease, and then we derive a

[2]For simplified notation, we removed the capacity argument for the *st*-graph and the superscript $+$ for the set of edges. However, the sets $\hat{\mathcal{E}}$ and $\hat{\mathcal{E}}^s$ contains directed edges. The superscript s in $\hat{\mathcal{E}}^s$ is used to restate the fact, that the lower-graph is a subgraph of the Ishikawa graph.

contradiction. Let f be the flow just before the first augmentation that decreases some shortest path distance, and let f' be the flow just afterward. Let v be the node with the minimum $d_{f'}(0,v)$ whose distance was decreased by the augmentation, so that $d_{f'}(0,v) < d_f(0,v)$. Let $p = 0 \rightsquigarrow u \rightarrow v$ be a shortest path from 0 to v in $\hat{\mathcal{G}}^s_{f'}$, so that $(u,v) \in \hat{\mathcal{E}}^s_{f'}$ and

$$d_{f'}(0,v) = \begin{cases} d_{f'}(0,u) & \text{if } u < v \text{ (infinite edge)} \\ d_{f'}(0,u) + 1 & \text{otherwise .} \end{cases} \tag{3.9}$$

Because of how we chose v, we know that the distance of node u from the source 0 did not decrease, *i.e.*,

$$d_{f'}(0,u) \geq d_f(0,u) \ . \tag{3.10}$$

We claim that $(u,v) \notin \hat{\mathcal{E}}^s_f$. Why? If we had $(u,v) \in \hat{\mathcal{E}}^s_f$, then we would also have

$$\begin{aligned} d_f(0,v) &\leq d_f(0,u) + 1 \ , \\ &\leq d_{f'}(0,u) + 1 \ , \\ &= d_{f'}(0,v) \ , \end{aligned} \tag{3.11}$$

which contradicts our assumption that $d_{f'}(0,v) < d_f(0,v)$. The above argument simply follows even if (u,v) is an infinite capacity edge. Hence $(u,v) \notin \hat{\mathcal{E}}^s_f$.

How can we have $(u,v) \notin \hat{\mathcal{E}}^s_f$ and $(u,v) \in \hat{\mathcal{E}}^s_{f'}$? Note that, in this case, (u,v) cannot be an infinite capacity edge. There can be two reasons:

1. A new lowest edge (u,v) is created due to the flow from v to u. This means that the augmentation must have increased the flow from v to u. The MEMF algorithm always augments flow along shortest paths, and therefore the shortest path from 0 to u in $\hat{\mathcal{G}}^s_f$ has (v,u) as its last edge. Therefore,

$$\begin{aligned} d_f(0,v) &= d_f(0,u) - 1 \ , \\ &\leq d_{f'}(0,u) - 1 \ , \\ &= d_{f'}(0,v) - 2 \ , \end{aligned} \tag{3.12}$$

which contradicts our assumption that $d_{f'}(0,v) < d_f(0,v)$.

2. A new edge (u,v) is created due to a saturating flow from u to v_1 for some $v_1 < v$. The MEMF algorithm always augments flow along shortest paths, and therefore the shortest path from 0 to v_1 in $\hat{\mathcal{G}}^s_f$ has (u,v_1) as its last edge. Since

$d_f(0,v) \le d_f(0,v_1)$, due to the upward infinite capacity edges, we have,

$$\begin{aligned} d_f(0,v) &\le d_f(0,v_1) \;, \\ &= d_f(0,u) + 1 \;, \\ &\le d_{f'}(0,u) + 1 \;, \\ &= d_{f'}(0,v) \;, \end{aligned} \tag{3.13}$$

which contradicts our assumption that $d_{f'}(0,v) < d_f(0,v)$.

We conclude that our assumption that such a node v exists is incorrect. □

The next theorem bounds the number of iterations of the MEMF algorithm.

Theorem 3.4.2. *If the MEMF algorithm (Algorithm 3.1) is run on the Ishikawa graph* $\hat{\mathcal{G}} = (\hat{\mathcal{V}} \cup \{0,1\}, \hat{\mathcal{E}})$ *with source* 0 *and sink* 1, *then the total number of augmentations performed by the algorithm is* $\mathcal{O}(|\hat{\mathcal{V}}||\hat{\mathcal{E}}|)$.

Proof. We say that an edge (u,v) in a residual lower-graph $\hat{\mathcal{G}}^s_f$ is *critical* on an augmenting path p if the residual capacity of p is the residual capacity of (u,v), *i.e.*, if $c_f(p) = c_f(u,v)$. After we have augmented flow along an augmenting path, any critical edge on the path disappears from the residual graph. Moreover, at least one edge on any augmenting path must be critical. We will show that each of the $|\hat{\mathcal{E}}|$ edges can become critical at most $|\hat{\mathcal{V}}|/2 + 1$ times. Furthermore, note that an infinite capacity edge cannot be critical at any point of the algorithm.

Let u and v be nodes in $\hat{\mathcal{V}} \cup \{0,1\}$ that are connected by an edge in $\hat{\mathcal{E}}^s$. Since augmenting paths are shortest paths, when (u,v) is critical for the first time, we have

$$d_f(0,v) = d_f(0,u) + 1 \;. \tag{3.14}$$

Once the flow is augmented, the edge (u,v) disappears from the residual graph. Since we maintain the lowest-cross-edge property, there cannot be an edge (u,v_1) in $\hat{\mathcal{G}}^s_f$ for some $v_1 < v$. Therefore, the edge (u,v) cannot reappear later on another augmenting path until after the flow from u to v_1 for some $v_1 \le v$ is decreased, which occurs only if (v_1,u) appears on an augmenting path. If f' is the flow when this event occurs, then we have

$$d_{f'}(0,u) = d_{f'}(0,v_1) + 1 \;. \tag{3.15}$$

Since $d_{f'}(0,v) \le d_{f'}(0,v_1)$, due to the upward infinite capacity edges, and $d_f(0,v) \le d_{f'}(0,v)$ by Lemma 3.4.1, we have

$$\begin{aligned} d_{f'}(0,u) &= d_{f'}(0,v_1) + 1 \;, \\ &\ge d_{f'}(0,v) + 1 \;, \\ &\ge d_f(0,v) + 1 \;, \\ &= d_f(0,u) + 2 \;. \end{aligned} \tag{3.16}$$

Consequently, from the time (u, v) becomes critical to the time when it next becomes critical, the distance of u from the source increases by at least 2. The distance of u from the source is initially at least 0. The intermediate nodes on a shortest path from 0 to u cannot contain 0, u or 1 (since (u, v) on an augmenting path implies that $u \neq 1$). Therefore, until u becomes unreachable from the source, if ever, its distance is at most $|\hat{\mathcal{V}}|$. Thus, after the first time that (u, v) becomes critical, it can become critical at most $|\hat{\mathcal{V}}|/2$ times more, for a total of $|\hat{\mathcal{V}}|/2 + 1$ times. Since there are $\mathcal{O}(|\hat{\mathcal{E}}|)$ pairs of nodes that can have an edge between them in a residual graph, the total number of critical edges during the entire execution of the MEMF algorithm is $\mathcal{O}(|\hat{\mathcal{V}}||\hat{\mathcal{E}}|)$. Each augmenting path has at least one critical edge, and hence the theorem follows. □

Let us analyze the time complexity of each subroutine of Algorithm 3.1 below. Note that both the subroutines *shortest-augmenting-path* and *augment* runs in $\mathcal{O}(m\,\ell)$ time, since, in the worst case, both subroutines need to check each edge in the lower-graph. However, *compute-edges* requires to run the flow reconstruction algorithm which takes $\mathcal{O}(\ell^3)$ time for each variable pair (i, j) (assuming a small max-flow problem with $2\,\ell$ nodes and ℓ^2 edges, solved using the most efficient algorithm [Orlin, 2013], see Section 3.3.1). Also *lower-graph* requires $\mathcal{O}(\ell^2)$ time for each variable pair (i, j), since it needs to check each of the Ishikawa edges. Hence, the worst case running time of each iteration (*i.e.*, augmentation step) is $\mathcal{O}(m\,\ell + \kappa\,(\ell^3 + \ell^2)) = \mathcal{O}(m\,\ell + m\,\ell^3) = \mathcal{O}(m\,\ell^3)$, where κ is the maximum number of flow-reconstructions (*i.e.*, saturated cross edges) at an augmentation step. Since the number of augmentations is bounded by $\mathcal{O}(|\hat{\mathcal{V}}||\hat{\mathcal{E}}|)$, the worst case running time of the entire execution of the MEMF algorithm is $\mathcal{O}(n\,\ell\,m\,\ell^2\,m\,\ell^3) = \mathcal{O}(n\,m^2\,\ell^6)$. This is $\mathcal{O}(\ell)$ slower than the standard Edmonds-Karp algorithm on the Ishikawa graph. Note, however, that MEMF requires $\mathcal{O}(\ell)$ less memory.

3.5 Efficient Algorithm

In the previous section, we have provided a general purpose polynomial time max-flow algorithm that is also memory efficient. However, for computer vision applications, the BK algorithm [Boykov and Kolmogorov, 2004] is shown to be significantly faster than the standard max-flow implementations, even though it lacks the polynomial time guarantee. The basic idea is to maintain a search tree throughout the algorithm instead of building the search tree from scratch at each iteration.

Motivated by this, we also propose doing search-tree-recycling similarly to the BK algorithm. Since we lose the polynomial time guarantee, for increased efficiency, we further simplify the Ishikawa graph. In particular, we find an augmenting path in a *block-graph*, that amalgamates the nodes in each column into blocks. Since an augmenting path in our block-graph corresponds to a collection of augmenting paths in the Ishikawa graph, our algorithm converges in fewer iterations than the BK algorithm.

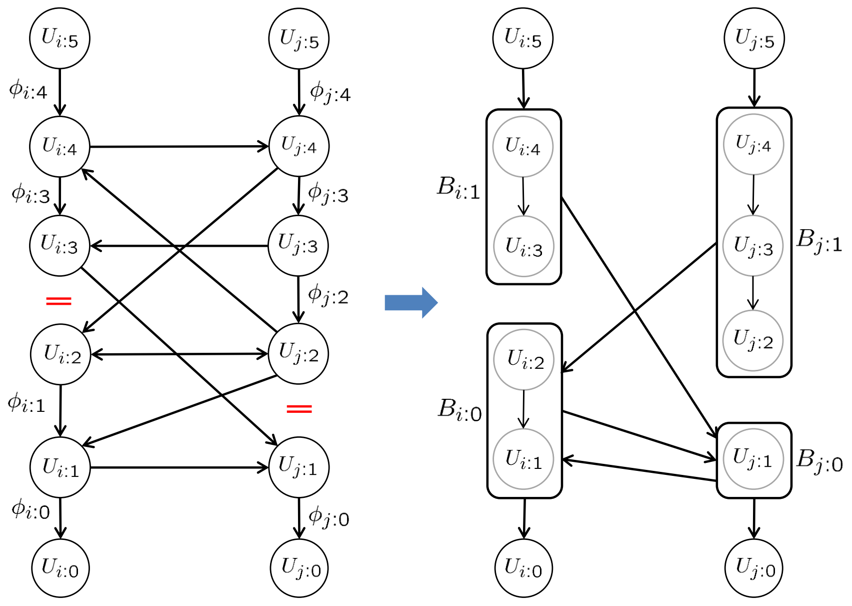


Figure 3.4: *To find an augmenting path in a memory efficient manner, we propose a simplified representation of the Ishikawa graph in terms of blocks corresponding to consecutive non-zero edges in each column i.*

3.5.1 Efficiently Finding an Augmenting Path

As mentioned above, we find an augmenting path in a block-graph[3], whose construction is detailed below.

Given the parameters $\boldsymbol{\phi}$, we rely on the fact that there exists a label λ such that $\phi_{i:\lambda} = 0$ for each $i \in \mathcal{V}$. In fact, it is easy to see that in each column i, if all $\phi_{i:\lambda}$ are positive, then there exists a *trivial* augmenting path from $U_{i:\ell}$ to $U_{i:0}$, and the minimum along the column can be subtracted from each $\phi_{i:\lambda}$. Now, at each column i, we partition the nodes $U_{i:\lambda}$ for all $\lambda \in \{1, \cdots, \ell - 1\}$ into a set of *blocks*, such that each node in a block is connected with positive edges $\phi_{i:\lambda}$. Let us denote these blocks by $B_{i:\gamma}$, where γ is indexed from bottom to top starting from 0. Note that there is no edge from $B_{i:\gamma+1}$ to $B_{i:\gamma}$. As depicted by Figure 3.4, our block-graph then contains only the blocks and the edges between the blocks.

The edges in the block-graph are obtained as follows. Let us consider a directed edge $(i, j) \in \mathcal{E}^+$. We add an edge $B_{i:\gamma} \rightarrow B_{j:\delta}$, where δ is the smallest value such that $\phi_{ij:\lambda\mu} > 0$ for some $U_{i:\lambda} \in B_{i:\gamma}$ and $U_{j:\mu} \in B_{j:\delta}$. While doing this, we also enforce that there is no edge $B_{i:\gamma'} \rightarrow B_{j:\delta'}$ such that $\gamma' > \gamma$ and $\delta' < \delta$. Meaning, if there is an edge $B_{i:\gamma'} \rightarrow B_{j:\delta'}$ for some $\gamma' > \gamma$ and $\delta' < \delta$, then the new edge $B_{i:\gamma} \rightarrow B_{j:\delta}$ will not be added. The reasoning behind this is that, because of the upward infinite-capacity edges between the nodes $U_{i:\lambda}$ and $U_{i:\lambda+1}$, we have the following:

1. If a node $U_{j:\mu}$ can be reached from $U_{i:\lambda}$ through positive edges, then the nodes

[3]We called this a *simplified graph* in [Ajanthan et al., 2016].

$U_{j:\mu'}$, for all $\mu' \geq \mu$, can also be reached.

2. If a node $U_{j:\mu}$ can be reached from $U_{i:\lambda}$ through positive edges, then it can also be reached from the nodes $U_{i:\lambda'}$, for all $\lambda' \leq \lambda$.

Hence, an edge $B_{i:\gamma} \to B_{j:\delta}$ indicates the fact that there is some positive flow possible from any node $U_{i:\lambda} \in B_{i:\gamma'}$, for all $\gamma' \leq \gamma$, to any node $U_{j:\mu} \in B_{j:\delta'}$, for all $\delta' \geq \delta$. In other words, the set of edges obtained by this procedure is sufficient.

Now, the relationship between augmenting paths in the original Ishikawa graph and in our block-graph can be characterized by the following theorem.

Theorem 3.5.1. *Given the Ishikawa graph, there is an augmenting path in the block-graph if and only if there exists an augmenting path in the Ishikawa graph.*

Proof. The basic idea of this proof is the same as the proof of Theorem 3.4.1. First, we prove that, if there is an augmenting path in the block-graph, then there exists an augmenting path in the Ishikawa graph. It is clear that an augmenting path in the block-graph contains an edge from node 0 to a block and then a sequence of edges $B_{i:\gamma} \to B_{j:\delta}$ and finally an edge from a block to node 1. Note that an edge from node 0 to a block $B_{i:\gamma}$ corresponds to a positive edge $e_{i:\ell-1}$ in the Ishikawa graph; similarly an edge from a block $B_{j:\delta}$ to node 1 corresponds to a positive edge $e_{j:0}$. Now, consider an edge $B_{i:\gamma} \to B_{j:\delta}$ in the augmenting path. Corresponding to this, there exists a positive edge $e_{ij:\lambda\mu}$ such that $U_{i:\lambda} \in B_{i:\gamma'}$ for some $\gamma' \geq \gamma$ and $U_{j:\mu} \in B_{j:\delta}$ in the Ishikawa graph. Furthermore, along the column i, there are upward infinite capacity edges, and nodes corresponding to a block are also connected with positive bidirectional edges. Hence, there exists an augmenting path in the Ishikawa graph, corresponding to the augmenting path in the block-graph.

Now, we prove the converse. Consider an augmenting path in the Ishikawa graph. The path may contain a sequence of positive edges $e_{i:\lambda}$, $e_{ij:\lambda\mu}$ and infinite capacity edges $e_{ii:\lambda\lambda+1}$. Note that, by construction, the $e_{i:\lambda}$ edges either will be in the same block $B_{i:\gamma}$ in the block-graph, or will be between a block and node 0 or node 1. Furthermore, the infinite capacity edges either will be in the same block, or there will be an edge $B_{i:\gamma} \to B_{j:\delta}$ in the block-graph to represent them. Finally, if $e_{ij:\lambda\mu}$ is a positive edge, then, by construction of the block-graph, there exists an edge $B_{i:\gamma} \to B_{j:\delta'}$ where $U_{i:\lambda} \in B_{i:\gamma}$ and $U_{j:\mu} \in B_{j:\delta}$ with $\delta' \leq \delta$. Hence, if there is an augmenting path in the Ishikawa graph, then there exists an augmenting path in the block-graph. □

Note that the block-graph can only be used to find an augmenting path; the quantity of the maximum permissible flow cannot be determined in this graph. Therefore, the capacity of an edge $B_{i:\gamma} \to B_{j:\delta}$ is not important, but it is important to have these edges. Note also that the block-graph is constructed incrementally for each edge $(i, j) \in \mathcal{E}$. Hence, it only requires us to store the Ishikawa graph parameters $\boldsymbol{\phi}_{ij}$ corresponding to the edge (i, j). Furthermore, since the block-graph $\hat{\mathcal{G}}^b$ is sparse, an augmenting path can be found fairly quickly.

Furthermore, similarly to the BK algorithm, we find an augmenting path P^b using BFS and maintain the search tree throughout the algorithm, by repairing it whenever the block-graph is updated. However, since the block-graph needs to be reconstructed after each augmentation, for simplicity, we maintain a single tree[4]. More specifically, we grow the search tree from source (node 0), in a breadth first manner, and if sink (node 1) is reached, then the augmenting path P^b is found.

3.5.2 Augmentation in the Block-Graph

Now, given an augmenting path P^b in the block-graph, we want to push the maximum permissible flow through it. More specifically, since P^b corresponds to a set of augmenting paths $\{p^b\}$ in the Ishikawa graph, we push the maximum flow through each path p^b, until no such path exists. This could be achieved by constructing the subgraph $\hat{\mathcal{G}}^p$ of the Ishikawa graph corresponding to the augmenting path P^b, and then finding each of the augmenting path p^b by searching in $\hat{\mathcal{G}}^p$. This would require us to either store $\hat{\mathcal{G}}^p$ (not memory efficient) or call the flow reconstruction algorithm too many times.

Instead, we propose breaking down the augmentation operation in the block-graph into a sequence of flow-loops and a subtraction along a column. Then, the maximum flow through the path can be pushed in a greedy manner, by pushing the maximum flow through each flow-loop. Before describing this procedure in detail, we introduce the following definitions.

Definition 3.5.1. A flow-loop $m(\lambda, \mu, \alpha)$ in the Ishikawa graph is defined as the following sequence of operations: First, a value α is pushed down the left column from $U_{i:\ell}$ to $U_{i:\lambda}$, then across from $U_{i:\lambda}$ to $U_{j:\mu}$, and finally up the right column from $U_{j:\mu}$ to $U_{j:\ell}$. Thus, applying the flow-loop $m(\lambda, \mu, \alpha)$ corresponds to replacing ϕ by $\phi + \Delta$, where

$$\begin{aligned}
\Delta_{i:\lambda'} &= -\alpha \quad \forall \lambda' \geq \lambda \ , \\
\Delta_{ij:\lambda\mu} &= -\alpha \ , \\
\Delta_{ji:\mu\lambda} &= \alpha \ , \\
\Delta_{j:\mu'} &= \alpha \quad \forall \mu' \geq \mu \ .
\end{aligned}$$

Definition 3.5.2. A flow-loop $\tilde{m}(\gamma, \delta, \alpha)$ in the block-graph $\hat{\mathcal{G}}^b$ is defined by the following sequence of operations: First a value α is pushed down the left column from $U_{i:\ell}$ to $B_{i:\gamma}$, then across from $B_{i:\gamma}$ to $B_{j:\delta}$, and finally up the right column from $B_{j:\delta}$ to $U_{j:\ell}$. The exact procedure is illustrated in Figure 3.5.

Note that, for a flow-loop $\tilde{m}(\gamma, \delta, \alpha)$ to be permissible, block $B_{i:\gamma}$ must contain node $U_{i:\ell-1}$. Note also that the flow-loop $\tilde{m}(\gamma, \delta, \alpha)$ can be thought of as a summation of flow-loops $m(\lambda, \mu, \alpha')$, where $U_{i:\lambda} \in B_{i:\gamma}$ and $U_{j:\mu} \in B_{j:\delta'}$, for all $\delta' \geq \delta$ (see Figure 3.5).

[4]The BK algorithm maintains two trees, source-tree and sink-tree, but we only maintain the source-tree.

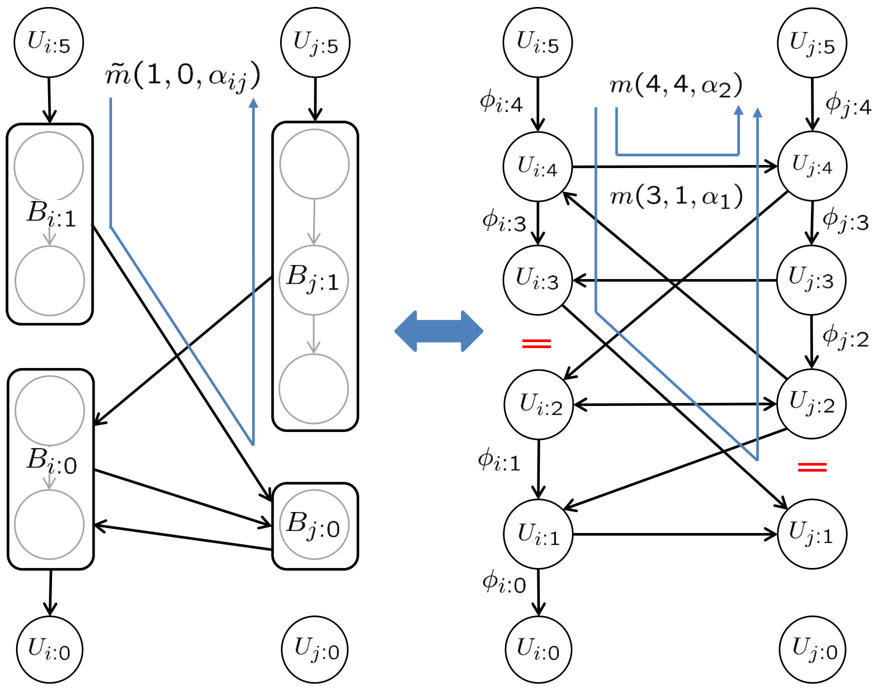


Figure 3.5: *An example flow-loop $\tilde{m}(1,0,\alpha_{ij})$ in the block-graph (**left**) is equivalent to the summation of two flow-loops $m(3,1,\alpha_1)$ and $m(4,4,\alpha_2)$ in the Ishikawa graph (**right**), with $\alpha_{ij} = \alpha_1 + \alpha_2$.*

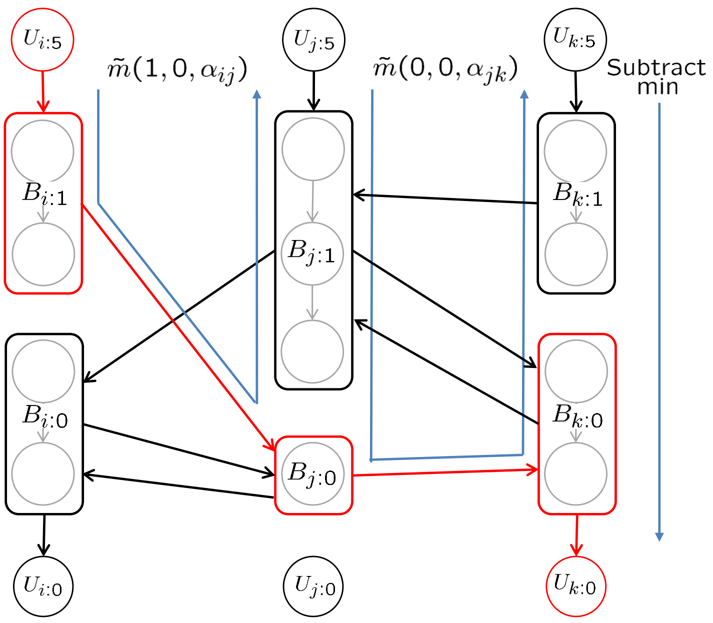


Figure 3.6: *An augmentation operation is broken down into a sequence of flow-loops $\tilde{m}(\gamma,\delta,\alpha)$, and a subtraction along the column k. The augmenting path P_s is highlighted in red.*

Given these definitions, one can easily see that the augmentation operation along the path P^b can be broken down into a sequence of flow-loops $\tilde{m}(\gamma, \delta, \alpha)$ and a subtraction along the last column k, as illustrated in Figure 3.6. Now, we push the maximum permissible flow through P^b, using the following greedy approach.

For each edge $B_{i:\gamma} \rightarrow B_{j:\delta}$ that is part of the path P^b, we apply a flow-loop $\tilde{m}(\gamma, \delta, \alpha_{ij})$, where α_{ij} is the maximum permissible flow through the edge $B_{i:\gamma} \rightarrow B_{j:\delta}$. In fact, applying this flow-loop translates to reconstructing the Ishikawa edge capacities $\boldsymbol{\phi}_{ij}$ corresponding to edge (i, j) and then applying flow-loops $m(\lambda, \mu, \alpha')$ for all $\lambda \geq \check{\lambda}$ and $\mu \geq \check{\mu}$, starting from $\check{\lambda}$ and $\check{\mu}$, until no permissible flow-loop $m(\lambda, \mu, \alpha')$ exists, with $\check{\lambda}$ and $\check{\mu}$ the smallest values such that $U_{i:\lambda} \in B_{i:\gamma}$ and $U_{j:\mu} \in B_{j:\delta}$. Finally, in the last column k, all the values $\phi_{k:\lambda}$ are positive, and the minimum along column k is subtracted from each $\phi_{k:\lambda}$. Note that, after this procedure, no augmenting path exists[5] in P^b and therefore this approach pushes the maximum permissible flow through the path P^b.

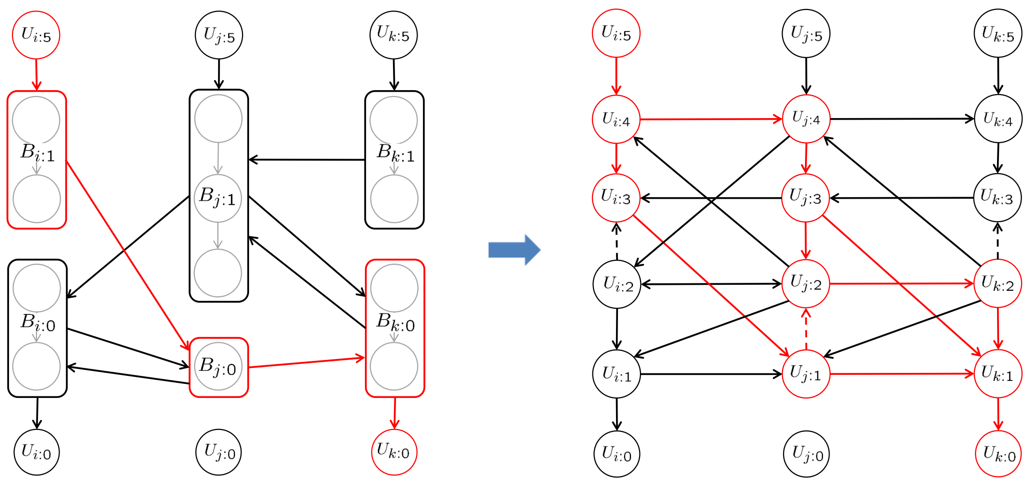
In the above explanation, we did not specifically discuss the scenario where P^b contains the same edge $(i, j) \in \mathcal{E}$ more than once, *e.g.*, $P^b = 0 \rightsquigarrow B_{i:\gamma} \rightarrow B_{j:\delta} \rightsquigarrow B_{j:\delta'} \rightarrow B_{i:\gamma'} \rightsquigarrow 1$. In fact this situation is not different and can be handled exactly the same way as before. This is mainly because of the construction of the block-graph, which guarantees $\gamma' < \gamma$ and $\delta' \leq \delta$. This means, even in this situation, the maximum flow through P^b is still positive and it can be pushed using the above mentioned procedure.

Remark. Note that the augmenting path P^b in the block-graph corresponds to a Directed Acyclic Graph (DAG) in the Ishikawa graph. See Figure 3.7 for an example. Now, the task is to push the maximum permissible flow through it. Since a DAG in an st-graph corresponds to a linear multi-label graph ($i - j - k$ for the DAG shown in Fig. 3.7), pushing the maximum-flow through it can be done using message-passing (in other words dynamic programming) in a single pass through this linear graph[6]. Since augmentation can be performed in a single pass, only ℓ^2 values (the Ishikawa or multi-label graph edges corresponding to variable pair (i, j)) are needed at a time, and, hence, it can be done in a memory efficient manner. Note that, even in our algorithm, the augmentation step is performed in a single pass.

Since, for each edge (i, j), we do not store all the $2\,\ell^2$ capacities, but only the $2\,\ell$ exit-flows Σ, augmentation must then also update these values. Fortunately, there is a direct relation between the flow-loops and Σ. To see this, let us consider the example flow-loop $\tilde{m}(1, 0, \alpha_{ij})$ shown in Figure 3.5. Applying this flow-loop updates

[5]After the application of this procedure there is no permissible flow-loop of the form $\tilde{m}(\gamma, \delta, \alpha)$ possible.

[6]More detail on the equivalence between max-flow and message-passing can be found in Section 3.6. Furthermore, for optimality of message-passing on linear graphs, see Section 2.3.4.4.



(a) Augmenting path in the block-graph (b) Corresponding directed acyclic graph

Figure 3.7: *Augmenting path in the block-graph (**left**) and its corresponding directed acyclic graph in the Ishikawa graph (**right**) are highlighted in red. Here the dashed arrows denote the upward infinite capacity edges.*

the corresponding exit-flows as

$$\begin{aligned} \Sigma_{ij:3} &= \Sigma_{ij:3} + \alpha_1 \ , \\ \Sigma_{ji:1} &= \Sigma_{ji:1} - \alpha_1 \ , \\ \Sigma_{ij:4} &= \Sigma_{ij:4} + \alpha_2 \ , \\ \Sigma_{ji:4} &= \Sigma_{ji:4} - \alpha_2 \ . \end{aligned} \tag{3.17}$$

Similar updates can be done for all flow-loops in our procedure. Note that the edge $B_{i:\gamma} \to B_{j:\delta}$ represents a collection of possible paths from all the nodes $U_{i:\lambda} \in B_{i:\gamma}$ to all the nodes $U_{j:\mu} \in B_{j:\delta'}$, for all $\delta' \geq \delta$. Therefore, unlike in the full Ishikawa graph, after applying a flow-loop, the portion of the graph $\hat{\mathcal{G}}^b_{ij}$ corresponding to edge $(i, j) \in \mathcal{E}$ needs to be reconstructed. This, however, can be done in a memory efficient manner, since it only involves one edge (i, j) at a time.

3.5.3 Summary

Our Memory Efficient Max-Flow (MEMF) method is summarized in Algorithm 3.2. Let us briefly explain the subroutines below.

block-graph. Given the initial Ishikawa parameters $\boldsymbol{\phi}^0$, this subroutine constructs the block-graph by amalgamating nodes into blocks as described in Section 3.5.1. If the input to the subroutine is the Ishikawa capacities $\boldsymbol{\phi}_{ij}$ corresponding to the edge $(i, j) \in \mathcal{E}$, then it constructs the block-graph portion $\hat{\mathcal{G}}^b_{ij}$.

Algorithm 3.2 Memory Efficient Max-Flow (MEMF) - Efficient Version

```
Require: $\boldsymbol{\phi}^0$                                    ▷ Initial Ishikawa capacities
  $\Sigma \leftarrow 0, T \leftarrow \emptyset$                    ▷ Initialize exit-flows and search tree
  $\hat{\mathcal{G}}^b \leftarrow$ block-graph($\boldsymbol{\phi}^0$)    ▷ Initial block-graph
  repeat
    $(T, P^b) \leftarrow$ augmenting-path($\hat{\mathcal{G}}^b, T$)          ▷ Section 3.5.1
    $\Sigma \leftarrow$ augment($P^b, \boldsymbol{\phi}^0, \Sigma$)          ▷ Section 3.5.2
    for each edge $(i,j) \in \mathcal{E}$ affected by augmentation do
      $\boldsymbol{\phi}_{ij} \leftarrow$ compute-edges($\boldsymbol{\phi}^0, \Sigma, i, j$)   ▷ Section 3.3.1
      $\hat{\mathcal{G}}^b_{ij} \leftarrow$ block-graph($\boldsymbol{\phi}_{ij}, i, j$)        ▷ Section 3.5.1
    end for
    $T \leftarrow$ repair-tree($T, \hat{\mathcal{G}}^b$)                ▷ Repair search tree
  until no augmenting paths possible
  return get-labelling($T$)                          ▷ Read from search tree
```

augmenting-path. Given the block-graph $\hat{\mathcal{G}}^b$ and the search tree T, this subroutine finds an augmenting path P^b by growing the search tree, as discussed in Section 3.5.1.

augment. Given the path P^b, this subroutine pushes the maximum permissible flow through it by applying flow-loops $\tilde{m}(\gamma, \delta, \alpha)$ and then subtracting the minimum from the last column, as discussed in Section 3.5.2.

compute-edges. This is the same subroutine as in Algorithm 3.1. (see Section 3.4.3).

repair-tree. This subroutine is similar to the *adoption* stage of the BK algorithm. Given the reconstructed block-graph, the search tree T is repaired by checking for valid *parent* for each *orphan* node. See Section 3.2.3 in [Boykov and Kolmogorov, 2004] for more detail.

get-labelling. This subroutine directly reads the optimal labelling from the search tree T.

As discussed Section 3.4.3, the space complexity of our algorithm is $\mathcal{O}(m\,\ell)$. For the rest of the chapter, this efficient version of the algorithm is referred to as MEMF.

3.6 Equivalence with Message Passing

In this section, we give a more insightful interpretation of our max-flow algorithm, by showing equivalence with the min-sum message passing algorithm (see Section 2.3.4.3 for an overview). Note that this equivalence was observed in [Tarlow et al., 2011] for binary submodular MRFs. Here we discuss it in the context of multi-label submodular MRFs and also show the relationship between the set of exit-flows and the set of messages. In particular, we first show that our max-flow algorithm is in fact solving the *optimal message passing* problem (2.85). Later, we point out the relationship between the set of exit-flows and the set of messages. To this end, let us first recall the optimal message passing problem.

