

PERSISTENCY ALGORITHMS FOR EFFICIENT INFERENCE IN MARKOV RANDOM FIELDS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

王晨 (Chen Wang)

August 2018

© 2018 Chen Wang
ALL RIGHTS RESERVED

PERSISTENCY ALGORITHMS FOR EFFICIENT INFERENCE IN MARKOV

RANDOM FIELDS

王晨 (Chen Wang), Ph.D.

Cornell University 2018

Markov Random Fields (MRFs) have achieved great success in a variety of computer vision problems, including image segmentation, stereo estimation, optical flow and image denoising, during the past 20 years. Despite the inference problem being NP-hard, a large number of approximation algorithms, e.g., graphcuts, have been studied, although all of these methods are computationally expensive. We observed that most problems in practice contains a large easy part and a small hard part. Therefore, in this thesis, we investigated a few persistency-based approaches which could compute optimal labeling for a large set of variables efficiently and reduce the scale of the problem that the expensive inference algorithms need to solve.

In particular, we will explore two different lines of research. The first direction focuses on generalizing the sufficient local condition to check persistency on a set of variables as opposed to a single variable in previous works, and provides a hierarchical relaxation to trade-off between efficiency and effectiveness. The second direction gives a discriminative view of persistency, which allow us to label more variables optimally with a small cost to label a few wrongly.

This thesis will present a literature study of persistency used for MRF inference, the mathematical formalization of the algorithms and the experimental results for both the first-order and higher-order MRF inference problems.

BIOGRAPHICAL SKETCH

王晨 (Chen Wang) was born and raised in Nanjing, Jiangsu, China.

Chen is currently a PhD student at Cornell University, Ithaca, NY, USA, focusing on combinatorial optimization algorithms with applications in computer vision, supervised by Professor Ramin Zabih. Before starting PhD, he graduated with a Bachelor of Science in Computer Science degree from Fudan University, Shanghai, China, in 2012, supervised by Professor Junping Zhang. Before the undergraduate study, he graduated from Jinling High School, Nanjing, Jiangsu, China, in 2008.

Chen joined Google Research, New York, NY, USA, since July 2016, supervised by Professor Ramin Zabih. Before that, he was a software engineer intern at Google, Mountain View, CA, USA, Spring 2016, supervised by Dr. Breet Allen. He was a software engineer intern at Google, Mountain View, CA, USA, Summer 2015, supervised by Dr. Aleksey Golovinskiy. He was a research intern at Microsoft Research New England, Boston, MA, US, Summer 2014, supervised by Professor Sham Kakade. He was a research intern at Microsoft Research Asia, Beijing, China, Spring and Summer 2011, supervised by Dr. Yunhua Hu and Dr. Hang Li.

献给我挚爱的妻子贾勤。感谢她对我的爱与支持。

献给我的父亲王俊浩，母亲李凤霞和爷爷王德云。

感谢他们对我的养育之恩，使我成为了现在的自己。

Dedicated to my beloved wife 贾勤 (Qin Jia) for her love and support.

Dedicated to my parents 王俊浩 (Junhao Wang) and 李凤霞 (Fengxia Li),
my grandfather 王德云 (Deyun Wang) for making me be who I am.

ACKNOWLEDGEMENTS

First, I want to express my grateful appreciation to my PhD advisor, Professor Ramin Zabih, who gave me 100% support for the past six years. Ramin is a great advisor, manager, senior researcher and friend. Thank you for setting the perfect environment for me to grow as a researcher. It is you who gives me full freedom and encourage me to explore whatever project I am interested in. It is you who gives me useful hints and connects me to correct people when I get stuck in research. It is you who removes all the bureaucratic barriers and makes us concentrate on our own work. It is also you who teaches us how to achieve a good work-life balance, and helps us in all aspects beyond research. I am so grateful to have you as my PhD advisor.

I would also like to thank my committee members Professor Robert Kleinberg and Professor David Shmoys, as well as Professor David Williamson. I feel so fortunate to go to Bobby's analysis of algorithms course and two Davids' combinatorial algorithm seminar, and Professor David Williamson's combinatorial algorithm course in my early stage of PhD study. It is also my great honor to be Bobby's TA for the analysis of algorithms course. This experience, as well as two Davids' fantastic textbook on *The Design of Approximation Algorithms* laid a solid foundation for my research.

I want to thank Professor Endre Boros, who introduced the nice topic persistency and autarky to me. Thank you for teaching me everything you know about pseudo-Boolean optimization and giving me extremely important feedback on my research works.

Professor Junping Zhang, thank you for introduce this great computer vision and machine learning area to me, teaches me all the rudiments in this field, allows me to join your lab to gain early research experience when I was an un-

dergraduate.

Akshay Bhat, Richard Bowen, Alexander Fix, Charles Herrmann, Emil Keyder, Kimberly Wilber, thank you all for making our Cornell group and Google group a great place to work. It is my honor to work with you all various projects, co-author on a couple of papers. I learned a lot from all of you. I also want to thank Charles Herrmann and Richard Bowen again for their help to proofread and polish the language of this thesis.

I also want to thank my internal and external colleagues Professor Serge Belongie, Xilun Chen, Professor Noah Snavely, Yang Yuan, Yizhou Zhang (Cornell), Dror Aiger, Brett Allen, Aaron Archer, Aleksey Golovinskiy, Zhuoliang Kang, Sanjiv Kumar, Vivek Kwatra, Ce Liu, Nathan Silberman (Google), Bogdan Savchynskyy (Heidelberg) and Meng Tang (Waterloo), as well as the anonymous reviewers for their valuable feedback and help during this thesis research.

This thesis research has been funded by NSF grant IIS-1161282, IIS-1161860, IIS-1447473 and by a Google Faculty Research Award.

A special thank goes to U.S. Consulate General in Shanghai, who puts my US visa application under view and holds me in China for one extra month. The majority of the thesis writing was done during that period.

And finally, my wife Qin Jia, parents Junhao Wang and Fengxia Li, grandparents Deyun Wang and Yueying Wu, maternal grandmother Jilan Bao, parents-in-law Jian Jia and Zhaoxia Mo, and all the other family members. Thank you all for your love and support. I cannot become the person who I am today without you. I love you all!

TABLE OF CONTENTS

Biographical Sketch	iii
Dedication	iv
Acknowledgements	v
Table of Contents	vii
List of Tables	x
List of Figures	xi
1 Introduction	1
1.1 Notation	2
1.2 Pixel labeling problems in vision	3
1.2.1 Examples of pixel labeling problems in vision	4
1.3 Markov Random Fields: Solving pixel labeling problems	7
1.3.1 Maximum A-Posteriori (MAP) inference	7
1.3.2 Case study: Image denoising	9
1.3.3 Modeling likelihood functions and prior probability	11
1.3.4 Markov Random Fields (MRFs)	15
1.3.5 Reduction to energy minimization problem	18
1.4 Hardness of MRF inference problem	21
1.5 Approximate inference for Markov Random Fields	23
1.5.1 Evaluation of approximate inference	23
1.6 Persistency of Markov Random Fields	26
1.6.1 Advantages of persistency algorithm	28
2 Mathematical Background	30
2.1 Submodular functions	31
2.1.1 Diminishing marginal gains and attractive priors	31
2.1.2 Equivalent definitions of submodularity	33
2.1.3 Generalized submodularity to multilabel MRFs	34
2.2 Pseudo-Boolean optimization	35
2.2.1 Pseudo-Boolean optimization and binary MRF inference	35
2.2.2 Multilinear polynomial representation	36
2.2.3 Posiform representation	37
2.3 Persistency and autarky	38
2.4 Persistency algorithm preliminary	40
2.4.1 Dead-end elimination (DEE) and variants	41
2.4.2 Quadratic pseudo-Boolean optimization (QPBO)	43
2.4.3 Kovtun’s algorithm	47
2.4.4 Multilabel QPBO (MQPBO)	49
2.4.5 Partial optimality by Pruning (PBP)	50
2.4.6 Maximum persistency	53
2.4.7 Maximum persistency via iterative relaxed inference (IRI)	56
2.5 Graphcuts algorithm and move-making techniques	56

2.5.1	Binary pairwise submodular MRFs and st-MINcUT	57
2.5.2	Expansion move for multilabel MRFs	61
2.5.3	Energy truncation and QPBO for non-submodular MRFs	63
2.5.4	Sum-of-submodular (SoS) fusion for higher-order MRFs	64
3	Related work	72
3.1	Markov Random Fields in computer vision	72
3.2	Exact inference for certain sub-class of MRFs	77
3.3	Persistency algorithms for MRFs	78
3.3.1	Local condition-based persistency approaches	78
3.3.2	Flow-based persistency approaches	79
3.3.3	MRF/LP-based persistency approaches	81
3.4	Persistency for other combinatorial optimization problems	82
3.5	Approximate inference algorithms for pairwise MRFs	83
3.5.1	Graphcuts and move-making algorithms	83
3.5.2	Message passing algorithms	85
3.5.3	Linear programming relaxation algorithms	86
3.5.4	Combinatorial algorithms	87
3.6	Approximate inference algorithms for higher-order MRFs	88
3.7	Persistency as a pre-processing for approximate inference	90
4	Persistency relaxation for first-order MRFs	91
4.1	Motivation	91
4.1.1	Outline of the chapter	94
4.2	Persistency decision problem	94
4.2.1	Base relaxation of persistency	95
4.2.2	Independent local minimum labeling	98
4.2.3	k -condition and hierarchical relaxation	100
4.2.4	Approximating the k -condition	102
4.3	Persistency construction problem	105
4.3.1	ILM labeling construction	107
4.4	Theoretical connection to previous works	108
4.4.1	Connection to DEE	108
4.4.2	Connection to MRF/LP-based persistency approaches	111
4.5	Experimental results	112
4.5.1	Datasets and experimental environment	113
4.5.2	Approaches	114
4.5.3	Measurement	115
4.5.4	Experimental results on speedup and persistency ratio	116
4.5.5	Effectiveness on the greedy ILM labeling construction	118
4.5.6	Sensitivity analysis on the graph structure	118
4.5.7	Sensitivity analysis on number of iterations τ	119
4.5.8	Overall running time break down	120
4.5.9	Persistency relaxation for multilabel MRFs	123

4.5.10	Detailed experimental results	124
5	Discriminative persistency for first-order MRFs	137
5.1	Motivation	137
5.1.1	Outline of the chapter	139
5.2	Discriminative view of persistency	140
5.2.1	Comparison between persistency and autarky	140
5.2.2	Discriminative criterion	142
5.3	Discriminative persistency algorithm	145
5.3.1	Approximating underlying probability p	145
5.3.2	Efficiently checking our discriminative criterion	146
5.3.3	Overall algorithm	148
5.3.4	Generalized efficient check of our discriminative criterion	150
5.4	Theoretical analysis	150
5.4.1	Running time analysis	151
5.4.2	Per-instance performance bounds	152
5.4.3	Worst case performance bounds	154
5.5	Experimental results	156
5.5.1	Datasets and experimental environment	157
5.5.2	Approaches	158
5.5.3	Measurement	159
5.5.4	Parameter setup	160
5.5.5	Overall performance of discriminative persistency	161
5.5.6	Experimental results on speedup and recall values	164
5.5.7	Experimental results on energy and precision values	166
5.5.8	Experimental results on precision/recall trade-off	167
5.5.9	Investigation on parameter sensitivity	168
5.5.10	Experimental results with a typical parameter setup	169
5.5.11	Comparison to other MRF inference algorithm	171
5.5.12	Visualization results	173
5.5.13	Discriminative persistency for multilabel MRFs	174
5.5.14	Experimental results on worst case bound parameter ϵ	175
6	Discriminative persistency for higher-order MRFs	178
6.1	Motivation	178
6.2	Generalized persistency conditions for higher-order MRFs	181
6.2.1	Generalized Dead-end Elimination	181
6.2.2	Generalized discriminative persistency	183
6.3	Experimental results	185
6.3.1	Experiments setup	185
6.3.2	Overall performance of discriminative persistency	187
6.3.3	Experimental results on parameter sensitivity	190
	Bibliography	193

LIST OF TABLES

4.1	Comparison of persistency algorithms	92
4.2	Datasets description	113
4.3	Experimental results on Brain-MRI dataset	125
4.4	Experimental results on Color Segmentation dataset (\mathcal{N}_4 version)	127
4.5	Experimental results on Color Segmentation dataset (\mathcal{N}_8 version)	130
4.6	Experimental results on Inpainting dataset (\mathcal{N}_8 version)	133
4.7	Experimental results on Middlebury dataset	134
4.8	Experimental results on Scene Decomposition dataset	136
5.1	An example to show the restriction of the conservative persis- tency conditions	138
5.2	Datasets description	158
5.3	Comparison between discriminative persistency and other per- sistency algorithms	162
5.4	Precision/recall value vs. κ	167
5.5	Parameters chosen from the leave-one-out procedure	169
5.6	Performance of discriminative persistency with typical parameters	170
5.7	Comparison between discriminative persistency and other ap- proximate inference algorithms	171
5.8	Preliminary experimental results on multilabel MRFs	175
5.9	Experimental results with different ϵ on Color-Seg- \mathcal{N}_4 dataset . .	176
6.1	Experimental results for higher-order discriminative persistency	187
6.2	Experimental results for DisPer-ApproxSum with different κ . .	190
6.3	Experimental results for DisPer-Rank with different ρ	191

LIST OF FIGURES

1.1	An example of denoising problem	4
1.2	An example of stitching problem	5
1.3	An example of segmentation problem	6
1.4	An example of stereo problem	6
1.5	An example of optical flow problem	7
1.6	Case study: Image denoising	9
1.7	Graph representation of neighborhood system	13
2.1	Overview of persistency algorithms	41
2.2	The flow network G_ϕ corresponding to the posiform ϕ in Example 2.4.1	46
2.3	Residual network in Example 2.4.1	47
2.4	Illustration of boundary and interior	51
2.5	Basic building blocks to reduce binary pairwise submodular MRFs to st-MIN CUT	59
2.6	Illustration of expansion move algorithm	61
2.7	Intuition of higher-order capacity constraint in SoS-flow	66
2.8	An example of augmenting path for SoS-flow	68
2.9	An example of st-SoS-cut	69
2.10	Intuition of SoS upper bound approximation	70
4.1	High-level comparison between our methods and DEE	93
4.2	Visualization of the terms used for persistency relaxation	97
4.3	Performance of PR-based methods	116
4.4	Persistent variables ratio vs. stopping parameter τ	120
4.5	Overall running time vs. stopping parameter τ	121
4.6	Overall running time break down on Color-Seg- \mathcal{N}_4	122
4.7	Experimental results of PR for multilabel MRFs	123
4.8	Experimental results of PR for induced binary MRFs	124
5.1	Diagram to illustrate the discriminative criterion for persistency	143
5.2	Energy vs. time curves for instance T_{ed} on stereo dataset	163
5.3	Energy vs. time curves for instance T_{su} on stereo dataset	163
5.4	Energy vs. time curves for instance V_{en} on stereo dataset	164
5.5	Speedup-recall scatter on Color-Seg- \mathcal{N}_8 dataset	165
5.6	Energy vs. time curves compared with approximate inference algorithms on denoising-sq dataset	172
5.7	Visualization of stereo instance T_{eddy}	174
6.1	Higher-order stereo	178
6.2	Higher-order image denoising	179
6.3	Higher-order semantic segmentation	180
6.4	Energy vs. time curves for instance 106024 on FoE dataset	189

6.5	Energy vs. time curves for instance 299086 on FoE dataset . . .	189
6.6	Energy vs. time curves for DisPer-ApproxSum with different κ .	191
6.7	Energy vs. time curves for DisPer-Rank with different ρ	192

CHAPTER 1

INTRODUCTION

Markov Random Fields (MRFs) have achieved huge success for computer vision tasks in the past few decades. They are widely used in applications such as image segmentation, stereo, etc [63, 136].

The MRF inference problem is defined over n variables $\mathbf{x} = (x_1, \dots, x_n)$, where each x_i is drawn from a discrete label set \mathcal{L}_i . There is an energy function $f(x)$ that we wish to minimize given a set of parameters θ ; θ characterizes the *unary costs* $\theta_i : \mathcal{L}_i \mapsto \mathbb{R}$ and the *higher-order costs* $\theta_C : \prod_{i \in C} \mathcal{L}_i \mapsto \mathbb{R}$. The energy function is

$$f(\mathbf{x}) = \sum_{i \in V} \theta_i(x_i) + \sum_{C \in \mathcal{C}} \theta_C(\mathbf{x}_C) \quad (1.1)$$

where $\mathcal{G} = (V, C)$ is the hypergraph representation of the MRF.

The MRF inference problem is to find $\mathbf{x}^* = \operatorname{argmin}_{\mathbf{x}} f(\mathbf{x})$, which is equivalent to finding the Maximum A-Posteriori (MAP) estimation of the underlying probabilistic model derived from the MRF. Unfortunately the MRF inference problem is NP-hard even when $|\mathcal{L}| = 2$ (i.e. binary labels) [79].

A popular approach to the inference problem is to try to find the optimal labeling for a subset of the variables [31, 62, 71, 75, 77, 83, 84, 85, 119, 126, 129, 134, 144, 145, 148]. A partial labeling that holds in every global minimizer is said to be *persistent* [12]. An optimal labeling for a subset of the variables can be used to reduce the difficulty of the inference problem.

The goal of the first three chapters is to introduce the mathematical background of Markov Random Fields, and provide an overall survey of the MRF inference algorithms, in particular the persistency-based approaches.

1.1 Notation

We will describe common notation used in the thesis here for easy reference. All special notation will be introduced at the point of first usage.

We will write scalar variables as lowercase Latin letters, e.g., x, y, z . We will use bold lowercase letters to represent vectors, e.g., $\mathbf{x}, \mathbf{y}, \mathbf{z}$. We usually refer to the entries of a vector by its index, e.g., x_i is the i -th entry of \mathbf{x} . We will use capital letters to write sets, for example, $x \in X, y \in Y, z \in Z$. We will use \mathbf{x}_S to represent a subvector of \mathbf{x} , with the indices in set S . Given two subvectors \mathbf{x}_A and \mathbf{x}_B where $A \cap B = \emptyset$, we define $\mathbf{x}_A \oplus \mathbf{x}_B$ to be the composition of \mathbf{x}_A and \mathbf{x}_B . Let $\mathbf{y} = \mathbf{x}_A \oplus \mathbf{x}_B$ when $A \cap B = \emptyset$, then we have $y_i = (\mathbf{x}_A)_i$ when $i \in A$ and $y_i = (\mathbf{x}_B)_i$ when $i \in B$. We also use $[n] = 0, 1, \dots, n$ to represent the set of natural numbers up to n . The Iverson bracket $\llbracket P(x) \rrbracket = 1$ when the predicate $P(x)$ is true, and $\llbracket P(x) \rrbracket = 0$ otherwise. For example, if $f(x) := \llbracket x \geq 0 \rrbracket$, we have $f(10) = 1, f(-2) = 0$.

For undirected graphs, we will use $\mathcal{G} = (V, E)$ to represent a graph \mathcal{G} by its vertex set V and edge set $E \subseteq V \times V$. We will interchangeably use the term vertex and node. Edges are also referred to as an unordered pair $e = (u, v) \in E$. We will denote $\mathcal{N}(u) = \{v \mid (u, v) \in E\}$ as all the neighbors (a.k.a., adjacent vertices) of vertex u . We will also use $d(u) = |\mathcal{N}(u)|$ as the degree of node u .

For hypergraphs, we use the notation $\mathcal{G} = (V, C)$. Note that the only notation change is that we use cliques $C \subseteq V$ to represent hyperedges, and C is the set of all the cliques we have in the graph.

For probabilities, we will use $p(X), p(X, Y), p(X|Y)$ for the distribution, joint distribution and conditional distribution respectively.

For optimization problems $\min_{\mathbf{x}} f(\mathbf{x})$ or $\max_{\mathbf{x}} f(\mathbf{x})$, the optimizer is denoted as \mathbf{x}^* , while the set of all the optimizers is denoted as X^* . The optimum value is referred as $\text{OPT} = f(\mathbf{x}^*)$.

We usually refer to a digital image as \mathcal{I} . We assume an image comprises $H \times W$ pixels, and we denote the set of pixels as P . For grayscale images, we discretize the intensity of each pixel p into 256 levels, i.e., $\mathcal{I}_p \in \{0, \dots, 255\}$. For color images, we represent each pixel by its RGB values, i.e., $\mathcal{I}_p \in \{0, \dots, 255\}^3$. For some vision applications, we may group adjacent pixels into superpixels. We can partition the pixel set into disjoint subsets, $P = \bigcap_i P_i$, and let each P_i represent a superpixel.

1.2 Pixel labeling problems in vision

There are a variety of computer vision problems involving inferring values for each pixel or superpixel. We call this category of problems *pixel labeling problems*. In labeling problems, we need to infer multiple labels simultaneously, (mostly) at the pixel level. This is in contrast to problems like image classification, which only predicts a single variable, e.g., whether we have dog or person in a given image.

Formally, we define the *pixel labeling problem* in computer vision as:

Definition 1.2.1 (pixel labeling problem). A pixel labeling problem is a problem of assigning a set of variables $\mathbf{x} = \{x_i\}$ indexed by $i \in V$ where we have a one-to-one correspondence between V and the pixel set P or the superpixel set $\mathcal{P} = \{P_i\}$ ¹.

¹Due to the one-to-one correspondence, we may just use x_p to refer to the label for pixel p directly without ambiguity.

Each x_i takes its value from a discrete *label set* \mathcal{L}_i . We define $\mathcal{L} = \prod_{i \in V} \mathcal{L}_i$ to be the *label set* of the pixel labeling problem.

1.2.1 Examples of pixel labeling problems in vision

Pixel labeling problem cover a large portion of fundamental computer vision problems. In this section, we will introduce a few widely studied pixel labeling problems in computer vision. The purpose of this section is to:

- motivate the topic of this thesis by real useful applications,
- give concrete examples for pixel labeling problems in computer vision,
- familiarize readers with the problems used for evaluation in this thesis.

Example 1.2.1 (denoising). The *image denoising* task is to restore “true” pixel values from given noisy input images. Here, the variable set V is exactly the set of pixels, the label set $\mathcal{L}_i = \{0, \dots, 255\}$ (for grayscale images) is the restored, denoised image. An example is shown in Figure 1.1.



Figure 1.1: An example of denoising problem [57] © IEEE. (Left) Ideal clean source image. (Center) Noisy input image. (Right) Denoising result.

Example 1.2.2 (stitching). *Image stitching* (also known as panorama creation or photomontage) is the task of compositing n input images with overlap into a single large output image without creating visible stitching artifacts. In this

problem, the variable set V is identical to the pixel set in the output image, and the label set $\mathcal{L}_i = [n]$ represents an index of the input source image, indicating from which source image we will copy the pixel value in the final output image. An example is shown in Figure 1.2.

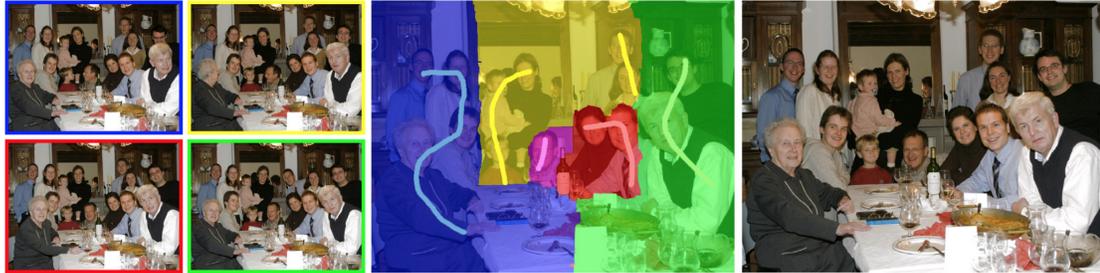


Figure 1.2: An example of stitching problem [1] © ACM. (Left) 4 input source images indexed by colors. Note that no single image has all the people with perfect expression. (Center) User inputs hard constraint via line strokes, meaning pixels under line strokes must select the corresponding source images. The color map apart from the user input line strokes visualizes the stitching result, indicating which source image contributes to the pixels values in the stitching result. (Right) Stitching result. Note that all the people in the image look good. Furthermore, we also obtain smooth transition between different source images.