3.6.1 The Optimal Message Passing Problem

As discussed in Section 2.3.4.3, the optimal message passing problem (2.85) takes the following form,

$$\begin{aligned} \max_{\boldsymbol{\theta}'} \sum_{i \in \mathcal{V}} \min_{\lambda \in \mathcal{L}} & \theta'_{i:\lambda} \,, \\ \text{s.t.} \quad & E_{\boldsymbol{\theta}'} \equiv E_{\boldsymbol{\theta}} \,, \\ & \theta'_{ij:\lambda\mu} \geq 0 \quad \forall\, (i,j) \in \mathcal{E},\, \lambda, \mu \in \mathcal{L} \,. \end{aligned} \tag{3.18}$$

Here, $E_{\boldsymbol{\theta}}$ denotes the multi-label energy function (3.1) defined by the parameter vector $\boldsymbol{\theta}$ and the shorthand $\theta_{i:\lambda} = \theta_i(\lambda)$ and $\theta_{ij:\lambda\mu} = \theta_{ij}(\lambda, \mu)$ is used. Furthermore, the notation $E_{\boldsymbol{\theta}'} \equiv E_{\boldsymbol{\theta}}$ means that $E_{\boldsymbol{\theta}'}(\mathbf{x}) = E_{\boldsymbol{\theta}}(\mathbf{x})$ for all labellings $\mathbf{x}$. Recall that the Ishikawa edge capacities $\boldsymbol{\phi}$ and the energy parameters $\boldsymbol{\theta}$ are related according to the formula (3.4).

If $E_{\boldsymbol{\theta}} \equiv E_{\boldsymbol{\theta}'}$, then $\boldsymbol{\theta}'$ is a reparametrization of $\boldsymbol{\theta}$, and the conditions for this equivalence are characterized by the reparametrization lemma (see Lemma 2.3.4). We restate it here for completeness.

Lemma 3.6.1. *Two energy functions $E_{\boldsymbol{\theta}}$ and $E_{\boldsymbol{\theta}'}$ are equivalent if and only if there exist messages $m_{ji:\lambda}$ and $m_{ij:\mu}$ for $(i,j) \in \mathcal{E}$ and $\lambda, \mu \in \mathcal{L}$ such that*

$$\begin{aligned} \theta'_{ij:\lambda\mu} &= \theta_{ij:\lambda\mu} - m_{ji:\lambda} - m_{ij:\mu} \,, \\ \theta'_{i:\lambda} &= \theta_{i:\lambda} + \sum_{(k,i) \in \mathcal{E}^+} m_{ki:\lambda} \,. \end{aligned} \tag{3.19}$$

The optimality of min-sum message passing for the case of multi-label submodular MRFs was observed in [Werner, 2007; Kolmogorov and Wainwright, 2005]. We now show that our max-flow algorithm is in fact solving the above message passing problem.

3.6.2 Max-Flow Solves the Optimal Message Passing Problem

Let us state our claim as a theorem.

Theorem 3.6.1. *The max-flow algorithm on the Ishikawa graph solves the optimal message passing problem* (3.18) *for multi-label submodular MRFs.*

A usual max-flow algorithm finds augmenting paths and pushes maximum flow through them. Before giving the proof, let us first characterize the notion of an augmenting path below.

The Notion of an Augmenting Path. Let us consider an augmenting path in the Ishikawa graph. If it is a *trivial* augmenting path, *i.e.*, it is an augmenting path along a column from nodes $U_{i:\ell}$ to $U_{i:0}$, then pushing the maximum flow along the path translates to subtracting the minimum value from each $\phi_{i:\lambda}\,;\,\lambda \in \mathcal{L}$. Therefore, the notion of a trivial augmenting path is,

$$\phi_{i:\lambda} > 0 \quad \forall\, \lambda \in \mathcal{L}\;, \tag{3.20}$$

for some $i \in \mathcal{V}$.

Let us consider a more interesting augmenting path in the Ishikawa graph, which contains at least one cross edge $e_{ij:\lambda\mu} \in \hat{\mathcal{E}}_c^+$. From the discussion in Section 3.5.2, one can easily see that the augmentation operation can be broken down into a sequence of flow-loops $m(\lambda, \mu, \alpha)$ (see Definition 3.5.1) and a subtraction along a column. This intuitively suggests that an augmenting path in the Ishikawa graph can be translated to a trivial augmenting path by passing flow around loops, *i.e.*, they differ by a *null* flow. From Lemma 3.3.1, a null flow corresponds to a reparametrization of $\boldsymbol{\phi}$ and two such energy functions are equivalent. Hence, the notion of any augmenting path can be characterized as, finding a set of reparametrizations that makes $\phi_{i:\lambda}$ positive for all $\lambda \in \mathcal{L}$ for some $i \in \mathcal{V}$.

Now we prove our theorem.

Proof. Note that the max-flow algorithm iteratively finds augmenting paths and pushes flow through them, without making the edge capacities negative. It proceeds until no augmenting path exists. From the argument above, it becomes clear that any augmenting path can be translated to a trivial augmenting path by passing flow around loops (*i.e.*, reparametrization). Furthermore, pushing permissible flow through a trivial augmenting path is simply a subtraction of the minimum value $\min_{\lambda\in\mathcal{L}} \phi_{i:\lambda}$ from each $\phi_{i:\lambda}\,;\,\lambda \in \mathcal{L}$. Therefore the max-flow algorithm solves the following optimization problem:

$$\begin{aligned} \max_{\boldsymbol{\phi}'} \; & \sum_{i\in\mathcal{V}} \min_{\lambda\in\mathcal{L}} \phi'_{i:\lambda}\;, \\ \text{s.t.} \quad & E_{\boldsymbol{\phi}'} \equiv E_{\boldsymbol{\phi}}\;, \\ & \phi'_{ij:\lambda\mu} \geq 0 \quad \forall\, e_{ij:\lambda\mu} \in \hat{\mathcal{E}}_c^+\;. \end{aligned} \tag{3.21}$$

Note that, given the Ishikawa edge capacities $\boldsymbol{\phi}$, the corresponding energy parameters $\boldsymbol{\theta}$ can be calculated using the following formula,

$$\begin{aligned} \theta_{i:\lambda} &= \phi_{i:\lambda}\;, \\ \theta_{ij:\lambda\mu} &= \sum_{\substack{\lambda'>\lambda \\ \mu'\leq\mu}} \phi_{ij:\lambda'\mu'} + \sum_{\substack{\lambda'\leq\lambda \\ \mu'>\mu}} \phi_{ji:\mu'\lambda'}\;. \end{aligned} \tag{3.22}$$

For multi-label submodular MRFs, by construction, $E_{\boldsymbol{\theta}} \equiv E_{\boldsymbol{\phi}} \equiv E_{\boldsymbol{\phi}'} \equiv E_{\boldsymbol{\theta}'}$. Furthermore, for all $(i,j) \in \mathcal{E}$ and $\lambda, \mu \in \mathcal{L}$, $\theta_{ij:\lambda\mu} \geq 0$, if $\phi_{ij:\lambda\mu} \geq 0$ for all cross edges $\hat{\mathcal{E}}_c^+$.

Hence, Eq. (3.21) can be equivalently written as

$$\begin{aligned}
&\max_{\boldsymbol{\theta}'} \sum_{i\in\mathcal{V}} \min_{\lambda\in\mathcal{L}} \theta'_{i:\lambda}\ , \\
&\text{s.t.}\quad E_{\boldsymbol{\theta}'} \equiv E_{\boldsymbol{\theta}}\ , \\
&\qquad \theta'_{ij:\lambda\mu} \geq 0 \quad \forall\,(i,j)\in\mathcal{E},\ \lambda,\mu\in\mathcal{L}\ .
\end{aligned} \tag{3.23}$$

This is exactly the same as the optimal message passing problem (3.18). □

3.6.3 Flow-Loop as a Reparametrization

From Lemmas 3.3.1 and 3.6.1, it is clear that a flow-loop corresponds to a reparametrization of the multi-label energy function. In this section, we find the equivalent reparametrization of a flow-loop $m(\lambda,\mu,\alpha)$. This will later allow us to understand the relationship between the set of exit-flows and the set of messages. Let us now state and prove our theorem.

Theorem 3.6.2. *Applying a flow-loop $m(\lambda,\mu,\alpha)$ in the Ishikawa graph is equivalent to a reparametrization of the multi-label energy function, with messages*

$$\begin{aligned}
m_{ji:\lambda'} &= -\alpha \quad \forall\,\lambda' \geq \lambda\ , \\
m_{ij:\mu'} &= \alpha \quad \forall\,\mu' \geq \mu\ .
\end{aligned} \tag{3.24}$$

Proof. Let the Ishikawa parameters be $\boldsymbol{\phi}$ and the energy parameters be $\boldsymbol{\theta}$ and assume that the flow is applied between columns i and j. Also, after the flow, the parameters are $\boldsymbol{\phi}'$ and $\boldsymbol{\theta}'$ respectively. Since $\boldsymbol{\theta}$ can be calculated from $\boldsymbol{\phi}$ using Eq. (3.4), $E_{\boldsymbol{\theta}} \equiv E_{\boldsymbol{\phi}}$. Similarly $E_{\boldsymbol{\phi}'} \equiv E_{\boldsymbol{\theta}'}$. Also from Lemma 3.3.1, $E_{\boldsymbol{\phi}} \equiv E_{\boldsymbol{\phi}'}$. Hence, $E_{\boldsymbol{\theta}} \equiv E_{\boldsymbol{\phi}} \equiv E_{\boldsymbol{\phi}'} \equiv E_{\boldsymbol{\theta}'}$. Now, from Definition 3.5.1,

$$\begin{aligned}
\phi'_{i:\lambda'} &= \phi_{i:\lambda'} - \alpha \quad \forall\,\lambda' \geq \lambda\ , \\
\phi'_{j:\mu'} &= \phi_{j:\mu'} + \alpha \quad \forall\,\mu' \geq \mu\ , \\
\phi'_{ij:\lambda\mu} &= \phi_{ij:\lambda\mu} - \alpha\ , \\
\phi'_{ji:\mu\lambda} &= \phi_{ji:\mu\lambda} + \alpha\ .
\end{aligned} \tag{3.25}$$

Substituting in Eq. (3.4),

$$\begin{aligned}
\theta'_{i:\lambda'} &= \theta_{i:\lambda'} - \alpha \quad \forall\,\lambda' \geq \lambda\ , \\
\theta'_{j:\mu'} &= \theta_{j:\mu'} + \alpha \quad \forall\,\mu' \geq \mu\ , \\
\theta'_{ij:\lambda'\mu'} &= \theta_{ij:\lambda'\mu'} - \alpha \quad \forall\,\lambda' < \lambda,\ \mu' \geq \mu\ , \\
\theta'_{ij:\lambda'\mu'} &= \theta_{ij:\lambda'\mu'} + \alpha \quad \forall\,\lambda' \geq \lambda,\ \mu' < \mu\ .
\end{aligned} \tag{3.26}$$

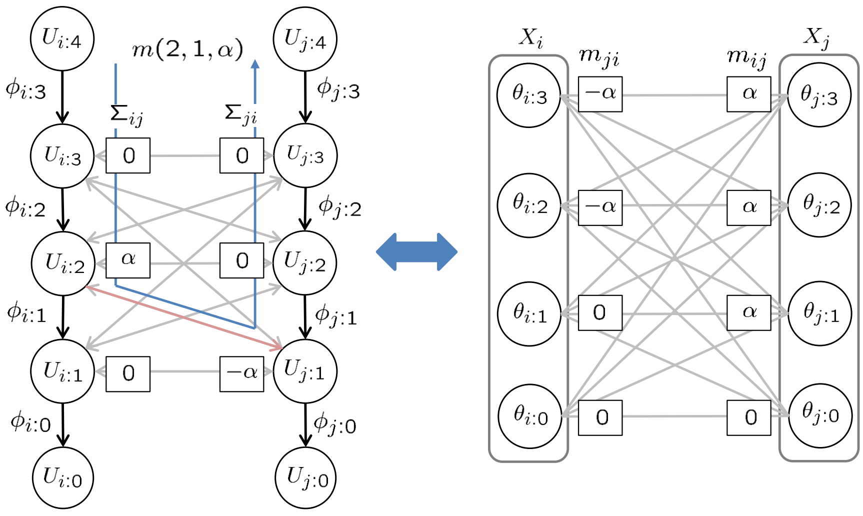


Figure 3.8: *A flow $m(2,1,\alpha)$ in the Ishikawa graph (**left**) and its equivalent reparametrization in the multi-label graph (**right**) - the energy parameters can be represented in a multi-label graph (see Section 2.3.1). Note that the exit-flow vectors (Σ_{ij}, Σ_{ji}) and the corresponding message vectors ($\mathbf{m}_{ji}, \mathbf{m}_{ij}$) are shown next to the nodes.*

Now, since $E_{\boldsymbol\theta} \equiv E_{\boldsymbol\theta'}$, by Lemma 3.6.1, there exist messages $m_{ji:\lambda}$ and $m_{ij:\mu}$ such that

$$\begin{aligned} \theta'_{ij:\lambda\mu} &= \theta_{ij:\lambda\mu} - m_{ji:\lambda} - m_{ij:\mu} \ , \\ \theta'_{i:\lambda} &= \theta_{i:\lambda} + \sum_{(k,i)\in\mathcal{E}^+} m_{ki:\lambda} \ . \end{aligned} \tag{3.27}$$

With a little bit of calculation, one can see that the messages take the following form

$$\begin{aligned} m_{ji:\lambda'} &= -\alpha \quad \forall\, \lambda' \geq \lambda \ , \\ m_{ij:\mu'} &= \alpha \quad \forall\, \mu' \geq \mu \ . \end{aligned} \tag{3.28}$$

Note that, for a permissible flow $m(\lambda,\mu,\alpha)$, the parameters $\boldsymbol\phi'$ and $\boldsymbol\theta'$ are non-negative. □

This equivalence is shown in Figure 3.8 for an example flow-loop $m(2,1,\alpha)$. Note that, as shown in the figure, the flow α through an edge $\phi_{ij:\lambda\mu}$ may be recorded in the set of exit-flows Σ. Furthermore, as shown in the figure, the relationship between the set of exit-flows and the set of messages can be written as

$$\begin{aligned} \Sigma_{ij:\lambda} &= m_{ji:\lambda-1} - m_{ji:\lambda} \quad \forall\, \lambda \in \{1,\cdots,\ell-1\} \ , \\ \Sigma_{ji:\mu} &= m_{ij:\mu-1} - m_{ij:\mu} \quad \forall\, \mu \in \{1,\cdots,\ell-1\} \ . \end{aligned} \tag{3.29}$$

Here, note that the exit-flow $\Sigma_{ij:\lambda}$ is the sum of flow passed from node $U_{i:\lambda}$ to all nodes in column j. Whereas $m_{ji:\lambda}$ is the message passed from edge (j,i) to node $X_{i:\lambda}$ in the multi-label graph.

3.7 Related Work

As we discussed in Chapter 2, the approaches that have been proposed to minimize multi-label submodular MRFs can be roughly grouped into two categories: those based on max-flow and those based on an LP relaxation of the problem. Below, we briefly review representative techniques in each category.

3.7.1 Max-Flow-based Methods

The most popular method to minimize a multi-label submodular MRF energy is to construct the Ishikawa graph [Ishikawa, 2003] and then apply a max-flow algorithm to find the min-cut solution. Broadly speaking, as mentioned in Section 2.2.4, there are three different kinds of max-flow algorithms: those relying on finding augmenting paths [Ford and Fulkerson, 1962], the push-relabel approach [Goldberg and Tarjan, 1988] and the pseudo-flow techniques [Chandran and Hochbaum, 2009]. Even though numerous implementations are available, the BK method [Boykov and Kolmogorov, 2004] is arguably the fastest implementation for 2D and sparse 3D graphs. Recently, for dense problems, the IBFS algorithm [Goldberg et al., 2011] was shown to outperform the BK method in a number of experiments [Verma and Batra, 2012]. Furthermore, for multi-label submodular MRFs with convex unary potentials an efficient algorithm is presented in [Hochbaum, 2001]. For arbitrary unary potentials, all the above-mentioned algorithms, however, require the same order of storage as the Ishikawa graph and hence scale poorly. Two approaches have nonetheless been studied to scale the max-flow algorithms. The first one explicitly relies on the N-D grid structure of the problem at hand [Delong and Boykov, 2008; Jamriška et al., 2012]. The second one makes use of distributed computing [Shekhovtsov and Hlaváč, 2013; Strandmark and Kahl, 2010; Vineet and Narayanan, 2008]. Unfortunately, both these approaches require additional resources (disk space or clusters) to run max-flow on an Ishikawa graph. By contrast, our algorithm lets us efficiently minimize the energy of much larger Ishikawa-type graphs on a standard computer. Furthermore, using the method of [Strandmark and Kahl, 2010], it can also be parallelized.

3.7.2 LP Relaxation-based Methods

One memory-efficient way to minimize a multi-label submodular MRF energy consists of formulating the problem as a linear program and then maximize the dual using message-passing techniques [Wainwright et al., 2005] (see Section 2.3.4). Many such algorithms have been studied [Kolmogorov, 2006; Komodakis et al., 2011; Savchynskyy et al., 2012; Werner, 2007]. Even though these algorithms are good at approximating the optimal solution (also theoretically optimal for multi-label submodular MRFs [Kolmogorov and Wainwright, 2005; Werner, 2007]), as evidenced by the comparison of [Kappes et al., 2015] and by our experiments, they usually take much longer to converge to the optimal solution than max-flow-based techniques.

Figure 3.9: *Left and right images of the stereo instance from the KITTI dataset. The images are of size* 1241 × 376*, and we set the number of labels to 40. This image pair was chosen arbitrarily as a representative of the dataset.*

3.8 Experiments

We evaluated our algorithm on the problems of stereo correspondence estimation and image inpainting. For stereo correspondence estimation, we employed six instances from the Middlebury dataset [Scharstein and Szeliski, 2002, 2003]: Tsukuba, Venus, Sawtooth, Map, Cones and Teddy, and one instance from the KITTI dataset [Geiger et al., 2013] (see Figure 3.9). For Tsukuba and Venus, we used the unary potentials of [Szeliski et al., 2008], and for all other stereo cases, those of [Birchfield and Tomasi, 1998]. For inpainting, we used the Penguin and House images employed in [Szeliski et al., 2008], and we used the same unary potentials as in [Szeliski et al., 2008]. In all the above cases, we used pairwise potentials that can be expressed as

$$\theta_{ij}(x_i, x_j) = \gamma_{ij}\,\theta(|x_i - x_j|)\ , \tag{3.30}$$

where, unless stated otherwise, the regularizer $\theta(|x_i - x_j|)$ is the quadratic function. Furthermore, we employed a 4-connected neighbourhood structure in all our experiments.

We compare our results with two max-flow implementations: the BK algorithm [Boykov and Kolmogorov, 2004] and Excesses Incremental Breadth First Search (EIBFS) [Goldberg et al., 2015] (which we ran on the Ishikawa graph), and three LP relaxation-based algorithms: Tree Reweighted Message Passing (TRWS) [Kolmogorov, 2006], Subgradient-based Dual Decomposition (DDSG) [Komodakis et al., 2011] and the Adaptive Diminishing Smoothing algorithm (ADSal) [Savchynskyy et al., 2012]. For DDSG and ADSal, we used the Opengm [Andres et al., 2012] implementations. For the other algorithms, we employed the respective authors' implementations.

In practice, we only ran the BK algorithm and EIBFS if the graph could be stored in RAM. Otherwise, we provide an estimate of their memory requirement. For LP relaxation-based methods, unless they converged, we ran the algorithms either for 10000 iterations, or for 50000 seconds, whichever occurred first. Note that the running times reported for our algorithm include graph construction. All our experiments were conducted on a 3.4 GHz i7-4770 CPU with 16 GB RAM.

The memory consumption and running times of the algorithms are provided in Table 3.1. Altogether, our algorithm lets us solve much larger problems than the BK algorithm and EIBFS, and is an order of magnitude faster than state-of-the-art message-passing algorithms.

	Problem	ℓ	BK	EIBFS	DDSG	ADSal	TRWS	MEMF
Memory [MB]	Tsukuba	16	3195	2495	258	252	287	**211**
	Venus	20	7626	5907	424	418	638	**396**
	Sawtooth	20	7566	5860	415	415	633	**393**
	Map	30	6454	4946	**171**	208	494	219
	Cones	60	*72303	*55063	**657**	939	5024	1200
	Teddy	60	*72303	*55063	**659**	939	5025	1200
	KITTI	40	*88413	*67316	**1422**	1802	6416	2215
	Penguin	256	*173893	*130728	236	1123	**215**	663
	House	256	*521853	*392315	689	2389	**643**	1986
Time [s]	Tsukuba	16	14	**4**	>9083	>7065	198	28
	Venus	20	35	**9**	>18156	1884	206	59
	Sawtooth	20	31	**8**	>16238	10478	455	35
	Map	30	57	**9**	>9495	>1679	187	36
	Cones	60	-	-	>50000	>17866	1095	**364**
	Teddy	60	-	-	>50000	>50000	6766	**2055**
	KITTI	40	-	-	>50000	>50000	>45408	**18665**
	Penguin	256	-	-	>50000	>50000	>50000	**6504**
	House	256	-	-	>50000	>50000	>50000	**9001**

Table 3.1: *Memory consumption and running time comparison with state-of-the-art baselines for quadratic regularizer (see paragraph 2 of Section 3.8, for details on the algorithms). A "*" indicates a memory estimate, and ">" indicates that the algorithm did not converge to the optimum within the specified time. Note that our algorithm has a memory consumption $\mathcal{O}(\ell)$ times lower than the max-flow-based methods and is an order of magnitude faster than message-passing algorithms. Compared to EIBFS, our algorithm is only 4 – 7 times slower, but requires 12 – 23 times less memory, which makes it applicable to more realistic problems. In all stereo problems, TRWS cached the pairwise potentials in an array for faster retrieval, but in the case of inpainting, it was not possible due to excessive memory requirement.*

3.8.1 MEMF Analysis

In this section, we empirically analyze various properties of our algorithm. First, note that, at each iteration, *i.e.*, at each augmentation step, our algorithm performs more computation than standard max-flow. Therefore, we would like our algorithm to find short augmenting paths and to converge in fewer iterations than standard max-flow. Below, we analyze these two properties empirically.

In Figure 3.10, we show the distribution of the lengths of the augmenting paths found by our algorithm for the Tsukuba stereo instance. Note that the median length is only 5. As a matter of fact, the maximum length observed over all our experiments was 1073 for the KITTI data. Nevertheless, even in that image, the median length was only 15. Note that, since our algorithm finds augmenting paths in the block-graph, the path lengths are not directly comparable to those found by other max-flow-based methods. In terms of number of augmentations, we found that our algorithm only

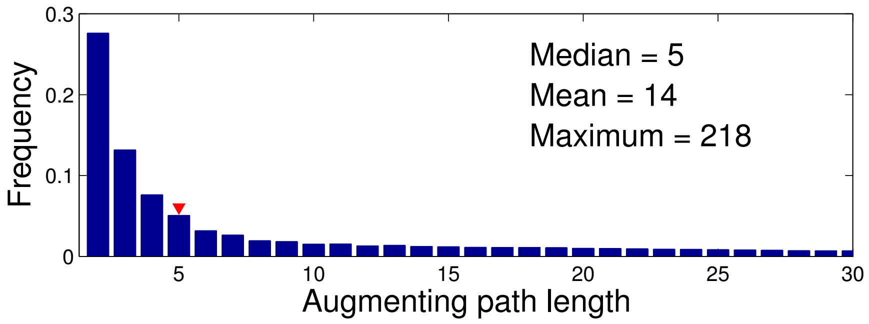


Figure 3.10: *Lengths of augmenting paths found by our algorithm for the Tsukuba stereo instance (see Section 3.8.1). Each bar indicates the proportion of paths of a certain length. For example, out of all augmenting paths* 28% *of them were of length* 2. *The red arrow indicates the median length.*

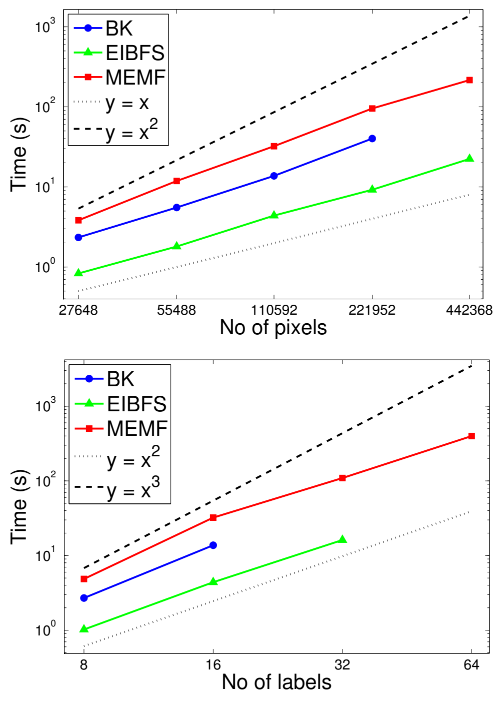
required between 35% and 50% of the total number of augmentations of the BK algorithm.

Next, we fixed the number of labels but varied the image size and compare the running times of the max-flow algorithms, for Tsukuba and Penguin instances in top row of Figure 3.11. Similarly, we fixed the image size but varied the number of labels and report the running times in bottom row of Figure 3.11. By doing this, we try to estimate the empirical time complexity of our algorithm. Note that, similar to other max-flow algorithms, MEMF exhibited near-linear performance with respect to the image size and near-cubic performance with respect to the number of labels, in these experiments.

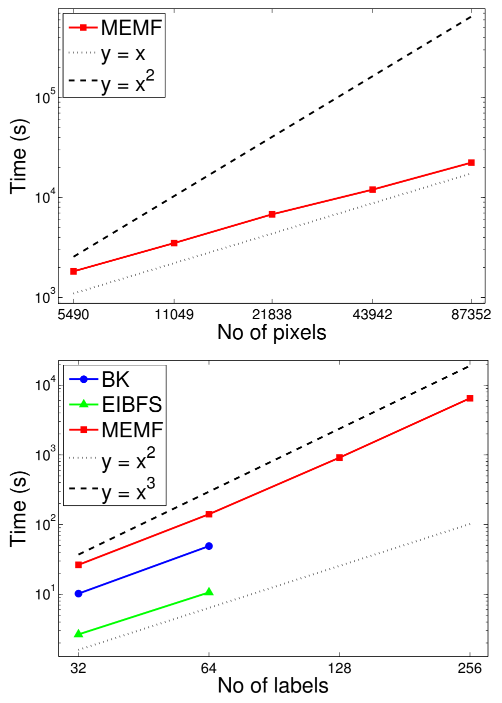
Finally, we report the percentage of time taken by each subroutine of our algorithm, for Tsukuba and Penguin instances in Figure 3.12. Note that the individual time complexities of the subroutines *compute-edges* and *block-graph* are $\mathcal{O}(\ell^3)$ and $\mathcal{O}(\ell^2)$, respectively. Therefore, they become dominant when the number of labels is large, and hence the corresponding percentages of time are high, particularly for Penguin.

3.8.2 Robust Regularizer

Since robust regularizers are highly effective in computer vision, we tested our algorithm by choosing Huber loss function (see Eq. (2.43)) as the regularizer. The results are summarized in Table 3.2. Note that the Ishikawa graph for a Huber regularizer is significantly smaller, *i.e.*, the number of edges per variable pair is $\mathcal{O}(\kappa\,\ell)$, instead of $\mathcal{O}(\ell^2)$ (see Section 2.3.2). Even in this case, our algorithm lets us solve much larger problems than the BK algorithm and EIBFS, and is an order of magnitude faster than state-of-the-art message-passing algorithms.



(a) Tsukuba



(b) Penguin

Figure 3.11: *Running time plots (in logarithmic scale) by changing the image size (**top**) and by changing the number of labels (**bottom**), for Tsukuba and Penguin (see Section 3.8.1). The dashed lines provide the reference slopes. Note that all algorithms exhibited near-linear performance with respect to the number of pixels and near-cubic performance with respect to the number of labels, but MEMF required $\mathcal{O}(\ell)$ less memory. The plots of BK algorithm and EIBFS are not complete, since we could not run them due to excessive memory requirement.*

3.8.3 Parallelization

We parallelized our algorithm based on the dual-decomposition technique of [Strandmark and Kahl, 2010] and evaluated it on the same six stereo instances from the Middlebury dataset [Scharstein and Szeliski, 2002, 2003]. The relative times t_m/t_s, where t_m stands for the multi-thread time and t_s for the single-thread one, are shown in Figure 3.13 for two and four threads. In this experiment, for all problems, the image grid was split vertically into two and four equally-sized blocks, respectively. Note that this splitting strategy is fairly arbitrary, and may affect the performance of the multi-threaded algorithm. In fact finding better splits may itself be a possible future direction.

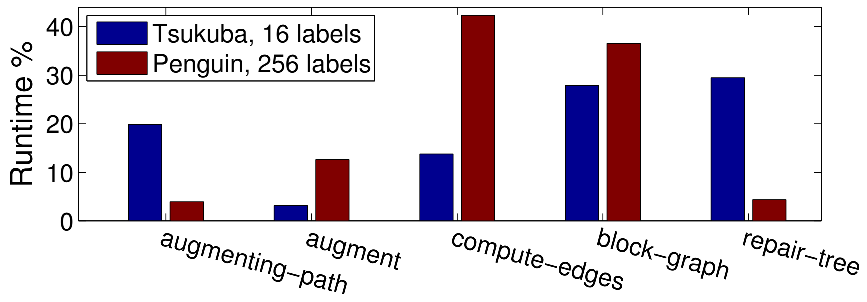


Figure 3.12: *Percentage of time taken by each subroutine (see Section 3.8.1). Note that, for Penguin, due to large number of labels, the percentages of time spend on compute-edges and block-graph are high.*

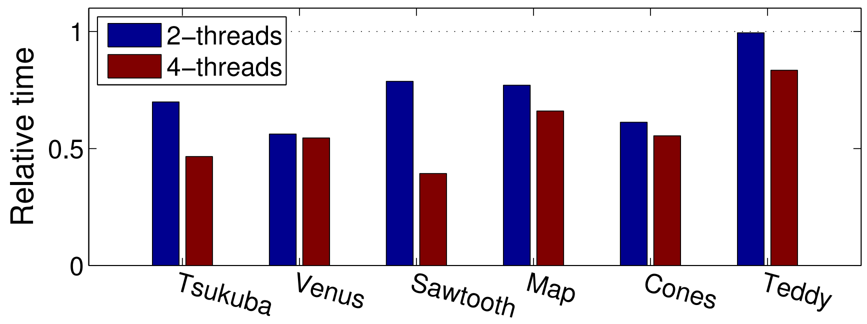


Figure 3.13: *Our algorithm can be accelerated using the parallel max-flow technique (see Section 3.8.3). The relative times ranged from 0.56 to 0.99 with 2-threads and from 0.39 to 0.83 with 4-threads. For Teddy, in the case of 2-threads, the multi-threaded algorithm performs almost the same as the single-threaded algorithm, which may be due to bad splits.*

Problem			Memory [MB]				Time [s]			
Name	ℓ	κ	BK	EIBFS	TRWS	MEMF	BK	EIBFS	TRWS	MEMF
Tsukuba	16	4	1715	1385	287	**211**	8	**3**	198	28
Venus	20	4	3375	2719	638	**396**	17	**5**	211	57
Sawtooth	20	4	3348	2698	633	**393**	15	**4**	467	34
Map	30	6	2680	2116	494	**219**	22	**5**	>2953	36
Cones	60	20	*42155	*32167	5025	**1200**	-	-	1118	**363**
Teddy	60	20	*42155	*32167	5025	**1200**	-	-	6879	**2064**
KITTI	40	10	*42161	*32627	6416	**2215**	-	-	>30165	**18923**
Penguin	256	25	*33487	*25423	**215**	663	-	-	>50000	**6277**
House	256	25	*100494	*76295	**643**	1986	-	-	>50000	**8568**

Table 3.2: *Memory consumption and running time comparison with state-of-the-art baselines for Huber regularizer (see Section 3.8.2). Here, ℓ is the number of labels and κ is the Huber value. A "*" indicates a memory estimate, and ">" indicates that the algorithm did not converge to the optimum within the specified time. Note that our algorithm has a much lower memory consumption than the max-flow-based methods and is an order of magnitude faster than message-passing algorithms. Compared to EIBFS, our algorithm is 7 – 11 times slower, but requires 7 – 10 times less memory, which makes it applicable to more realistic problems. In all stereo problems, TRWS cached the pairwise potentials in an array for faster retrieval, but in the case of inpainting, it was not possible due to excessive memory requirement.*

3.9 Discussion

We have introduced a variant of the max-flow algorithm that can minimize multi-label submodular MRF energies optimally while requiring much less storage. Furthermore, our experiments have shown that our algorithm is an order of magnitude faster than state-of-the-art methods. We, therefore, believe that our algorithm constitutes the method of choice to minimize Ishikawa type graphs when the complete graph cannot be stored in memory.

Even though our MEMF algorithm converges in approximately the same time as that of the BK method, it is $4-7$ times slower than the EIBFS algorithm. Note that the EIBFS algorithm is an improved version of the BK method which combines push-relabel techniques with the augmenting path based BK algorithm (see [Goldberg et al., 2015]). Due to the similarity of MEMF to the BK method, we believe that MEMF can also be accelerated in a similar fashion. This would not only make MEMF significantly faster but also improve its worst case time complexity.

On the other hand, the equivalence of MEMF with min-sum message passing indicates interesting directions to pursue. For instance, MEMF can be used to obtain min-marginals similarly to [Kohli and Torr, 2008]. Furthermore, MEMF can be combined with a message-passing algorithm, such as TRWS [Kolmogorov, 2006] to obtain a hybrid algorithm. Potentially, such an algorithm can be designed in a way that, it has desirable characteristics of both algorithms, *i.e.*, good at approximating the true energy (similar to TRWS) while being fast to converge (similar to max-flow).

In the next chapter, we introduce an approximate algorithm for a certain class of non-submodular MRFs which minimizes a multi-label submodular energy at each iteration. As shown in Section 4.6.2.1, this allows us to tackle much larger non-submodular MRFs using MEMF in this approximate technique.

Chapter 4

Iteratively Reweighted Graph-Cut for Multi-Label MRFs with Non-Convex Priors

In the previous chapter, we discussed a memory efficient max-flow algorithm to optimally solve multi-label submodular MRFs (*e.g.*, MRFs with convex priors). In this chapter, we present an approximate graph-cut algorithm for multi-label MRFs with a certain class of non-convex priors. We show that, by iteratively minimizing a multi-label submodular energy function, we can approximately minimize MRFs with robust non-convex priors. Furthermore, we discuss its relationship to the majorize-minimize framework. This chapter is based on our work [Ajanthan et al., 2015] with some extensions.

4.1 Introduction

In this chapter, we introduce an algorithm to minimize the energy of multi-label Markov random fields with non-convex edge priors. As discussed in Chapter 2, in general, minimizing a multi-label MRF energy function is NP-hard. While in rare cases a globally optimal solution can be obtained in polynomial time, *e.g.*, in the presence of convex priors [Ishikawa, 2003], in most scenarios one has to rely on an approximate algorithm (see Chapter 2 for a review). Even though graph-cut-based algorithms [Boykov et al., 2001] have proven successful for specific problems (*e.g.*, *metric* priors), there does not seem to be a single algorithm that performs well with different non-convex priors such as the truncated quadratic, the Cauchy function and the corrupted Gaussian, which are widely acknowledged as highly effective in computer vision.

Here, we propose to fill this gap and introduce an iterative graph-cut-based algorithm to minimize multi-label MRF energies with a certain class of non-convex priors. Our algorithm iteratively minimizes a weighted surrogate energy function that is easier to optimize, with weights computed from the solution at the previous iteration. We show that, under suitable conditions on the non-convex priors, and as long as the weighted surrogate energy can be decreased, our approach guarantees that the true energy decreases at each iteration.

More specifically, we consider MRF energies with arbitrary data terms and where the non-convex priors are concave functions of some convex priors over pairs of nodes. In this scenario, and when the label set is linearly ordered, the solution at each iteration of our algorithm can be obtained by applying the Ishikawa algorithm [Ishikawa, 2003]. Since the resulting solution is optimal, our algorithm guarantees that our MRF energy decreases. Furthermore, our MEMF algorithm described in Chapter 3 can be applied instead of the standard Ishikawa method to tackle large scale problems.

Note that, since our algorithm iteratively approximates the true multi-label MRF energy using a surrogate energy, it can be categorized as a move-making algorithm. Here, the Ishikawa graph construction at each iteration defines the associated move-function (see Section 2.3.3 for details on move-making algorithms). Moreover, our method is inspired by the Iteratively Reweighted Least Squares (IRLS) algorithm which is well-known for continuous optimization. To the best of our knowledge, this is the first time that such a technique is transposed to the MRF optimization scenario.

We demonstrate the effectiveness of our algorithm on the problems of stereo correspondence estimation and image inpainting. Our experimental evaluation shows that our method consistently outperforms other state-of-the-art graph-cut-based algorithms [Boykov et al., 2001; Veksler, 2012], and, in most scenarios, yields lower energy values than TRWS [Kolmogorov, 2006], which was shown to be one of the best-performing multi-label approximate energy minimization methods [Szeliski et al., 2008; Kappes et al., 2015]. Our code is available at https://github.com/tajanthan/irgc.

4.2 Iteratively Reweighted Minimization

Given a set $\mathcal{X}$ and functions $f_a : \mathcal{X} \to \mathcal{D}$ and $h_a : \mathcal{D} \to \mathbb{R}$, where $\mathcal{D} \subset \mathbb{R}$, let us assume that we want to minimize an objective function of the form

$$C_h(\mathbf{x}) = \sum_{a=1}^{N} h_a \circ f_a(\mathbf{x}) \ , \tag{4.1}$$

where $\circ$ denotes the function composition. In addition, without loss of generality, assume that we have a method to minimize a weighted cost function of the form

$$C_w(\mathbf{x}) = \sum_{a=1}^{N} w_a \, f_a(\mathbf{x}) \ . \tag{4.2}$$

For instance, in the IRLS algorithm, $f_a(\mathbf{x})$ is a squared cost.

Our goal is to study the conditions under which C_h can be minimized by iteratively minimizing C_w. To this end, we first give the definition of a supergradient, which we will rely upon in the following discussion.

Definition 4.2.1. Let $\mathcal{D}$ be a subset of $\mathbb{R}$. A *supergradient* of a function $h : \mathcal{D} \to \mathbb{R}$ at a point c is a value $h^s(c) \in \mathbb{R}$ such that

$$h(d) \leq h(c) + (d - c)\, h^s(c) \;, \tag{4.3}$$

for any point $d \in \mathcal{D}$.

A supergradient h^s is called a strict supergradient if the inequality is strict for any point $d \neq c$. If the function is differentiable, then the supergradient at a point is unique and equal to the derivative. A concave function defined on a subset of the real numbers has a supergradient at each interior point.

In [Aftab and Hartley, 2015], the following lemma was provided to study the behavior of iteratively reweighted minimization methods.

Lemma 4.2.1. *Let $h(\cdot)$ be a concave function defined on a subset $\mathcal{D}$ of the real numbers and $h^s(c_a)$ be a supergradient at c_a. If c_a and d_a in $\mathcal{D}$ satisfy*

$$\sum_{a=1}^{N} d_a\, h^s(c_a) \leq \sum_{a=1}^{N} c_a\, h^s(c_a) \;, \tag{4.4}$$

then

$$\sum_{a=1}^{N} h(d_a) \leq \sum_{a=1}^{N} h(c_a) \;. \tag{4.5}$$

If the first inequality is strict, so is the second.

Proof. Since h^s is a supergradient,

$$h(d_a) \leq h(c_a) + (d_a - c_a)\, h^s(c_a) \;, \tag{4.6}$$

for all a. Summing over a gives,

$$\sum_{a=1}^{N} h(d_a) \leq \sum_{a=1}^{N} h(c_a) + \underbrace{\sum_{a=1}^{N} (d_a - c_a)\, h^s(c_a)}_{\leq 0} \;. \tag{4.7}$$

The last sum is non-positive by hypothesis, which completes the proof. □

The proof of this lemma is illustrated in Figure 4.1. Note that Lemma 4.2.1 only considers the case where the function h is the same for all the elements in the sum. This is in contrast with our definition of the cost in Eq. (4.1), where we want to allow h to be indexed on a. To handle this more general scenario, we introduce the following lemma.

Lemma 4.2.2. *Given a set $\mathcal{X}$, functions $f_a : \mathcal{X} \to \mathcal{D}$ and concave functions $h_a : \mathcal{D} \to \mathbb{R}$, with $\mathcal{D} \subset \mathbb{R}$, such that,*

$$\sum_{a=1}^{N} w_a^t\, f_a(\mathbf{x}^{t+1}) \leq \sum_{a=1}^{N} w_a^t\, f_a(\mathbf{x}^t) \;, \tag{4.8}$$

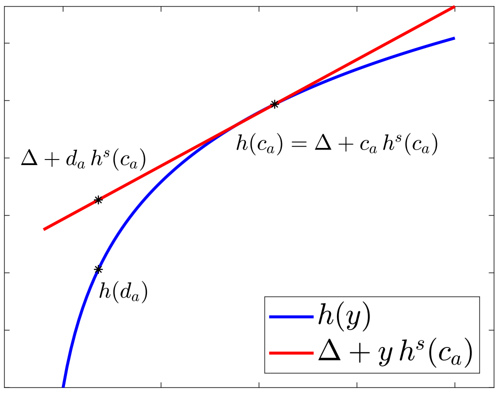


Figure 4.1: *Illustration of Lemma 4.2.1. Here, $\Delta = h(c_a) - c_a\, h^s(c_a)$. Note that, $d_a\, h^s(c_a) \leq c_a\, h^s(c_a)$ implies $h(d_a) \leq h(c_a)$. Furthermore, our iteratively reweighted algorithm, in fact, minimizes an upper bound (shown in red) of $h(y)$ at each iteration. See Section 4.2.1 for more detail.*

where $w_a^t = h_a^s(f_a(\mathbf{x}^t))$ and $\mathbf{x}^t$ is the estimate of $\mathbf{x}$ at iteration t, then

$$\sum_{a=1}^{N} h_a \circ f_a(\mathbf{x}^{t+1}) \leq \sum_{a=1}^{N} h_a \circ f_a(\mathbf{x}^t)\ . \tag{4.9}$$

If the first inequality is strict, so is the second.

Proof. Let us define $c_a = f_a(\mathbf{x}^t)$ and $d_a = f_a(\mathbf{x}^{t+1})$. Since h_a^s is a supergradient,

$$h_a(d_a) \leq h_a(c_a) + (d_a - c_a)\, h_a^s(c_a)\ , \tag{4.10}$$

for all a. Summing over a gives,

$$\sum_{a=1}^{N} h_a(d_a) \leq \sum_{a=1}^{N} h_a(c_a) + \sum_{a=1}^{N} (d_a - c_a)\, h_a^s(c_a)\ . \tag{4.11}$$

The sum $\sum_{a=1}^{N} (d_a - c_a)\, h^s(c_a) = \sum_{a=1}^{N} w_a^t\, f_a(\mathbf{x}^{t+1}) - \sum_{a=1}^{N} w_a^t\, f_a(\mathbf{x}^t)$ is non-positive by hypothesis, which completes the proof. □

It is important to note that this lemma holds for discrete subsets $\mathcal{D}$, as well as continuous ones, and that the functions h_a do not need to be differentiable.

Therefore, for concave functions h_a, by choosing the supergradients of h_a as weights at each iteration, we can minimize the objective function $C_h(\mathbf{x})$ of Eq. (4.1) by iteratively minimizing the cost $C_w(\mathbf{x})$ of Eq. (4.2). This general procedure is summarized in Algorithm 4.1.

Algorithm 4.1 is applicable to any minimization problem, as long as the objective function takes the form of Eq. (4.1) with concave functions h_a. Furthermore, to minimize the surrogate cost of Eq. (4.2), any algorithm (either exact or approximate) can be used, as long as it decreases this cost.

Algorithm 4.1 Iteratively Reweighted Minimization

```
$C_h(\mathbf{x}) \leftarrow \sum_{a=1}^{N} h_a \circ f_a(\mathbf{x})$          ▷ Concave functions $h_a$
Initialize $\mathbf{x}$
**repeat**
    $w_a^t \leftarrow h_a^s(f_a(\mathbf{x}^t))$
    $\mathbf{x}^{t+1} \leftarrow \underset{\mathbf{x}}{\text{argmin}} \sum_{a=1}^{N} w_a^t f_a(\mathbf{x})$
**until** convergence of $C_h(\mathbf{x})$
**return** $\mathbf{x}^{t+1}$
```

4.2.1 Relationship to the Majorize-Minimize Framework

In the Majorize-Minimize (MM) framework [Hunter and Lange, 2004], the original function is minimized by iteratively minimizing a *majorizing* function, which is defined below.

Definition 4.2.2. Let $C(\mathbf{y})$ and $\hat{C}(\mathbf{y} \mid \mathbf{y}^t)$ be two real-valued functions, where $\mathbf{y}^t$ is the estimate of $\mathbf{y}$ at iteration t and the form $\hat{C}(\mathbf{y} \mid \mathbf{y}^t)$ depends on $\mathbf{y}^t$. If

$$\begin{aligned} \hat{C}(\mathbf{y} \mid \mathbf{y}^t) &\geq C(\mathbf{y}) \quad \forall\, \mathbf{y}\,, \\ \hat{C}(\mathbf{y}^t \mid \mathbf{y}^t) &= C(\mathbf{y}^t)\,, \end{aligned} \tag{4.12}$$

then $\hat{C}(\mathbf{y} \mid \mathbf{y}^t)$ is said to *majorize* the function $C(\mathbf{y})$.

In our setting, our algorithm minimizes the true cost $C(\mathbf{x}) = C_h(\mathbf{x})$ by iteratively minimizing a surrogate cost $\tilde{C}(\mathbf{x}) = C_w(\mathbf{x})$. For simplicity, by defining $y_a = f_a(\mathbf{x})$, we can write the surrogate cost as

$$\tilde{C}(\mathbf{y}) = \sum_{a=1}^{N} h_a^s(y_a^t)\, y_a\,. \tag{4.13}$$

We would like to point out that this surrogate cost is a majorizing function of the true cost, up to a constant.

Theorem 4.2.1. *Let $C(\mathbf{y})$ be the true cost and $\tilde{C}(\mathbf{y})$ (Eq. (4.13)) be the surrogate cost. The function $\hat{C}(\mathbf{y} \mid \mathbf{y}^t)$ defined as*

$$\hat{C}(\mathbf{y} \mid \mathbf{y}^t) = \tilde{C}(\mathbf{y}) + \underbrace{\sum_{a=1}^{N} \left(h_a(y_a^t) - h_a^s(y_a^t)\, y_a^t\right)}_{\text{constant}}\,, \tag{4.14}$$

majorizes $C(\mathbf{y})$.

Proof.

$$
\begin{aligned}
\hat{C}(\mathbf{y} \mid \mathbf{y}^t) &= \tilde{C}(\mathbf{y}) + \sum_{a=1}^{N} \left(h_a(y_a^t) - h_a^s(y_a^t)\, y_a^t\right) , \qquad (4.15)\\
&= \sum_{a=1}^{N} \left(h_a(y_a^t) + (y_a - y_a^t)\, h_a^s(y_a^t)\right) ,\\
&\geq \sum_{a=1}^{N} h_a(y_a) \quad \text{since } h_a^s \text{ is a supergradient} ,\\
&= C(\mathbf{y}) .
\end{aligned}
$$

Note that the inequality is tight when $\mathbf{y} = \mathbf{y}^t$. Hence, $\hat{C}(\mathbf{y} \mid \mathbf{y}^t)$ majorizes $C(\mathbf{y})$. □

Therefore, our iteratively reweighted algorithm can also be thought of as a majorize-minimize algorithm.

4.3 An Iteratively Reweighted Scheme for MRFs

Recall that our goal is to tackle the problem of MRF energy minimization. Here, we show how this can be achieved by exploiting Algorithm 4.1.

To this end, let us first consider an MRF energy in its most general form (see Section 2.1) as a summation of *clique potentials*,

$$E(\mathbf{x}) = \sum_{a=1}^{|\mathcal{C}|} \theta_a(\mathbf{x}_a) , \qquad (4.16)$$

where $\mathcal{C}$ is the set of *cliques* in the graph (*i.e.*, the groups of connected nodes). Here, $\mathbf{x}_a$ represents the set of variables corresponding to the nodes in clique a, and $\theta_a : \mathcal{L}^{|\mathbf{x}_a|} \to \mathbb{R}$ is the energy (or potential) function associated with clique a, where $\mathcal{L}$ is the label set.

Let us now assume that each potential function can be written as

$$\theta_a(\mathbf{x}_a) = h_a \circ f_a(\mathbf{x}_a) , \qquad (4.17)$$

where h_a is a concave function and f_a an arbitrary one. This lets us rewrite the MRF energy of Eq. (4.16) as

$$E(\mathbf{x}) = \sum_{a=1}^{|\mathcal{C}|} h_a \circ f_a(\mathbf{x}_a) , \qquad (4.18)$$

which has the form of Eq. (4.1)[1]. Therefore, we can employ Algorithm 4.1 to minimize

[1]Note that $f_a(\mathbf{x}_a)$ can be equivalently written as $f_a(\mathbf{x})$, where the variables $x_i \notin \mathbf{x}_a$ (*i.e.*, not in clique a) simply have no effect on the function.

$E(\mathbf{x})$, and iteratively minimize the surrogate energy

$$\tilde{E}(\mathbf{x}) = \sum_{a=1}^{|\mathcal{C}|} w_a f_a(\mathbf{x}_a) \ , \tag{4.19}$$

with weight w_a taken as the supergradient of h_a evaluated using the previous estimate of $\mathbf{x}$.

It is important to note, however, that for this algorithm to be effective, the minimization of $\tilde{E}(\mathbf{x})$ at each iteration must be relatively easy, and at least guarantee that the surrogate energy decreases. Furthermore, while in practice any existing MRF energy minimization algorithm (either exact or approximate) can be utilized to minimize $\tilde{E}(\mathbf{x})$, the quality of the overall solution found by our algorithm may vary accordingly. In the next section, we discuss a special case of this general MRF energy minimization algorithm, which, as depicted in our experiments, is effective in many scenarios.

4.4 Iteratively Reweighted Graph-Cut

In this section, we introduce an iterative algorithm for the case of multi-label MRFs with pairwise node interactions. In particular, we propose to make use of the Ishikawa method [Ishikawa, 2003] at each iteration of our algorithm. The Ishikawa method yields an optimal solution under the following two conditions[2]: 1) the label set must be ordered; 2) the pairwise potential must be a convex function of the label difference. In practice, such convex priors have limited power due to their poor ability to model noise. In contrast, while still relying on the first condition, our algorithm allows us to generalize to non-convex priors, and in particular to robust norms that have proven effective in computer vision. Furthermore, our MEMF algorithm described in Chapter 3 can be used at each iteration to tackle large scale problems.

4.4.1 MRF with Pairwise Interactions

In an MRF with pairwise node interactions, the energy can be expressed as

$$E(\mathbf{x}) = \sum_{i \in \mathcal{V}} \theta_i(x_i) + \sum_{(i,j) \in \mathcal{E}} \theta_{ij}(x_i, x_j) \ , \tag{4.20}$$

where θ_i and θ_{ij} denote the unary potentials and pairwise potentials, respectively. Here, $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of edges in the MRF graph.

As discussed in Section 4.3, to be able to make use of Algorithm 4.1, we need to have potential functions of the form given in Eq. (4.17). Under this assumption, we

[2]In fact, as discussed in Section 2.3.2, the Ishikawa algorithm can optimally solve multi-label submodular MRFs.

can then rewrite the energy of Eq. (4.20) as

$$E(\mathbf{x}) = \sum_{i \in \mathcal{V}} h_u \circ f_i(x_i) + \sum_{(i,j) \in \mathcal{E}} h_b \circ f_{ij}(x_i, x_j) \,, \tag{4.21}$$

where h_u and h_b are concave functions.

Following Algorithm 4.1, we minimize this energy by iteratively minimizing a surrogate energy of the form (at iteration $t+1$)

$$\tilde{E}(\mathbf{x}) = \sum_{i \in \mathcal{V}} h_u^s \left(f_i(x_i^t)\right) f_i(x_i) + \sum_{(i,j) \in \mathcal{E}} h_b^s \left(f_{ij}(x_i^t, x_j^t)\right) f_{ij}(x_i, x_j) \,,$$

where h_u^s and h_b^s are the supergradients of h_u and h_b, respectively, and x_i^t denotes the estimate of x_i at iteration t.

Since our goal is to employ the Ishikawa algorithm to minimize $\tilde{E}(\mathbf{x})$, we need to define the different functions h_u, h_b, f_i and f_{ij} so as to satisfy the requirements of this algorithm. To this end, for the unary potential, we choose h_u to be the identity function. That is,

$$\theta_i(x_i) = h_u \circ f_i(x_i) = f_i(x_i) \,. \tag{4.22}$$

This implies that no reweighting is required for the unary potentials, since the supergradient of h_u is always 1. The Ishikawa algorithm having no specific requirement on the form of the unary term, f_i can be any arbitrary function.

In contrast, for the pairwise potentials, the Ishikawa algorithm requires f_{ij} to be a convex function of the label difference. That is, for a convex function g defined on a subset of $\mathbb{R}$,

$$f_{ij}(x_i, x_j) = g(|x_i - x_j|) \,. \tag{4.23}$$

Such a pairwise term is referred to as a convex prior in Section 2.3.2. In addition, because the energy $\tilde{E}(\mathbf{x})$ depends on a *weighted* sum of pairwise terms, we need the weights to satisfy some conditions. More precisely, to be able to use the max-flow algorithm within the Ishikawa method, the weights need to be non-negative. Since these weights are computed from the supergradient of h_b, this translates into a requirement for h_b to be *non-decreasing*. Note that, in the context of smoothness potentials in an MRF, this requirement comes at virtually no cost, since we hardly expect the potentials to decrease as the label difference increases.

Under these conditions, the surrogate energy to be minimized by the Ishikawa method at each iteration of our algorithm can be written as

$$\tilde{E}(\mathbf{x}) = \sum_{i \in \mathcal{V}} f_i(x_i) + \sum_{(i,j) \in \mathcal{E}} w_{ij}^t \, g(|x_i - x_j|) \,, \tag{4.24}$$

where g is a convex function, and $w_{ij}^t = h_b^s\Big(f_{ij}(x_i^t, x_j^t)\Big)$, with h_b a concave, non-decreasing function. In the first iteration, we set the weights w_{ij}^0 to some constant

Algorithm 4.2 Iteratively Reweighted Graph-Cut (IRGC)

```
$E(\mathbf{x}) \leftarrow \sum_{i\in\mathcal{V}} f_i(x_i) + \sum_{(i,j)\in\mathcal{E}} h_b \circ g\left(|x_i - x_j|\right)$    ▷ $g$ convex, $h_b$ non-decreasing concave
$w^0_{ij} \leftarrow 0.5$    ▷ Initialize the weights
repeat
    if $t \neq 0$ then
        $w^t_{ij} \leftarrow h^s_b\left(g\left(\left|x^t_i - x^t_j\right|\right)\right)$    ▷ Update the weights
    end if
    $\mathbf{x}^{t+1} \leftarrow \operatorname{argmin}_{\mathbf{x}} \sum_{i\in\mathcal{V}} f_i(x_i) + \sum_{(i,j)\in\mathcal{E}} w^t_{ij}\, g(|x_i - x_j|)$    ▷ Ishikawa algorithm
until $E(\mathbf{x}^{t+1}) = E(\mathbf{x}^t)$    ▷ Convergence of $E(\mathbf{x})$
return $\mathbf{x}^{t+1}$
```

value[3] to make the algorithm independent of any initial estimate $\mathbf{x}^0$. Our overall Iteratively Reweighted Graph-Cut (IRGC) algorithm is summarized in Algorithm 4.2.

Ishikawa Algorithm. Here, we briefly give the Ishikawa graph construction and refer the interested reader to Section 2.3.2 for more detail. Let the label set $\mathcal{L} = \{0, 1, \ldots, \ell - 1\}$. The Ishikawa graph is an st-graph $\hat{\mathcal{G}} = (\hat{\mathcal{V}} \cup \{0, 1\}, \hat{\mathcal{E}}^+, \boldsymbol{\phi})$. Here, there are ℓ nodes corresponding to each pixel $i \in \mathcal{V}$, denoted with $U_{i:\lambda}$ for all $\lambda \in \mathcal{L}$, arranged in a column. In addition, it has directed edges connecting each node, whose capacities are denoted with $\phi_{ij:\lambda\mu}$. For convenience, the edge capacities along each column i is denoted with $\phi_{i:\lambda} = \phi_{ii:\lambda+1\lambda}$. See Figure 4.2 for an Ishikawa graph corresponding to convex priors.

As discussed in Section 2.3.2, for convex priors, the Ishikawa edge capacities take the following form,

$$\begin{aligned} \phi_{i:\lambda} &= f_i(\lambda)\,, \\ \phi_{ij:\lambda\mu} &= \begin{cases} 0 & \text{if } \lambda < \mu \\ \frac{w^t_{ij}}{2} g''(|\lambda - \mu|) & \text{if } \lambda = \mu \\ w^t_{ij}\, g''(|\lambda - \mu|) & \text{if } \lambda > \mu \end{cases}\,, \end{aligned} \tag{4.25}$$

where $g''(|\delta|) = g(|\delta + 1|) + g(|\delta - 1|) - 2\,g(|\delta|)$, which is non-negative for a convex function g.

In our scenario, with our condition that w^t_{ij} be non-negative, the Ishikawa graph contains no negative edges. Therefore, the global minimum of the corresponding energy can be found in polynomial time using the max-flow algorithm. Note that, for a sparsely connected graph, *e.g.*, 4 or 8-connected neighbourhood, the memory requirement of a general Ishikawa graph is $\mathcal{O}(n\,\ell^2)$, where $n = |\mathcal{V}|$ and $\ell = |\mathcal{L}|$. However if g is linear, then the memory required drops to $\mathcal{O}(n\,\ell)$.

[3]We set $w^0_{ij} = \epsilon$, where $0 \leq \epsilon \leq 1$, so that the effect of the edge terms is smaller for the first estimate. In our experiments, we found $\epsilon = 0.5$ to work well and thus always use this value.

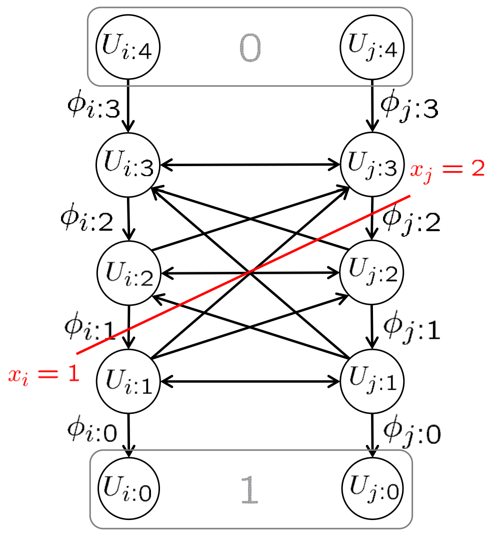


Figure 4.2: *Example of an Ishikawa graph for convex priors. Note that only upward cross edges are non-zero. The graph incorporates edges with infinite capacity from $U_{i:\lambda}$ to $U_{i:\lambda+1}$, not shown in the graph. Here the cut corresponds to the labeling $\mathbf{x} = \{1, 2\}$ where the label set $\mathcal{L} = \{0, 1, 2, 3\}$.*

4.4.2 Choice of Functions g and h_b

While, in Section 4.4.1, we have defined conditions on the functions g and h_b (*i.e.*, g convex and h_b concave, non-decreasing) for our algorithm to be applicable with the Ishikawa algorithm, these conditions still leave us a lot of freedom in the actual choice of these functions. Here, we discuss several such choices, with special considerations on the memory requirement of the resulting algorithm.

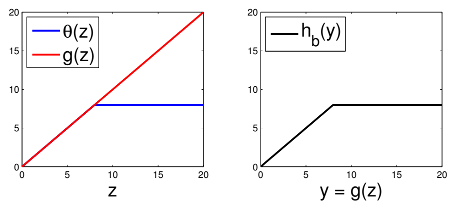
In the context of computer vision problems with ordered label sets, *e.g.*, stereo and inpainting, it is often important to make use of robust estimators as pairwise potentials to better account for discontinuities, or outliers. Many such robust estimators belong to the family of functions with a single inflection point in $\mathbb{R}^+$. In other words, they can be generally defined as non-decreasing functions $\theta_{ij}(|\lambda - \mu|)$, such that for a given $\kappa \geq 0$, $\theta(z)$ is convex if $z \leq \kappa$, and concave otherwise[4]. Such functions include the truncated linear, the truncated quadratic and the Cauchy function $\theta(z) = \kappa^2/2 \log\left(1 + (z/\kappa)^2\right)$ [Hartley and Zisserman, 2003]. Note that any convex or concave function on $\mathbb{R}^+$ also belongs to this family.

For a given such function θ, according to our algorithm, we need to write

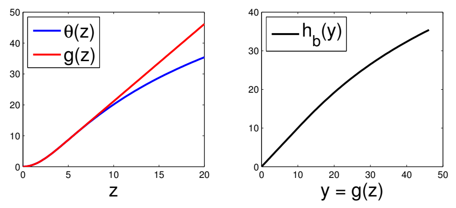
$$\theta(z) = h_b \circ g(z) \ , \tag{4.26}$$

with a concave, non-decreasing function h_b and a convex function g. Note that the Ishikawa graph structure is determined by the function g. Therefore, to make the graph as sparse as possible, and thus limit the required memory, we need to choose

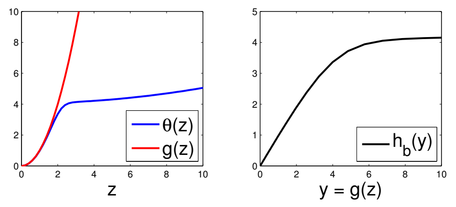
[4]Here $z = |\lambda - \mu|$, where $\lambda, \mu \in \mathcal{L}$.



(a) θ - Truncated linear



(b) θ - Cauchy function



(c) θ - Corrupted Gaussian

Figure 4.3: *Plots of θ, g and h_b with $\theta(z) = h_b \circ g(z)$, when θ is* ***(a)*** *the truncated linear,* ***(b)*** *the Cauchy function and* ***(c)*** *the corrupted Gaussian. Here g is convex and h_b is concave. In* ***(a)*** *and* ***(b)****, the functions g and h_b are derived from Table 4.1. In* ***(c)****, $g(z) = z^2$ and $h_b(y) = \theta(\sqrt{y})$.*

	$y = g(z)$		$h_b(y)$
$z \leq \kappa$	$\theta(z)$	$y \leq \theta(\kappa)$	y
$z \geq \kappa$	$\theta'(\kappa)(z-\kappa) + \theta(\kappa)$	$y \geq \theta(\kappa)$	$\theta\left(\frac{y+\kappa\,\theta'(\kappa)-\theta(\kappa)}{\theta'(\kappa)}\right)$

Table 4.1: *Functions g and h_b corresponding to a given $\theta(z)$, such that $\theta(z)$ is convex if $z \leq \kappa$ and concave otherwise. It can easily be verified that $\theta(z) = h_b \circ g(z)$, and that g is convex and h_b is concave, as well as that both functions are non-decreasing, because θ is non-decreasing. Here $\theta'(\kappa)$ is the derivative of θ at κ, or its left derivative $\theta'(\kappa^-)$ if θ is not differentiable at κ. See Figure 4.3(a-b) for example plots.*

g such that the second order difference $g''(z)$ is zero for as many values z as possible. Table 4.1 gives the functions g and h_b such that $g''(z)$ is zero $\forall z \geq \kappa$ while satisfying Eq. (4.26) and the necessary conditions on h_b and g. Figure 4.3(a-b) provide the plots corresponding to the truncated linear and Cauchy function. For a function g derived according to Table 4.1, the memory requirement of the Ishikawa graph is $\mathcal{O}(n\,\kappa\,\ell)$.

Note that our method is not limited to the family of functions described above. As an example, we consider the case of another robust estimator, the corrupted Gaussian function $\theta_G(z) = -\log(\alpha \exp(-z^2) + (1-\alpha)\exp(-z^2/\beta^2)/\beta)$ [Hartley and Zisserman, 2003], which does not follow the definitions of the functions described before. However, since $\theta_G(\sqrt{\cdot})$ is concave, we can minimize $\theta_G(z)$ by choosing $g(z) = z^2$ and $h_b(y) = \theta_G(\sqrt{y})$. The corresponding plots for the corrupted Gaussian are provided in Figure 4.3(c).

4.4.3 Hybrid Strategy

While our algorithm guarantees that the energy value decreases at each iteration, it remains prone to getting trapped in local minima (with respect to the iterative reweighting scheme). Here, we propose a hybrid optimization strategy that combines IRGC with a different minimization technique, and helps us escape from some of the local minima of the energy.

In particular, here we make use of α-expansion [Boykov et al., 2001] as an additional minimization technique. At each iteration of our algorithm, instead of updating $\mathbf{x}^t \to \mathbf{x}^{t+1}$ in one step, our hybrid strategy performs the following steps:

1. Update $\mathbf{x}^t \to \mathbf{x}'$ by minimizing the surrogate energy using the Ishikawa algorithm.

2. Improve the new estimate $\mathbf{x}' \to \mathbf{x}^{t+1}$ using one pass of α-expansion with the true energy, such that $E(\mathbf{x}^{t+1}) \leq E(\mathbf{x}')$.

For non-metric pairwise potentials, for which regular α-expansion does not apply, we truncate the non-submodular terms as suggested in [Rother et al., 2005]. Note that this still guarantees that the energy will decrease. We found that the variety in the optimization strategy arising from this additional α-expansion step was effective to overcome local minima. Since both algorithms guarantee to decrease the energy $E(\mathbf{x})$ at each step, our hybrid algorithm also decreases the energy at each iteration. In our experiments, we refer to this hybrid algorithm as IRGC+expansion.

Note that other methods, such as $\alpha\beta$-swap, or any algorithm that guarantees to improve the given estimate can be employed. Alternatively, one could exploit a fusion move strategy [Lempitsky et al., 2010] to combine the estimates obtained by the two different algorithms. However, this would come at an additional computation cost, and, we believe, goes beyond the scope of this work.

4.5 Related Work

We review the past work on two different aspects of our work in order to highlight our contributions.

4.5.1 Approximate MRF Optimization Algorithms

As discussed in Chapter 2, approximate MRF energy minimization methods can be categorized into two groups. The first class of such methods consists of move-making techniques (Section 2.3.3) that were inspired by the success of the graph cut algorithm at solving binary problems in computer vision. These techniques include α-expansion, $\alpha\beta$-swap [Boykov et al., 2001] and multi-label moves [Veksler, 2012; Torr and Kumar, 2009; Jezierska et al., 2011]. The core idea of these methods is to reduce the original multi-label problem to a sequence of binary graph cut problems, called surrogate problems. Each graph cut problem can then be solved either optimally by the max-flow algorithm [Boykov and Kolmogorov, 2004] if the resulting binary energy is submodular, or approximately via a roof dual technique [Boros and Hammer, 2002] otherwise. Among these methods, the closest one to our IRGC algorithm is the multi-label smooth swap introduced in [Veksler, 2012]. This method minimizes a convex upper bound similar to IRGC, but only applicable to truncated convex priors. In spite of that, as we mentioned previously, and evidenced by our experiments, there does not seem to be a single move-making algorithm that perform well with different robust non-convex priors.

The second type of approximate energy minimization methods consists of message passing algorithms (usually based on the LP relaxation, see Section 2.3.4), such as belief propagation (BP) [Felzenszwalb and Huttenlocher, 2006], tree reweighted message passing (TRW) [Wainwright et al., 2005; Kolmogorov, 2006] and the dual decomposition-based approach of [Komodakis et al., 2011], which TRW is a special case of. Among them, the TRWS algorithm [Kolmogorov, 2006] was shown to be one of the best performing method [Szeliski et al., 2008; Kappes et al., 2015] and we compare it against our algorithm with favourable results.

4.5.2 IRLS-based Methods

As mentioned earlier, our algorithm is inspired by the IRLS method. Recently, several methods similarly motivated by the IRLS have been proposed to minimize different objective functions. For instance, in [Aftab et al., 2015], the L_q norm (for $1 \leq q < 2$) was minimized by iteratively minimizing a weighted L_2 cost function. In [Ochs et al., 2013], an iterated L_1 algorithm was introduced to optimize non-convex functions that are the sum of convex data terms and concave smoothness terms. More recently, a general formulation (not restricted to weighted L_2 or L_1 minimization) was studied, together with the conditions under which such iteratively reweighted algorithms ensure the cost to decrease [Aftab and Hartley, 2015]. In our work, the iteratively reweighted idea is transposed to the discrete MRF optimization scenario.