Example 1.2.3 (segmentation). *Image segmentation* is the task of partitioning pixels into regions corresponding to distinct objects or parts of the scene. In this problem, we can either have our variable set V map to the whole pixel set, or we can pre-cluster pixels into superpixels in the color space, and map V to superpixels. The label set \mathcal{L}_i also depends on the applications. The most common cases include foreground extraction, where $\mathcal{L}_i = \{foreground, background\}$, or color segmentation where we want to cluster the pixels into n parts based on appearance (so $\mathcal{L}_i = [n]$), or semantic segmentation which determines which object each pixel belongs to in a pre-defined semantic label set, e.g., $\mathcal{L}_i = \{person, sky, grass, tree, car, unknown\}$. An example of image segmentation is provided in Figure 1.3.

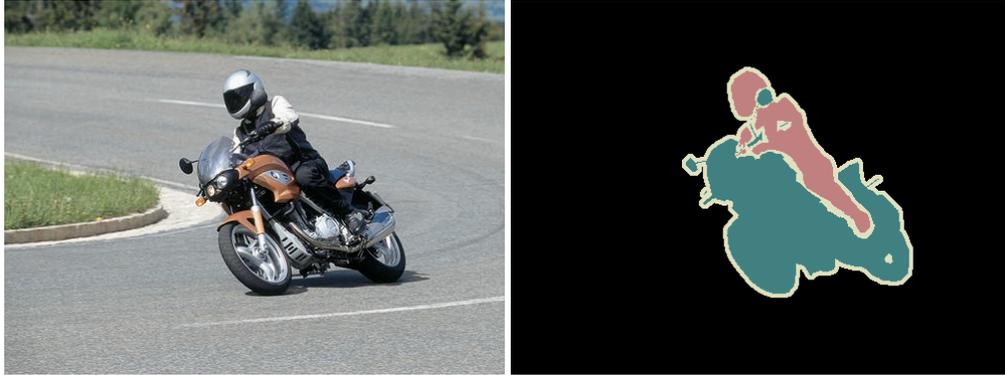


Figure 1.3: An example of segmentation problem [32] © Springer. (Left) Input image. (Right) Segmentation result (black: background, green: motorcycle, pink: person, white: unknown).

Example 1.2.4 (stereo). In the *stereo reconstruction* problem, we want to estimate the depth of the scene from a stereo pair (images of the same scene with slightly different view angles). In this problem, the variable set V is the set of all pixels. Without loss of generality, we can rectify the input images so that all the scene content only moves horizontally between the stereo pair. Therefore, our label set \mathcal{L}_i could be the discretized disparity (horizontal displacement) or the depth. Usually the discretization is finer at the depth close to the camera and coarser at the depth far away from the camera. One example is illustrated in Figure 1.4.



Figure 1.4: An example of stereo problem [121] © IEEE. (Left and Center) The stereo pair. Note that the second image moves slightly to right, and we have parallax: the relative position changes between the bear and the house. This parallax is caused by depth. (Right) Output disparity map. Darker pixels means farther away from the camera.

Example 1.2.5 (optical Flow). *Optical flow* is the task of tracking motion of ob-

jects between images (often two consecutive frames in a video). Given a pair of images, we want to infer a dense (or semi-dense) flow field (dx, dy) indicating the motion between frames. In this problem, the variable set V could be the whole pixel set to get a dense flow field, or we could map V to a downsampled grid to get a semi-dense flow field. Our label set \mathcal{L}_i usually is a discretized 2D motion (dx, dy) . An example of optical flow problem is provided in Figure 1.5.

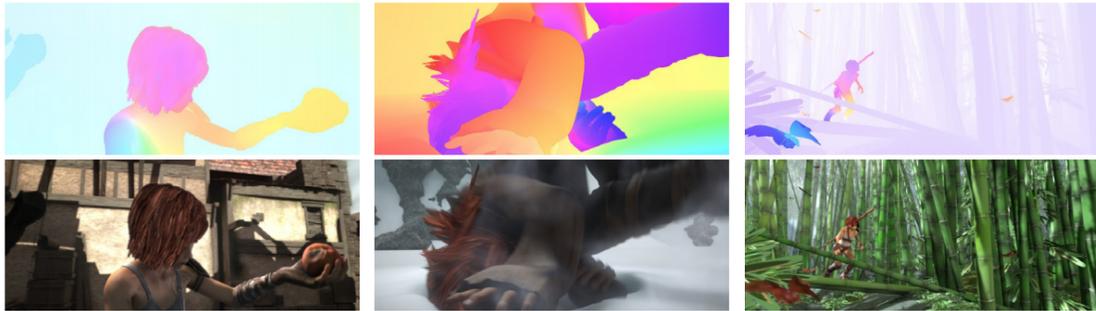


Figure 1.5: An example of optical flow problem [19] © Springer. (Top row) Visualization of the flow fields, in which color indicates direction according to a 360° color wheel, and hue indicates magnitudes. (Bottom row) Example of frames.

1.3 Markov Random Fields: Solving pixel labeling problems

We have formally defined the pixel labeling problem in Section 1.2. Now, we will talk about how to solve this problem.

1.3.1 Maximum A-Posteriori (MAP) inference

The most successful approach to solve pixel labeling problems is *probabilistic inference*.

Formally speaking, we want to infer the value of our labels $\mathbf{x} \in \mathcal{L}$ based on our observations $\mathbf{y} \in \mathcal{Y}$. For example, in the stereo problem, our \mathbf{x} is the disparity map and \mathbf{y} is the pair of input images. We assume there is an underlying joint probability distribution $p(\mathbf{x}, \mathbf{y})$. It is not necessary for us to consider the joint distribution directly, it is more useful to consider the conditional probability $p(\mathbf{x} | \mathbf{y})$. This conditional probability is also referred as the *posterior probability*, since it is the probability after we observe \mathbf{y} . The Maximum A-Posteriori (MAP) criteria infer our labeling \mathbf{x} by maximizing the posterior probability

$$\hat{\mathbf{x}}_{\text{MAP}} := \operatorname{argmax}_{\mathbf{x} \in \mathcal{L}} p(\mathbf{x} | \mathbf{y}). \quad (1.2)$$

In other words, we perform our inference on \mathbf{x} by maximizing the probability given the observation \mathbf{y} . This is Bayesian optimal when we have the 0 – 1 loss (Section 5.7.1.1, [105]), and empirically successful with other loss functions in practice.

In practice, we usually don't model the posterior probability $p(\mathbf{x} | \mathbf{y})$ directly. It is much easier to model the reversed conditional probability $p(\mathbf{y} | \mathbf{x})$, which is known as *likelihood function*, the probability of the observation given the underlying labels. Then we can apply Bayes' rule

$$p(\mathbf{x} | \mathbf{y}) = \frac{p(\mathbf{y} | \mathbf{x})p(\mathbf{x})}{p(\mathbf{y})}. \quad (1.3)$$

$p(\mathbf{x})$ doesn't depend on our observation, so it is called the *prior probability*. $p(\mathbf{y}) = \sum_{\mathbf{x} \in \mathcal{L}} p(\mathbf{y} | \mathbf{x})p(\mathbf{x})$ is called the *partition function* and usually denoted as Z . It is an important term in the learning problem. However, in our MAP inference problem, we want to infer the value of \mathbf{x} with a fixed \mathbf{y} . So $p(\mathbf{y})$ is a constant, which can be ignored. Therefore, we can rewrite our MAP criteria as

$$\hat{\mathbf{x}}_{\text{MAP}} := \operatorname{argmax}_{\mathbf{x} \in \mathcal{L}} p(\mathbf{x} | \mathbf{y}) = \operatorname{argmax}_{\mathbf{x} \in \mathcal{L}} \frac{p(\mathbf{y} | \mathbf{x})p(\mathbf{x})}{p(\mathbf{y})} = \operatorname{argmax}_{\mathbf{x} \in \mathcal{L}} p(\mathbf{y} | \mathbf{x})p(\mathbf{x}). \quad (1.4)$$

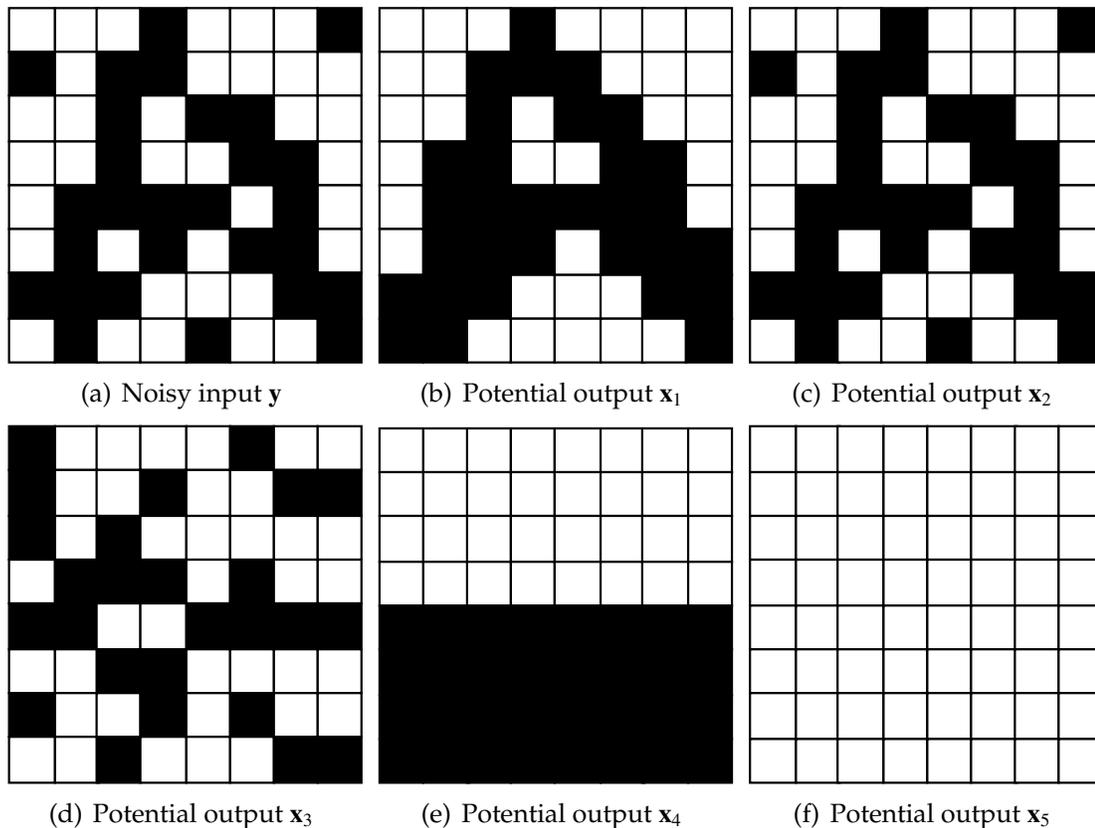


Figure 1.6: Case study: Image denoising. The input image is a noisy letter A. There are also 5 potential denoising results, which are the clean letter A, noisy letter A, a random result, a half-white half-black and a pure white image.

1.3.2 Case study: Image denoising

We have introduced the probability inference framework and the MAP criteria in the previous section. Note that it is usually computationally challenging or even infeasible to model the joint distribution and posterior probability directly. However, we may model the likelihood function and the prior probability approximately in practice and provide close enough approximation to the joint distribution via Bayes' rule. In this section, we will explore several desired properties of the likelihood function and prior probability we want via a concrete motivating application: image denoising.

For the likelihood function $p(\mathbf{y} | \mathbf{x})$, it is reasonable to assume our sensor samples the intensity for each pixel independently. Furthermore, we can assume the observed intensity at pixel i only depends on the ground truth intensity value at pixel i . The simplest model is just to have the observed intensity y_i follow a Gaussian distribution around the ground truth intensity x_i , where the parameter σ_1 could be learned from training data. Therefore, we have

$$p(\mathbf{y} | \mathbf{x}) := \prod_i p(y_i | x_i) = \prod_i \mathcal{N}(\|y_i - x_i\|, \sigma_1). \quad (1.5)$$

This probability is a fidelity term that enforces our output be similar to the input image. Therefore, among all the 5 potential output shown in Figure 1.6, we have $p(\mathbf{y} | \mathbf{x}_2) \gg p(\mathbf{y} | \mathbf{x}_1) \gg p(\mathbf{y} | \mathbf{x}_3) \approx p(\mathbf{y} | \mathbf{x}_4) \approx p(\mathbf{y} | \mathbf{x}_5)$, since \mathbf{x}_2 is exactly \mathbf{y} , while \mathbf{x}_1 looks much similar to \mathbf{y} compared against all the other potential outputs.

It is trivial to see that we cannot assume that the prior probability $p(\mathbf{x})$ is the uniform distribution over \mathbf{x} . Otherwise, the MAP inference criteria is equivalent to maximizing the likelihood function $p(\mathbf{y} | \mathbf{x})$. Using the likelihood function we defined above, the optimal label we get is $\hat{\mathbf{x}}_{\text{MAP}} = \mathbf{y}$, i.e., we will always output the input noisy image. This just means that without any prior knowledge, the best way to denoise an image is to do nothing, since it fits the likelihood function claiming each observed pixel intensity should be close to its true intensity.

This demonstrates the importance of the prior probability in the MAP inference framework. We know that not all images are equally possible in practice. It is also computationally challenging to model the global dependency between \mathbf{x} directly. In practice, we usually decompose \mathbf{x} into small pieces and rely on the local dependency to approximate the global dependency. For example, one important prior property we want \mathbf{x} hold is called *spatial locality*, i.e., the information x_i has at pixel i should be highly correlated to its nearby pixels. For the

image denoising task, we can decompose $p(\mathbf{x})$ over all pairs of adjacent pixels in the 4-connected grid, which makes up our edge set E . Again, we can assume the intensity difference between adjacent pixels $|x_i - x_j|$ follow another Gaussian distribution, where the parameter σ_2 could be learned from training data.

$$p(\mathbf{x}) := \prod_{(i,j) \in E} p_{ij}(x_i, x_j) = \prod_{(i,j) \in E} \mathcal{N}(\|x_i - x_j\|, \sigma_2). \quad (1.6)$$

Again, using the example shown in Figure 1.6, we have $p(\mathbf{x}_5) \gg p(\mathbf{x}_4) \gg p(\mathbf{x}_1) > p(\mathbf{x}_2) \gg p(\mathbf{x}_3)$, since our prior probability here claims that we prefer to have small intensity transitions. In other words, the constant image will achieve the highest prior probability while the random image will achieve the lowest prior probability.

Now, we have shown that if we only have the likelihood function or if we only have the prior probability, we won't get the desired output for the denoising task. However, if we put them together, we will have $p(\mathbf{x}_1 | \mathbf{y}) \gg p(\mathbf{x}_2 | \mathbf{y}) \gg p(\mathbf{x}_5 | \mathbf{y}) \approx p(\mathbf{x}_5 | \mathbf{y}) \gg p(\mathbf{x}_3 | \mathbf{y})$. Although the first potential output \mathbf{x}_1 minimizes neither the likelihood function $p(\mathbf{y} | \mathbf{x})$ nor the prior probability $p(\mathbf{x})$, it achieves the best trade-off between these two terms. Therefore, MAP inference will pick \mathbf{x}_1 as our inference result, which is desired.

1.3.3 Modeling likelihood functions and prior probability

We have explored the choice of likelihood function and prior probability through a concrete image denoising example in the previous section. Now, we will discuss the likelihood function and prior probability in the general case.

The space of $p(\mathbf{y} | \mathbf{x})$ and $p(\mathbf{x})$ is too big to be modeled directly. We can

choose a much simpler but still plausible approximation in practice. One important observation from vision applications is the *spatial locality* property: that the label x_i and observation y_i at pixel i depend only or mostly on nearby pixels' labeling. Therefore, we can make several independence assumptions to both the likelihood function $p(\mathbf{y} | \mathbf{x})$ and the prior probability $p(\mathbf{x})$.

Definition 1.3.1 (separable likelihood). The likelihood function is a separable likelihood if we can write $p(\mathbf{y} | \mathbf{x})$ as

$$p(\mathbf{y} | \mathbf{x}) := \prod_i p(\mathbf{y}_{\mathcal{N}(i)} | x_i), \quad (1.7)$$

where $\mathcal{N}(i)$ is the neighborhood of pixel i .

In this thesis, we will model the likelihood as separable. This likelihood is called separable likelihood because it is a separable function over variables x_i . Note that each x_i may contribute not only to a single y_i in this model, but also to a local neighbor region $\mathcal{N}(i)$. This is a straightforward generalization of the model we used in the denoising task shown in Section 1.3.2.

From the denoising example, we see that we want to rely on the prior probability $p(\mathbf{x})$ to enforce spatial locality of the labels. For vision applications, we can assume each label x_i only depends on the labels of nearby pixels.

Now, we will define the most widely used neighborhood structures for vision applications. The key idea is to employ the special 2D grid structure of pixels.

Example 1.3.1 (4-connected neighborhood \mathcal{N}_4). In this neighborhood, $E = \{(p, q) | p \neq q, \|p - q\|_1 \leq 1\}$ is the set of all pairs of pixels within Manhattan distance 1. That is, we define the neighbors of pixel p be its left, right, top, and bottom adjacent pixels.

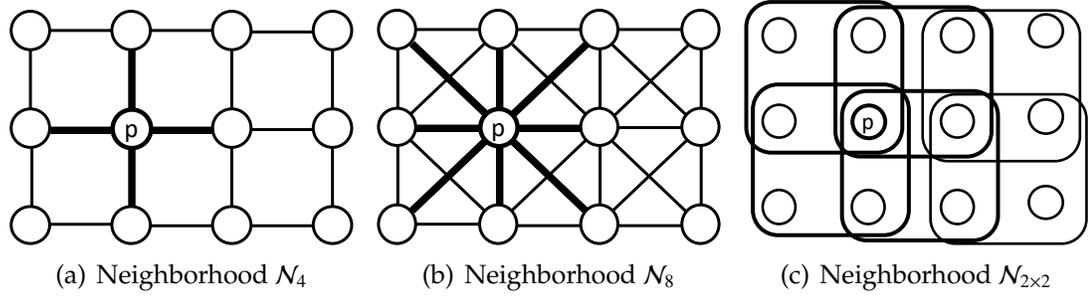


Figure 1.7: Graph representation of neighborhood system. Edge and cliques shown in bold highlight the neighbors of pixel p .

Example 1.3.2 (8-connected neighborhood \mathcal{N}_8). In this neighborhood, $E = \{(p, q) \mid p \neq q, \|p - q\|_\infty \leq 1\}$ be the set of all pairs of pixels within Chebyshev distance 1. That is, in addition to the four pixels in \mathcal{N}_4 , we also include the four diagonal adjacent pixels as pixel p 's neighbors.

Remark. When we have the edge set E defined in \mathcal{N}_4 or \mathcal{N}_8 above, we can also let vertex set $V = P$. Then we can derive the graph representation $\mathcal{G} = (V, E)$ of the given neighborhood system. Examples of the graph representations for \mathcal{N}_4 and \mathcal{N}_8 are shown in Figure 1.7(a) and Figure 1.7(b).

Definition 1.3.2 (pairwise dependency). In general, we could represent the probability dependency between a pair of variables (a.k.a., *pairwise dependency of prior probability*) by an arbitrary graph, not limited to \mathcal{N}_4 and \mathcal{N}_8 above. With such a graph representation $\mathcal{G} = (V, E)$, we can decompose our prior probability $p(\mathbf{x})$ as

$$p(\mathbf{x}) := \prod_{(i,j) \in E} p(x_i, x_j). \quad (1.8)$$

As a straightforward generalization of the pairwise dependency, we can use a clique $C \subseteq V$ to replace the pair of variables, and describe the dependency for the variables \mathbf{x}_C in clique C by probability $p(\mathbf{x}_C)$. In this way, we can define the *higher-order dependency of prior probability* formally.

Definition 1.3.3 (higher-order dependency). We can represent the probability dependency between sets of variables (a.k.a., *higher-order dependency*) by an arbitrary hypergraph $\mathcal{G} = (V, C)$. Then we can decompose our prior probability $p(\mathbf{x})$ as

$$p(\mathbf{x}) := \prod_{C \in \mathcal{C}} p(\mathbf{x}_C). \quad (1.9)$$

In computer vision, a frequently used higher-order dependency is defined by local patches. An example of the (2×2) -patch neighborhood system is shown in Figure 1.7(c).

Example 1.3.3 ($(a \times b)$ -patch neighborhood $\mathcal{N}_{a \times b}$). Let vertex set $V = P$, C be the set of all the $(a \times b)$ subgrids in the $H \times W$ grid, we can represent the this $(a \times b)$ -patch neighborhood system $\mathcal{N}_{a \times b}$ as a hypergraph $\mathcal{G} = (V, C)$.

Remark. We have primarily focused on examples where our variable set V is identically the pixel set P in the image. However, Definition 1.3.2 and Definition 1.3.3 are defined on general (hyper)graph structures. The definition of pairwise dependency and higher-order dependency is still valid when our variable set V is a superpixel set \mathcal{P} or other objects.

The probabilistic independency derived from Definition 1.3.2 and Definition 1.3.3 is referred as *Markov property*, and this model is referred as *Markov Random Fields*, which is the main topic of this thesis. Therefore, we will use the next section to formally define and discuss it.

1.3.4 Markov Random Fields (MRFs)

Markov Random Fields (MRFs, also known as undirected graphical model) is a powerful tool to model conditional dependency among a large set of random variables. Compared with a Bayesian network (also known as directed graphical model), it has advantages that it is both conceptually and computationally easier to model the desired independence properties and it is easier to make local modification of the whole model.

Each Markov Random Field has an underlying graph structure $\mathcal{G} = (V, E)$, where each vertex in V represent a random variable, and we add an edge between two vertices if we want to describe the conditional dependency between them. Now, we can define the probability over this graph.

Definition 1.3.4 (MRF factorization). Given a graph $\mathcal{G} = (V, E)$, we define the clique set C as all fully-connected subgraphs in \mathcal{G} :

$$C := \{C \subseteq V \mid (i, j) \in E, \forall i, j \in C\}. \quad (1.10)$$

Definition 1.3.5 (Gibbs distribution). A distribution p is a *Gibbs distribution* parameterized by the set of *cliques* $C = \{C_1, C_2, \dots, C_k\}$ and *clique probabilities* $\mathcal{P} = \{p_1, p_2, \dots, p_k\}$ if it follows

$$p(\mathbf{x}) := \frac{1}{Z} \prod_i p_i(\mathbf{x}_{C_i}), \quad (1.11)$$

where

$$Z := \sum_{\mathbf{x}} \prod_i p_i(\mathbf{x}_{C_i}) \quad (1.12)$$

is the normalizing constant referred as the *partition function*.

One advantage of MRF is that we can easily describe the independency given the graph structure. Intuitively, probabilistic dependency flows along the undi-

rected paths in the graph, and it is blocked if we condition on the intervening vertices. We can formally define the independency as following.

Definition 1.3.6 (global independency). We say a vertex set Z separates X and Y if X and Y are not connected in $(V - Z, E)$. This is denoted as $\text{sep}_{\mathcal{G}}(X, Y | Z)$. Then we can define the *global independency* w.r.t. \mathcal{G} as

$$\mathcal{I}_g(\mathcal{G}) = \{(X \perp Y | Z) | \text{sep}_{\mathcal{G}}(X, Y | Z)\}. \quad (1.13)$$

Definition 1.3.7 (local independency, Markov property). Given a vertex set X , define its neighbor $\mathcal{N}(X) = \cup_{x \in X} \mathcal{N}(x) - X$ be all the vertices not in X but adjacent to any vertex in X . Then we can define the *local independency* w.r.t. \mathcal{G} as

$$\mathcal{I}_\ell(\mathcal{G}) = \{(X \perp V - X - \mathcal{N}(X) | \mathcal{N}(X)) | X \subseteq V\}. \quad (1.14)$$

Both the global independency and local independency describes the independency condition on the adjacent vertices from different aspects. Actually, we can prove the equivalence between them from the following theorem.