4.6 Experiments

We evaluated our algorithm on the problems of stereo correspondence estimation and image inpainting. In those cases, the pairwise potentials typically depend on additional constant weights, and can thus be written as

$$\theta_{ij}(x_i, x_j) = \gamma_{ij}\,\theta(|x_i - x_j|)\ , \tag{4.27}$$

where γ_{ij} are the constant weights. As long as $\gamma_{ij} \geq 0$, our algorithm is unaffected by these weights, in the sense that we can simply multiply our weights w_{ij}^t by these additional constant weights. Note that since the main purpose of this work is to evaluate the performance of our algorithm on different MRF energy functions, we used different smoothing costs $\theta(\cdot)$ for different problem instances without tuning the weights γ_{ij} for the specific smoothing costs.

Problem	γ_{ij}	$\theta(\cdot)$	κ
Teddy	$\begin{cases} 30 & \text{if } \nabla_{ij} \leq 10 \\ 10 & \text{otherwise} \end{cases}$	Truncated linear	8
Map	4		6
Sawtooth	20	Truncated quadratic	3
Venus	50		3
Cones	10	Cauchy function	8
Tsukuba	$\begin{cases} 40 & \text{if } \nabla_{ij} \leq 8 \\ 20 & \text{otherwise} \end{cases}$		2

Table 4.2: *Pairwise potential $\theta_{ij}(x_i, x_j) = \gamma_{ij}\,\theta(|x_i - x_j|)$ used for the stereo problems (see Section 4.6.1). Here $\theta(z)$ is convex if $z \leq \kappa$ and concave otherwise, and ∇_{ij} denotes the absolute intensity difference between the pixels i and j in the left image.*

We compare our results with those of α-expansion, $\alpha\beta$-swap [Boykov et al., 2001], multi-label swap [Veksler, 2012] and TRWS [Kolmogorov, 2006]. For fairer comparison, we improved the results of TRWS using α-expansion which is denoted as TRWS+expansion. For α-expansion, we used the max-flow algorithm when the pairwise potentials were *metric*, and the QPBOP algorithm [Boros and Hammer, 2002; Rother et al., 2007] (denoted as α-expansionQ) otherwise. In the latter case, if a node in the binary problem is unlabeled then the previous label is retained. For our comparison, we used the publicly available implementation of α-expansion, $\alpha\beta$-swap, QPBO and TRWS, and implemented the multi-label swap algorithm as described in [Veksler, 2012]. In our algorithm, the max-flow implementation[5] of [Boykov and Kolmogorov, 2004] is used to find the min-cut solution on the Ishikawa graph.

All the algorithms were initialized by assigning the label 0 to all the nodes (note that in [Veksler, 2012] multi-label swap was initialized using α-expansion). For multi-label swap the parameter t was fixed to 2 in all our experiments (see [Veksler, 2012] for details). The energy values presented in the following sections were obtained at convergence of the different algorithms, except for TRWS which we ran for 100 iterations and chose the best energy value[6]. All our experiments were conducted on a 3.4GHz i7-4770 CPU with 16 GB RAM, and no effort was made to exploit the multiple cores.

4.6.1 Stereo Correspondence

Given a pair of rectified images (one left and one right), stereo correspondence estimation aims to find the disparity map, which specifies the horizontal displacement of each pixel between the two images with respect to the left image. For this task, we employed the same six instances from the Middlebury dataset [Scharstein and

[5]The EIBFS algorithm [Goldberg et al., 2015] is published after this work and EIBFS is shown to be faster than the BK method on dense graphs.

[6]While the energy of TRWS decreases slightly by running more iterations, the algorithm becomes very slow.

Algorithm	Teddy		Map	
	E[10^3]	T[s]	E[10^3]	T[s]
$\alpha\beta$-swap	2708.1	35	149.5	2
α-expansion	2664.6	21	144.4	2
Multi-label swap	5502.3	236	470.6	13
TRWS	2652.7	318	143.0	34
TRWS+expansion	**2646.8**	326	**142.9**	35
IRGC	2687.8	65	144.0	9
IRGC+expansion	2650.3	44	143.2	4

Table 4.3: *Comparison of the minimum energies (E) and execution times (T) for stereo problems with truncated linear prior (see Section 4.6.1). IRGC+expansion found the lowest energy or virtually the same energy as TRWS+expansion and it was 8 times faster than TRWS+expansion.*

Algorithm	Sawtooth		Venus		Cones		Tsukuba	
	E[10^3]	T[s]	E[10^3]	T[s]	E[10^3]	T[s]	E[10^3]	T[s]
$\alpha\beta$-swap	1079.5	7	3219.7	8	4489.9	135	409.1	6
α-expansionQ	1067.5	15	3201.5	16	2480.7	183	403.3	8
Multi-label swap	1660.6	103	5740.1	163	-	-	-	-
TRWS	1038.8	52	3098.6	50	2304.8	311	395.8	19
TRWS+expansion	**1034.4**	56	3083.4	53	2303.7	322	**395.5**	21
IRGC	1042.1	93	3081.4	49	**2301.4**	397	397.3	20
IRGC+expansion	1034.9	32	**3078.8**	26	**2301.4**	204	396.1	14

Table 4.4: *Comparison of the minimum energies (E) and execution times (T) for stereo problems with semi-metric priors (see Section 4.6.1). IRGC+expansion found the lowest energy or virtually the same energy as TRWS+expansion. Note that IRGC outperformed all other graph-cut-based algorithms and found a lower energy than TRWS for Venus and Cones.*

Szeliski, 2002, 2003] as in Chapter 3: Teddy, Map, Sawtooth, Venus, Cones and Tsukuba. For Tsukuba and Venus, we used the unary potentials of [Szeliski et al., 2008], and for the other cases, those of [Birchfield and Tomasi, 1998]. The pairwise potentials are summarized in Table 4.2. Note that we do not explicitly model occlusions, which should be handled by our robust potentials.

The final energies and execution times corresponding to the stereo problems are summarized in Tables 4.3 and 4.4. The disparity maps found using our IRGC+expansion algorithm and energy vs time plots of the algorithms for some of the problems are shown in Figure 4.4 and Figure 4.5(a-c), respectively. Note that, in most cases, IRGC outperforms the other graph-cut-based algorithms. Note also that IRGC+expansion yields the lowest energy or an energy that is virtually the same as the lowest one.

To illustrate the fact that our algorithm can also exploit priors that are not first convex and then concave, we employed a corrupted Gaussian pairwise potential

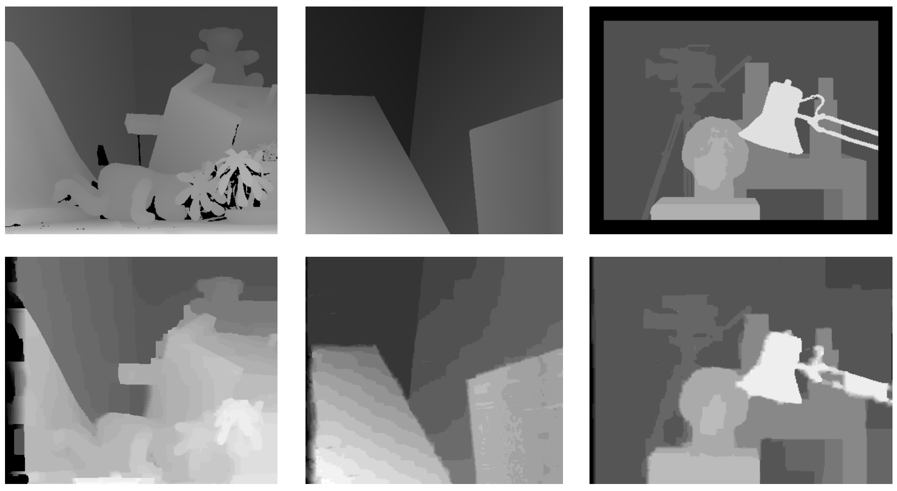

(a) Teddy, Truncated linear (b) Venus, Truncated quadratic (c) Tsukuba, Cauchy function

Figure 4.4: *Disparity maps obtained with IRGC+expansion (see Section 4.6.1). The corresponding ground-truth is shown above each disparity map.*

(with parameters $\alpha = 0.75$ and $\beta = 50$) on the Tsukuba stereo pair, and the results are shown in Table 4.5.

4.6.2 Image Inpainting

Image inpainting tackles the problem of filling in the missing pixel values of an image, while simultaneously denoising the observed pixel values. In our experiments, as in Chapter 3, we used the Penguin and House images employed in [Szeliski et al., 2008]. Due to memory limitation, however, we down-sampled the labels from 256 to 128 for Penguin and from 256 to 64 for House. We used the same unary potential as in [Szeliski et al., 2008], *i.e.*, $f_i(x_i) = (I_i - x_i)^2$ if the intensity I_i is observed, and $f_i(x_i) = 0$ otherwise. As pairwise potentials, we employed the truncated quadratic cost $\theta_{ij}(x_i, x_j) = \gamma_{ij} \min\left((x_i - x_j)^2, \kappa^2\right)$. For Penguin, $\gamma_{ij} = 20$ and $\kappa = 10$, and for House $\gamma_{ij} = 5$ and $\kappa = 15$. The final energies and execution times are summarized in Table 4.6, with the inpainted images shown in Figure 4.6. Furthermore, in Figure 4.5d, we show the energy as a function of time for the different algorithms. Note that, for both images, our IRGC+expansion method outperforms other graph-cut-based algorithms and performs similarly to TRWS+expansion.

4.6.2.1 Tackling Large Scale Problems

Since the MEMF algorithm (see Chapter 3) can simply replace standard max-flow in Ishikawa-type graphs, we replaced the BK method with our MEMF procedure in

Algorithm	Tsukuba	
	E[10^3]	T[s]
$\alpha\beta$-swap	568.3	10
α-expansionQ	555.2	10
TRWS	550.0	20
TRWS+expansion	**548.9**	24
IRGC	614.3	55
IRGC+expansion	549.3	40

Table 4.5: *Comparison of the minimum energies (E) and execution times (T) on Tsukuba with a corrupted Gaussian prior (see Section 4.6.1). While IRGC was trapped in a local minimum, IRGC+expansion found a lower energy than TRWS.*

Algorithm	House		Penguin	
	E[10^3]	T[s]	E[10^3]	T[s]
$\alpha\beta$-swap	2488.9	14	4562.8	10
α-expansionQ	2510.0	531	4486.4	18
Multi-label swap	**2399.9**	1457	4520.6	395
TRWS	2400.1	113	4269.5	138
TRWS+expansion	2400.0	116	**4230.7**	141
IRGC	**2399.9**	155	4696.9	1045
IRGC+expansion	**2399.9**	108	4238.9	222

Table 4.6: *Comparison of minimum energies (E) and execution times (T) for the truncated quadratic prior on two inpainting problems (see Section 4.6.2). On House, IRGC and multi-label swap also achieved the same lowest energy, but the latter was roughly 15 times slower than IRGC+expansion. On Penguin, while IRGC was trapped in a local minimum, IRGC+expansion was able to find a lower energy than TRWS and yield an energy similar to TRWS+expansion.*

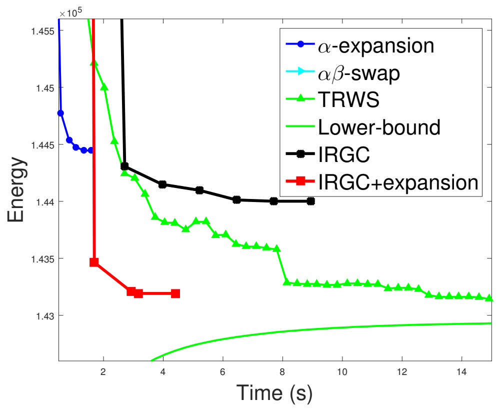
the IRGC algorithm[7]. This lets us tackle much larger problems. In particular, we can now compute inpainting results using all 256 labels, as opposed to the down-sampled label sets. The results of the IRGC+expansion algorithm, with the BK method and with MEMF are summarized in Table 4.7.

4.6.3 Summary

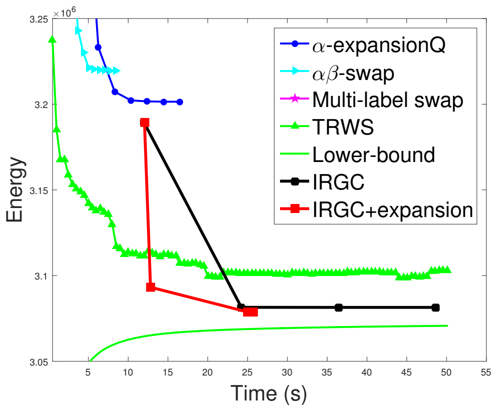
To evaluate the quality of the minimum energies, we followed the strategy of [Szeliski et al., 2008], which makes use of the lower bound found by TRWS. This quality measure is computed as

$$Q = \frac{E - E_b}{E_b} 100\% \,, \tag{4.28}$$

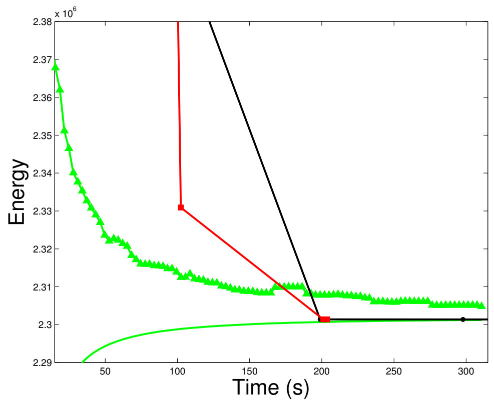
[7]In fact, the MEMF algorithm is developed after this work and hence, the BK method is used to obtain the min-cut solution of the Ishikawa graph.



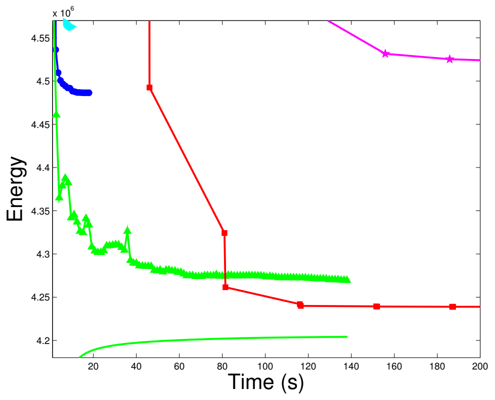
(a) Map, Truncated linear



(b) Venus, Truncated quadratic



(c) Cones, Cauchy function



(d) Penguin, Truncated quadratic

Figure 4.5: *Energy vs time (seconds) plots for the algorithms for* ***(a) - (c)*** *some stereo problems (see Section 4.6.1) and* ***(d)*** *an inpainting problem (see Section 4.6.2). The plots are zoomed-in to show the finer details. IRGC+expansion algorithm outperformed all the other algorithms and found the lowest energy within 2 – 5 iterations. IRGC found a lower energy than α-expansion for Map.*

where E_b is the largest lower bound of TRWS and E is the minimum energy found by an algorithm. In Table 4.8, we compare the resulting values of our algorithms with TRWS, which, from the previous surveys [Szeliski et al., 2008; Kappes et al., 2015] was found to be the best-performing baseline. Note that our IRGC+expansion algorithm yields a better quality measure than TRWS on average. TRWS+expansion yields a slightly better average quality measure than IRGC+expansion, namely 0.1873%. Note, however, that our algorithm was 1.5 – 8 times faster than TRWS+expansion on stereo problems, except for the corrupted Gaussian prior.

4.6.4 IRGC Analysis

To study the behavior of our algorithm for different choices of $\theta(\cdot)$ (*i.e.*, pairwise prior) and κ (*i.e.*, parameter of the prior), we employed the Tsukuba pair for stereo

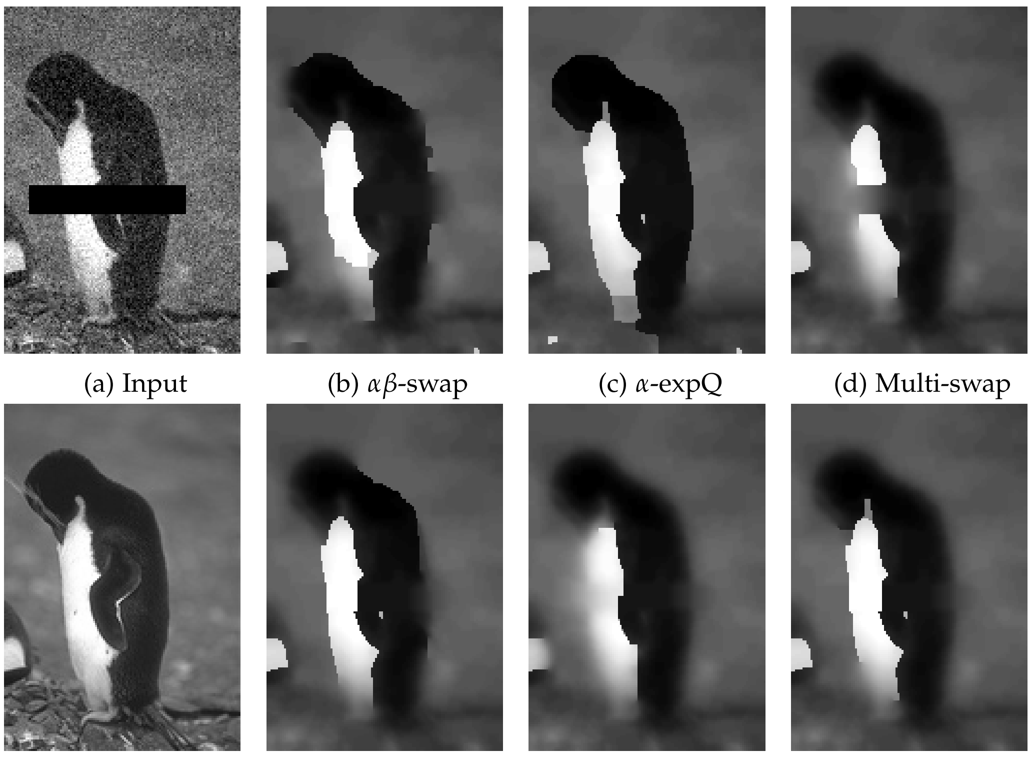


Figure 4.6: *Inpainted images for Penguin (see Section 4.6.2). Note that multi-label swap was not able to completely inpaint the missing pixels. IRGC+expansion produced smoother results than QPBOP-based α-expansion (see the bottom of the penguin) while preserving the finer details compared to TRWS (see the neck of the penguin). See Table 4.6 for the energy values.*

correspondence and the Penguin image for inpainting. We chose the truncated linear, the truncated quadratic and the Cauchy function for θ and three different κ values for each prior. The results are summarized in Tables 4.9 and 4.10.

In summary, the behavior observed in the previous sections holds: Our IRGC+expansion algorithm is able to find the lowest energy, or virtually the same energy as the best-performing method, irrespective of the employed prior and of the value of κ. A more detailed analysis shows that, when the pairwise potential is the truncated linear or the truncated quadratic, both IRGC and multi-label swap are susceptible to be trapped in local minima if the truncating value κ is small. For large values of κ, both the algorithms perform better and find minimum energies closer to the best-performing methods. In addition, for the Cauchy pairwise potential, IRGC behaves very similarly to IRGC+expansion for all values of κ.

Problem			Memory [MB]		Time [s]	
Name	ℓ	κ	BK	MEMF	BK	MEMF
Penguin	128	10	4471	**332**	**224**	2566
House	64	15	8877	**498**	**106**	409
Penguin	256	20	*17143	**663**	-	**17748**
House	256	60	*137248	**1986**	-	**19681**

Table 4.7: *Memory consumption and running time comparison of IRGC+expansion with either the BK method or our MEMF algorithm as subroutine (see Section 4.6.2.1). Here, the regularizer is the truncated quadratic function with truncation value κ^2. A "*" indicates a memory estimate. Compared to the BK method, MEMF is only 4 – 11 times slower but requires 13 – 18 times less memory, which makes it applicable to much larger MRFs.*

Problem	TRWS	IRGC	IRGC+exp.
Teddy	0.3040%	1.6289%	**0.2102%**
Map	**0.0511%**	0.7387%	0.1728%
Sawtooth	0.6452%	0.9621%	**0.2616%**
Venus	0.9096%	0.3498%	**0.2625%**
Cones	0.1551%	**0.0065%**	0.0074%
Tsukuba	**0.0910%**	0.4678%	0.1679%
Tsu. cor. Gaussian	0.3926%	12.1226%	**0.2736%**
House	0.0154%	**0.0058%**	**0.0058%**
Penguin	1.5556%	11.7218%	**0.8259%**
Average	0.4577%	3.1116%	**0.2431%**

Table 4.8: *Quality of the minimum energies according to Eq.* (4.28) *(see Section 4.6.3). IRGC+expansion clearly yields better quality energies than TRWS on average.*

4.7 Discussion

We have introduced an Iteratively Reweighted Graph-Cut algorithm that can minimize multi-label MRF energies with arbitrary data terms and non-convex priors. We have shown that, while the basic algorithm sometimes gets trapped in local minima, our hybrid version consistently outperforms (or performs virtually as well as) state-of-the-art MRF energy minimization techniques. We, therefore, believe our algorithm constitutes the first move-making algorithm to effectively tackle MRFs with robust non-convex priors.

Note that our algorithm has the flexibility in subroutine algorithm choices. For instance, for the hybrid version, fusion moves [Lempitsky et al., 2010] or any other move-making algorithm can be used instead of α-expansion. This could potentially improve the solution quality of the hybrid version of our algorithm. On the other hand, instead of the Ishikawa algorithm, the surrogate energy can be optimized approximately using the convex expansion technique of [Carr and Hartley, 2009] (see Section 2.3.3.2 for an overview). This would improve the running time and mem-

	Algorithm	$\kappa = 2$		$\kappa = 3$		$\kappa = 6$	
		E	T[s]	E	T[s]	E	T[s]
Tr. linear	$\alpha\beta$-swap	404,073	5.7	465,658	7.6	550,770	7.2
	α-expansion	403,886	4.4	465,123	3.6	548,783	3.8
	Multi-label swap	572,519	10.1	581,673	16.5	550,084	38.0
	TRWS	**402,593**	23.7	**464,184**	23.1	548,764	21.7
	IRGC	432,727	11.1	492,037	10.7	552,532	12.1
	IRGC+expansion	403,997	6.6	465,105	6.9	**548,669**	10.8
Tr. quadratic	$\alpha\beta$-swap	537,628	7.7	684,198	11.6	709,384	15.9
	α-expansionQ	524,563	9.4	640,186	9.7	701,684	15.9
	Multi-label swap	657,389	39.7	724,209	59.9	619,579	134.5
	TRWS	**519,565**	23.1	612,864	21.3	620,231	19.0
	IRGC	609,013	58.7	619,363	29.4	**619,176**	30.4
	IRGC+expansion	521,812	20.0	**609,513**	22.9	**619,176**	21.9
Cauchy	$\alpha\beta$-swap	409,072	14.1	541,955	16.2	612,809	18.5
	α-expansionQ	403,293	14.5	444,085	20.8	489,803	48.0
	TRWS	**395,786**	20.9	418,871	19.4	430,886	18.4
	IRGC	397,276	29.7	**418,629**	26.0	**430,597**	26.8
	IRGC+expansion	396,090	23.0	**418,629**	23.1	**430,597**	20.2

Table 4.9: *Comparison of the minimum energies (E) and execution times (T) for Tsukuba with different values of κ (see Section 4.6.4). IRGC+expansion found the lowest energy or virtually the same energy as TRWS. For the truncated linear prior, IRGC+expansion was 2 – 5 times faster than TRWS. For the truncated quadratic and Cauchy priors, IRGC outperformed other graph-cut-based algorithms in all most all the cases.*

	Algorithm	$\kappa = 4$		$\kappa = 10$		$\kappa = 15$	
		E	T[s]	E	T[s]	E	T[s]
Tr. linear	$\alpha\beta$-swap	1,822,240	13.4	2,544,290	16.9	2,766,680	16.1
	α-expansion	1,798,950	8.2	2,544,210	6.3	2,765,380	6.3
	Multi-label swap	2,110,930	12.9	2,587,730	36.1	2,782,300	76.6
	TRWS	**1,797,470**	151.1	**2,537,430**	151.2	**2,761,240**	150.6
	IRGC	1,977,010	44.5	2,574,190	46.5	2,786,550	55.2
	IRGC+expansion	1,803,150	43.7	2,542,980	33.6	2,764,810	28.1
Tr. quadratic	$\alpha\beta$-swap	2,929,130	27.2	4,562,820	40.1	5,336,120	76.3
	α-expansionQ	2,889,260	18.4	4,486,430	40.6	5,411,470	73.8
	Multi-label swap	3,184,930	70.2	4,520,570	446.1	5,511,780	1336.5
	TRWS	**2,847,580**	156.4	4,269,540	155.8	4,878,610	156.0
	IRGC	3,145,380	512.3	4,696,940	1660.0	5,508,910	282.1
	IRGC+expansion	2,861,400	177.0	**4,238,860**	353.3	**4,868,400**	325.9
Cauchy	$\alpha\beta$-swap	3,237,880	53.7	4,319,210	103.0	4,644,580	104.6
	α-expansionQ	3,212,320	56.3	4,248,640	109.1	4,632,250	140.0
	TRWS	**3,148,650**	150.0	4,086,660	150.3	4,364,480	150.1
	IRGC	3,152,870	673.0	**4,084,930**	1248.0	**4,364,470**	1313.9
	IRGC+expansion	3,151,270	232.7	4,087,540	296.3	4,367,270	248.3

Table 4.10: *Comparison of the minimum energies (E) and execution times (T) for Penguin with different values of κ (see Section 4.6.4). IRGC+expansion found the lowest energy or virtually the same energy as TRWS. For the truncated linear prior, IRGC+expansion was 4 – 5 times faster than TRWS. For the Cauchy prior, both IRGC and IRGC+expansion outperformed other graph-cut-based algorithms but IRGC+expansion was faster to converge.*

ory usage while compromising on the solution quality. Such an algorithm would be useful in resource-scarce environments, such as embedded systems or mobile devices. Furthermore, one may replace the Ishikawa algorithm with the primal-dual approach of [Pock et al., 2008; Mollenhoff et al., 2016]. This would further extend the applicability of IRGC to continuous label spaces and enables the use of Graphics Processing Unit (GPU) to parallelize the algorithm.

Finally, as discussed in the chapter, IRGC really is a special case of an iteratively reweighted approach to MRFs and even continuous energy minimization. Therefore, we believe, our approach can be extended to other types of MRF problems and we intend to study useful extensions in the future.

In the next chapter, we address fully connected CRFs and present an efficient linear programming relaxation-based algorithm. This algorithm is shown to be the first efficient and effective minimization of the dense CRF energy function.

Chapter 5

Efficient Linear Programming for Dense CRFs

So far, we have discussed an optimal max-flow algorithm and an approximate move-making algorithm for sparsely connected MRFs. In this chapter, we present an efficient LP relaxation-based algorithm for fully connected CRFs[1]. Specifically, we present a block-coordinate descent algorithm to minimize the LP relaxation of a dense CRF with Gaussian pairwise potentials. We show that each iteration of our algorithm is *linear* in the number of pixels and labels. To this end, we also discuss a modification to the permutohedral lattice based filtering method [Adams et al., 2010], which enables us to perform approximate Gaussian filtering with ordering constraints in linear time. This chapter is based on our work [Ajanthan et al., 2017a] and the extended version is available in [Ajanthan et al., 2017b]. *The work presented in this chapter was conducted under the supervision of Prof. Philip Torr and As.Prof Pawan Kumar, during my visit at the Torr Vision Group at the University of Oxford, from 4th July 2016 to 4th December 2016.*

5.1 Introduction

In the past few years, the dense Conditional Random Field (CRF) with Gaussian pairwise potentials has become popular for multi-class image-based semantic segmentation. At the origin of this popularity lies the use of an efficient filtering method [Adams et al., 2010], which was shown to lead to a linear time mean-field inference strategy [Krähenbühl Philipp, 2011] (see Section 2.3.5 for a brief review on mean-field). Recently, this filtering method was exploited to minimize the dense CRF energy using other, typically more effective, continuous relaxation methods [Desmaison et al., 2016a]. Among the relaxations considered in [Desmaison et al., 2016a], the Linear Programming (LP) relaxation provides strong theoretical guarantees on the quality of the solution [Kleinberg and Tardos, 2002; Kumar et al., 2009].

Note that the LP relaxation-based algorithms discussed in Section 2.3.4 exploit the sparsity of the CRF via the tree decomposition technique. In the fully connected

[1]As mentioned in Section 2.1.1, from the optimization perspective, both MRF and CRF models are identical.

case, they would yield *quadratic* complexity in the number of pixels per iteration. Clearly, this would lead to prohibitively large running time. In [Desmaison et al., 2016a], the LP was minimized via projected subgradient descent. While relying on the filtering method, computing the subgradient was shown to be *linearithmic* in the number of pixels, but not *linear*. Moreover, even with the use of a line search strategy, the algorithm required a large number of iterations to converge, making it inefficient.

We introduce an iterative LP minimization algorithm for a dense CRF with Gaussian pairwise potentials which has *linear* time complexity per iteration. To this end, instead of relying on a standard subgradient technique, we propose to make use of the proximal method [Parikh and Boyd, 2014]. The resulting proximal problem has a smooth dual, which can be efficiently optimized using block-coordinate descent. We show that each block of variables can be optimized efficiently. Specifically, for one block, the problem decomposes into significantly smaller subproblems, each of which is defined over a single pixel. For the other block, the problem can be optimized via the Frank-Wolfe algorithm [Frank and Wolfe, 1956; Lacoste-Julien et al., 2012] (often referred to as *conditional gradient descent*). We show that the conditional gradient required by this algorithm can be computed efficiently. In particular, we modify the filtering method of [Adams et al., 2010] such that the conditional gradient[2] can be computed in a time *linear* in the number of pixels and labels. Besides this linear complexity, our approach has two additional benefits. First, it can be initialized with the solution of a faster, less accurate algorithm, such as mean-field [Krähenbühl Philipp, 2011] or the Difference of Convex (DC) relaxation of [Desmaison et al., 2016a], thus speeding up convergence. Second, the optimal step size of our iterative procedure can be obtained analytically, thus preventing the need to rely on an expensive line search procedure.

We demonstrate the effectiveness of our algorithm on the MSRC and Pascal VOC 2010 [Everingham et al., 2010] segmentation datasets. The experiments evidence that our algorithm is significantly faster than the state-of-the-art LP minimization technique of [Desmaison et al., 2016a]. Furthermore, it yields assignments whose energies are much lower than those obtained by other competing methods [Desmaison et al., 2016a; Krähenbühl Philipp, 2011]. Altogether, our framework constitutes the first efficient and effective minimization algorithm for dense CRFs with Gaussian pairwise potentials. Our code is available at https://github.com/oval-group/DenseCRF.

5.2 Preliminaries

Let us first provide some background on the dense CRF model and its LP relaxation. Then, we briefly review two convex optimization techniques which would be useful to explain our algorithm.

[2]Due to the use of the filtering method, only an approximate conditional gradient can be computed. However, this approximate conditional gradient is empirically shown to be collinear to the exact conditional gradient for a wide range of values [Desmaison et al., 2016b].

5.2.1 Dense CRF Energy Function

Let us recall the energy function associated with a pairwise MRF (2.4),

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

where $x_i \in \mathcal{L}$ with $|\mathcal{L}| = \ell$. Here, $\mathcal{V} = \{1, \ldots, n\}$ is the set of vertices and $\mathcal{E}$ is the set of undirected edges in the underlying MRF graph. For a dense CRF, the set $\mathcal{E}$ encodes the fully connected graph, *i.e.*,

$$\mathcal{E} = \{(i,j) \mid i, j \in \mathcal{V}, i \neq j\} \,. \tag{5.2}$$

By substituting for $\mathcal{V}$ and $\mathcal{E}$, the energy associated with a pairwise dense CRF can be written as

$$E(\mathbf{x}) = \sum_{i=1}^{n} \theta_i(x_i) + \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \theta_{ij}(x_i, x_j) \,, \tag{5.3}$$

where θ_i and θ_{ij} denote the *unary potentials* and *pairwise potentials*, respectively. The unary potentials define the data cost and the pairwise potentials the smoothness cost. Furthermore, we denote the vector of unary potentials as $\boldsymbol{\theta}_u = \{\theta_{i:\lambda} \mid i \in \mathcal{V}, \lambda \in \mathcal{L}\}$.

5.2.2 Gaussian Pairwise Potentials

Similarly to [Desmaison et al., 2016a; Krähenbühl Philipp, 2011], we consider Gaussian pairwise potentials, which have the following form:

$$\theta_{ij}(x_i, x_j) = \theta(x_i, x_j) \sum_{c} w^{(c)} \, k\left(\mathbf{f}_i^{(c)}, \mathbf{f}_j^{(c)}\right) \,, \tag{5.4}$$
$$k(\mathbf{f}_i, \mathbf{f}_j) = \exp\left(\frac{-\|\mathbf{f}_i - \mathbf{f}_j\|^2}{2}\right) \,.$$

Here, $\theta(x_i, x_j)$ is referred to as the *label compatibility* function and the mixture of Gaussian kernels[3] as the *pixel compatibility* function. The non-negative weights $w^{(c)}$ define the mixture coefficients, and $\mathbf{f}_i^{(c)} \in \mathbb{R}^{\mathrm{d}^{(c)}}$ encodes features associated to the random variable X_i, where $d^{(c)}$ is the feature dimension. For semantic segmentation, each pixel in an image corresponds to a random variable. In practice, as in [Desmaison et al., 2016a; Krähenbühl Philipp, 2011], we then use the position and RGB values of a pixel as features, and assume the label compatibility function to be the *Potts model*, *i.e.*,

$$\theta(x_i, x_j) = \mathbb{1}[x_i \neq x_j] = \begin{cases} 1 & \text{if } x_i \neq x_j \\ 0 & \text{if } x_i = x_j \,. \end{cases} \tag{5.5}$$

[3]Note that, in this definition, we assume that the features $\mathbf{f}_i$ and $\mathbf{f}_j$ have already been normalized by the filter standard deviation σ. Hence σ is not explicitly written.

These potentials have proven useful to obtain fine grained labellings in segmentation tasks [Krähenbühl Philipp, 2011].

5.2.3 Integer Programming Formulation

In the multi-label graph representation (see Eq. (2.27)), a labelling is represented by defining indicator variables $x_{i:\lambda} \in \{0,1\}$, where $x_{i:\lambda} = 1$ if and only if $x_i = \lambda$. Using this notation, the energy minimization problem can be written as the following Integer Program (IP):

$$\begin{aligned} \min_{\mathbf{x}} \quad E(\mathbf{x}) &= \sum_i \sum_\lambda \theta_{i:\lambda}\, x_{i:\lambda} + \sum_{i,j\neq i} \sum_{\lambda,\mu} \theta_{ij:\lambda\mu}\, x_{i:\lambda}\, x_{j:\mu}\ , \\ \text{s.t.} \quad \sum_\lambda x_{i:\lambda} &= 1 \qquad \forall\, i \in \mathcal{V}\ , \\ x_{i:\lambda} &\in \{0,1\} \quad \forall\, i \in \mathcal{V}, \quad \forall\, \lambda \in \mathcal{L}\ . \end{aligned} \tag{5.6}$$

Here, we use the shorthand $\theta_{i:\lambda} = \theta_i(\lambda)$ and $\theta_{ij:\lambda\mu} = \theta_{ij}(\lambda,\mu)$. The first set of constraints ensure that each random variable is assigned exactly one label. Note that the value of the objective function is equal to the energy of the labelling encoded by $\mathbf{x}$.

5.2.4 Linear Programming Relaxation

By relaxing the binary constraints of the indicator variables in (5.6) and using the fact that the label compatibility function is the Potts model, the linear programming relaxation [Kleinberg and Tardos, 2002] of (5.6) is defined as

$$\begin{aligned} \min_{\mathbf{y}} \quad \tilde{E}(\mathbf{y}) &= \sum_i \sum_\lambda \theta_{i:\lambda}\, y_{i:\lambda} + \sum_{i,j\neq i} \sum_\lambda K_{ij} \frac{|y_{i:\lambda} - y_{j:\mu}|}{2}\ , \\ \text{s.t.} \quad \mathbf{y} \in \mathcal{S} &= \left\{ \mathbf{y} \;\middle|\; \begin{array}{l} \sum_\lambda y_{i:\lambda} = 1,\, i \in \mathcal{V} \\ y_{i:\lambda} \geq 0,\, i \in \mathcal{V},\, \lambda \in \mathcal{L} \end{array} \right\}, \end{aligned} \tag{5.7}$$

where $K_{ij} = \sum_c w^{(c)}\, k\left(\mathbf{f}_i^{(c)}, \mathbf{f}_j^{(c)}\right)$. Here $\mathcal{S}$ denotes the set of *real labellings* (see Definition 2.3.5). For integer labellings, the LP objective $\tilde{E}(\cdot)$ has the same value as the IP objective $E(\cdot)$.

The above relaxation is the same as the standard LP relaxation[4] [Chekuri et al., 2004; Werner, 2007] for the Potts model and it provides an integrality gap of 2. The result in [Manokaran et al., 2008] means that it is unlikely (unless the Unique Games Conjecture is false) that a better relaxation can be designed for this problem. Using standard solvers to minimize this LP would require the introduction of $\mathcal{O}(n^2)$ variables (see Eq. (5.20)), making it intractable. Therefore the non-smooth objective of Eq. (5.7) has to be optimized directly. This was handled using projected subgradient

[4]This is exactly the same as the linear programming over the local polytope discussed in Section 2.3.4.2 for the case where the label compatibility is the Potts model.

descent in [Desmaison et al., 2016a], which also turns out to be inefficient in practice. In this chapter, we introduce an efficient algorithm to tackle this problem while maintaining *linear* scaling in both space and time complexity.

5.2.5 Proximal Minimization Algorithm

Proximal algorithms are well known in the convex optimization literature and many popular algorithms, such as Alternating Direction Method of Multipliers (ADMM) [Boyd et al., 2011] are special instances of them. Here, we give a brief overview, and we refer the interested reader to [Parikh and Boyd, 2014].

Proximal algorithms rely on the use of the *proximal operator*, which is defined below.

Definition 5.2.1. Let $f : \mathbb{R}^N \to \mathbb{R} \cup \{\infty\}$ be a *closed proper convex function*[5]. Then, the *proximal operator* $\text{prox}_{\eta f} : \mathbb{R}^N \to \mathbb{R}^N$ of f with parameter $\eta > 0$ is defined as

$$\text{prox}_{\eta f}(\mathbf{v}) = \underset{\mathbf{u}}{\text{argmin}}\, f(\mathbf{u}) + \frac{1}{2\eta}\|\mathbf{u} - \mathbf{v}\|^2 , \tag{5.8}$$

where $\|\cdot\|$ is the standard Euclidean norm.

In a proximal minimization algorithm, at each iteration r, the proximal operator is applied, *i.e.*,

$$\mathbf{u}^{r+1} = \text{prox}_{\eta f}(\mathbf{u}^r) , \tag{5.9}$$

where $\mathbf{u}^r$ denotes the estimate of $\mathbf{u}$ at iteration r. From this update equation, it is evident that the proximal algorithm guarantees monotonic decrease in the function value. In addition to that, it converges to the minimum of the function f.

Theorem 5.2.1. *The point $\mathbf{u}^*$ minimizes f if and only if*

$$\mathbf{u}^* = \text{prox}_{\eta f}(\mathbf{u}^*) . \tag{5.10}$$

Proof. This can be proven by applying the first order convexity condition of f. See [Parikh and Boyd, 2014]. □

Accelerated Version. Note that the basic proximal algorithm can be accelerated by including an extrapolation step in the algorithm. For a simple version, the update equation takes the following form,

$$\begin{aligned}\tilde{\mathbf{u}}^r &= \mathbf{u}^r + \omega^r(\mathbf{u}^r - \mathbf{u}^{r-1}) , \\ \mathbf{u}^{r+1} &= \text{prox}_{\eta f}(\tilde{\mathbf{u}}^r) ,\end{aligned} \tag{5.11}$$

where one simple choice of ω^r to guarantee convergence is $\omega^r = r/(r+3)$ [Parikh and Boyd, 2014].

[5]A function is closed, proper and convex, if its *epigraph* $\text{epi}(f) = \{(\mathbf{u}, t) \in \mathbb{R}^N \times \mathbb{R} \mid f(\mathbf{u}) \leq t\}$ is a nonempty closed convex set [Parikh and Boyd, 2014].

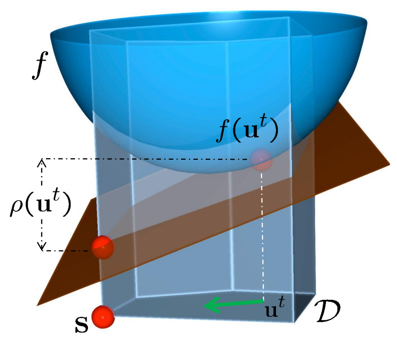


Figure 5.1: *An iteration of the Frank-Wolfe algorithm illustrated using a 3D example. Image from [Lacoste-Julien et al., 2012]. Here,* $\mathbf{u}^t$ *is the estimate of* $\mathbf{u}$ *at iteration* t*,* $\mathbf{s}$ *is the conditional gradient and* $\rho(\mathbf{u}^t)$ *denotes the duality gap. Furthermore, the green arrow illustrates the Frank-Wolfe update* (5.15). *(Best viewed in color)*

In this chapter, we discuss how the proximal algorithm can be utilized to design an efficient block-coordinate algorithm for the LP relaxation of a dense CRF energy function.

5.2.6 The Frank-Wolfe Algorithm

The Frank-Wolfe algorithm [Frank and Wolfe, 1956] is an iterative algorithm to optimize a constrained convex optimization problem of the form,

$$\min_{\mathbf{u}\in\mathcal{D}} f(\mathbf{u}) \ , \tag{5.12}$$

where $f : \mathbb{R}^N \to \mathbb{R}$ is a convex and continuously differentiable function and the domain $\mathcal{D} \subset \mathbb{R}^N$ is convex and *compact*[6]. This algorithm is also known as the *conditional gradient method*. At each iteration, the algorithm computes the conditional gradient, by minimizing the first order Taylor approximation of the function f around $\mathbf{u}^t$, where $\mathbf{u}^t$ is the estimate at iteration t. Hence, the conditional gradient can be written as

$$\begin{aligned} \mathbf{s} &= \operatorname*{argmin}_{\hat{\mathbf{s}}\in\mathcal{D}} \tilde{f}(\hat{\mathbf{s}}) \ , \qquad (5.13)\\ &= \operatorname*{argmin}_{\hat{\mathbf{s}}\in\mathcal{D}} f(\mathbf{u}^t) + \left\langle \hat{\mathbf{s}} - \mathbf{u}^t, \nabla f(\mathbf{u}^t)\right\rangle \ , \\ &= \operatorname*{argmin}_{\hat{\mathbf{s}}\in\mathcal{D}} \left\langle \hat{\mathbf{s}}, \nabla f(\mathbf{u}^t)\right\rangle \ . \end{aligned}$$

[6]A topological space $\mathcal{X}$ is called *compact* if every open cover has a finite subcover. Furthermore, a set $\mathcal{D} \subset \mathbb{R}^N$ is compact if and only if, it is closed and bounded [Kelley, 1975].

Once the conditional gradient is computed, the step size δ can be obtained using line search,

$$\delta = \operatorname*{argmin}_{\hat{\delta} \in [0,1]} f(\hat{\delta}\,\mathbf{s} + (1-\hat{\delta})\mathbf{u}^t)\ . \tag{5.14}$$

In fact, in cases where the optimal step size is difficult to compute, one can use the step size $\delta = 2/(t+2)$, which guarantees convergence [Frank and Wolfe, 1956; Lacoste-Julien et al., 2012]. Now, the update equation takes the following form,

$$\mathbf{u}^{t+1} = \delta\,\mathbf{s} + (1-\delta)\mathbf{u}^t\ . \tag{5.15}$$

From this, it is clear that the algorithm guarantees monotonic decrease of the function value. Furthermore, due to the convexity of f, at each iteration of the Frank-Wolfe algorithm a linearization duality gap[7] is available. In particular, the duality gap is defined as

$$\begin{aligned} \rho(\mathbf{u}^t) &= f(\mathbf{u}^t) - \tilde{f}(\mathbf{s})\ , \\ &= \max_{\hat{\mathbf{s}} \in \mathcal{D}} \left\langle \mathbf{u}^t - \hat{\mathbf{s}}, \nabla f(\mathbf{u}^t) \right\rangle\ . \end{aligned} \tag{5.16}$$

The property of the duality gap is characterized by the following Lemma.

Lemma 5.2.1. *Let $\rho(\mathbf{u}^t)$ be the duality gap defined in Eq.* (5.16) *and $\mathbf{u}^*$ be the point where the minimum of f is attained. Then,*

$$\rho(\mathbf{u}^t) \geq f(\mathbf{u}^t) - f(\mathbf{u}^*)\ . \tag{5.17}$$

Proof. Due to the convexity of f,

$$\tilde{f}(\hat{\mathbf{s}}) = f(\mathbf{u}^t) + \left\langle \hat{\mathbf{s}} - \mathbf{u}^t, \nabla f(\mathbf{u}^t) \right\rangle \leq f(\hat{\mathbf{s}})\ , \tag{5.18}$$

for all $\hat{\mathbf{s}} \in \mathcal{D}$. Now, by minimizing over $\hat{\mathbf{s}} \in \mathcal{D}$ in both sides and rearranging the terms, the desired result is obtained. □

See Figure 5.1 for an illustration of this lemma. Hence, the duality gap $\rho(\mathbf{u}^t)$ provides a stopping criterion, *i.e.*, for a chosen tolerance $\epsilon > 0$, if $\rho(\mathbf{u}^t) \leq \epsilon$, the algorithm can be terminated. The complete algorithm is summarized in Algorithm 5.1.

The Frank-Wolfe algorithm is a crucial component of our LP minimization algorithm, in the sense that one block of variables are optimized by it. The use of Frank-Wolfe ensures linear space complexity of our algorithm. In addition, by modifying the efficient filtering method [Adams et al., 2010], we show that the conditional gradient can be computed in a time linear in the number of pixels and labels. Hence, overall, our algorithm maintains linear scaling in both space and time complexity.

[7] In fact the gap defined in Eq. (5.16) is a special case of Fenchel duality gap [Lacoste-Julien et al., 2012] and hence the name "duality gap".

Algorithm 5.1 The Frank-Wolfe Algorithm [Frank and Wolfe, 1956]

```
Require: $\mathbf{u}^0 \in \mathcal{D}$ and tolerance $\epsilon$
  for $t \leftarrow 0 \ldots T$ do
    $\mathbf{s} \leftarrow \operatorname{argmin}_{\hat{\mathbf{s}} \in \mathcal{D}} \langle \hat{\mathbf{s}}, \nabla f(\mathbf{u}^t) \rangle$        ▷ Conditional gradient
    $\delta = \frac{2}{t+2}$ or optimize $\delta$ by line search        ▷ Step size
    $\mathbf{u}^{t+1} \leftarrow (1-\delta)\mathbf{u}^t + \delta\,\mathbf{s}$        ▷ Update
    if $\rho(\mathbf{u}^t) \leq \epsilon$ then        ▷ Stopping condition
      break
    end if
  end for
```

5.3 Proximal Minimization for LP Relaxation

Our goal is to design an efficient minimization strategy for the LP relaxation in (5.7). To this end, we propose to use the proximal minimization algorithm (see Section 5.2.5). This guarantees monotonic decrease in the objective value, enabling us to leverage faster, less accurate methods for initialization. Furthermore, the additional quadratic regularization term makes the dual problem smooth, enabling the use of more sophisticated optimization methods. In the remainder of this chapter, we detail this approach and show that each iteration has linear time complexity. In practice, our algorithm converges in a small number of iterations, thereby making the overall approach computationally efficient.

Note that the LP objective function $\tilde{E}(\mathbf{y})$ can be shown to be a closed proper convex function. Now, given the current estimate of the solution $\mathbf{y}^r \in \mathcal{S}$, the proximal update equation can be written as

$$\begin{aligned} \min_{\mathbf{y}} \quad & \tilde{E}(\mathbf{y}) + \frac{1}{2\eta}\|\mathbf{y} - \mathbf{y}^r\|^2 \;, \\ \text{s.t.} \quad & \mathbf{y} \in \mathcal{S} \;, \end{aligned} \tag{5.19}$$

where η sets the strength of the proximal term.

Note that (5.19) consists of piecewise linear terms and a quadratic regularization term. Specifically, the piecewise linear term comes from the pairwise term $|y_{i:\lambda} - y_{j:\lambda}|$ in (5.7) that can be reformulated as $\max\{y_{i:\lambda} - y_{j:\lambda}, y_{j:\lambda} - y_{i:\lambda}\}$. The proximal term $\|\mathbf{y} - \mathbf{y}^r\|^2$ provides the quadratic regularization. In this section, we introduce a new algorithm that is tailored to this problem. In particular, we optimally solve the Lagrange dual of (5.19) in a block-wise fashion.

5.3.1 Dual Formulation

Let us first write the proximal problem (5.19) in the standard form by introducing auxiliary variables $z_{ij:\lambda}$.

$$
\begin{aligned}
\min_{\mathbf{y},\mathbf{z}} \quad & \sum_i \sum_\lambda \theta_{i:\lambda}\, y_{i:\lambda} + \sum_{i,j \neq i} \sum_\lambda \frac{K_{ij}}{2} z_{ij:\lambda} + \frac{1}{2\eta} \|\mathbf{y} - \mathbf{y}^r\|^2 \;, && \text{(5.20a)}\\
\text{s.t.} \quad & z_{ij:\lambda} \geq y_{i:\lambda} - y_{j:\lambda} \quad \forall\, i, j \neq i \quad \forall\, \lambda \in \mathcal{L}\;, && \text{(5.20b)}\\
& z_{ij:\lambda} \geq y_{j:\lambda} - y_{i:\lambda} \quad \forall\, i, j \neq i \quad \forall\, \lambda \in \mathcal{L}\;, && \text{(5.20c)}\\
& \sum_\lambda y_{i:\lambda} = 1 \quad \forall\, i \in \mathcal{V}\;, && \text{(5.20d)}\\
& y_{i:\lambda} \geq 0 \quad \forall\, i \in \mathcal{V} \quad \forall\, \lambda \in \mathcal{L}\;. && \text{(5.20e)}
\end{aligned}
$$

We introduce three blocks of dual variables. Namely, $\boldsymbol{\alpha} = \{\alpha^1_{ij:\lambda}, \alpha^2_{ij:\lambda} \mid i, j \neq i, \lambda \in \mathcal{L}\}$ for the constraints in Eqs. (5.20b) and (5.20c), $\boldsymbol{\beta} = \{\beta_i \mid i \in \mathcal{V}\}$ for Eq. (5.20d) and $\boldsymbol{\gamma} = \{\gamma_{i:\lambda} \mid i \in \mathcal{V}, \lambda \in \mathcal{L}\}$ for Eq. (5.20e), respectively. The vector $\boldsymbol{\alpha}$ has $p = 2n(n-1)\ell$ elements. Here, we introduce two matrices that will be useful to write the dual problem compactly.

Definition 5.3.1. Let $A \in \mathbb{R}^{\mathrm{n}\ell \times \mathrm{p}}$ and $B \in \mathbb{R}^{\mathrm{n}\ell \times \mathrm{n}}$ be two matrices such that

$$
\begin{aligned}
(A\boldsymbol{\alpha})_{i:\lambda} &= -\sum_{j \neq i} \left(\alpha^1_{ij:\lambda} - \alpha^2_{ij:\lambda} + \alpha^2_{ji:\lambda} - \alpha^1_{ji:\lambda} \right) \;, && \text{(5.21)}\\
(B\boldsymbol{\beta})_{i:\lambda} &= \beta_i \;.
\end{aligned}
$$

We can now state our first proposition.

Proposition 5.3.1. Given matrices $A \in \mathbb{R}^{\mathrm{n}\ell \times \mathrm{p}}$ and $B \in \mathbb{R}^{\mathrm{n}\ell \times \mathrm{n}}$ and dual variables $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma})$.

1. The Lagrange dual of (5.20) takes the following form:

$$
\min_{\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma}} \quad g(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}) = \frac{\eta}{2} \|A\boldsymbol{\alpha} + B\boldsymbol{\beta} + \boldsymbol{\gamma} - \boldsymbol{\theta}_u\|^2 + \langle A\boldsymbol{\alpha} + B\boldsymbol{\beta} + \boldsymbol{\gamma} - \boldsymbol{\theta}_u, \mathbf{y}^r \rangle - \langle \mathbf{1}, \boldsymbol{\beta} \rangle \;, \tag{5.22}
$$

$$
\begin{aligned}
\text{s.t.} \qquad & \gamma_{i:\lambda} \geq 0 \quad \forall\, i \in \mathcal{V} \quad \forall\, \lambda \in \mathcal{L}\;,\\
& \boldsymbol{\alpha} \in \mathcal{C} = \left\{ \boldsymbol{\alpha} \;\middle|\; \begin{array}{l} \alpha^1_{ij:\lambda} + \alpha^2_{ij:\lambda} = \frac{K_{ij}}{2}, \forall\, i, j \neq i, \forall\, \lambda \in \mathcal{L} \\ \alpha^1_{ij:\lambda}, \alpha^2_{ij:\lambda} \geq 0, \forall\, i, j \neq i, \forall\, \lambda \in \mathcal{L} \end{array} \right\} \;.
\end{aligned}
$$

2. The primal variables $\mathbf{y}$ satisfy

$$
\mathbf{y} = \eta \left(A\boldsymbol{\alpha} + B\boldsymbol{\beta} + \boldsymbol{\gamma} - \boldsymbol{\theta}_u \right) + \mathbf{y}^r \;. \tag{5.23}
$$

Let us first analyze the properties of matrices A and B and then turn to the derivation of the dual.

Proposition 5.3.2. Let $\boldsymbol{\omega} \in \mathrm{I\!R}^{n\ell}$. Then, for all $i \neq j$ and $\lambda \in \mathcal{L}$,

$$\begin{aligned}\left(A^T\boldsymbol{\omega}\right)_{ij:\lambda^1} &= \omega_{j:\lambda} - \omega_{i:\lambda}\ ,\\ \left(A^T\boldsymbol{\omega}\right)_{ij:\lambda^2} &= \omega_{i:\lambda} - \omega_{j:\lambda}\ .\end{aligned} \tag{5.24}$$

Here, the index $ij : \lambda^1$ denotes the element corresponding to $\alpha^1_{ij:\lambda}$.

Proof. This can be easily proven by inspecting the matrix A. □

Proposition 5.3.3. The matrix $B \in \mathrm{I\!R}^{n\ell\times n}$ defined in Eq. (5.21) satisfies the following properties:

1. Let $\boldsymbol{\omega} \in \mathrm{I\!R}^{n\ell}$. Then, $\left(B^T\boldsymbol{\omega}\right)_i = \sum_{\lambda\in\mathcal{L}} \omega_{i:\lambda}$ for all $i \in \mathcal{V}$.

2. $B^TB = \ell I$, where $I \in \mathrm{I\!R}^{n\times n}$ is the identity matrix.

3. BB^T is a block diagonal matrix, with each block $\left(BB^T\right)_i = \mathbf{1}$ for all $i \in \mathcal{V}$, where $\mathbf{1} \in \mathrm{I\!R}^{\ell\times\ell}$ is the matrix of all ones.

Proof. Note that, from Eq. (5.21), the matrix B simply repeats the elements β_i for ℓ times. In particular, for $\ell = 3$, the matrix B has the following form:

$$B = \begin{bmatrix} 1 & 0 & \cdots & \cdots & \cdots & 0 \\ 1 & \vdots & & & & \vdots \\ 1 & 0 & \cdots & \cdots & \cdots & \vdots \\ 0 & 1 & & & & \vdots \\ \vdots & 1 & & & & \vdots \\ \vdots & 1 & & & & \vdots \\ \vdots & 0 & & & & \vdots \\ \vdots & \vdots & & & & 0 \\ \vdots & \vdots & & & & 1 \\ \vdots & \vdots & & & & 1 \\ 0 & \cdots & \cdots & \cdots & 0 & 1 \end{bmatrix} . \tag{5.25}$$

Therefore, multiplication by B^T amounts to summing over the labels. From this, the other properties can be proven easily. □

Now we prove Proposition 5.3.1.

Proof. The Lagrangian associated with the primal problem (5.20) can be written as [Boyd and Vandenberghe, 2009]:

$$\max_{\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma}} \min_{\mathbf{y},\mathbf{z}} L(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma},\mathbf{y},\mathbf{z}) = \sum_i \sum_\lambda \theta_{i:\lambda}\, y_{i:\lambda} + \sum_{i,j\neq i} \sum_\lambda \frac{K_{ij}}{2} z_{ij:\lambda} + \frac{1}{2\eta} \sum_i \sum_\lambda \left(y_{i:\lambda} - y^r_{i:\lambda}\right)^2 \tag{5.26}$$

$$+ \sum_{i,j\neq i} \sum_\lambda \alpha^1_{ij:\lambda} \left(y_{i:\lambda} - y_{j:\lambda} - z_{ij:\lambda}\right) + \sum_{i,j\neq i} \sum_\lambda \alpha^2_{ij:\lambda} \left(y_{j:\lambda} - y_{i:\lambda} - z_{ij:\lambda}\right)$$

$$+ \sum_i \beta_i \left(1 - \sum_\lambda y_{i:\lambda}\right) - \sum_i \sum_\lambda \gamma_{i:\lambda}\, y_{i:\lambda}\ ,$$

$$\text{s.t.} \qquad \alpha^1_{ij:\lambda}, \alpha^2_{ij:\lambda} \geq 0 \quad \forall\, i, j \neq i \quad \forall\, \lambda \in \mathcal{L}\ ,$$

$$\gamma_{i:\lambda} \geq 0 \quad \forall\, i \in \mathcal{V} \quad \forall\, \lambda \in \mathcal{L}\ .$$

Note that the dual problem is obtained by minimizing the Lagrangian over the primal variables $(\mathbf{y}, \mathbf{z})$. With respect to $\mathbf{z}$, the Lagrangian is linear and when $\nabla_{\mathbf{z}} L(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma},\mathbf{y},\mathbf{z}) \neq 0$, the minimization in $\mathbf{z}$ yields $-\infty$. This situation is not useful as the dual function is unbounded. Therefore we restrict ourselves to the case where $\nabla_{\mathbf{z}} L(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma},\mathbf{y},\mathbf{z}) = 0$. By differentiating with respect to $\mathbf{z}$ and setting the derivatives to zero, we obtain

$$\alpha^1_{ij:\lambda} + \alpha^2_{ij:\lambda} = \frac{K_{ij}}{2} \quad \forall\, i, j \neq i \quad \forall\, \lambda \in \mathcal{L}\ . \tag{5.27}$$

The minimum of the Lagrangian with respect to $\mathbf{y}$ is attained when $\nabla_{\mathbf{y}} L(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma},\mathbf{y},\mathbf{z}) = 0$. Before differentiating with respect to $\mathbf{y}$, let us rewrite the Lagrangian using Eq. (5.27) and reorder the terms:

$$L(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma},\mathbf{y},\mathbf{z}) = \sum_i \sum_\lambda (\theta_{i:\lambda} - \beta_i - \gamma_{i:\lambda})\, y_{i:\lambda} + \frac{1}{2\eta} \sum_i \sum_\lambda \left(y_{i:\lambda} - y^r_{i:\lambda}\right)^2 \tag{5.28}$$

$$+ \sum_{i,j\neq i} \sum_\lambda \left(\alpha^1_{ij:\lambda} - \alpha^2_{ij:\lambda}\right) y_{i:\lambda} + \sum_{i,j\neq i} \sum_\lambda \left(\alpha^2_{ji:\lambda} - \alpha^1_{ji:\lambda}\right) y_{i:\lambda} + \sum_i \beta_i\ .$$

Now, by differentiating with respect to $\mathbf{y}$ and setting the derivatives to zero, we get

$$\frac{1}{\eta}\left(y_{i:\lambda} - y^r_{i:\lambda}\right) = -\sum_{j\neq i} \left(\alpha^1_{ij:\lambda} - \alpha^2_{ij:\lambda} + \alpha^2_{ji:\lambda} - \alpha^1_{ji:\lambda}\right) + \beta_i + \gamma_{i:\lambda} - \theta_{i:\lambda} \quad \forall\, i \in \mathcal{V} \quad \forall\, \lambda \in \mathcal{L}\ . \tag{5.29}$$

Using Eq. (5.21), the above equation can be written in vector form as

$$\frac{1}{\eta}\left(\mathbf{y} - \mathbf{y}^r\right) = A\boldsymbol{\alpha} + B\boldsymbol{\beta} + \boldsymbol{\gamma} - \boldsymbol{\theta}_u\ . \tag{5.30}$$

This proves Eq. (5.23). Now, using Eqs. (5.27) and (5.30), the dual problem can be

Algorithm 5.2 Proximal Minimization of LP (PROX-LP)

```
Require: Initial solution $\mathbf{y}^0 \in \mathcal{S}$ and the dual objective $g$
  for $r \leftarrow 0 \dots R$ do
    $A\boldsymbol{\alpha}^0 \leftarrow \mathbf{0}, \quad \boldsymbol{\beta}^0 \leftarrow \mathbf{0}, \quad \boldsymbol{\gamma}^0 \leftarrow \mathbf{0}$        ▷ Feasible initialization
    for $t \leftarrow 0 \dots T$ do
      $(\boldsymbol{\beta}^t, \boldsymbol{\gamma}^t) \leftarrow \operatorname{argmin}_{\boldsymbol{\beta},\boldsymbol{\gamma}} g(\boldsymbol{\alpha}^t, \boldsymbol{\beta}, \boldsymbol{\gamma})$        ▷ Section 5.3.2.1
      $\tilde{\mathbf{y}}^t \leftarrow \eta (A\boldsymbol{\alpha}^t + B\boldsymbol{\beta}^t + \boldsymbol{\gamma}^t - \boldsymbol{\theta}_u) + \mathbf{y}^r$        ▷ Current (infeasible) primal solution
      $A\mathbf{s}^t \leftarrow$ conditional gradient of $g$, computed using $\tilde{\mathbf{y}}^t$        ▷ Section 5.3.2.2
      $\delta \leftarrow$ optimal step size given $(\mathbf{s}^t, \boldsymbol{\alpha}^t, \tilde{\mathbf{y}}^t)$        ▷ Section 5.3.2.2
      $A\boldsymbol{\alpha}^{t+1} \leftarrow (1-\delta)A\boldsymbol{\alpha}^t + \delta A\mathbf{s}^t$        ▷ Frank-Wolfe update on $\boldsymbol{\alpha}$
    end for
    $\mathbf{y}^{r+1} \leftarrow P_{\mathcal{S}}(\tilde{\mathbf{y}}^t)$        ▷ Project the primal solution to the feasible set $\mathcal{S}$
  end for
```

written as

$$\min_{\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma}} g(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma}) = \frac{\eta}{2}\|A\boldsymbol{\alpha} + B\boldsymbol{\beta} + \boldsymbol{\gamma} - \boldsymbol{\theta}_u\|^2 + \langle A\boldsymbol{\alpha} + B\boldsymbol{\beta} + \boldsymbol{\gamma} - \boldsymbol{\theta}_u, \mathbf{y}^r\rangle - \langle \mathbf{1}, \boldsymbol{\beta}\rangle \,, \quad (5.31)$$

$$\text{s.t.} \qquad \gamma_{i:\lambda} \geq 0 \quad \forall i \in \mathcal{V} \quad \forall \lambda \in \mathcal{L}\,,$$

$$\boldsymbol{\alpha} \in \mathcal{C} = \left\{ \boldsymbol{\alpha} \;\middle|\; \begin{array}{l} \alpha^1_{ij:\lambda} + \alpha^2_{ij:\lambda} = \frac{K_{ij}}{2}, \forall i, j \neq i, \forall \lambda \in \mathcal{L} \\ \alpha^1_{ij:\lambda}, \alpha^2_{ij:\lambda} \geq 0, \forall i, j \neq i, \forall \lambda \in \mathcal{L} \end{array} \right\} .$$

Here, $\mathbf{1}$ denotes the vector of all ones of appropriate dimension. Note that we converted our problem to a minimization one by changing the sign of all the terms. This proves Eq. (5.22). □

5.3.2 Algorithm

The dual problem (5.22), in its standard form, can only be tackled using projected gradient descent. However, by separating the variables based on the type of the feasible domains, we propose an efficient block-coordinate descent approach. Each of these blocks are amenable to more sophisticated optimization, resulting in a computationally efficient algorithm. As the dual problem is strictly convex and smooth, the optimal solution is still guaranteed[8]. For $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$, the problem decomposes over the pixels, as shown in Section 5.3.2.1, therefore making it efficient. The minimization with respect to $\boldsymbol{\alpha}$ is over a compact domain, which can be efficiently tackled using the Frank-Wolfe algorithm (see Section 5.2.6). Our complete algorithm is summarized in Algorithm 5.2. In the following sections, we discuss each step in more detail.

[8]Note that, as empirically shown in [Desmaison et al., 2016b], the conditional gradient computed by our algorithm is, in fact, collinear to the exact conditional gradient for a wide range of image sizes and filter standard deviations. Therefore, the update direction of our algorithm is correct but the step size may not be optimal. Thus, we can expect that the approximation introduced by the filtering method does not hinder the convergence our algorithm.

5.3.2.1 Optimizing over $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$

We first turn to the problem of optimizing over $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ while $\boldsymbol{\alpha}^t$ is fixed. Since the dual variable $\boldsymbol{\beta}$ is unconstrained, the minimum value of the dual objective g is attained when $\nabla_{\boldsymbol{\beta}} g(\boldsymbol{\alpha}^t, \boldsymbol{\beta}, \boldsymbol{\gamma}) = 0$.

Proposition 5.3.4. If $\nabla_{\boldsymbol{\beta}} g(\boldsymbol{\alpha}^t, \boldsymbol{\beta}, \boldsymbol{\gamma}) = 0$, then $\boldsymbol{\beta}$ satisfy

$$\boldsymbol{\beta} = B^T \left(A\boldsymbol{\alpha}^t + \boldsymbol{\gamma} - \boldsymbol{\theta}_u\right)/\ell \ . \tag{5.32}$$

Proof. By differentiating the dual objective g with respect to $\boldsymbol{\beta}$ and setting the derivatives to zero, we obtain the above equation. Note that, from Proposition 5.3.3, $B^T\mathbf{y}^r = \mathbf{1}$ since $\mathbf{y}^r \in \mathcal{S}$ (defined in Eq. (5.7)), and $B^TB = \ell I$. Both these identities are used to simplify the above equation. □

Note that, now, $\boldsymbol{\beta}$ is a function of $\boldsymbol{\gamma}$. We therefore substitute $\boldsymbol{\beta}$ in (5.22) and minimize over $\boldsymbol{\gamma}$. Interestingly, the resulting problem can be optimized independently for each pixel, with each subproblem being an ℓ dimensional Quadratic Program (QP) with non-negativity constraints, where ℓ is the number of labels.

Proposition 5.3.5. The optimization over $\boldsymbol{\gamma}$ decomposes over pixels, where, for a pixel i, we have a QP of the form

$$\begin{aligned} \min_{\boldsymbol{\gamma}_i} \quad & \frac{1}{2}\boldsymbol{\gamma}_i^T Q \boldsymbol{\gamma}_i + \left\langle \boldsymbol{\gamma}_i, Q\left((A\boldsymbol{\alpha}^t)_i - \boldsymbol{\theta}_i\right) + \mathbf{y}_i^r \right\rangle \ , \\ \text{s.t.} \quad & \boldsymbol{\gamma}_i \geq \mathbf{0} \ . \end{aligned} \tag{5.33}$$

Here, $\boldsymbol{\gamma}_i$ denotes the vector $\{\gamma_{i:\lambda} \mid \lambda \in \mathcal{L}\}$ and $Q = \eta\,(I - \mathbf{1}/\ell) \in \mathbb{R}^{\ell\times\ell}$, with I the identity matrix and $\mathbf{1}$ the matrix of all ones.

Let us first define a matrix D and analyze its properties. This will be useful to prove Proposition 5.3.5.

Definition 5.3.2. Let $D \in \mathbb{R}^{n\ell\times n\ell}$ be a matrix that satisfies

$$D = I - \frac{BB^T}{\ell} \ , \tag{5.34}$$

where B is defined in Eq. (5.21).

Proposition 5.3.6. The matrix D satisfies the following properties:

1. D is block diagonal, with each block matrix $D_i = I - \mathbf{1}/\ell$, where $I \in \mathbb{R}^{\ell\times\ell}$ is the identity matrix and $\mathbf{1} \in \mathbb{R}^{\ell\times\ell}$ is the matrix of all ones.

2. $D^TD = D$.