Theorem 1.3.1 ([72]). We say $p \models \mathcal{I}$ if the given probability distribution p satisfies the probabilistic independency defined in \mathcal{I} . Then for any graph \mathcal{G} and distribution p , we have $p \models \mathcal{I}_g(\mathcal{G}) \Leftrightarrow p \models \mathcal{I}_\ell(\mathcal{G})$.

We can easily shown the Gibbs distribution defined over graph \mathcal{G} satisfies the desired Markov property defined above. Furthermore, the Hammersley-Clifford theorem claims the opposite direction that all the positive distribution satisfying Markov property can be written in the form of a Gibbs distribution.

Theorem 1.3.2 (Hammersley-Clifford [51, 72]). A strictly positive probability distribution p satisfies Markov property over graph \mathcal{G} if and only if it can be written as the Gibbs distribution over graph \mathcal{G} .

Remark. In some definition of the factorization over graph \mathcal{G} , the clique set C only contains maximal cliques. Actually, both representations are equivalent. On one hand, we can absorb the clique function for smaller cliques into the clique functions of larger cliques. Given clique $C_1 \not\subseteq C_2 \subseteq V$, we can rewrite $\tilde{p}_{C_2}(\mathbf{x}_{C_2}) = p_{C_1}(\mathbf{x}_{C_1})p_{C_2}(\mathbf{x}_{C_2})$. On the other hand, given Gibbs distribution only defined on maximal cliques, we can append uniform distribution as a dummy term for the non-maximal cliques to get the Gibbs distribution over all the cliques. We prefer to keep the non-maximal clique in the Gibbs distribution representation since we may define different clique functions over C_1 and its superset C_2 separately. It is mathematically equivalent but conceptually easier to understand the model if we don't mix them together.

Remark. For the same sake in the previous remark, it is also not necessary to list all the cliques in a Gibbs distribution in practice. If we don't have a special clique function defined over non-maximal clique C , we can safely absorb this uniform distribution (i.e., a constant) into any maximal clique function containing C and simplify the Gibbs distribution. In general, we can write the Gibbs distribution only over the cliques we care about in practice. We will discuss this issue again in Section 1.3.5.

Remark. Although we talked about the Markov property of the prior probability $p(\mathbf{x})$ in this section. Note that for the separable likelihood $p(\mathbf{y} | \mathbf{x}) = \prod_i p(\mathbf{y}_{N(i)|x_i})$, each term only depends on a singleton clique $C = \{x_i\}$ with the fixed observation \mathbf{y} . Therefore, $p(\mathbf{y} | \mathbf{x})p(\mathbf{x})$ can also be represented as a Markov Random Field.

1.3.5 Reduction to energy minimization problem

It is a common trick to use the negative log operator to convert the product of strictly positive probability into a sum. Applying this trick to (1.3), we get:

$$-\log(p(\mathbf{x} | \mathbf{y})) = -\log(p(\mathbf{y} | \mathbf{x})) - \log(p(\mathbf{x})) + \log(p(\mathbf{y})). \quad (1.15)$$

Definition 1.3.8 (data term).

$$f_{\text{data}}(\mathbf{x}) := -\log(p(\mathbf{y} | \mathbf{x})) \quad (1.16)$$

is referred to as the *data term* for our pixel labeling problem. It is called the data term because it depends on our observation data \mathbf{y} .

Definition 1.3.9 (prior term).

$$f_{\text{prior}}(\mathbf{x}) := -\log(p(\mathbf{x})) \quad (1.17)$$

is referred to as *prior term* of our pixel labeling problem. It is called the prior term because it purely depends on our prior knowledge of the labels \mathbf{x} , and not our observation \mathbf{y} .

Definition 1.3.10 (energy).

$$f(\mathbf{x}) := f_{\text{data}}(\mathbf{x}) + f_{\text{prior}}(\mathbf{x}). \quad (1.18)$$

The sum of the data term and prior term is referred to as the *energy* function of the pixel labeling problem. It is also known as the *potential* function in the literature. This name comes from the Ising model [58], one of the very first MRF applications, used in statistical mechanics. In this model, $f(\mathbf{x})$ has a physical connection to the energy of the atomic spin configuration.

Since $-\log(\cdot)$ is a strictly monotonically decreasing function, and $p(\mathbf{y})$ is a constant with fixed \mathbf{y} , our MAP inference criteria (1.4) is equivalent to:

$$\operatorname{argmax}_{\mathbf{x} \in \mathcal{L}} p(\mathbf{x} | \mathbf{y}) = \operatorname{argmin}_{\mathbf{x} \in \mathcal{L}} f(\mathbf{x}). \quad (1.19)$$

With the definition of separable likelihood function and Gibbs distribution of the prior probability, we could further expand the definition of the energy function $f(\mathbf{x})$, and define the MRF inference problem, the central topic of this thesis.

Definition 1.3.11 ((higher-order) MRF inference problem). The *(Higher-order) MRF inference problem* minimizes the energy function defined over hypergraph $\mathcal{G} = (V, C)$ in the following form:

$$f(\mathbf{x}) := \sum_{i \in V} \theta_i(x_i) + \sum_{C \in \mathcal{C}} \theta_C(\mathbf{x}_C). \quad (1.20)$$

Mathematically, each singleton $\{x_i\}$ is also a clique. However, since a lot of work has focused specifically on singleton cliques in relation to the inference problem, we usually write them apart explicitly and assume C only contains cliques with more than 2 variables. One special case of the general MRF inference problem is when $|C| = 2, \forall C \in C$. We call this case a pairwise MRF inference problem.

Definition 1.3.12 (pairwise MRF inference problem). The *pairwise MRF inference problem* minimizes the energy function defined over graph $\mathcal{G} = (V, E)$ in the following form:

$$f(\mathbf{x}) := \sum_{i \in V} \theta_i(x_i) + \sum_{(i,j) \in E} \theta_{ij}(x_i, x_j). \quad (1.21)$$

Definition 1.3.13 (unary terms, pairwise terms, higher-order terms). θ_i in (1.20) and (1.21) is referred as *unary terms* of the MRF. θ_{ij} in (1.21) is referred as *pairwise*

terms of the MRF. θ_c in (1.20) is referred as *higher-order terms* of the MRF (this may also include pairwise terms).

Remark. Some references in the literature interchangeably use the term unary terms and data terms, and pairwise/higher-order terms and prior terms. However, we want to differentiate between these two sets of concepts in this thesis. When we talk about data terms and prior terms, it means that the term comes from the likelihood function or the prior probability. When we talk about unary terms, pairwise terms, and higher-order terms, it only indicates how many free variables we have in that function. With the separable likelihood assumption, all the data terms are unary terms. However, the opposite direction is not true. For example, we may have a prior knowledge that we prefer to have black pixels over white pixels for each single pixel, which yields a unary prior term.

There are a few advantages of using the energy rather than the probability.

- We can use $\exp(\cdot)$ to convert any given energy functions back to its probabilistic interpretation. So we don't lose the representation ability.
- In practice, sometimes it is easier to model the problem from the energy point of view. This allows us to compare the multiple given label configurations and determine which is more likely. For example, *Potts model* penalizes label smoothness by assigning 0 cost for same labels and constant cost $c > 0$ for different labels in the prior terms.
- It is also conceptually easier for humans to handle the additive model rather than the multiplicative model.
- It is easier to make local modifications of energy rather than probability. For example, we used the Gaussian distribution to model the label smoothness in our image denoising example before. However, to improve

the robustness of the model, we often allow an intensity change over a certain threshold to be equally likely, since large intensity changes happen across object boundaries. If we formalize the problem as energy, we can add the cap directly. However, if we use a probabilistic interpretation, we need to have a complicated normalization to make sure that we still meet the requirements for a probability.

Therefore, for the rest of the thesis, we will downplay the probabilistic aspect of Markov Random Fields a little bit, and view the optimization problem of the energy in the form (1.20) and (1.21). This is the MRF inference problem we want to address in this thesis.

1.4 Hardness of MRF inference problem

It is known that the general MRF inference is a computationally challenging problem. We will present a few hardness analyses of this problem in this section. A more detailed study on the hardness of the MRF inference problem can be found in [95].

In the previous section, we presented the MRF inference problem as an optimization problem. In order to study hardness, we need to work on its variant *decision problem*.

Definition 1.4.1 (decision problem of MRF inference, MRFINFERENCE). The decision problem of MRF inference is defined as given energy function $f(\mathbf{x})$ and energy value e , whether we have a label $\hat{\mathbf{x}}_{\text{MAP}}$ such that $f(\hat{\mathbf{x}}_{\text{MAP}}) \leq e$.

We can show that MRFINFERENCE is NP-hard by reducing the well known

NP-complete problem MAXCUT to this problem.

Definition 1.4.2 (MAXCUT). Given a graph $\mathcal{G} = (V, E)$ and a value k , determine whether we can partition V into disjoint set $V = S + T$ such that the cut value $\text{cut}(S, T) \geq k$, where

$$\text{cut}(S, T) := |\{(s, t) \mid s \in S, t \in T, (s, t) \in E\}|. \quad (1.22)$$

Theorem 1.4.1. MRFINFERENCE is NP-hard.

Proof. Given arbitrary MAXCUT problem instance defined by graph $\mathcal{G} = (V, E)$ and threshold k . We can define the MRFINFERENCE problem instance over the same graph $\mathcal{G} = (V, E)$ and define the following pairwise energy function $f(\mathbf{x}) = \sum_{(i,j) \in E} \theta_{ij}(x_i, x_j)$:

$$\theta_{ij}(x_i, x_j) = \begin{cases} -1, & x_i \neq x_j, \\ 0, & x_i = x_j. \end{cases} \quad (1.23)$$

We also define the label set $\mathcal{L}_i = \{0, 1\}$ and define the threshold of the MRFINFERENCE problem $e = -k$.

We can construct the one-to-one correspondence between a MAXCUT solution and a MRFINFERENCE solution in polynomial time. Let $v_i \in S$ if and only if $x_i = 0$, it is easy to show we have $\text{cut}(S, T) = -f(\mathbf{x})$ with this construction.

Therefore, any certificate $\hat{\mathbf{x}}$ of the MRFINFERENCE such that $f(\hat{\mathbf{x}}) \leq e = -k$ will also give us a certificate that we have a cut at least k in \mathcal{G} . On the other hand, the infeasibility of achieving $f(\mathbf{x}) \leq e$ also proves there is no cut with value at least k in \mathcal{G} . In sum, we can reduce the NP-complete MAXCUT problem to the MRFINFERENCE problem in polynomial time, which concludes MRFINFERENCE is NP-hard. \square

1.5 Approximate inference for Markov Random Fields

We have discussed the hardness of the MRF inference problem in the previous section. This fact says it is impossible to solve the general MRF inference problem optimally in polynomial time, unless $P = NP$. We will discuss a few special families of MRFs, which can be solved optimally, in Chapter 3 and Chapter 2. In this section, we will introduce the high-level framework to apply approximate inference algorithms for Markov Random Fields, and the methods to evaluate them.

Approximate inference algorithm performs the optimization of the MRF inference problem in the best of effort flavor. The label $\hat{\mathbf{x}}$ computed by the approximate inference algorithm is usually referred to as the *approximated solution* of the problem. The goal is get the energy function of the MRF $f(\hat{\mathbf{x}})$ as close to the optimal energy $f(\mathbf{x})$ as possible. Please refer to Williamson and Shmoys [147] for a good textbook to cover fundamental of approximation algorithms.

1.5.1 Evaluation of approximate inference

Approximation bound analysis

The most widely used theoretical guarantee of the approximate inference algorithm is the *multiplicative* and *additive* error bound of the algorithm, defined as following.