Proof. From Proposition 5.3.3, the matrix BB^T is block diagonal with each block $\left(BB^T\right)_i = \mathbf{1}$. Therefore D is block diagonal with each block matrix $D_i = I - \mathbf{1}/\ell$. Note that the block matrices D_i are identical. The second property can be proven using simple matrix algebra. □

Now we turn to the proof of Proposition 5.3.5.

Proof. By substituting $\boldsymbol{\beta}$ in the dual problem (5.22) with Eq. (5.32), the optimization problem over γ takes the following form:

$$\min_{\gamma} g(\boldsymbol{\alpha}^t, \gamma) = \frac{\eta}{2}\|D(A\boldsymbol{\alpha}^t + \gamma - \boldsymbol{\theta}_u)\|^2 + \left\langle D(A\boldsymbol{\alpha}^t + \gamma - \boldsymbol{\theta}_u), \mathbf{y}^r\right\rangle + \frac{1}{\ell}\left\langle \mathbf{1}, A\boldsymbol{\alpha}^t + \gamma - \boldsymbol{\theta}_u\right\rangle \ , \tag{5.35}$$

$$\text{s.t.} \qquad \gamma \geq \mathbf{0}\ ,$$

where $D = I - \frac{BB^T}{\ell}$.

Note that, since $\mathbf{y}^r \in \mathcal{S}$, from Proposition 5.3.3, $B^T\mathbf{y}^r = \mathbf{1}$. Using this fact, the identity $D^TD = D$, and by removing the constant terms, the optimization problem over γ can be simplified:

$$\begin{aligned} \min_{\gamma} g(\boldsymbol{\alpha}^t, \gamma) &= \frac{\eta}{2}\gamma^T D\gamma + \langle \gamma, \eta D(A\boldsymbol{\alpha}^t - \boldsymbol{\theta}_u) + \mathbf{y}^r\rangle\ , \\ \text{s.t.} \qquad & \gamma \geq \mathbf{0}\ . \end{aligned} \tag{5.36}$$

Furthermore, since D is block diagonal from Proposition 5.3.6, we obtain

$$\min_{\gamma \geq \mathbf{0}} g(\boldsymbol{\alpha}^t, \gamma) = \sum_i \min_{\gamma_i \geq \mathbf{0}} \frac{\eta}{2}\gamma_i^T D_i \gamma_i + \langle \gamma_i, \eta D_i\left((A\boldsymbol{\alpha}^t)_i - \boldsymbol{\theta}_i\right) + \mathbf{y}_i^r\rangle\ , \tag{5.37}$$

where the notation γ_i denotes the vector $\{\gamma_{i:\lambda} \mid \lambda \in \mathcal{L}\}$ and $\boldsymbol{\theta}_i = \{\theta_{i:\lambda} \mid \lambda \in \mathcal{L}\}$. By substituting $Q = \eta\, D_i$, the QP associated with each pixel i can be written as

$$\min_{\gamma_i \geq \mathbf{0}} \quad \frac{1}{2}\gamma_i^T Q\gamma_i + \left\langle \gamma_i, Q\left((A\boldsymbol{\alpha}^t)_i - \boldsymbol{\theta}_i\right) + \mathbf{y}_i^r\right\rangle\ . \tag{5.38}$$

□

Each of these ℓ dimensional quadratic programs are optimized using the iterative algorithm of [Xiao and Chen, 2014]. Before we give the update equation, let us first write our problem in the form used in [Xiao and Chen, 2014]. For a given $i \in \mathcal{V}$, this yields

$$\min_{\gamma_i \geq \mathbf{0}} \frac{1}{2}\gamma_i^T Q\gamma_i - \langle \gamma_i, \mathbf{h}_i\rangle\ , \tag{5.39}$$

where

$$\begin{aligned} Q &= \eta\left(I - \frac{\mathbf{1}}{\ell}\right)\ , \\ \mathbf{h}_i &= -Q\left((A\boldsymbol{\alpha}^t)_i - \boldsymbol{\theta}_i\right) - \mathbf{y}_i^r\ . \end{aligned} \tag{5.40}$$

Hence, at each iteration, the element-wise update equation has the following form:

$$\gamma_{i:\lambda} = \gamma_{i:\lambda} \left[\frac{2\,(Q^{-}\gamma_i)_\lambda + h^{+}_{i:\lambda} + \epsilon}{(|Q|\gamma_i)_\lambda + h^{-}_{i:\lambda} + \epsilon} \right] , \tag{5.41}$$

where $Q^{-} = \max(-Q, 0)$, $|Q| = \text{abs}(Q)$, $h^{+}_{i:\lambda} = \max(h_{i:\lambda}, 0)$ and $h^{-}_{i:\lambda} = \max(-h_{i:\lambda}, 0)$ and $0 < \epsilon \ll 1$. These max and abs operations are element-wise. We refer the interested reader to [Xiao and Chen, 2014] for more detail on this update rule.

Note that, even though the matrix Q has ℓ^2 elements, the multiplication by Q can be performed in $\mathcal{O}(\ell)$. In particular, the multiplication by Q can be decoupled into a multiplication by the identity matrix and a matrix of all ones, both of which can be performed in linear time. Similar observations can be made for the matrices Q^{-} and $|Q|$. Hence, the time complexity of the above update is $\mathcal{O}(\ell)$. Consequently, the overall time complexity of optimizing over γ is $\mathcal{O}(n\,\ell)$. Once the optimal γ is computed for a given $\boldsymbol{\alpha}^t$, the corresponding optimal $\boldsymbol{\beta}$ is given by Eq. (5.32).

5.3.2.2 Optimizing over $\boldsymbol{\alpha}$

We now turn to the problem of optimizing over $\boldsymbol{\alpha}$ given $\boldsymbol{\beta}^t$ and γ^t. To this end, we use the Frank-Wolfe algorithm (see Section 5.2.6), which has the advantage of being projection free. Furthermore, for our specific problem, we show that the required conditional gradient can be computed efficiently and the optimal step size can be obtained analytically.

Recall that the Frank-Wolfe algorithm requires the feasible domain to be convex and compact.

Lemma 5.3.1. *The feasible domain $\mathcal{C}$ in Eq.* (5.22) *is convex and compact.*

Proof. Since $\mathcal{C}$ is defined using linear inequalities it is convex. Note that the feasible set $\mathcal{C}$ is separable, *i.e.*, it can be written as $\mathcal{C} = \prod_{i,\,j \neq i,\,\lambda \in \mathcal{L}} \mathcal{C}_{ij:\lambda}$, with

$$\mathcal{C}_{ij:\lambda} = \left\{ (\alpha^1_{ij:\lambda}, \alpha^2_{ij:\lambda}) \mid \alpha^1_{ij:\lambda} + \alpha^2_{ij:\lambda} = K_{ij}/2, \alpha^1_{ij:\lambda}, \alpha^2_{ij:\lambda} \geq 0 \right\} . \tag{5.42}$$

Now, the set $\mathcal{C}_{ij:\lambda}$ is a line satisfying $\alpha^1_{ij:\lambda} + \alpha^2_{ij:\lambda} = K_{ij}/2$ on the positive quadrant of $\mathbb{R}^2$, which is compact. Hence, $\mathcal{C}$ is compact. □

Conditional Gradient Computation. The conditional gradient with respect to $\boldsymbol{\alpha}$ is obtained by solving the following linearization problem

$$\mathbf{s} = \underset{\hat{\mathbf{s}} \in \mathcal{C}}{\text{argmin}} \left\langle \hat{\mathbf{s}}, \nabla_{\boldsymbol{\alpha}} g(\boldsymbol{\alpha}^t, \boldsymbol{\beta}^t, \gamma^t) \right\rangle . \tag{5.43}$$

Here, $\nabla_{\boldsymbol{\alpha}} g(\boldsymbol{\alpha}^t, \boldsymbol{\beta}^t, \gamma^t)$ denotes the gradient of the dual objective function with respect to $\boldsymbol{\alpha}$ evaluated at $(\boldsymbol{\alpha}^t, \boldsymbol{\beta}^t, \gamma^t)$.

Proposition 5.3.7. The conditional gradient $\mathbf{s}$ satisfies

$$(A\mathbf{s})_{i:\lambda} = -\sum_j \left(K_{ij} \mathbb{1}[\tilde{y}^t_{i:\lambda} \geq \tilde{y}^t_{j:\lambda}] - K_{ij} \mathbb{1}[\tilde{y}^t_{i:\lambda} \leq \tilde{y}^t_{j:\lambda}] \right) \,, \tag{5.44}$$

where $\tilde{\mathbf{y}}^t = \eta \left(A\boldsymbol{\alpha}^t + B\boldsymbol{\beta}^t + \gamma^t - \boldsymbol{\theta}_u \right) + \mathbf{y}^r$ using Eq. (5.23).

Proof. The conditional gradient with respect to $\boldsymbol{\alpha}$ is obtained by solving the following linearization problem:

$$\mathbf{s} = \operatorname*{argmin}_{\hat{\mathbf{s}} \in \mathcal{C}} \left\langle \hat{\mathbf{s}}, \nabla_{\boldsymbol{\alpha}} g(\boldsymbol{\alpha}^t, \boldsymbol{\beta}^t, \gamma^t) \right\rangle \,, \tag{5.45}$$

where

$$\nabla_{\boldsymbol{\alpha}} g(\boldsymbol{\alpha}^t, \boldsymbol{\beta}^t, \gamma^t) = A^T \tilde{\mathbf{y}}^t \,, \tag{5.46}$$

with $\tilde{\mathbf{y}}^t = \eta \left(A\boldsymbol{\alpha}^t + B\boldsymbol{\beta}^t + \gamma^t - \boldsymbol{\theta}_u \right) + \mathbf{y}^r$ using Eq. (5.23). Note that $\tilde{\mathbf{y}}^t$ may be primal infeasible, *i.e.*, $\tilde{\mathbf{y}}^t \notin \mathcal{S}$, however, $\mathbf{y}^r \in \mathcal{S}$.

Note that, from Eq. (5.42), the feasible set $\mathcal{C}$ is separable. Therefore, the conditional gradient can be computed separately, corresponding to each set $\mathcal{C}_{ij:\lambda}$. This yields

$$\begin{aligned} \min_{\hat{s}^1_{ij:\lambda}, \hat{s}^2_{ij:\lambda}} \quad & \hat{s}^1_{ij:\lambda} \nabla_{\alpha^1_{ij:\lambda}} g(\boldsymbol{\alpha}^t, \boldsymbol{\beta}^t, \gamma^t) + \hat{s}^2_{ij:\lambda} \nabla_{\alpha^2_{ij:\lambda}} g(\boldsymbol{\alpha}^t, \boldsymbol{\beta}^t, \gamma^t) \,, \\ \text{s.t.} \quad & \hat{s}^1_{ij:\lambda} + \hat{s}^2_{ij:\lambda} = K_{ij}/2 \,, \\ & \hat{s}^1_{ij:\lambda}, \hat{s}^2_{ij:\lambda} \geq 0 \,, \end{aligned} \tag{5.47}$$

where, using Proposition 5.3.2, the gradients can be written as:

$$\begin{aligned} \nabla_{\alpha^1_{ij:\lambda}} g(\boldsymbol{\alpha}^t, \boldsymbol{\beta}^t, \gamma^t) &= \tilde{y}^t_{j:\lambda} - \tilde{y}^t_{i:\lambda} \,, \\ \nabla_{\alpha^2_{ij:\lambda}} g(\boldsymbol{\alpha}^t, \boldsymbol{\beta}^t, \gamma^t) &= \tilde{y}^t_{i:\lambda} - \tilde{y}^t_{j:\lambda} \,. \end{aligned} \tag{5.48}$$

Hence, the minimum is attained at:

$$\begin{aligned} s^1_{ij:\lambda} &= \begin{cases} K_{ij}/2 & \text{if } \tilde{y}^t_{i:\lambda} \geq \tilde{y}^t_{j:\lambda} \\ 0 & \text{otherwise} \,, \end{cases} \\ s^2_{ij:\lambda} &= \begin{cases} K_{ij}/2 & \text{if } \tilde{y}^t_{i:\lambda} \leq \tilde{y}^t_{j:\lambda} \\ 0 & \text{otherwise} \,. \end{cases} \end{aligned} \tag{5.49}$$

Now, from Eq. (5.21), $A\mathbf{s}$ takes the following form:

$$(A\mathbf{s})_{i:\lambda} = -\sum_{j\neq i}\left(\frac{K_{ij}}{2}\mathbb{1}[\tilde{y}^t_{i:\lambda} \geq \tilde{y}^t_{j:\lambda}] - \frac{K_{ij}}{2}\mathbb{1}[\tilde{y}^t_{i:\lambda} \leq \tilde{y}^t_{j:\lambda}] + \frac{K_{ji}}{2}\mathbb{1}[\tilde{y}^t_{j:\lambda} \leq \tilde{y}^t_{i:\lambda}] - \frac{K_{ji}}{2}\mathbb{1}[\tilde{y}^t_{j:\lambda} \geq \tilde{y}^t_{i:\lambda}]\right) , \tag{5.50}$$
$$= -\sum_{j}\left(K_{ij}\mathbb{1}[\tilde{y}^t_{i:\lambda} \geq \tilde{y}^t_{j:\lambda}] - K_{ij}\mathbb{1}[\tilde{y}^t_{i:\lambda} \leq \tilde{y}^t_{j:\lambda}]\right) .$$

Here, we used the symmetry of the kernel matrix K to obtain this result. Note that the second equation is a summation over $j \in \mathcal{V}$. This is true due to the identity $K_{ii}\mathbb{1}[\tilde{y}^t_{i:\lambda} \geq \tilde{y}^t_{i:\lambda}] - K_{ii}\mathbb{1}[\tilde{y}^t_{i:\lambda} \leq \tilde{y}^t_{i:\lambda}] = 0$ when $j = i$. □

Note that Eq. (5.44) has the same form as the LP subgradient (Eq. (20) in [Desmaison et al., 2016a]). This is not a surprising result. In fact, it has been shown that, for certain problems, there exists a duality relationship between subgradients and conditional gradients [Bach, 2015]. To compute this subgradient, the state-of-the-art algorithm proposed in [Desmaison et al., 2016a] has a time complexity linearithmic in the number of pixels. Unfortunately, since this constitutes a critical step of both our algorithm and that of [Desmaison et al., 2016a], such a linearithmic cost greatly affects their efficiency. In Section 5.4, however, we show that this complexity can be reduced to linear, thus effectively leading to a speedup of an order of magnitude in practice.

Optimal Step Size. One of the main difficulties of using an iterative algorithm, whether subgradient or conditional gradient descent, is that its performance depends critically on the choice of the step size. Here, we can analytically compute the optimal step size that results in the maximum decrease in the objective for the given descent direction.

Proposition 5.3.8. The optimal step size δ satisfies

$$\delta = P_{[0,1]}\left(\frac{\langle A\boldsymbol{\alpha}^t - A\mathbf{s}^t, \tilde{\mathbf{y}}^t\rangle}{\eta\|A\boldsymbol{\alpha}^t - A\mathbf{s}^t\|^2}\right) . \tag{5.51}$$

Here, $P_{[0,1]}$ denotes the projection to the interval $[0,1]$, that is, clipping the value to lie in $[0,1]$.

Proof. The optimal step size δ gives the maximum decrease in the objective function g given the descent direction $\mathbf{s}^t$. This can be formulated as the following optimization problem:

$$\begin{aligned} \min_{\delta} \quad & \frac{\eta}{2}\left\|A\boldsymbol{\alpha}^t + \delta\left(A\mathbf{s}^t - A\boldsymbol{\alpha}^t\right) + B\boldsymbol{\beta}^t + \boldsymbol{\gamma}^t - \boldsymbol{\theta}_u\right\|^2 \\ & + \left\langle A\boldsymbol{\alpha}^t + \delta\left(A\mathbf{s}^t - A\boldsymbol{\alpha}^t\right) + B\boldsymbol{\beta}^t + \boldsymbol{\gamma}^t - \boldsymbol{\theta}_u), \mathbf{y}^r\right\rangle - \langle \mathbf{1}, \boldsymbol{\beta}\rangle \ , \\ \text{s.t.} \quad & \delta \in [0,1] \ . \end{aligned} \tag{5.52}$$

Note that the above function is optimized over the scalar variable δ and the minimum is attained when the derivative is zero. Hence, setting the derivative to zero, we have

$$\begin{aligned} 0 &= \eta \left\langle \delta \left(A\mathbf{s}^t - A\boldsymbol{\alpha}^t \right) + A\boldsymbol{\alpha}^t + B\boldsymbol{\beta}^t + \boldsymbol{\gamma}^t - \boldsymbol{\theta}_u, A\mathbf{s}^t - A\boldsymbol{\alpha}^t \right\rangle + \left\langle \mathbf{y}^r, A\mathbf{s}^t - A\boldsymbol{\alpha}^t \right\rangle \ , \quad (5.53) \\ \delta &= \frac{\left\langle A\boldsymbol{\alpha}^t - A\mathbf{s}^t, \eta \left(A\boldsymbol{\alpha}^t + B\boldsymbol{\beta}^t + \boldsymbol{\gamma}^t - \boldsymbol{\theta}_u \right) + \mathbf{y}^r \right\rangle}{\eta \|A\boldsymbol{\alpha}^t - A\mathbf{s}^t\|^2} \ , \\ \delta &= \frac{\left\langle A\boldsymbol{\alpha}^t - A\mathbf{s}^t, \tilde{\mathbf{y}}^t \right\rangle}{\eta \|A\boldsymbol{\alpha}^t - A\mathbf{s}^t\|^2} \ . \end{aligned}$$

In fact, if the optimal δ is out of the interval $[0, 1]$, the value is simply truncated to be in $[0, 1]$. □

Memory Efficiency. For a dense CRF, the dual variable $\boldsymbol{\alpha}$ requires $\mathcal{O}(n^2\,\ell)$ storage, which becomes infeasible since n is the number of pixels in an image. Note, however, that $\boldsymbol{\alpha}$ always appears in the product $\tilde{\boldsymbol{\alpha}} = A\boldsymbol{\alpha}$ in Algorithm 5.2. Therefore, we only store the variable $\tilde{\boldsymbol{\alpha}}$, which reduces the storage complexity to $\mathcal{O}(n\,\ell)$.

5.3.2.3 Summary

To summarize, our method has four desirable qualities of an efficient iterative algorithm. First, it can benefit from an initial solution obtained by a faster but less accurate algorithm, such as mean-field or DC relaxation. Second, with our choice of a quadratic proximal term, the dual of the proximal problem can be efficiently optimized in a block-wise fashion. Specifically, the dual variables $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ are computed efficiently by minimizing one small QP (of dimension the number of labels) for each pixel independently. The remaining dual variable $\boldsymbol{\alpha}$ is optimized using the Frank-Wolfe algorithm, where the conditional gradient is computed in linear time, and the optimal step size is obtained analytically. Overall, the time complexity of one iteration of our algorithm is $\mathcal{O}(n\,\ell)$. To the best of our knowledge, this constitutes the first LP minimization algorithm for dense CRFs that has linear time iterations. We denote this algorithm as PROX-LP.

5.4 Fast Conditional Gradient Computation

The algorithm described in the previous section assumes that the conditional gradient (Eq. (5.44)) can be computed efficiently. Note that Eq. (5.44) contains two terms that are similar up to sign and order of the label constraint in the indicator function. To simplify the discussion, let us focus on the first term and on a particular label i, which we will not explicitly write in the remainder of this section. The second term in Eq. (5.44) and the other labels can be handled in the same manner. With these simplifications, we need to efficiently compute an expression of the form

$$\forall\, i \in \mathcal{V}, \quad u'_i = \sum_j k(\mathbf{f}_i, \mathbf{f}_j)\, u_j\, \mathbb{1}[y_i \geq y_j] \ , \tag{5.54}$$

with $y_i, y_j \in [0,1]$ and $\mathbf{f}_i, \mathbf{f}_j \in \mathbb{R}^{\mathrm{d}}$ for all $i, j \in \mathcal{V}$. Note that, in Eq. (5.44), the value u_j was assumed to be 1, but here we consider the general case where $u_j \in \mathbb{R}$.

The usual way of speeding up computations involving such Gaussian kernels is by using the efficient filtering method [Adams et al., 2010]. This approximate method has proven accurate enough for similar applications [Desmaison et al., 2016a; Krähenbühl Philipp, 2011]. In our case, due to the ordering constraint $\mathbb{1}[y_i \geq y_j]$, the symmetry is broken and the direct application of the filtering method is impossible. In [Desmaison et al., 2016a], the authors tackled this problem using a divide-and-conquer strategy, which lead to a time complexity of $\mathcal{O}(d^2 n \log(n))$. In practice, this remains a prohibitively high run time, particularly since gradient computations are performed many times over the course of the algorithm. Here, we introduce a more efficient method.

Specifically, we show that the term in Eq. (5.54) can be computed in $\mathcal{O}(Hdn)$ time (where H is a small constant defined in Section 5.4.2), at the cost of additional storage. In practice, this leads to a speedup of one order of magnitude. Below, we first briefly review the original filtering algorithm and then explain our modified algorithm that efficiently handles the ordering constraints.

5.4.1 Original Filtering Method

In this section, we assume that the reader is familiar with the permutohedral lattice based filtering method [Adams et al., 2010] and only a brief overview is provided. We refer the interested reader to the original paper [Adams et al., 2010] and the thesis [Adams, 2011] for more detail. Furthermore, empirical evaluation on the approximation error introduced by this filtering method in the context of dense CRFs can be found in [Desmaison et al., 2016b]. In short, the approximation introduces a constant multiplicative factor for a wide range of image sizes and filter standard deviations.

In [Adams et al., 2010], each pixel $i \in \mathcal{V}$ is associated with a tuple $(\mathbf{f}_i, u_i)$, which we call a *feature point*. The elements of this tuple are the feature $\mathbf{f}_i \in \mathbb{R}^{\mathrm{d}}$ and the value $u_i \in \mathbb{R}$. At the beginning of the algorithm, the feature points are embedded in a d-dimensional hyperplane tessellated by the *permutohedral lattice* (see Figure 5.2). The vertices of this permutohedral lattice are called *lattice points*, denoted by $\mathcal{P}$ and each lattice point l is associated with a scalar value $\bar{u}_l$. Furthermore, the neighbouring feature points of a lattice point l is denoted by $N(l)$ and the neighbouring lattice points of a feature point i is denoted by $\bar{N}(i)$. These neighbourhoods are explained in Figure 5.2. In addition, the *barycentric weight*[9] between the lattice point l and feature point j is denoted with w_{lj}. Finally, the set of feature point scores is denoted by $\mathcal{Y} = \{y_j \mid j \in \mathcal{V}\}$, their set of values is denoted by $\mathcal{U} = \{u_j \mid j \in \mathcal{V}\}$ and the set of lattice point values is denoted by $\bar{\mathcal{U}} = \{\bar{u}_l \mid l \in \mathcal{P}\}$.

[9]The barycentric weights are based on the barycentric coordinate system introduced in [Hille, 2005]. This is a coordinate system in which the location of a point of a simplex (a triangle, tetrahedron, etc.) is specified as the center of mass, or barycenter, of usually unequal masses placed at its vertices. Barycentric interpolation was shown to be exponentially cheaper than multi-linear interpolation [Adams et al., 2010].

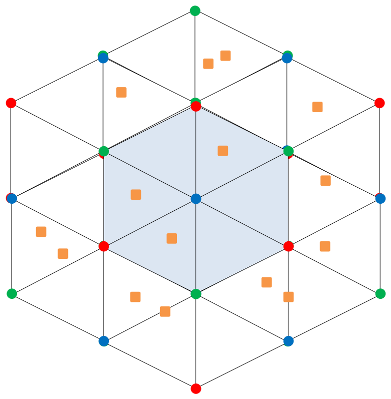


Figure 5.2: *A* 2*-dimensional hyperplane tessellated by the permutohedral lattice. The feature points are denoted with squares and the lattice points with circles. The neighbourhood of the center lattice point is shaded and, for a feature point, the neighbouring lattice points are the vertices of the enclosing triangle.*

Once the permutohedral lattice is constructed, the algorithm performs three main steps: *splatting*, *blurring* and *slicing*. During splatting, for each lattice point, the values of the neighbouring feature points are accumulated using barycentric interpolation. Next, during blurring, the values of the lattice points are convolved with a one dimensional truncated Gaussian kernel along each feature dimension separately. Finally, during slicing, the resulting values of the lattice points are propagated back to the feature points using the same barycentric weights. These steps are explained graphically in the top row of Figure 5.3. The pseudocode of the algorithm is given in Algorithm 5.3. The time complexity of this algorithm is $\mathcal{O}(dn)$ [Adams et al., 2010; Krähenbühl Philipp, 2011], and the complexity of the permutohedral lattice creation $\mathcal{O}(d^2n)$. Since the approach in [Desmaison et al., 2016a] creates multiple lattices at every iteration, the overall complexity of this approach is $\mathcal{O}(d^2 n \log(n))$.

Note that, in this original algorithm, there is no notion of score y_i associated with each pixel. In particular, during splatting, the values u_i are accumulated to the neighbouring lattice points without considering their scores. Therefore, this algorithm cannot be directly applied to handle our ordering constraint $\mathbb{1}[y_i \geq y_j]$.

5.4.2 Modified Filtering Method

We now introduce a filtering-based algorithm that can handle ordering constraints. To this end, we uniformly discretize the continuous interval $[0, 1]$ into H different discrete bins, or levels. Note that each pixel, or feature point, belongs to exactly one of these bins, according to its corresponding score. In particular, each bin $h \in \{0, \dots, H-1\}$ is associated with an interval which is identified as $\left[\frac{h}{H-1}, \frac{h+1}{H-1}\right)$. Given the score y_j of the feature point j, its bin/level can be identified as

$$h_j = \lfloor y_j (H-1) \rfloor \ , \tag{5.55}$$

Algorithm 5.3 Original Filtering Method [Adams et al., 2010]

```
Require: Set of lattice points $\mathcal{P}$ and the set of feature point values $\mathcal{U}$
  $\mathcal{U}' \leftarrow \mathbf{0} \quad \bar{\mathcal{U}} \leftarrow \mathbf{0} \quad \bar{\mathcal{U}}' \leftarrow \mathbf{0}$        ▷ Initialization
  for each $l \in \mathcal{P}$ do                                              ▷ Splatting
      for each $j \in N(l)$ do
          $\bar{u}_l \leftarrow \bar{u}_l + w_{lj}\, u_j$
      end for
  end for
  $\bar{\mathcal{U}}' \leftarrow k \otimes \bar{\mathcal{U}}$                    ▷ Blurring
  for each $i \in \mathcal{V}$ do                                              ▷ Slicing
      for each $l \in \bar{N}(i)$ do
          $u'_i \leftarrow u'_i + w_{li}\, \bar{u}'_l$
      end for
  end for
```

where $\lfloor \cdot \rfloor$ denotes the standard floor function. Note that the last bin (with bin id $H-1$) is associated with the interval $[1, \cdot)$. Since $y_j \leq 1$, this bin contains the feature points whose scores are exactly 1. We then propose to instantiate H permutohedral lattices, one for each level $h \in \{0, \dots, H-1\}$. In other words, at each level h, there is a lattice point l, whose value we denote by $\bar{u}_{l:h}$.

To handle the ordering constraints, we then modify the splatting step in the following manner. A feature point belonging to bin q is splat to the permutohedral lattices corresponding to levels $q \leq h < H$. Blurring is then performed independently in each individual permutohedral lattice. This guarantees that a feature point will only influence the values of the feature points that belong to the same level or higher ones. In other words, a feature point j influences the value of a feature point i only if $y_i \geq y_j$. Finally, during the slicing step, the value of a feature point belonging to level q is recovered from the q^{th} permutohedral lattice. Our algorithm is depicted graphically in the bottom row of Figure 5.3. Its pseudocode is provided in Algorithm 5.4. In this algorithm, we denote the set of values corresponding to all the lattice points at level h as $\bar{\mathcal{U}}_h = \{u_{l:h} \mid l \in \mathcal{P}\}$.

Note that, while discussed for constraints of the form $\mathbb{1}[y_i \geq y_j]$, this algorithm can easily be adapted to handle $\mathbb{1}[y_i \leq y_j]$ constraints, which are required for the second term in Eq. (5.44). In particular, one needs to change the interval identified by the bin h to: $\left(\frac{h-1}{H-1}, \frac{h}{H-1}\right]$. Using this fact, one can easily derive the splatting and slicing equations for the $\mathbb{1}[y_i \leq y_j]$ constraint.

Overall, our modified filtering method has a time complexity of $\mathcal{O}(Hdn)$ and a space complexity of $\mathcal{O}(Hdn)$. Note that the complexity of the lattice creation is still $\mathcal{O}(d^2 n)$ and can be reused for each of the H instances. Moreover, as opposed to the method in [Desmaison et al., 2016a], this operation is performed only once, during the initialization step. In practice, we were able to choose H as small as 10, thus achieving a substantial speedup compared to the divide-and-conquer strategy of [Desmaison et al., 2016a]. By discretizing the interval $[0, 1]$, we add another level

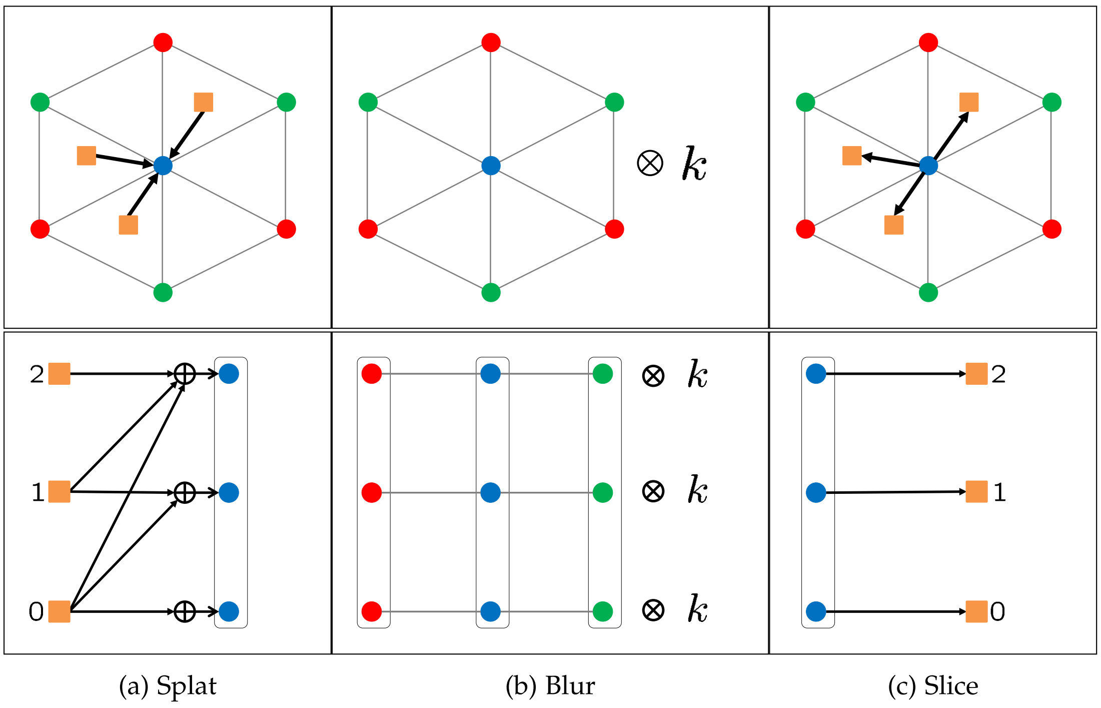


Figure 5.3: **Top row:** *Original filtering method. The barycentric interpolation is denoted by an arrow and k here is the truncated Gaussian kernel. During splatting, for each lattice point, the values of the neighbouring feature points are accumulated using barycentric interpolation. Next, the lattice points are blurred with k along each dimension. Finally, the values of the lattice points are given back to the feature points using the same barycentric weights.* **Bottom row:** *Our modified filtering method. Here, $H = 3$, and the figure therefore illustrates 3 lattices. We write the bin number of each feature point next to the point. Note that, at the splatting step, the value of a feature point is accumulated to its neighbouring lattice points only if it is above or equal to the feature point level. Then, blurring is performed at each level independently. Finally, the resulting values are recovered from the lattice points at the feature point level.*

of approximation to the overall algorithm. However, this approximation can be eliminated by using a dynamic data structure, which we briefly explain below.

5.4.2.1 Adaptive Version

Here, we briefly explain the adaptive version of our modified algorithm, which replaces the fixed discretization with a dynamic data structure. Effectively, discretization boils down to storing a vector of length H at each lattice point. Instead of such a fixed-length vector, one can use a dynamic data structure that grows with the number of different scores encountered at each lattice point in the splatting and blurring steps. In the worst case, *i.e.*, when all the neighbouring feature points have different

Algorithm 5.4 Modified Filtering Method

```
Require: Lattice points $\mathcal{P}$, feature values $\mathcal{U}$, discrete levels $H$ and scores $\mathcal{Y}$
  $\mathcal{U}' \leftarrow \mathbf{0} \quad \bar{\mathcal{U}} \leftarrow \mathbf{0} \quad \bar{\mathcal{U}}' \leftarrow \mathbf{0}$        ▷ Initialization
  for each $l \in \mathcal{P}$ do        ▷ Splatting
      for each $j \in N(l)$ do
          $h_j \leftarrow \lfloor y_j\,(H-1) \rfloor$
          for each $h \in \{h_j \ldots H-1\}$ do        ▷ Splat at the feature point level and above
              $\bar{u}_{l:h} \leftarrow \bar{u}_{l:h} + w_{lj}\,u_j$
          end for
      end for
  end for
  for each $h \in \{0 \ldots H-1\}$ do        ▷ Blurring at each level independently
      $\bar{\mathcal{U}}'_h \leftarrow k \otimes \bar{\mathcal{U}}_h$
  end for
  for each $i \in \mathcal{V}$ do        ▷ Slicing
      $h_i \leftarrow \lfloor y_i\,(H-1) \rfloor$
      for each $l \in \bar{N}(i)$ do
          $u'_i \leftarrow u'_i + w_{li}\,\bar{u}'_{l:h_i}$        ▷ Slice at the feature point level
      end for
  end for
```

scores, the maximum number of values to store at a lattice point is

$$H = \max_{l \in \mathcal{P}} |N^2(l)| \,, \tag{5.56}$$

where $|N^2(l)|$ denotes the the number of feature points in the union of neighbourhoods of the lattice point l and its neighbouring lattice points (the vertices of the shaded hexagon in Figure 5.2). In our experiments, we observed that $|N^2(l)|$ is usually less than 100, with an average around 10. Empirically, however, we found this dynamic version to be slightly slower than the static one. We conjecture that this is due to the static version benefiting from better compiler optimization. Furthermore, both the versions obtained results with similar accuracy and therefore we used the static one for all our experiments.

5.5 Related Work

We review the past work on three different aspects of our work in order to highlight our contributions.

5.5.1 Dense CRF

The fully-connected CRF has become increasingly popular for semantic segmentation. It is particularly effective at preventing oversmoothing, thus providing better accuracy at the boundaries of objects. As a matter of fact, in a complementary direction, many methods have now proposed to combine dense CRFs with convolutional neural networks [Chen et al., 2014; Schwing and Urtasun, 2015; Zheng et al., 2015] to achieve state-of-the-art performance on segmentation benchmarks.

The main challenge that had previously prevented the use of dense CRFs is their computational cost at inference, which, naively, is $\mathcal{O}(n^2)$ per iteration. In the case of Gaussian pairwise potential, the efficient filtering method of [Adams et al., 2010] proven to be key to the tractability of inference in the dense CRF. While an approximate method, the accuracy of the computation proven sufficient for practical purposes. This was first observed in [Krähenbühl Philipp, 2011] for the specific case of mean-field inference. More recently, several continuous relaxations, such as QP, DC and LP, were also shown to be applicable to minimizing the dense CRF energy by exploiting this filtering procedure in various ways [Desmaison et al., 2016a]. Unfortunately, while tractable, minimizing the LP relaxation, which is known to provide the best approximation to the original labelling problem, remained too slow in practice [Desmaison et al., 2016a]. Our algorithm is faster both theoretically and empirically. Furthermore, and as evidenced by our experiments, it yields lower energy values than any existing dense CRF inference strategy.

5.5.2 LP Relaxation

There are two ways to relax the integer program (5.6) to a linear program, depending on the label compatibility function: 1) the standard LP relaxation [Chekuri et al., 2004; Werner, 2007]; and 2) the LP relaxation specialized to the Potts model [Kleinberg and Tardos, 2002]. There are many notable works on minimizing the standard LP relaxation on sparse CRFs. This includes the algorithms that directly make use the dual of this LP [Kolmogorov, 2006; Komodakis et al., 2011; Wainwright et al., 2005] (see Section 2.3.4 for a review) and those based on a proximal minimization framework [Meshi et al., 2015; Ravikumar et al., 2008]. Unfortunately, all of the above algorithms exploit the sparsity of the problem, and they would yield an $\mathcal{O}(n^2)$ cost per iteration in the fully-connected case. In this work, we focus on the Potts model based LP relaxation for dense CRFs and provide an algorithm whose iterations have time complexity $\mathcal{O}(n)$. Even though we focus on the Potts model, as pointed out in [Desmaison et al., 2016a], this LP relaxation can be extended to general label compatibility functions using a hierarchical Potts model [Kumar and Koller, 2009].

5.5.3 Frank-Wolfe

The optimization problem of structural support vector machines (SVM) has a form similar to our proximal problem. The Frank-Wolfe algorithm [Frank and Wolfe, 1956] was shown to provide an effective and efficient solution to such a problem via block-coordinate optimization [Lacoste-Julien et al., 2012]. Several works have recently focused on improving the performance of this algorithm [Osokin et al., 2016; Shah et al., 2015] and extended its application domain [Krishnan et al., 2015]. Our work draws inspiration from this structural SVM literature, and makes use of the Frank-Wolfe algorithm to solve a subtask of our overall LP minimization method. Efficiency, however, could only be achieved thanks to our modification of the efficient filtering procedure to handle ordering constraints.

To the best of our knowledge, our approach constitutes the first LP minimization algorithm for dense CRFs to have linear time iterations. Our experiments demonstrate the importance of this result on both speed and labelling quality. Being fast, our algorithm can be incorporated in any end-to-end learning framework, such as [Zheng et al., 2015]. We therefore believe that it will have a significant impact on future semantic segmentation results, and potentially in other application domains.

5.6 Experiments

In this section, we first discuss two variants that further speedup our algorithm and some implementation details. We then turn to the empirical results.

5.6.1 Accelerated Variants

Empirically we observed that our algorithm can be accelerated by restricting the optimization procedure to affect only relevant subsets of labels and pixels. These subsets can be identified from an intermediate solution of PROX-LP (Algorithm 5.2). In particular, we remove the label λ from the optimization if $y_{i:\lambda} < 0.01$ for all pixels i. In other words, the score of a label λ is insignificant for all the pixels. We denote this version as PROX-LP$_\ell$. Similarly, we optimize over a pixel only if it is *uncertain* in choosing a label. Here, a pixel i is called uncertain if $\max_\lambda y_{i:\lambda} < 0.95$. In other words, no label has a score higher than 0.95. The intuition behind this strategy is that, after a few iterations of PROX-LP$_\ell$, most of the pixels are labelled correctly, and we only need to fine tune the few remaining ones. In practice, we limit this restricted set to 10% of the total number of pixels. We denote this accelerated algorithm as PROX-LP$_{\text{acc}}$. As shown in our experiments, PROX-LP$_{\text{acc}}$ yields a significant speedup at virtually no loss in the quality of the results.

5.6.2 Implementation Details

In practice, we initialize our algorithm with the solution of the best continuous relaxation algorithm, which is called DC$_{\text{neg}}$ in [Desmaison et al., 2016a]. The parameters of our algorithm, such as the proximal regularization constant η and the stopping criterion, are chosen manually. A small value of η leads to easier minimization of the proximal problem, but also yields smaller steps at each proximal iteration. We found $\eta = 0.1$ to work well in all our experiments. We fixed the maximum number of proximal steps (R in Algorithm 5.2) to 10, and each proximal step is optimized for a maximum of 5 Frank-Wolfe iterations (T in Algorithm 5.2). In all our experiments the number of levels H is fixed to 10.

Dataset	Algorithm	$w^{(1)}$	σ_1	$w^{(2)}$	$\sigma_{2:s}$	$\sigma_{2:c}$
MSRC	MF	7.467846	1.000000	4.028773	35.865959	11.209644
	DC_{neg}	2.247081	3.535267	1.699011	31.232626	7.949970
Pascal	MF	100.000000	1.000000	74.877398	50.000000	5.454272
	DC_{neg}	0.500000	3.071772	0.960811	49.785678	1.000000

Table 5.1: *Parameters tuned for MF and DC_{neg} on the MSRC and Pascal validation sets using Spearmint [Snoek et al., 2012] (see Section 5.6.3).*

5.6.3 Segmentation Results

We evaluated our algorithm on the MSRC and Pascal VOC 2010 [Everingham et al., 2010] segmentation datasets, and compare it against mean-field inference (MF) [Krähenbühl Philipp, 2011], the best performing continuous relaxation method of [Desmaison et al., 2016a] (DC_{neg}) and the subgradient-based LP minimization method of [Desmaison et al., 2016a] (SG-LP). Note that, in [Desmaison et al., 2016a], the LP was initialized with the DC_{neg} solution and optimized for 5 iterations. Furthermore, the LP optimization was performed on a subset of labels identified by the DC_{neg} solution in a similar manner to the one discussed in Section 5.6.1. We refer to this algorithm as SG-LP_{ℓ}. For all the baselines, we employed the respective authors' implementations that were obtained from the web or through personal communication. Furthermore, for all the algorithms, the integral labelling is computed from the fractional solution using the *argmax* rounding scheme.

For both datasets, we used the same splits and unary potentials as in [Krähenbühl Philipp, 2011]. The pairwise potentials were defined using two kernels: a spatial kernel and a bilateral one [Krähenbühl Philipp, 2011]. Our pixel compatibility function can be written as

$$K_{ab} = w^{(1)} \exp\left(-\frac{|\mathbf{p}_a - \mathbf{p}_b|^2}{\sigma_1}\right) + w^{(2)} \exp\left(-\frac{|\mathbf{p}_a - \mathbf{p}_b|^2}{\sigma_{2:s}} - \frac{|\mathbf{I}_a - \mathbf{I}_b|^2}{\sigma_{2:c}}\right) , \tag{5.57}$$

where $\mathbf{p}_a$ denotes the (x, y) position of pixel a measured from top left and $\mathbf{I}_a$ denotes the (r, g, b) values of pixel a. Note that there are 5 learnable parameters: $w^{(1)}, \sigma_1, w^{(2)}, \sigma_{2:s}, \sigma_{2:c}$. For each method, these kernel parameters were cross validated on validation data using Spearmint [Snoek et al., 2012] (with a budget of 2 days). To be able to compare energy values, we then evaluated all methods with the same parameters. In other words, for each dataset, each method was run several times with different parameter values. The final parameter values for MF and DC_{neg} are given in Table 5.1. Note that, on MSRC, cross-validation was performed on the less accurate ground truth provided with the original dataset. Nevertheless, we evaluated all methods on the accurate ground truth annotations provided by [Krähenbühl Philipp, 2011].

5.6.3.1 Results with the Parameters Tuned for $\mathrm{DC}_{\mathrm{neg}}$

The results for the parameters tuned for $\mathrm{DC}_{\mathrm{neg}}$ on the MSRC and Pascal datasets are given in Table 5.2. Here MF5 denotes the mean-field algorithm run for 5 iterations. In Figure 5.4, we show the assignment energy as a function of time for an image in MSRC (the tree image in Figure 5.5) and for an image in Pascal (the sheep image in Figure 5.5). Furthermore, we provide some of the segmentation results in Figure 5.5. Since the resulting segmentations of all versions of our algorithm are visually indistinguishable, only the results for the fully accelerated version ($\mathrm{PROX\text{-}LP}_{\mathrm{acc}}$) are shown.

In summary, $\mathrm{PROX\text{-}LP}_{\ell}$ obtains the lowest integral energy in both datasets. Furthermore, our fully accelerated version is the fastest LP minimization algorithm and always outperforms the baselines by a great margin in terms of energy. From Figure 5.5, we can see that $\mathrm{PROX\text{-}LP}_{\mathrm{acc}}$ marks most of the crucial pixels (*e.g.*, object boundaries) as *uncertain*, and optimizes over them efficiently and effectively. Note that, on top of being fast, $\mathrm{PROX\text{-}LP}_{\mathrm{acc}}$ obtains the highest accuracy in MSRC for the parameters tuned for $\mathrm{DC}_{\mathrm{neg}}$.

5.6.3.2 Results with the Parameters Tuned for MF

To ensure consistent behaviour across different energy parameters, we ran the same experiments for the parameters tuned for MF. In this setting, all versions of our algorithm again yield significantly lower energies than the baselines. For this parameter setting, the respective timing plots and segmentation results are given in Figs. 5.6 and 5.7, and the quantitative results are summarized in Table 5.3.

Interestingly, for the parameters tuned for MF, even though our algorithm obtains much lower energies, MF yields the best segmentation accuracy. In fact, one can argue that the parameters tuned for MF do not model the segmentation problem accurately, but were tuned such that the inaccurate MF inference yields good results. Note that, in the Pascal dataset, when tuned for MF, the Gaussian mixture coefficients are very high (see Table 5.1). In such a setting, $\mathrm{DC}_{\mathrm{neg}}$ ended up classifying all pixel in most images as background. In fact, $\mathrm{SG\text{-}LP}_{\ell}$ was able to improve over $\mathrm{DC}_{\mathrm{neg}}$ in only 1% of the images, whereas all our versions improved over $\mathrm{DC}_{\mathrm{neg}}$ in roughly 25% of the images. Furthermore, our accelerated versions could not get any advantage over the standard version and resulted in similar run times. Note that, in most of the images, the *uncertain* pixels are in fact the entire image, as shown in Figure 5.7.

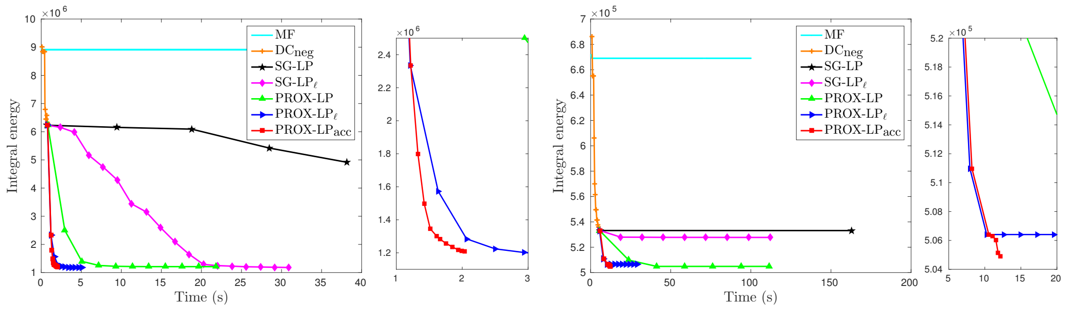


Figure 5.4: *Assignment energy as a function of time with the parameters tuned for DC$_{neg}$ for an image in (**left**) MSRC and (**right**) Pascal (see Section 5.6.3.1). A zoomed-in version is shown next to each plot. Except MF, all other algorithms are initialized with DC$_{neg}$. Note that PROX-LP clearly outperforms SG-LP$_\ell$ by obtaining much lower energies in fewer iterations. Furthermore, the accelerated versions of our algorithm obtain roughly the same energy as PROX-LP but significantly faster.*

	Algorithm	MF5	MF	DC$_{neg}$	SG-LP$_\ell$	PROX-LP	PROX-LP$_\ell$	PROX-LP$_{acc}$	Avg. E ($\times 10^3$)	Avg. T (s)	Acc.	IoU
MSRC	MF5	-	0	0	0	0	0	0	8078.0	**0.2**	79.33	52.30
	MF	96	-	0	0	0	0	0	8062.4	0.5	79.35	52.32
	DC$_{neg}$	96	96	-	0	0	0	0	3539.6	1.3	83.01	57.92
	SG-LP$_\ell$	96	96	90	-	3	1	1	3335.6	13.6	83.15	58.09
	PROX-LP	96	96	94	92	-	13	45	1274.4	23.5	83.99	**59.66**
	PROX-LP$_\ell$	96	96	95	94	81	-	61	**1189.8**	6.3	83.94	59.50
	PROX-LP$_{acc}$	96	96	95	94	49	31	-	1340.0	3.7	**84.16**	59.65
Pascal	MF5	-	13	0	0	0	0	0	1220.8	0.8	79.13	27.53
	MF	2	-	0	0	0	0	0	1220.8	**0.7**	79.13	27.53
	DC$_{neg}$	99	99	-	-	0	0	0	629.5	3.7	80.43	28.60
	SG-LP$_\ell$	99	99	95	-	5	12	12	617.1	84.4	80.49	**28.68**
	PROX-LP	99	99	95	84	-	32	50	507.7	106.7	80.63	28.53
	PROX-LP$_\ell$	99	99	86	86	64	-	43	**502.1**	22.1	**80.65**	28.29
	PROX-LP$_{acc}$	99	99	86	86	46	39	-	507.7	14.7	80.58	28.45

Table 5.2: *Results on the MSRC and Pascal datasets with the parameters tuned for DC$_{neg}$ (see Section 5.6.3.1). We show: the percentage of images where the row method strictly outperforms the column one on the final integral energy, the average integral energy over the test set, the average run time, the segmentation accuracy and the intersection over union score. Note that all versions of our algorithm obtain much lower energies than the baselines. Interestingly, while our fully accelerated version does slightly worse in terms of energy, it is the best in terms of the segmentation accuracy in MSRC.*

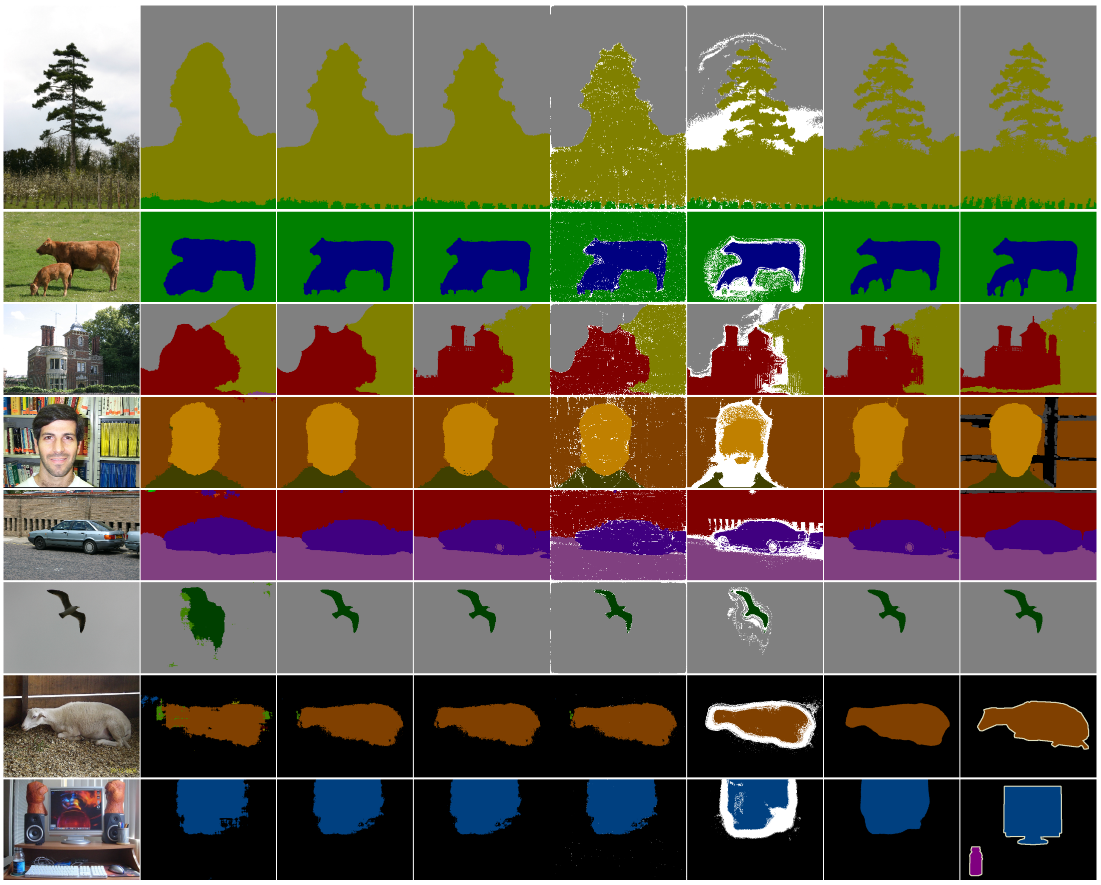


Figure 5.5: *Results with the parameters tuned for DC$_{neg}$ for images in (**top six rows**) MSRC and (**bottom two rows**) Pascal (see Section 5.6.3.1). The uncertain pixels identified by DC$_{neg}$ and PROX-LP$_{acc}$ are marked in white. Note that all versions of our algorithm obtain visually good segmentations similar to that of PROX-LP$_{acc}$ (not shown). In addition, even though DC$_{neg}$ is less accurate (the percentatge of uncertain pixels for DC$_{neg}$ is usually less than 1%) in predicting uncertain pixels, our algorithm marks most of the crucial pixels (object boundaries and shadows) as uncertain. Furthermore, in the MSRC images, the improvement of PROX-LP$_{acc}$ over the baselines is clearly visible and the final segmentation is virtually the same as the accurate ground truth.*

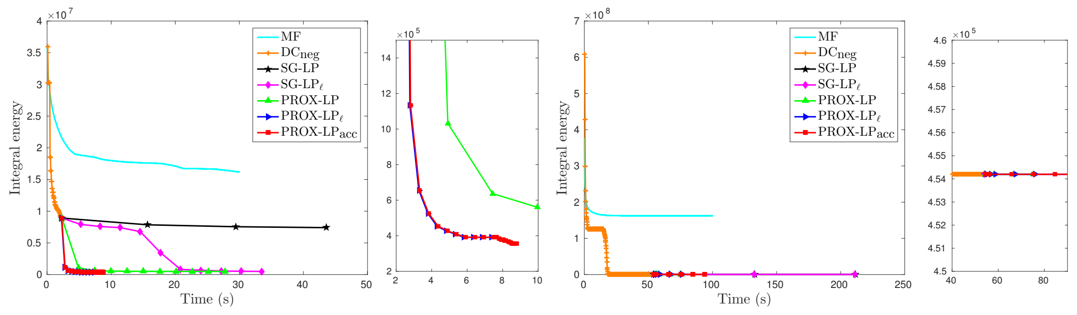


Figure 5.6: *Assignment energy as a function of time with the parameters tuned for MF for an image in (**left**) MSRC and (**right**) Pascal (see Section 5.6.3.2). A zoomed-in version is shown next to each plot. Except for MF, all the algorithms were initialized with DC$_{neg}$. For the MSRC image, PROX-LP clearly outperforms SG-LP$_\ell$ by obtaining much lower energies in fewer iterations, and the accelerated versions of our algorithm obtain roughly the same energy as PROX-LP but significantly faster. For the Pascal image, however, no LP algorithm is able to improve over DC$_{neg}$. Note that, in the Pascal dataset, for the MF parameters, DC$_{neg}$ ended up classifying all pixels in most images as background (which yields low energy values) and no LP algorithm is able to improve over it.*

	Algorithm	MF5	MF	DC$_{neg}$	SG-LP$_\ell$	PROX-LP	PROX-LP$_\ell$	PROX-LP$_{acc}$	Avg. E ($\times 10^3$)	Avg. T (s)	Acc.	IoU
MSRC	MF5	-	0	0	0	0	0	0	2366.6	**0.2**	81.14	54.60
	MF	95	-	18	15	2	1	2	1053.6	13.0	**83.86**	**59.75**
	DC$_{neg}$	95	77	-	0	0	0	0	812.7	2.8	83.50	59.67
	SG-LP$_\ell$	95	80	48	-	2	0	1	800.1	37.3	83.51	59.68
	PROX-LP	95	93	95	93	-	35	46	265.6	27.3	83.01	58.74
	PROX-LP$_\ell$	95	94	94	94	59	-	43	**261.2**	13.9	82.98	58.62
	PROX-LP$_{acc}$	95	93	93	93	49	46	-	295.9	7.9	83.03	58.97
Pascal	MF5	-	-	1	1	0	0	0	40779.8	**0.8**	80.42	28.66
	MF	93	-	3	3	0	0	1	20354.9	21.7	**80.95**	**28.86**
	DC$_{neg}$	93	87	-	0	0	0	0	2476.2	39.1	77.77	14.93
	SG-LP$_\ell$	93	87	1	-	0	0	0	2474.1	414.7	77.77	14.92
	PROX-LP	94	90	24	24	-	4	9	1475.6	81.0	78.04	15.79
	PROX-LP$_\ell$	94	90	24	24	5	-	9	**1458.9**	82.7	78.04	15.79
	PROX-LP$_{acc}$	94	89	28	27	18	18	-	1623.7	83.9	77.86	15.18

Table 5.3: *Results on the MSRC and Pascal datasets with the parameters tuned for MF (see Section 5.6.3.2). We show: the percentage of images where the row method strictly outperforms the column one on the final integral energy, the average integral energy over the test set, the average run time, the segmentation accuracy and the intersection over union score. Note that all versions of our algorithm obtain much lower energies than the baselines. However, as expected, lower energy does not correspond to better segmentation accuracy, mainly due to the less accurate energy parameters. Furthermore, the accelerated versions of our algorithm are similar in run time and obtain similar energies compared to PROX-LP.*

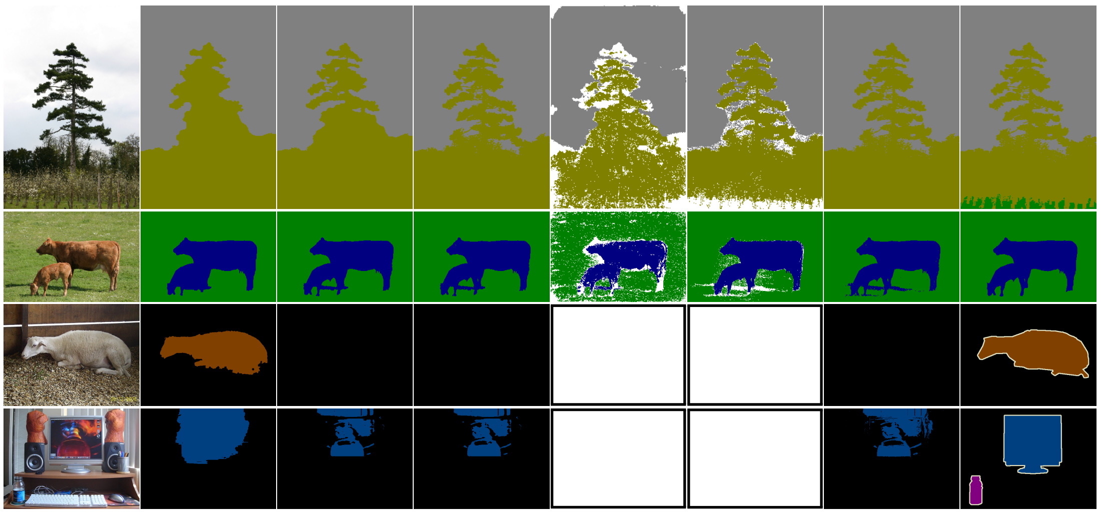

(a) Image
(b) MF
(c) DC_{neg}
(d) SG-LP_{ℓ}
(e) Uncert. (DC_{neg})
(f) Uncert. (ours)
(g) $\text{PROX-LP}_{\text{acc}}$
(h) Ground truth

Figure 5.7: *Results with the parameters tuned for MF for two images in (**top two rows**) MSRC and (**bottom two rows**) Pascal (see Section 5.6.3.2). The uncertain pixels identified by DC_{neg} and PROX-LP_{acc} are marked in white. Note that, in MSRC, PROX-LP_{acc} obtain visually good segmentations similar to MF (or better). In Pascal, the segmentation results are poor except for MF, even though we obtain much lower energies. We argue that, in this case, the energy parameters do not model the segmentation problem accurately.*

5.6.3.3 Summary

We have evaluated all the algorithms using two different parameter settings. Therefore, we summarize the best segmentation accuracy obtained by each algorithm and the corresponding parameter setting in Table 5.4. Note that, on MSRC, the best parameter setting for DC_{neg} corresponds to the parameters tuned for MF. This is a strange result but can be explained by the fact that, as mentioned in the main paper, cross-validation was performed using the less accurate ground truth provided with the original dataset, but evaluation using the accurate ground truth annotations provided by [Krähenbühl Philipp, 2011].

Furthermore, in contrast to MSRC, the segmentation results of our algorithm on the Pascal dataset are not the state-of-the-art, even with the parameters tuned for DC_{neg}. This may be explained by the fact, that due to the limited cross-validation, the energy parameters obtained for the Pascal dataset are not accurate. Therefore, even though our algorithm obtained lower energies, that was not reflected in the segmentation accuracy.

Algorithm	MSRC			Pascal		
	Parameters	Avg. T (s)	Acc.	Parameters	Avg. T (s)	Acc.
MF5	MF	**0.2**	81.14	MF	**0.8**	80.42
MF	MF	13.0	83.86	MF	21.7	**80.95**
DC_{neg}	MF	2.8	83.50	DC_{neg}	3.7	80.43
SG-LP_{ℓ}	MF	37.3	83.51	DC_{neg}	84.4	80.49
PROX-LP	DC_{neg}	23.5	83.99	DC_{neg}	106.7	80.63
PROX-LP_{ℓ}	DC_{neg}	6.3	83.94	DC_{neg}	22.1	80.65
$\text{PROX-LP}_{\text{acc}}$	DC_{neg}	3.7	**84.16**	DC_{neg}	14.7	80.58

Table 5.4: *Best segmentation results of each algorithm with their respective parameters, the average time on the test set and the segmentation accuracy (see Section 5.6.3.3). In MSRC, the best segmentation accuracy is obtained by PROX-LP$_{acc}$ and in Pascal it is by MF. Note that, on MSRC, the best parameter setting for DC$_{neg}$ corresponds to the parameters tuned for MF. This is due to the fact that cross-validation was performed on the less accurate ground truth but evaluation on the accurate ground truth annotations provided by [Krähenbühl Philipp, 2011]. Furthermore, the low segmentation performance of our algorithm on the Pascal dataset is may be due to less accurate energy parameters resulted from limited cross-validation.*

Our observation that a lower energy does not necessarily result in improved segmentation is an important one (similar behaviour was observed in [Desmaison et al., 2016a; Wang et al., 2015]). Indeed, this lack of correlation between the dense CRF energy and the segmentation accuracy highlights the importance of performing a more thorough analysis of the dense CRF model. Developing methods that fix this discrepancy is an interesting direction of future research, which would be aided by our LP relaxation solver. For example, an end-to-end training regime that utilizes our LP relaxation could provide a significant boost in segmentation accuracy.

5.6.4 PROX-LP Analysis

In this section we empirically analyze the properties of our algorithm by providing statistics of more images and also analyze the effect of the proximal regularization constant (η in Eq. (5.19)).

5.6.4.1 Statistics of More Images

To verify consistent behaviour of our algorithm across the whole dataset, we plot the assignment energy as a function of time for more images on MSRC dataset. See Fig 5.8. For this experiment, the parameters tuned for DC_{neg} were used. In summary, PROX-LP clearly outperforms SG-LP_{ℓ} by obtaining much lower energies in fewer iterations and the accelerated versions obtain roughly the same energy as PROX-LP but significantly faster. In short, the same behaviour as in Figure 5.4 is observed.

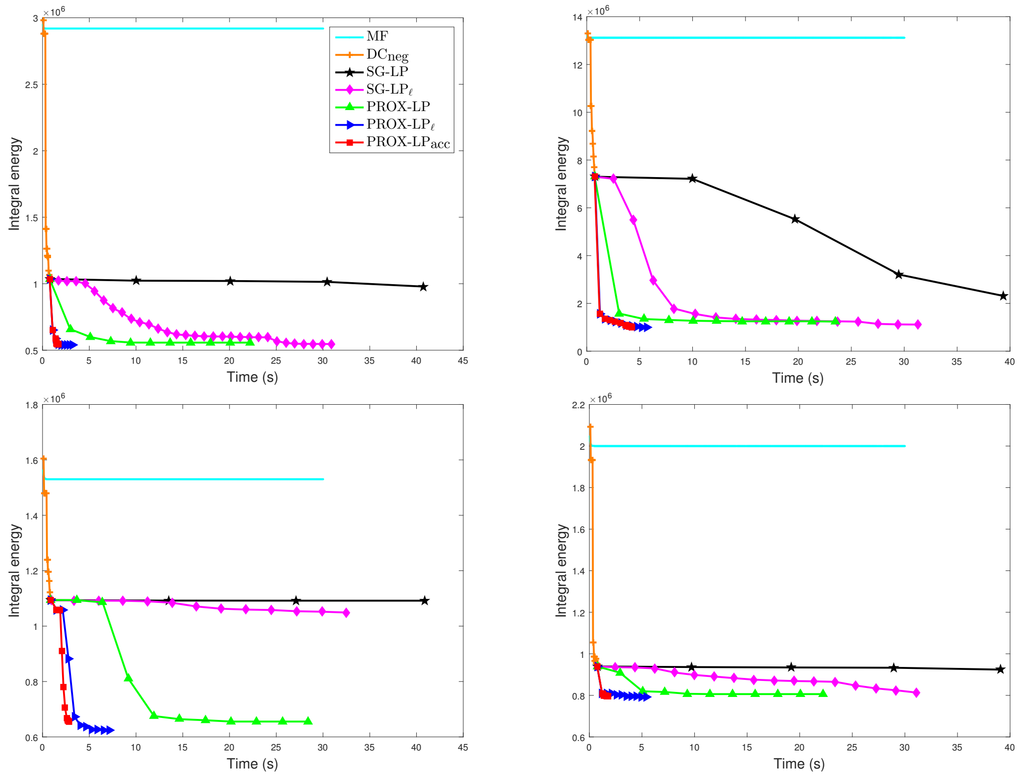


Figure 5.8: *Assignment energy as a function of time with the parameters tuned for DC$_{neg}$ for some images in (cow, building, person and car images shown in Figure 5.5) MSRC (see Section 5.6.4.1). Note that PROX-LP clearly outperforms SG-LP$_\ell$ by obtaining much lower energies in fewer iterations. Furthermore, the accelerated versions of our algorithm obtain roughly the same energy as PROX-LP but significantly faster. In short, the same behaviour as in Figure 5.4 is observed.*

5.6.4.2 Effect of the Proximal Regularization Constant

We plot the assignment energy as a function of time for an image in MSRC (the same image used to generate Figure 5.4) by varying the proximal regularization constant η. Here, we used the parameters tuned for DC$_{\text{neg}}$. The plot is shown in Figure 5.9. In summary, for a wide range of η, PROX-LP obtains similar energies with approximately the same run time.

5.6.5 Modified Filtering Method

We then compare our modified filtering method, described in Section 5.4, with the divide-and-conquer strategy of [Desmaison et al., 2016a]. To this end, we evaluated both algorithms on one of the Pascal VOC test images (the sheep image in Figure 5.5), but varying the image size, the number of labels and the Gaussian kernel standard deviation. Note that, to generate a plot for one variable, the other variables are fixed

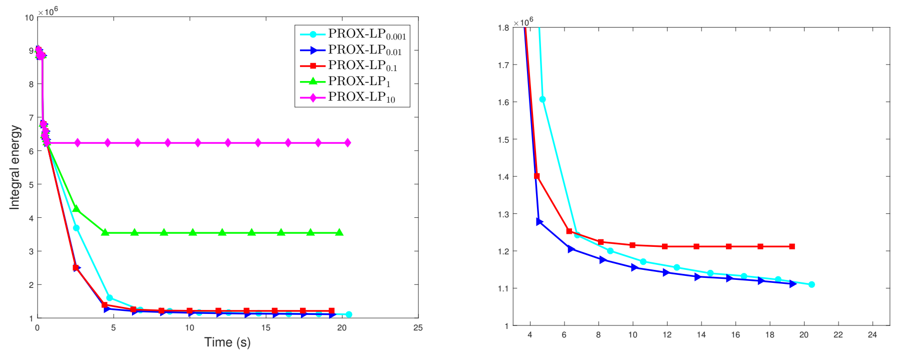


Figure 5.9: *Assignment energy as a function of time for an image in MSRC, for different values of η (see Section 5.6.4.2). The zoomed plot is shown on the right. Note that, for $\eta = 0.1, 0.01, 0.001$, PROX-LP obtains similar energies in approximately the same run time.*

to their respective standard values. The standard value for the number of pixels is 187500, for the number of labels 21, and for the standard deviation 1. For this experiment, the conditional gradients were computed from a random primal solution $\tilde{\mathbf{y}}^t$. In top two rows of Figure 5.10, we show the speedup of our modified filtering approach over the one of [Desmaison et al., 2016a] as a function of the number of pixels, labels and filter standard deviation. The timings were averaged over 10 runs, and we observed only negligible timing variations between the different runs.