Definition 1.5.1 (multiplicative bound and additive bound). If the approximation solution $\hat{\mathbf{x}}$ computed by the given approximate inference algorithm for ar-

bitrary strictly positive energy function $f(\mathbf{x}) > 0$, we have

$$f(\mathbf{x}^*) \leq f(\hat{\mathbf{x}}) \leq \alpha f(\mathbf{x}^*) + \epsilon, \quad (1.24)$$

we say this approximate inference algorithm is a (α, ϵ) -approx algorithm.

In the special case $\epsilon = 0$, $f(\hat{\mathbf{x}}) \leq \alpha f(\mathbf{x}^*)$, then $\alpha \geq 1$ is called the *multiplicative error bound* or the *approximation ratio* of the algorithm. We also denote it as α -approx algorithm in short.

In another special case $\alpha = 1$, $f(\hat{\mathbf{x}}) \leq f(\mathbf{x}^*) + \epsilon$, $\epsilon \geq 0$ is called the *additive error bound* of the algorithm.

Remark. We further enforce $f(\mathbf{x})$ to be strictly positive since it is conceptually easier to require our multiplicative factor $\alpha \geq 1$, i.e., $\alpha = 2$ means any approximation solution will not be two times larger than the optimal energy. Note that adding a constant to $f(\mathbf{x})$ will not change the optimizer of the problem. So we can always convert an arbitrary MRF inference problem to have a strictly positive energy function by adding a large enough constant. One common misunderstanding is whether adding a constant will affect the multiplicative bound analysis. For example, if $f(\mathbf{x}^*) = 5$ and our approximation solution $f(\hat{\mathbf{x}}) = 15$. Then our approximation is 3 times larger than the optimal solution. Suppose we have $f' = f + 10000$ and our inference algorithm is also invariant to adding a constant, we will have $f'(\mathbf{x}^*) = 10005$, $f'(\hat{\mathbf{x}}) = 10015$, with the approximation solution and optimal solution almost equal. So some people may ask whether it still makes sense to analyze the multiplicative bound. This is a wrong impression since our multiplicative bound must hold for any input function. We also need to consider the equivalent energy function f'' which pushes $f''(\mathbf{x}^*)$ towards 0. This will introduce a larger ratio between the approximation solution and the optimal solution. It also shows if we have an inference algorithm with bounded

approximation ratio, it cannot be invariant w.r.t. adding a constant. Otherwise, we can also subtract a constant to make $f(\mathbf{x}^*) \rightarrow 0^+$ and the approximation ratio go to infinity. Given this universal modifier over all the energy function, it still makes sense to study the multiplicative bound of the algorithm.

Remark. The multiplicative bound and the additive bound defined in Definition 1.5.1 are the properties of the algorithm before it is run on any concrete energy function. As a result, these are also called the *worst case bound* of the algorithm. There is another type of error bound we can only know after we solve the concrete energy function. One example of such a bound is the duality gap between the primal solution and the dual solution of the linear programming problem. The duality gap indicates how close we approximated the optimizer. We call this type *per-instance bound*. In this thesis, we use the worst case bound by default, and we will make our reference to per-instance bound explicitly; for example, we will say the duality gap is a per-instance additive bound for the linear programming problem.

Running time and space analysis

We will use the asymptotic analysis [28] to understand the running time and space complexity of the inference algorithm. Usually we will represent the scale of the MRF inference problem by the number of variables N , the number of edges/cliques M , the maximum size of the clique K and the number of labels $L := |\cup_{i \in V} \mathcal{L}_i|$, and represent the running time and space bound as a function of N, M, K, L .

We can also measure the actual running time and memory consumption in practice, and provide an empirical aspect of the evaluation.

1.6 Persistency of Markov Random Fields

We introduced the approximate inference for MRFs in the previous section, which computes $\hat{\mathbf{x}}$ that encourages the energy function $f(\hat{\mathbf{x}})$ to be close to $f(\mathbf{x}^*)$. We can also study algorithms that label a subset of variables, where we can prove optimality for each variable in that subset. It is known as the *persistency* of MRFs, which we will introduce in this section. Here is the formal definition of persistency.

Definition 1.6.1 (strong persistency). Given a MRF inference problem, $x_i = \ell$ is called *strong persistent* if and only if $x_i^* = \ell$ for all optimizers x^* of f .

Definition 1.6.2 (weak persistency). Given a MRF inference problem, $x_i = \ell$ is called *weak persistent* if and only if $x_i^* = \ell$ for any optimizers x^* of f .

Example 1.6.1. Suppose $V = \{x_1, x_2, x_3\}$, and $\mathcal{L}_i = \{0, 1\}$, and we have two optimizer $(0, 0, 1)$ and $(0, 1, 1)$. In this case, $x_1 = 0$ is a strong (which implies weak) persistent label of the problem. Both $x_2 = 0$ and $x_2 = 1$ are the weak persistent label, but we have no strong persistent label for x_2 . $x_3 = 1$ is a strong (which implies weak) persistent label.

Then we can get the following corollary from the definition directly.

Corollary 1.6.1. $x_i = \ell$ is a strong persistent label implies $x_i = \ell$ is also a weak persistent label.

Remark. Strong persistency is the desired property we want for optimization task, since we can fix the variable with its strong persistent label without affecting the optimization task. Weak persistent is ambiguous when we want to fix a label, but it is the correct property if we want to rule out a label, i.e., we do not need to consider a label if we can prove it cannot be weak persistent.

We may also refer a subvector of \mathbf{x} as a *partial labeling* of the MRF problem, and define its persistency based on the persistency of each single variable in that partial labeling.

Definition 1.6.3 (partial labeling). We will use \mathbf{x}_S to represent a subvector of \mathbf{x} with indices in S , where $S \subseteq V$. Let $\mathcal{L}_S = \prod_{i \in S} \mathcal{L}_i$ be the label space of \mathbf{x}_S . We will refer to \mathbf{x} and \mathbf{x}_S as a *full labeling* and *partial labeling* (w.r.t. S) respectively.

Definition 1.6.4 (persistent partial labeling). We will call the partial labeling \mathbf{x}_S strong persistent or weak persistent if every single variable x_i in \mathbf{x}_S is strong persistent or weak persistent.

Since the MRF inference problem is NP-hard, we cannot compute the persistent labels for each single variable. However, we can still compute persistent labels for a very large subset of all the variables, which proves the optimality of those variables. Even for the variables we cannot get a persistent label, we may still prove some labels cannot be a weak persistent label of that variable. So we can exclude them from the label set.

We can also combine the persistency algorithms and the approximate inference algorithm together. As a pre-processing step, we employ the persistency algorithm to simplify the problem that the approximate inference algorithm needs to solve. A high level description of the algorithm could be found in Algorithm 1.1.

In Algorithm 1.1, $x_A \oplus (x_i = \ell)$ means we concatenate variable $x_i = \ell$ to the existing partial labeling x_A . Note that $\mathcal{L}_i \leftarrow \{\ell\}$ on line 9 of the code is for notation simplicity; it indicates that we fix the label for one particular variable. In practice, we will remove that variable from V and project our energy function with

Algorithm 1.1: Persistency algorithm as pre-processing for MRF inference

```
1  $A \leftarrow \emptyset;$ 
2  $x_A \leftarrow \emptyset;$ 
3 repeat
4   for  $i \in V \setminus A$  do
5     for  $\ell \in \mathcal{L}_i$  do
6       if we can prove  $x_i = \ell$  is strong persistent then
7          $x_A \leftarrow x_A \oplus (x_i = \ell);$ 
8          $A \leftarrow A \cup \{i\};$ 
9          $\mathcal{L}_i \leftarrow \{\ell\};$ 
10        break;
11      else
12        if we can prove  $x_i = \ell$  cannot be weak persistent then
13           $\mathcal{L}_i \leftarrow \mathcal{L}_i \setminus \{\ell\};$ 
14        end
15      end
16    end
17  end
18 until converges or reaches iteration limit;
19 Solve the remaining problem using any approximate inference;
```

that fixed label. A formal definition will be discussed in Chapter 2. Note that every time we prove some persistency labels or rule out impossible optimal labels, we may have further information for other variables as well. So we can iteratively run this algorithm for several iterations or until convergence.

1.6.1 Advantages of persistency algorithm

There are several advantages of computing persistency labels.

- In the approximate inference framework, we focused more on the optimality of the energy overall, rather than the optimality of each variable. However, we know variables of MRFs for computer vision problems usually correspond to properties of individual pixel or patch. It might be inter-

esting to show optimality for such variables. For example, it is interesting to know pixel p must be at depth d in the stereo problem. Similarly it is interesting to know pixel q must be sky in the segmentation problem.

- The state-of-the-art persistency algorithm can prove persistency for almost all the labels in practice. When we combine such persistency algorithms and the approximate inference algorithms, usually we will have a simpler problem than the original one, since we provide additional information on persistency to the approximate inference. It usually results in lower energy compared to solving the whole problem with the approximate inference algorithm, and provides better quality for computer vision tasks.
- We can establish a trade-off between the running time of a persistency algorithm and the number of persistent labels it can find. Most MRF problems in practice contain a large portion of easy problems and a very small portion of hard problems. Another category of state-of-the-art persistency algorithm can compute a large enough persistent partial labeling very efficiently. Therefore, we can offload the easy part of the MRF inference problem from the computationally expensive approximate inference stage to the relatively cheap persistency stage, and achieve an overall faster running time.
- The definition of persistency is conservative, i.e., fix the strong persistent labels or exclude the labels which can be proved to be not weak persistent from the current MRF inference problem will never result in a mistake. It will only simplify the problem by feeding more information to the approximate inference stage. We also show in our work that by compromising the soundness of the persistency definition a little bit, we may benefit more from the persistency algorithm without hurting the quality too much.

CHAPTER 2

MATHEMATICAL BACKGROUND

In this chapter, we will cover the mathematical background needed to understand both the most relevant related work and the main results of this thesis. To recapitulate, we want to solve the pairwise MRF inference problem

$$\min_{\mathbf{x}} f(\mathbf{x}) := \sum_{i \in V} \theta_i(x_i) + \sum_{(i,j) \in E} \theta_{ij}(x_i, x_j), \quad (2.1)$$

defined over graph $\mathcal{G} = (V, E)$ and its general form of higher-order MRFs

$$\min_{\mathbf{x}} f(\mathbf{x}) := \sum_{i \in V} \theta_i(x_i) + \sum_{C \in \mathcal{C}} \theta_C(\mathbf{x}_C), \quad (2.2)$$

defined over hypergraph $\mathcal{G} = (V, C)$.

This chapter is organized in the following way. We will first introduce the concept of submodular function for set functions and binary MRFs, and the natural extension of submodular functions to multilabel MRFs in Section 2.1. It is an important condition to characterize certain sub-classes of MRFs which can be solved exactly. Then we will introduce the pseudo-Boolean optimization from the operation research community in Section 2.2, which provides the mathematical foundation for many of the partial optimality results we have for MRFs. We introduce persistency and autarky, the central topic of the whole thesis in Section 2.3. In Section 2.4, we conduct a literature survey and explain the high-level ideas of multiple representative persistency algorithms in the literature. Finally, we review the basis of graphcuts algorithm and move-making techniques in Section 2.5. Graphcuts algorithm is the most successful approximate MRF inference algorithm. We will employ it in conjunction with our proposed persistency algorithms to evaluate the performance of our methods in this thesis.

2.1 Submodular functions

We will introduce the basis of submodular functions in this section. Submodularity is an important property to characterize certain sub-classes of MRFs which can be optimized exactly.

2.1.1 Diminishing marginal gains and attractive priors

Submodular functions are usually defined as set functions $2^V \mapsto \mathbb{R}$. They capture the *diminishing marginal gain* property over discrete binary choices. Consider that we have a set function $f : 2^V \mapsto \mathbb{R}$, where V is the *ground set* of choices. We can pick an arbitrary subset $S \subseteq V$ with value $f(S)$. The *marginal gain* of adding element $i \notin S$ is described by $f(S \cup \{i\}) - f(S)$. We will use $f(S + i)$ as a shorthand for $f(S \cup \{i\})$. The diminishing marginal gain property says that we will have smaller marginal gain if we add i to a larger set $T \supseteq S$, i.e., $f(T + i) - f(T) \leq f(S + i) - f(S)$.