In summary, our modified filtering method is $10 - 65$ times faster than the state-of-the-art algorithm of [Desmaison et al., 2016a]. Furthermore, note that all versions of our algorithm operate in the region where the speedup is around $45 - 65$.

Similar plots for an MSRC image (the tree image in Figure 5.5) are shown in bottom two rows of Figure 5.10. In this case, speedup is around $15 - 32$, with around $23 - 32$ in the operating region of all versions of our algorithm.

5.7 Discussion

We have introduced the first LP minimization algorithm for dense CRFs with Gaussian pairwise potentials whose iterations are linear in the number of pixels and labels. Thanks to the efficiency of our algorithm and to the tightness of the LP relaxation, our approach yields much lower energy values than state-of-the-art dense CRF inference methods. Furthermore, our experiments have demonstrated that, with the right set of energy parameters, highly accurate segmentation results can be obtained with our algorithm. The speed and effective energy minimization of our algorithm make it a perfect candidate to be incorporated into an end-to-end learning framework, such as [Zheng et al., 2015]. This, we believe, will be key to further improving the accuracy of deep semantic segmentation architectures.

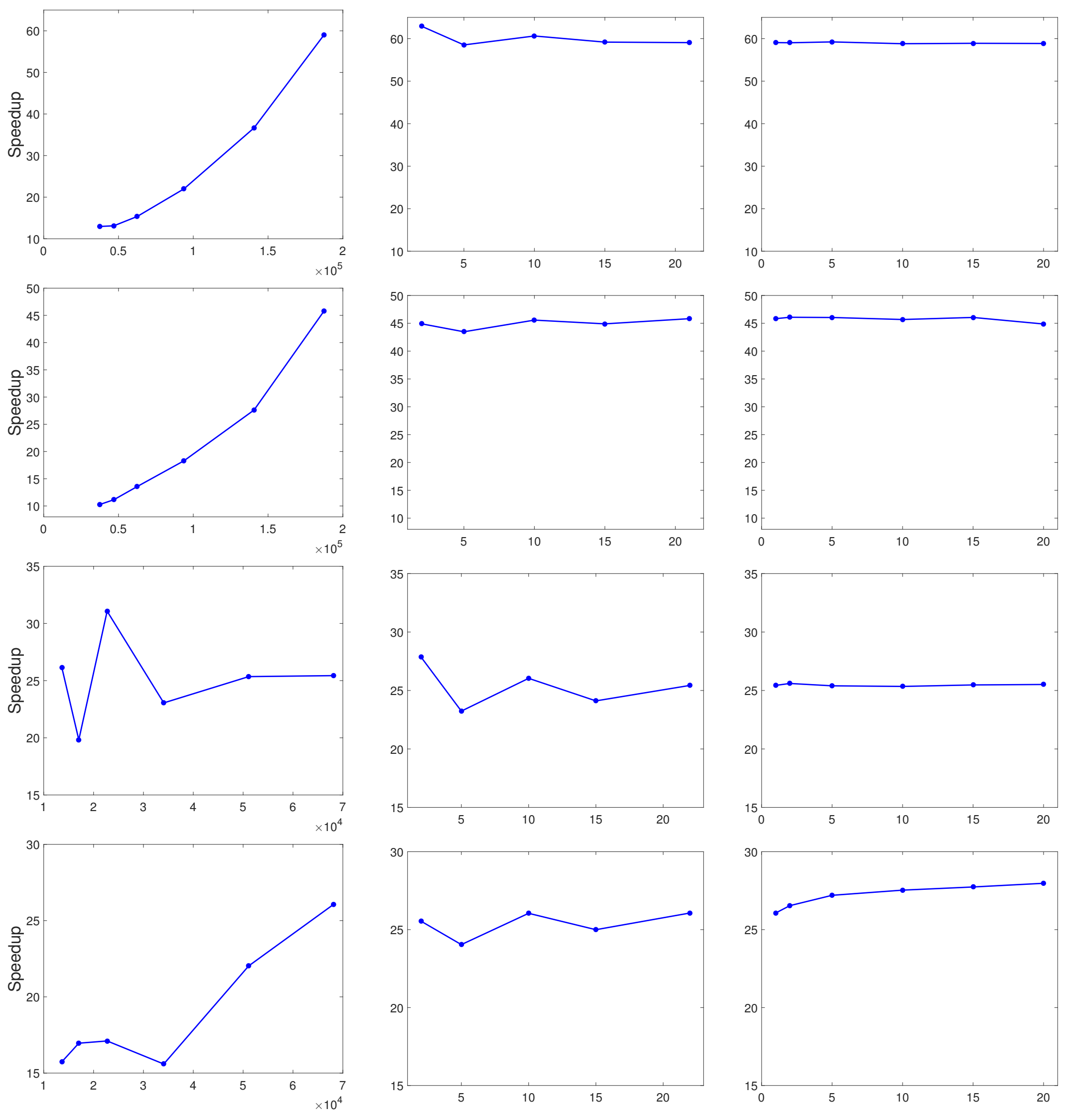
Note that, as discussed in the chapter, lower energy does not necessarily result in improved segmentation. This lack of correlation between the dense CRF energy and the segmentation accuracy is due to inaccurate energy parameters. Developing methods that fix this discrepancy is an interesting direction for future research, which would be aided by our LP relaxation solver. For example, as mentioned above, an end-to-end training regime that utilizes our LP relaxation could provide a significant boost in segmentation accuracy.

Furthermore, with simple heuristics, our PROX-LP$_{\text{acc}}$ algorithm converges significantly faster than the standard PROX-LP method without compromising on the solution quality. It would be interesting to think about additional heuristics that can further improve the convergence rates.

Currently, our algorithm can only handle fully connected CRFs where the label compatibility function is the Potts model. As mentioned in the chapter, our algorithm can be extended to general label compatibility functions using a hierarchical Potts model [Kumar and Koller, 2009]. When extending to more general label compatibility functions, the main challenge is to maintain linear time complexity per iteration. This would be an important extension of our LP minimization algorithm.

On the other hand, our modification to the permutohedral lattice based filtering method has wide applicability beyond what is discussed in this chapter. For instance, similarly to [Zhang et al., 2015], marginal inference on the dense CRF model can be performed and whose iterations can be made *linear* using our modified filtering method. In addition to this, we intend to explore possible applications of our modified filtering method in future.

In the next chapter, we summarize the contributions of the thesis and discuss possible extensions.



(a) Number of pixels (b) Number of labels (c) Filter standard deviation

Figure 5.10: *Speedup of our modified filtering method over the divide-and-conquer strategy of [Desmaison et al., 2016a] on an image in (**top two rows**) Pascal and (**bottom two rows**) MSRC (see Section 5.6.5). **First** and **third** rows correspond to spatial kernel ($d = 2$) and **second** and **fourth** rows correspond to bilateral kernel ($d = 5$). Note that our speedup grows with the number of pixels (except when $d = 2$ on MSRC, for which it is approximately constant) and is approximately constant with respect to the number of labels and filter standard deviation.*

Chapter 6

Conclusion

So far, we have described an exact max-flow algorithm and two approximate algorithms for MRF optimization with applications to computer vision. In this chapter, we summarize the contributions presented in this thesis and discuss possible future extensions.

6.1 Contributions

In this thesis, we have introduced three new algorithms for MRF optimization targeting computer vision applications, that are either more efficient (in terms of running time and/or memory usage) or more effective (in terms of solution quality), than the state-of-the-art methods. We summarize the contributions in each chapter below.

Chapter 3. In this chapter, we have introduced a variant of the max-flow algorithm (called MEMF) that can minimize multi-label submodular MRF energies optimally, while requiring much less storage. Specifically, our max-flow algorithm stores only two ℓ-dimensional vectors, called *exit-flows* (where ℓ is the number of labels) per variable pair instead of the $2\,\ell^2$ edge capacities of the standard max-flow algorithm. Consequently, we reduced the memory requirement of the max-flow algorithm by $\mathcal{O}(\ell)$ for Ishikawa type graphs, while not compromising on optimality. As a result, our approach lets us optimally solve much larger problems on a standard computer.

Furthermore, we proved the worst case time complexity of our algorithm by following the proof of the standard Edmonds-Karp algorithm. While requiring $\mathcal{O}(\ell)$ less memory, the polynomial time version of MEMF was shown to be $\mathcal{O}(\ell)$ times slower than Edmonds-Karp in the worst case. Later, we have shown that our MEMF algorithm actually solves the *optimal message passing* problem (*i.e.*, the dual of the LP relaxation of the discrete MRF) when the MRF is multi-label submodular. This allowed us to draw the equivalence between the set of exit-flows and the set of messages.

Finally, our experiments have shown that our algorithm is an order of magnitude faster than state-of-the-art methods. We, therefore, believe that our algorithm constitutes the method of choice to minimize Ishikawa type graphs when the complete graph cannot be stored in memory.

Chapter 4. In this chapter, we have introduced a move-making algorithm (called IRGC) for multi-label MRFs with robust non-convex priors. In particular, our algorithm iteratively approximates the original MRF energy with an appropriately weighted surrogate energy that is easier to minimize. We have shown that, under suitable conditions on the non-convex priors, and as long as the weighted surrogate energy can be decreased, our approach guarantees that the true energy decreases at each iteration. To this end, we considered the scenario where the weighted surrogate energy is multi-label submodular and showed that our algorithm then lets us handle of a large variety of non-convex priors. In addition to the basic version, we have presented a hybrid optimization strategy that combines IRGC with α-expansion.

Furthermore, we have shown that our IRGC algorithm is a special instance of the well known majorization-minimization framework. However, to the best of our knowledge, this is the first time such a technique is transposed to the MRF optimization scenario.

Finally, our experiments have shown that, while the basic algorithm sometimes gets trapped in local minima, our hybrid version consistently outperforms (or performs virtually as well as) state-of-the-art MRF energy minimization techniques. We, therefore, believe our algorithm constitutes the first move-making algorithm to effectively tackle MRFs with robust non-convex priors.

Chapter 5. In this chapter, we have introduced an efficient LP minimization algorithm (called PROX-LP) for dense CRFs with Gaussian pairwise potentials. In particular, we have developed a proximal minimization framework, where the dual of each proximal problem is optimized via block-coordinate descent. We have shown that each block of variables can be optimized in a time *linear* in the number of pixel and labels. Specifically, for one block, the problem decomposes into significantly smaller subproblems, each of which is defined over a single pixel. For the other block, the problem is optimized via conditional gradient descent. This had two advantages: 1) the *conditional gradient* can be computed in a time *linear* in the number of pixels and labels; and 2) the optimal step size can be computed analytically. To the best of our knowledge, this constitutes the first LP minimization algorithm for dense CRFs that has *linear* time iterations. Thanks to the efficiency of our algorithm and to the tightness of the LP relaxation, our approach yielded much lower energy values than state-of-the-art dense CRF inference methods.

Furthermore, to efficiently compute the conditional gradient required by our algorithm, we needed to perform approximate Gaussian filtering with ordering constraints. To accomplish this in linear time, we modified the permutohedral lattice based filtering method to handle ordering constraints. This modified filtering method has wide applicability beyond what is discussed in this thesis.

Finally, our experiments have demonstrated that, with the right set of energy parameters, highly accurate segmentation results can be obtained with our algorithm. The speed and effective energy minimization of our algorithm make it a perfect candidate to be incorporated into an end-to-end learning framework. This, we believe, will be key to further improving the accuracy of deep semantic segmentation architectures.

6.2 Future Directions

We now discuss some possible future directions which can extend the practical and theoretical implications of our work presented in this thesis.

MEMF

- Even though our MEMF algorithm converges in approximately the same time as that of the BK method, it is $4-7$ times slower than the EIBFS algorithm. Note that the EIBFS algorithm is an improved version of the BK method which combines push-relabel techniques with the augmenting path based BK algorithm (see [Goldberg et al., 2015]). Due to the similarity of MEMF to the BK method, we believe that MEMF can also be accelerated in a similar fashion. This would not only make MEMF significantly faster but also improve its worst case time complexity.
- The equivalence of MEMF with min-sum message passing indicates interesting future directions. For instance, MEMF can be used to obtain min-marginals similarly to [Kohli and Torr, 2008]. Furthermore, MEMF can be combined with a message-passing algorithm, such as TRWS to obtain a hybrid algorithm. Potentially, such an algorithm can be designed in a way that, it has desirable characteristics of both algorithms, *i.e.*, good at approximating the true energy (similar to TRWS) while being fast to converge (similar to max-flow).

IRGC

- Since the IRGC algorithm has the flexibility in subroutine algorithm choices, it would be interesting to study useful such choices.
  - For the hybrid version, fusion moves or any other move-making algorithm can be used instead of α-expansion. This could potentially improve the solution quality of the hybrid version of our algorithm.
  - Instead of the Ishikawa algorithm, the surrogate energy can be optimized approximately using the convex expansion technique of [Carr and Hartley, 2009] (see Section 2.3.3.2 for an overview). This would improve the running time and memory usage while compromising on the solution quality. This would be useful in resource-scarce environments, such as embedded systems or mobile devices.
  - One may replace the Ishikawa algorithm with the primal-dual approach of [Pock et al., 2008; Mollenhoff et al., 2016]. This would further extend the applicability of IRGC to continuous label spaces and enables the use of Graphics Processing Unit (GPU) to parallelize the algorithm.

- As discussed in Chapter 4, IRGC really is a special case of an iteratively reweighted approach to MRFs and even continuous energy minimization. Therefore, we believe, our approach can be extended to other types of MRF problems and it would be interesting to study useful such extensions.

PROX-LP

- As discussed in the chapter, lower energy does not necessarily result in improved segmentation. This lack of correlation between the dense CRF energy and the segmentation accuracy is due to inaccurate energy parameters. Developing methods that fix this discrepancy is an interesting direction for future research, which would be aided by our LP relaxation solver. For example, as mentioned earlier, an end-to-end training regime that utilizes our LP relaxation could provide a significant boost in segmentation accuracy.
- By using simple heuristics, our PROX-LP$_{\text{acc}}$ algorithm converges significantly faster than the standard PROX-LP method without compromising on the solution quality. It would be interesting to think about additional heuristics that can further improve the convergence rates.
- Currently, our algorithm can only handle fully connected CRFs where the label compatibility function is the Potts model. As mentioned in Chapter 5, our algorithm can be extended to general label compatibility functions using a hierarchical Potts model [Kumar and Koller, 2009]. When extending to more general label compatibility functions, the main challenge is to maintain linear time complexity per iteration. This would be an important extension of our LP minimization algorithm.
- As mentioned earlier, our modified filtering method has wide applicability beyond what is shown in this thesis. For instance, similarly to [Zhang et al., 2015], marginal inference on the dense CRF model can be performed and whose iterations can be made *linear* using our modified filtering method. But its application is not limited to this and it would be interesting to explore possible applications of our modified filtering method.

Bibliography

Adams, A.; Baek, J.; and Davis, M. A., Fast high-dimensional filtering using the permutohedral lattice. *Computer Graphics Forum*, (2010). (cited on pages 7, 105, 106, 111, 123, 124, 125, and 128)

Adams, A. B., 2011. *High-dimensional gaussian filtering for computational photography*. Stanford University. (cited on page 123)

Aftab, K. and Hartley, R., Convergence of iteratively re-weighted least squares to robust M-estimators. *WACV*, (2015). (cited on pages 83 and 93)

Aftab, K.; Hartley, R.; and Trumpf, J., Generalized Weiszfeld algorithms for L_q optimization. *PAMI*, (2015). (cited on page 93)

Ajanthan, T.; Desmaison, A.; Bunel, R.; Salzmann, M.; Torr, P. H. S.; and Kumar, M. P., Efficient linear programming for dense CRFs. *CVPR*, (2017). (cited on pages 7 and 105)

Ajanthan, T.; Desmaison, A.; Bunel, R.; Salzmann, M.; Torr, P. H. S.; and Kumar, M. P., Efficient linear programming for dense CRFs. *arXiv preprint arXiv:1611.09718*, (2017). (cited on pages 7 and 105)

Ajanthan, T.; Hartley, R.; and Salzmann, M., Memory efficient max-flow for multi-label submodular MRFs. *CVPR*, (2016). (cited on pages 7, 47, and 61)

Ajanthan, T.; Hartley, R.; and Salzmann, M., Memory efficient max-flow for multi-label submodular MRFs. *arXiv preprint arXiv:1702.05888*, (2017). (cited on pages 7 and 47)

Ajanthan, T.; Hartley, R.; Salzmann, M.; and Li, H., Iteratively reweighted graph cut for multi-label MRFs with non-convex priors. *CVPR*, (2015). (cited on pages 7 and 81)

Andres, B.; Beier, T.; and Kappes, J. H., OpenGM: A C++ library for discrete graphical models. *arXiv preprint arXiv:1206.0111*, (2012). (cited on page 73)

Bach, F., Duality between subgradient and conditional gradient methods. *SIAM Journal on Optimization*, (2015). (cited on page 121)

Bertsekas, D. P., 1999. *Nonlinear programming*. Athena Scientific Belmont. (cited on pages 13 and 42)

Besag, J., Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society*, (1974). (cited on page 10)

Birchfield, S. and Tomasi, C., A pixel dissimilarity measure that is insensitive to image sampling. *PAMI*, (1998). (cited on pages 73 and 95)

Boros, E. and Hammer, P. L., Pseudo-boolean optimization. *Discrete applied mathematics*, (2002). (cited on pages 15, 18, 20, 32, 52, 93, and 94)

Boyd, S.; Parikh, N.; Chu, E.; Peleato, B.; and Eckstein, J., Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, (2011). (cited on page 109)

Boyd, S. and Vandenberghe, L., 2009. *Convex optimization*. Cambridge university press. (cited on pages 13, 22, 33, 34, 36, and 115)

Boykov, Y. and Kolmogorov, V., An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *PAMI*, (2004). (cited on pages 14, 20, 48, 50, 60, 67, 72, 73, 93, and 94)

Boykov, Y.; Veksler, O.; and Zabih, R., Fast approximate energy minimization via graph cuts. *PAMI*, (2001). (cited on pages 5, 13, 14, 27, 31, 81, 82, 92, and 94)

Boykov, Y. Y. and Jolly, M.-P., Interactive graph cuts for optimal boundary & region segmentation of objects in ND images. *ICCV*, (2001). (cited on page 14)

Carr, P. and Hartley, R., Solving multilabel graph cut problems with multilabel swap. *DICTA*, (2009). (cited on pages 30, 100, and 143)

Chandran, B. G. and Hochbaum, D. S., A computational study of the pseudoflow and push-relabel algorithms for the maximum flow problem. *Operations research*, (2009). (cited on pages 20 and 72)

Chekuri, C.; Khanna, S.; Naor, J.; and Zosin, L., A linear programming formulation and approximation algorithms for the metric labeling problem. *SIAM Journal on Discrete Mathematics*, (2004). (cited on pages 5, 13, 108, and 128)

Chen, L.-C.; Papandreou, G.; Kokkinos, I.; Murphy, K.; and Yuille, A. L., Semantic image segmentation with deep convolutional nets and fully connected CRFs. *ICLR*, (2014). (cited on page 128)

Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; and Stein, C., 2001. *Introduction to algorithms*. MIT press. (cited on page 57)

Dahlhaus, E.; Johnson, D. S.; Papadimitriou, C. H.; Seymour, P. D.; and Yannakakis, M., The complexity of multiway cuts. *ACM symposium on Theory of computing*, (1992). (cited on page 13)

Dantzig, G., 2016. *Linear programming and extensions*. Princeton university press. (cited on page 34)

Delong, A. and Boykov, Y., A scalable graph-cut algorithm for ND grids. *CVPR*, (2008). (cited on page 72)

Desmaison, A.; Bunel, R.; Kohli, P.; Torr, P. H.; and Kumar, M. P., Efficient continuous relaxations for dense CRF. *ECCV*, (2016). (cited on pages 5, 105, 106, 107, 109, 121, 123, 124, 125, 128, 129, 130, 136, 137, 138, and 140)

Desmaison, A.; Bunel, R.; Kohli, P.; Torr, P. H.; and Kumar, M. P., Efficient continuous relaxations for dense CRF - Supplementary material. *ECCV*, (2016). (cited on pages 106, 116, and 123)

Edmonds, J. and Karp, R. M., Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM*, (1972). (cited on pages 20 and 54)

Everingham, M.; Van Gool, L.; Williams, C. K.; Winn, J.; and Zisserman, A., The pascal visual object classes (VOC) challenge. *IJCV*, (2010). (cited on pages 3, 106, and 130)

Felzenszwalb, P. F. and Huttenlocher, D. P., Efficient belief propagation for early vision. *IJCV*, (2006). (cited on page 93)

Ford, L. and Fulkerson, D. R., 1962. *Flows in networks*. Princeton University Press. (cited on pages 14, 15, 17, 18, 20, 47, 49, and 72)

Frank, M. and Wolfe, P., An algorithm for quadratic programming. *Naval research logistics quarterly*, (1956). (cited on pages 106, 110, 111, 112, and 128)

Fujishige, S., 2005. *Submodular functions and optimization*. Elsevier. (cited on page 15)

Geiger, A.; Lenz, P.; Stiller, C.; and Urtasun, R., Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research*, (2013). (cited on page 73)

Globerson, A. and Jaakkola, T. S., Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations. *NIPS*, (2008). (cited on page 44)

Goemans, M. X. and Williamson, D. P., Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, (1995). (cited on page 13)

Goldberg, A. V.; Hed, S.; Kaplan, H.; Kohli, P.; Tarjan, R. E.; and Werneck, R. F., Faster and more dynamic maximum flow by incremental breadth-first search. Springer. (cited on pages 20, 73, 78, 94, and 143)

Goldberg, A. V.; Hed, S.; Kaplan, H.; Tarjan, R. E.; and Werneck, R. F., Maximum flows by incremental breadth-first search. Springer. (cited on page 72)

Goldberg, A. V. and Tarjan, R. E., A new approach to the maximum-flow problem. *Journal of the ACM*, (1988). (cited on pages 20 and 72)

Grimmett, G. R., A theorem about random fields. *Bulletin of the London Mathematical Society*, (1973). (cited on page 10)

Hartley, R. and Zisserman, A., 2003. *Multiple view geometry in computer vision*. Cambridge University Press. (cited on pages 2, 90, and 91)

Hille, E., 2005. *Analytic function theory*. American Mathematical Society. (cited on page 123)

Hochbaum, D. S., An efficient algorithm for image segmentation, Markov random fields and related problems. *Journal of the ACM*, (2001). (cited on page 72)

Huber, P. J., Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, (1964). (cited on page 26)

Hunter, D. R. and Lange, K., A tutorial on MM algorithms. *The American Statistician*, (2004). (cited on page 85)

Ishikawa, H., Exact optimization for Markov random fields with convex priors. *PAMI*, (2003). (cited on pages 5, 14, 21, 22, 26, 47, 48, 72, 81, 82, and 87)

Jamriška, O.; Sỳkora, D.; and Hornung, A., Cache-efficient graph cuts on structured grids. *CVPR*, (2012). (cited on page 72)

Jezierska, A.; Talbot, H.; Veksler, O.; and Wesierski, D., A fast solver for truncated-convex priors: quantized-convex split moves. *Energy Minimization Methods in Computer Vision and Pattern Recognition*, (2011). (cited on page 92)

Kappes, J. H.; Andres, B.; Hamprecht, F. A.; Schnörr, C.; Nowozin, S.; Batra, D.; Kim, S.; Kausler, B. X.; Kröger, T.; Lellmann, J.; Komodakis, N.; Savchynskyy, B.; and Rother, C., A comparative study of modern inference techniques for structured discrete energy minimization problems. *IJCV*, (2015). (cited on pages 4, 13, 72, 82, 93, and 98)

Karmarkar, N., A new polynomial-time algorithm for linear programming. *ACM symposium on Theory of computing*, (1984). (cited on page 36)

Kelley, J. L., 1975. *General topology*. Springer. (cited on page 110)

Kleinberg, J. and Tardos, E., Approximation algorithms for classification problems with pairwise relationships: Metric labeling and Markov random fields. *Journal of the ACM*, (2002). (cited on pages 5, 13, 38, 105, 108, and 128)

Kohli, P. and Torr, P. H., Efficiently solving dynamic Markov random fields using graph cuts. *ICCV*, (2005). (cited on page 48)

Kohli, P. and Torr, P. H., Measuring uncertainty in graph cut solutions. *CVIU*, (2008). (cited on pages 78 and 143)

Koller, D. and Friedman, N., 2009. *Probabilistic graphical models: Principles and techniques*. MIT press. (cited on page 1)

Kolmogorov, V., Convergent tree-reweighted message passing for energy minimization. *PAMI*, (2006). (cited on pages 5, 39, 44, 48, 72, 73, 78, 82, 93, 94, and 128)

Kolmogorov, V. and Wainwright, M., On the optimality of tree-reweighted max-product message-passing. *UAI*, (2005). (cited on pages 68 and 72)

Kolmogorov, V. and Zabin, R., What energy functions can be minimized via graph cuts? *PAMI*, (2004). (cited on pages 4 and 14)

Komodakis, N. and Paragios, N., Beyond loose LP-relaxations: Optimizing MRFs by repairing cycles. *ECCV*, (2008). (cited on page 44)

Komodakis, N.; Paragios, N.; and Tziritas, G., MRF energy minimization and beyond via dual decomposition. *PAMI*, (2011). (cited on pages 14, 42, 44, 48, 72, 73, 93, and 128)

Komodakis, N.; Tziritas, G.; and Paragios, N., Performance vs computational efficiency for optimizing single and dynamic MRFs: Setting the state of the art with primal-dual strategies. *CVIU*, (2008). (cited on page 31)

Krähenbühl Philipp, V., Koltun, Efficient inference in fully connected CRFs with gaussian edge potentials. *NIPS*, (2011). (cited on pages 4, 5, 105, 106, 107, 108, 123, 124, 128, 130, 135, and 136)

Krishnan, R. G.; Lacoste-Julien, S.; and Sontag, D., Barrier Frank-Wolfe for marginal inference. *NIPS*, (2015). (cited on page 128)

Kullback, S. and Leibler, R. A., On information and sufficiency. *The annals of mathematical statistics*, (1951). (cited on page 44)

Kumar, M. P. and Koller, D., MAP estimation of semi-metric MRFs via hierarchical graph cuts. *UAI*, (2009). (cited on pages 128, 139, and 144)

Kumar, M. P.; Kolmogorov, V.; and Torr, P. H., An analysis of convex relaxations for MAP estimation of discrete MRFs. *JMLR*, (2009). (cited on pages 5, 13, 44, and 105)

Lacoste-Julien, S.; Jaggi, M.; Schmidt, M.; and Pletscher, P., Block-coordinate Frank-Wolfe optimization for structural SVMs. *ICML*, (2012). (cited on pages 106, 110, 111, and 128)

Lafferty, J.; McCallum, A.; Pereira, F.; et al., Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *ICML*, (2001). (cited on page 12)

Lempitsky, V.; Rother, C.; Roth, S.; and Blake, A., Fusion moves for Markov random field optimization. *PAMI*, (2010). (cited on pages 31, 92, and 100)

Manokaran, R.; Naor, J.; Raghavendra, P.; and Schwartz, R., SDP gaps and UGC hardness for multiway cut, 0-extension and metric labeling. *STOC*, (2008). (cited on page 108)

Meshi, O.; Mahdavi, M.; and Schwing, A., Smooth and strong: MAP inference with linear convergence. *NIPS*, (2015). (cited on page 128)

Mollenhoff, T.; Laude, E.; Moeller, M.; Lellmann, J.; and Cremers, D., Sublabel-accurate relaxation of nonconvex energies. *CVPR*, (2016). (cited on pages 103 and 143)

Muramatsu, M. and Suzuki, T., A new second-order cone programming relaxation for max-cut problems. *Journal of the Operations Research Society of Japan*, (2003). (cited on page 44)

Nemhauser, G. L. and Wolsey, L. A., 1988. *Integer programming and combinatorial optimization*. Springer. (cited on page 1)

Ochs, P.; Dosovitskiy, A.; Brox, T.; and Pock, T., An iterated l_1 algorithm for non-smooth non-convex optimization in computer vision. *CVPR*, (2013). (cited on page 93)

Olsson, C.; Eriksson, A. P.; and Kahl, F., Solving large scale binary quadratic problems: Spectral methods vs semidefinite programming. *CVPR*, (2007). (cited on page 44)

Orlin, J. B., Max-flows in $O(nm)$ time, or better. *ACM symposium on Theory of Computing*, (2013). (cited on page 60)

Osokin, A.; Alayrac, J.-B.; Lukasewitz, I.; Dokania, P. K.; and Lacoste-Julien, S., Minding the gaps for block Frank-Wolfe optimization of structured SVMs. *ICML*, (2016). (cited on page 128)

Parikh, N. and Boyd, S. P., Proximal algorithms. *Foundations and Trends in Optimization*, (2014). (cited on pages 106 and 109)

Pearl, J., 1988. *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. Morgan Kaufmann. (cited on pages 14, 32, 38, and 40)

Pock, T.; Schoenemann, T.; Graber, G.; Bischof, H.; and Cremers, D., A convex formulation of continuous multi-label problems. *ECCV*, (2008). (cited on pages 103 and 143)

Ravikumar, P.; Agarwal, A.; and Wainwright, M. J., Message-passing for graph-structured linear programs: Proximal projections, convergence and rounding schemes. *ICML*, (2008). (cited on pages 38 and 128)

Ravikumar, P. and Lafferty, J., Quadratic programming relaxations for metric labeling and Markov random field map estimation. *ICML*, (2006). (cited on pages 5 and 44)

Rother, C.; Kolmogorov, V.; Lempitsky, V.; and Szummer, M., Optimizing binary MRFs via extended roof duality. *CVPR*, (2007). (cited on page 94)

Rother, C.; Kumar, S.; Kolmogorov, V.; and Blake, A., Digital tapestry [automatic image synthesis]. *CVPR*, (2005). (cited on page 92)

Savchynskyy, B.; Schmidt, S.; Kappes, J. H.; and Schnörr, C., Efficient MRF energy minimization via adaptive diminishing smoothing. *UAI*, (2012). (cited on pages 48, 72, and 73)

Scharstein, D. and Szeliski, R., A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *IJCV*, (2002). (cited on pages 2, 73, 76, and 94)

Scharstein, D. and Szeliski, R., High-accuracy stereo depth maps using structured light. *CVPR*, (2003). (cited on pages 73, 76, and 95)

Schlesinger, D. and Flach, B., 2006. *Transforming an arbitrary minsum problem into a binary one*. TU, Fak. Informatik. (cited on pages 14, 21, and 47)

Schwing, A. G. and Urtasun, R., Fully connected deep structured networks. *arXiv preprint arXiv:1503.02351*, (2015). (cited on page 128)

Shah, N.; Kolmogorov, V.; and Lampert, C. H., A multi-plane block-coordinate Frank-Wolfe algorithm for training structural SVMs with a costly max-oracle. *CVPR*, (2015). (cited on page 128)

Shekhovtsov, A. and Hlaváč, V., A distributed min-cut/max-flow algorithm combining path augmentation and push-relabel. *IJCV*, (2013). (cited on page 72)

Snoek, J.; Larochelle, H.; and Adams, R. P., Practical bayesian optimization of machine learning algorithms. *NIPS*, (2012). (cited on page 130)

Sontag, D.; Meltzer, T.; Globerson, A.; Jaakkola, T.; and Weiss, Y., Tightening LP relaxations for MAP using message passing. *UAI*, (2008). (cited on page 44)

Strandmark, P. and Kahl, F., Parallel and distributed graph cuts by dual decomposition. *CVPR*, (2010). (cited on pages 48, 72, and 76)

Szeliski, R.; Zabih, R.; Scharstein, D.; Veksler, O.; Kolmogorov, V.; Agarwala, A.; Tappen, M.; and Rother, C., A comparative study of energy minimization methods for Markov random fields with smoothness-based priors. *PAMI*, (2008). (cited on pages 2, 12, 13, 73, 82, 93, 95, 96, 97, and 98)

Tarlow, D.; Givoni, I. E.; Zemel, R. S.; and Frey, B. J., Graph cuts is a max-product algorithm. *UAI*, (2011). (cited on page 67)

Torr, P. H., Solving Markov random fields using semi definite programming. *AISTATS*, (2003). (cited on page 44)

Torr, P. H. and Kumar, M. P., Improved moves for truncated convex models. *NIPS*, (2009). (cited on pages 31, 48, and 92)

Veksler, O., Multi-label moves for MRFs with truncated convex priors. *IJCV*, (2012). (cited on pages 31, 48, 82, 92, 93, and 94)

Verma, T. and Batra, D., Max-flow revisited: An empirical comparison of max-flow algorithms for dense vision problems. *BMVC*, (2012). (cited on page 72)

Vineet, V. and Narayanan, P., CUDA cuts: Fast graph cuts on the GPU. *CVPR Workshops*, (2008). (cited on page 72)

Wainwright, M. J.; Jaakkola, T. S.; and Willsky, A. S., MAP estimation via agreement on trees: Message-passing and linear programming. *Information Theory*, (2005). (cited on pages 14, 35, 37, 42, 43, 44, 72, 93, and 128)

Wainwright, M. J.; Jordan, M. I.; et al., Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, (2008). (cited on pages 13, 32, 37, and 44)

Wang, P.; Shen, C.; and van den Hengel, A., Efficient SDP inference for fully-connected CRFs based on low-rank decomposition. *CVPR*, (2015). (cited on page 136)

Werner, T., A linear programming approach to max-sum problem: A review. *PAMI*, (2007). (cited on pages 5, 37, 39, 42, 68, 72, 108, and 128)

Werner, T., Revisiting the linear programming relaxation approach to Gibbs energy minimization and weighted constraint satisfaction. *PAMI*, (2010). (cited on pages 40, 42, and 44)

Xiao, X. and Chen, D., Multiplicative iteration for nonnegative quadratic programming. *Numerical Linear Algebra with Applications*, (2014). (cited on pages 118 and 119)

Yedidia, J. S.; Freeman, W. T.; and Weiss, Y., Understanding belief propagation and its generalizations. *Exploring artificial intelligence in the new millennium*, (2003). (cited on page 11)

Yedidia, J. S.; Freeman, W. T.; Weiss, Y.; et al., Generalized belief propagation. *NIPS*, (2000). (cited on page 32)

Zhang, J.; Djolonga, J.; and Krause, A., Higher-order inference for multi-class log-supermodular models. *CVPR*, (2015). (cited on pages 139 and 144)

Zheng, S.; Jayasumana, S.; Romera-Paredes, B.; Vineet, V.; Su, Z.; Du, D.; Huang, C.; and Torr, P., Conditional random fields as recurrent neural networks. *ICCV*, (2015). (cited on pages 128, 129, and 138)

Index