Definition 2.1.1 (submodular functions). A function $f : 2^V \mapsto \mathbb{R}$ is *submodular* if for every $S \subseteq T \subseteq V$, and $i \in V \setminus T$, we have:

$$f(T + i) - f(T) \leq f(S + i) - f(S). \quad (2.3)$$

Similarly, we can define *supermodular functions* and *modular functions*.

Definition 2.1.2 (supermodular functions). A function $f : 2^V \mapsto \mathbb{R}$ is *supermodular* if for every $S \subseteq T \subseteq V$, and $i \in V \setminus T$, we have:

$$f(T + i) - f(T) \geq f(S + i) - f(S). \quad (2.4)$$

Definition 2.1.3 (modular functions). A function $f : 2^V \mapsto \mathbb{R}$ is *modular* if for every $S \subseteq T \subseteq V$, and $i \in V \setminus T$, we have:

$$f(T + i) - f(T) = f(S + i) - f(S). \quad (2.5)$$

We can easily show the following three properties.

Lemma 2.1.1. *f is a modular function if and only if it is both a submodular function and supermodular function.*

Lemma 2.1.2. *$f : 2^V \mapsto \mathbb{R}$ is a modular function if and only if it is a linear function over V , i.e., $\exists g : V \mapsto \mathbb{R}$ such that $f(S) = \sum_{i \in S} g(i)$.*

Lemma 2.1.3. *Non-negative linear combination of submodular functions are still submodular. Let f_1, f_2, \dots, f_k be k submodular functions, and $a_1, a_2, \dots, a_k \in \mathbb{R}$ are non-negative, then we have $f = \sum_{i=1}^k a_i f_i$ is submodular as well.*

We can trivially establish the connection between the binary MRFs and set functions, since we can view the binary labeling \mathbf{x} as a indicator variables of set S such that $x_i = \llbracket i \in S \rrbracket$. Therefore, the definition of submodularity can be naturally generalized to binary MRFs. Note that the unary terms θ_i is the linear part of the energy function. We will say a binary MRF energy function is sum-of-submodular (SoS) if each of its pairwise term θ_{ij} or higher-order term θ_C is submodular. It is easy to see SoS functions are also submodular functions (Lemma 2.1.3), but the opposite direction is not true. However, in the literature, when people say a binary MRF is submodular, it actually means that the energy function is sum-of-submodular (SoS) in most cases. Therefore, we will follow this convention from the literature.

Definition 2.1.4. We will say the binary MRF energy $f(\mathbf{x}) = \sum_{C \in \mathcal{C}} \theta_C(\mathbf{x}_C)$ is submodular if each term θ_C is submodular.

In particular, for pairwise terms θ_{ij} , submodularity is equivalent to $\theta_{ij}(0, 0) + \theta_{ij}(1, 1) \leq \theta_{ij}(0, 1) + \theta_{ij}(1, 0)$. This means that the pairwise term prefers to have x_i and x_j take the same label. In other words, submodularity corresponds to an *attractive* prior. This is a desired property for most vision applications to enforce spatial coherence. Similarly, supermodular terms corresponds to *repulsive* prior. A repulsive prior also makes sense for certain vision applications, e.g., in semantic segmentation we want to pixels across an object boundary to take different labels.

2.1.2 Equivalent definitions of submodularity

There are multiple equivalent definitions of submodularity besides the one provided in Definition 2.1.1. They are all useful in certain situations.

Theorem 2.1.4 (equivalent definition of submodularity). *The following three statements are equivalent:*

$$f(T + i) - f(T) \leq f(S + i) - f(S), \forall S \subseteq T \subseteq V, i \in V \setminus T, \quad (2.6)$$

$$f(S \cup T) + f(S \cap T) \leq f(S) + f(T), \forall S, T \subseteq V, \quad (2.7)$$

$$f(S) + f(S + i + j) \leq f(S + i) + f(S + j), \forall S \subseteq V, i, j \in V \setminus S, i \neq j. \quad (2.8)$$

Proof. (2.6) \Rightarrow (2.8) is trivial. We just need to let $T = S + j$ and note $S \subseteq S + j$, then we get (2.8) from (2.6) immediately.

For (2.8) \Rightarrow (2.7), first note that when $S \subseteq T$ or $T \subseteq S$, (2.7) holds trivially. Now let's consider $T \setminus S = \{i_1, \dots, i_a\}$ and $S \setminus T = \{j_1, \dots, j_b\}$, and $A_{k,k'} = S \cap T + i_1 +$

$\dots + i_{k-1} + j_1 + \dots + j_{k'-1}$. We can show the following by basic arithmetic.

$$\begin{aligned}
& \sum_{k=1}^a \sum_{k'=1}^b \left[f(A_{k,k'} + i_k + j_{k'}) - f(A_{k,k'} + i_k) - f(A_{k,k'} + j_{k'}) + f(A_{k,k'}) \right] \\
&= \sum_{k=1}^a \sum_{k'=1}^b f(A_{k+1,k'+1}) - \sum_{k=1}^a \sum_{k'=1}^b f(A_{k+1,k'}) - \sum_{k=1}^a \sum_{k'=1}^b f(A_{k,k'+1}) + \sum_{k=1}^a \sum_{k'=1}^b f(A_{k,k'}) \quad (2.9) \\
&= f(A_{a+1,b+1}) - f(A_{a+1,1}) - f(A_{1,b+1}) + f(A_{1,1}) \\
&= f(S \cup T) - f(S) - f(T) + f(S \cap T).
\end{aligned}$$

It is easy to show the first line above is non-positive due to (2.8). Therefore, we have $f(S \cup T) - f(S) - f(T) + f(S \cap T) \leq 0$, which is exactly (2.7).

(2.7) \Rightarrow (2.6) is also trivial, just note that $(S + i) \cap T = S$ and $(S + i) \cup T = T + i$ for $S \subseteq T \subseteq V$ and $i \in V \setminus T$. \square

2.1.3 Generalized submodularity to multilabel MRFs

It is known that the inference problem for binary pairwise submodular MRFs can be reduced to st-MIN CUT problem and solved exactly via max-flow [8]. Therefore, people have wondered how to generalize submodularity to multilabel MRFs and find the characterization of the solvable sub-classes.

Motivated by the equivalent definition of submodularity of binary pairwise MRFs that $\theta_{ij}(0,0) + \theta_{ij}(1,1) \leq \theta_{ij}(0,1) + \theta_{ij}(1,0)$, we define the *multilabel-submodularity* as following.

Definition 2.1.5. Given a total ordering among the label set (\mathcal{L}, \leq) , we define $\min(\mathbf{x}, \mathbf{y})$ and $\max(\mathbf{x}, \mathbf{y})$ to be the element-wise min/max vector based on \leq :

$$\min(\mathbf{x}, \mathbf{y})_i = \begin{cases} x_i, & x_i \leq y_i \\ y_i, & x_i \geq y_i \end{cases}, \quad \max(\mathbf{x}, \mathbf{y})_i = \begin{cases} y_i, & x_i \leq y_i \\ x_i, & x_i \geq y_i \end{cases}. \quad (2.10)$$

Definition 2.1.6 (multilabel-submodularity, [123]). Given a label set \mathcal{L} with total ordering defined over it (\mathcal{L}, \leq) , we say a higher-order term $\theta_C(\mathbf{x}_C)$ is *multilabel-submodular* if

$$\theta_C(\min(\mathbf{x}_C, \mathbf{y}_C)) + \theta_C(\max(\mathbf{x}_C, \mathbf{y}_C)) \leq \theta_C(\mathbf{x}_C) + \theta_C(\mathbf{y}_C). \quad (2.11)$$

Remark. As a special case of Definition 2.1.6, the multilabel pairwise term θ_{ij} is multilabel-submodular when

$$\theta_{ij}(\min(x_i, y_i), \min(x_j, y_j)) + \theta_{ij}(\max(x_i, y_i), \max(x_j, y_j)) \leq \theta_{ij}(x_i, x_j) + \theta_{ij}(y_i, y_j). \quad (2.12)$$

Similarly, we follow the convention that a given MRF energy is multilabel-submodular when it is actually a sum of multilabel-submodular energy. As we will see in Schlesinger and Flach [123], pairwise multilabel-submodular MRFs can be minimized exactly.

Definition 2.1.7. We say that the MRF energy $f(\mathbf{x}) = \sum_{C \in \mathcal{C}} \theta_C(\mathbf{x}_C)$ is a multilabel-submodular MRF energy if each term θ_C is multilabel-submodular.

2.2 Pseudo-Boolean optimization

In this section, we will review the basis of pseudo-Boolean functions (PBFs) and pseudo-Boolean optimization.

2.2.1 Pseudo-Boolean optimization and binary MRF inference

Pseudo-Boolean optimization has been studied in the operation research fields since 1960s [50]. Binary MRF inference problem (i.e., $|\mathcal{L}| = 2$) can be viewed

as a special case of the pseudo-Boolean optimization problem. *Persistency*, the central topic of this thesis, was also first proposed and studied in the context of pseudo-Boolean optimization [49]. Boros and Hammer [12] provide a good review on the history of pseudo-Boolean optimization. We adopt the notations used in [12] in this thesis.

Let $\mathbb{B} := \{0, 1\}$ be the set of Boolean values and \mathbb{R} be the set of real values. A function $f : \mathbb{B}^n \mapsto \mathbb{R}$ is called a *pseudo-Boolean function* (PBF). The *pseudo-Boolean optimization problem* is to minimize the given pseudo-Boolean function.

Definition 2.2.1 (pseudo-Boolean optimization problem, [12, 49]).

$$\min_{\mathbf{x} \in \mathbb{B}^n} f(\mathbf{x}). \quad (2.13)$$

Remark. It is trivial to see that the energy function for any binary MRF is also a pseudo-Boolean function. Therefore, the binary MRF inference problem is just a special case of the pseudo-Boolean optimization problem.

2.2.2 Multilinear polynomial representation

Definition 2.2.2 (multilinear polynomial representation of PBF, [12, 49]). Each pseudo-Boolean function f has a unique multilinear polynomial representation [49]:

$$f(\mathbf{x}) = \sum_{S \subseteq V} c_S \prod_{i \in S} x_i, \quad (2.14)$$

where $\mathbf{x} = (x_1, x_2, \dots, x_n)$.

Definition 2.2.3 (degree of PBF, [12]). The size of largest subset $S \subseteq V$ such that $c_S \neq 0$ is called the *degree* of f , and it is denoted as $\deg(f)$. We will call a PBF f linear (quadratic, cubic, etc) if $\deg(f) \leq 1$ ($\leq 2, 3$, etc.)

Example 2.2.1 (multilinear polynomial representation). Let's say we have two variables x_1, x_2 in our binary MRF. We have the following unary terms and pairwise terms

$$\begin{aligned}\theta_1(0) &= 5, & \theta_1(1) &= 10, \\ \theta_2(0) &= 10, & \theta_2(1) &= 0,\end{aligned}\tag{2.15}$$

$$\theta_{12}(0,0) = 3, \quad \theta_{12}(0,1) = -2, \quad \theta_{12}(1,0) = 0, \quad \theta_{12}(1,1) = 9.$$

We can then rewrite our energy function $f(x_1, x_2)$ as a PBF in the multilinear polynomial as:

$$\begin{aligned}f(x_1, x_2) &= \theta_1(x_1) + \theta_2(x_2) + \theta_{12}(x_1, x_2) \\ &= 5(1 - x_1) + 10x_1 + 10(1 - x_2) + 3(1 - x_1)(1 - x_2) - 2(1 - x_1)x_2 + 9x_1x_2 \\ &= 20 + 2x_1 - 13x_2 + 10x_1x_2.\end{aligned}\tag{2.16}$$

2.2.3 Posiform representation

For the pseudo-Boolean optimization problem, it is easier to consider an alternative representation of PBFs known as *posiforms*. We will define posiforms in this section and defer the connection between posiform representations and pseudo-Boolean optimization to Section 2.4.2. Let $\bar{x}_i = 1 - x_i$ be the *negations* and let $L = \{x_1, \bar{x}_1, \dots, x_n, \bar{x}_n\}$ be the set of *literals*. Then we can define the *posiform* as following:

Definition 2.2.4 (posiform representation of PBF, [12]).

$$\phi(\mathbf{x}) = a_0 + \sum_{T \subseteq L} a_T \prod_{u \in T} u,\tag{2.17}$$

where $a_T \geq 0$ when $T \neq \emptyset$. a_0 is the constant term of the posiform (which might be negative), denoted as $C(\phi)$.

Remark. It is customary to assume $a_T = 0$ when $\exists u$ such that $\{u, \bar{u}\} \subseteq T$, since otherwise we always have $\prod_{u \in T} u = 0$.

Example 2.2.2 (posiform representation, [10]). Here's an example of a PBF with its multilinear polynomial representation and several possible posiform representations.

$$\begin{aligned}
 f(\mathbf{x}) &= -2 - x_1 - x_2 - x_3 + x_1x_2 + x_1x_3 + x_2x_3 && \text{(multilinear polynomial)} \\
 &= -5 + \bar{x}_1 + \bar{x}_2 + \bar{x}_3 + x_1x_2 + x_1x_3 + x_2x_3 && \text{(quadratic posiform)} \\
 &= -4 + \bar{x}_3 + \bar{x}_1\bar{x}_2 + x_1x_3 + x_2x_3 && \text{(quadratic posiform)} \\
 &= -3 + x_1x_2x_3 + \bar{x}_1\bar{x}_2\bar{x}_3 && \text{(cubic posiform)}
 \end{aligned} \tag{2.18}$$

We can learn two things from this example, 1) posiform representation is not unique for the given PBF, 2) the degree of the posiform representation may not necessarily tie with the degree of the PBF.

2.3 Persistency and autarky

Partial optimality, or *persistency*, of the MRF inference problem is the central topic of this thesis. We have already defined *persistency* in Definition 1.6.1, 1.6.2 and 1.6.4. Actually, the concept of persistency was first studied in the context of pseudo-Boolean optimization [49]. It was adopted by the vision community for binary MRF inference [77, 115] in 2007, and then extended to the multilabel MRF inference [71].

Let's recapitulate this important concept.

Definition 2.3.1 (strong persistency). A partial labeling \mathbf{x}_S is strong persistent if

$$\mathbf{x}_S = \mathbf{x}_S^*, \quad \forall \mathbf{x}^* \in \operatorname{argmin}_{\mathbf{x}} f(\mathbf{x}). \tag{2.19}$$

Definition 2.3.2 (weak persistency). A partial labeling \mathbf{x}_S is weak persistent if

$$\exists \mathbf{x}^* \in \operatorname{argmin}_{\mathbf{x}} f(\mathbf{x}), \text{ s.t.}, \mathbf{x}_S = \mathbf{x}_S^*. \quad (2.20)$$

We will primarily focus on strong persistency in this thesis. Therefore, we usually just use the terminology persistency referring to strong persistency. It is trivial to show the following two properties of the strong persistent labels.

Lemma 2.3.1. *If the strong persistent labeling \mathbf{x}_S exists for given S , it must be unique.*

Lemma 2.3.2. *When two partial labeling \mathbf{x}_A and \mathbf{x}_B ($A \cap B = \emptyset$) are both strong persistent, their composition $\mathbf{x}_A \oplus \mathbf{x}_B$ is also strong persistent.*

We can see that persistency is the desired property we want for the MRF inference problem, since it determines the optimal value of a subset of the variables. However, we are facing a chicken and egg problem because the definition of persistent labels requires us to know the global minimizer \mathbf{x}^* in advance. Given the general hardness of the MRF inference problem, it is computationally infeasible to check persistency for a given partial labeling [12]. Therefore, we need to find another property which can be computed without knowledge of the global minimizer(s) \mathbf{x}^* .

The operation research community has also explored this problem and studied a property called *autarky*. This concept is first introduced for Boolean satisfiability (SAT) problem [104], then used by the pseudo-Boolean optimization study [11]. We follow a modern way [14] to define and generalize it from pseudo-Boolean functions to multilabel MRF energy functions as following.

Definition 2.3.3 (autarky, [14]). A partial labeling \mathbf{x}_S is an autarky if

$$f(\mathbf{x}_S \oplus \mathbf{z}_{V \setminus S}) < f(\mathbf{z}), \quad \forall \mathbf{z} \in \mathcal{L}, \text{ s.t. } \mathbf{x}_S \neq \mathbf{z}_S. \quad (2.21)$$

We use $\mathbf{x}_S \oplus \mathbf{z}_{V \setminus S}$ as syntactic sugar to represent substituting a partial labeling \mathbf{x}_S into a full labeling \mathbf{z} . The autarky property claims that the partial labeling \mathbf{x}_S is so appealing that no matter what labels we have for other variables, it is always the best choice for variables indexed by S . We can easily show autarky implies persistency by the following lemma.

Lemma 2.3.3. *Suppose \mathbf{x}_S is an autarky of $f(\mathbf{x})$, then \mathbf{x}_S is also strongly persistent.*

Proof. Suppose \mathbf{x}^* is minimizer of $f(\mathbf{x})$ such that $\mathbf{x}_S^* \neq \mathbf{x}_S$. When we overwrite \mathbf{x}^* using \mathbf{x}_S , since \mathbf{x}_S is an autarky, we must have: $f(\mathbf{x}_S \oplus \mathbf{x}_{V \setminus S}^*) < f(\mathbf{x}_S^* \oplus \mathbf{x}_{V \setminus S}^*) = f(\mathbf{x}^*)$, which is a contradiction. \square

2.4 Persistency algorithm preliminary

We will review the existing persistency algorithms in the literature for MRF inference problem. As we discussed in Section 2.3, due to the hardness of checking persistency directly, all existing persistency algorithms appear to check the autarky property as a sufficient condition. As a reminder, autarky states that overwriting an arbitrary labeling with this partial labeling will reduce the energy.

Since all of these algorithms check autarky instead of persistency, it is guaranteed that all the existing persistency algorithms will never wrongly label a variable. Therefore, the trade-off between these methods is the speed versus coverage, i.e., the overall running time versus the percentage of the variables the algorithm can label. We summarize the relationships among these methods in Figure 2.1, then we will present the high-level idea of these methods in the

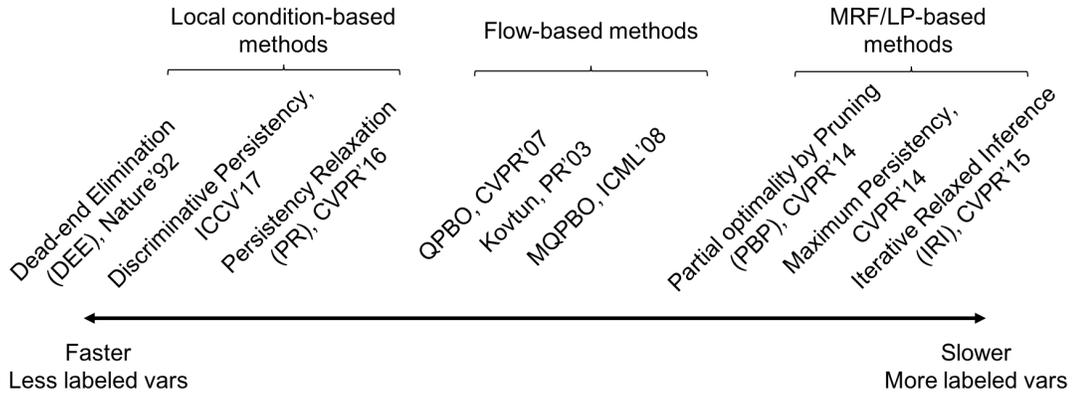


Figure 2.1: Overview of persistency algorithms.

ascending order according to their speed.

2.4.1 Dead-end elimination (DEE) and variants

Dead-end elimination (DEE) [31] and its variants [44, 97, 108, 141] are the fastest persistency algorithms in the literature. DEE checks the local conditions as a further relaxation of the autarky property, which can be examined very efficiently. But these methods cannot label too many variables in general. This line of research originally studied the specific protein side-chain structure prediction problem in the bio-chemistry community, which is a special case of MRF inference. Therefore, it did not draw the attention of the computer vision community until very recently [126, 145].

The original idea of DEE [31] is simple. It rules out $x_i = \ell$ to be persistent for a single variable when $\exists \ell' \in \mathcal{L}_i, \ell' \neq \ell$ such that:

$$\theta_i(\ell) + \sum_{j:(i,j) \in E} \min_{x_j \in \mathcal{L}_j} \theta_{ij}(\ell, x_j) > \theta_i(\ell') + \sum_{j:(i,j) \in E} \max_{x_j \in \mathcal{L}_j} \theta_{ij}(\ell', x_j). \quad (2.22)$$

The meaning behind this condition is label $x_i = \ell'$ is always a better choice than label $x_i = \ell$ even when all the adjacent pairwise terms around variable x_i takes

their best possible (minimum) energy value with $x_i = \ell$ and the worst possible (maximum) energy value with $x_i = \ell'$. This is a straightforward relaxation of the autarky property. Then DEE algorithm computes the persistent partial labeling via iteratively testing this condition over combinations of variables and labels and at each step eliminates impossible labels.

Goldstein [44] noticed that we can trivially get a tighter relaxation since it is unnecessary to allow x_j to take different values on the LHS and RHS in (2.22). Therefore, we have the following revised Goldstein condition:

$$\theta_i(\ell) - \theta_i(\ell') + \sum_{j:(i,j) \in E} \min_{x_j \in \mathcal{L}_j} (\theta_{ij}(\ell, x_j) - \theta_{ij}(\ell', x_j)) > 0. \quad (2.23)$$

The first term $\theta_i(\ell) - \theta_i(\ell')$ is the energy increment when we flip the label from ℓ' to ℓ for x_i , while $\min_{x_j \in \mathcal{L}_j} (\theta_{ij}(\ell, x_j) - \theta_{ij}(\ell', x_j))$ is the minimal possible energy increment for each adjacent pairwise term. We can prove $x_i = \ell$ cannot be persistent if this energy change is still strictly positive in the best possible situation. Since the Goldstein condition (2.23) is a no-brainer upgrade of the original DEE condition (2.22), when we mention the DEE algorithm in this paper, we refer to the DEE algorithm with Goldstein condition by default.

Voigt *et al.* [141] generalized Goldstein condition to eliminate a pair of labels ($x_i = r, x_j = t$) simultaneously when $\exists(x_i = u, x_j = v)$ such that:

$$\mathcal{E}_{ij}(r, s) - \mathcal{E}_{ij}(u, v) + \sum_{k:k \neq i,j} \min_{x_k \in \mathcal{L}_k} (\mathcal{E}_{ijk}(r, s, x_k) - \mathcal{E}_{ijk}(u, v, x_k)) > 0, \quad (2.24)$$

where $\mathcal{E}_{ij}(x_i, x_j) := \theta_i(x_i) + \theta_j(x_j) + \theta_{ij}(x_i, x_j)$ is the combined energy function for x_i and x_j , and $\mathcal{E}_{ijk}(x_i, x_j, x_k) = \mathcal{E}_{ik}(x_i, x_k) + \mathcal{E}_{jk}(x_j, x_k)$.

2.4.2 Quadratic pseudo-Boolean optimization (QPBO)

The operation research community has explored persistency for quadratic pseudo-Boolean optimization (QPBO) for decades [49] (see more recent results at [12]). These results were introduced to the computer vision community and applied to general binary MRF inference in 2007 [77, 115]. This category of methods rely on solving max-flow on an auxiliary flow network, which is twice as large¹. Then this underlying graph is used to compute persistency. We will review the main results in this section.

We have introduced in Definition 2.2.4 the posiform representation of pseudo-Boolean functions. It has a strong connection with the pseudo-Boolean optimization problem due to the following two lemmas.

Lemma 2.4.1 ([12]). *Suppose ϕ is a posiform representation of a pseudo-Boolean function f , then we have $C(\phi) \leq \min_{\mathbf{x}} f(\mathbf{x})$.*

Lemma 2.4.2 ([12]). *Every pseudo-Boolean function f can be represented as a posiform ϕ such that $C(\phi) = \min_{\mathbf{x}} f(\mathbf{x})$.*

Lemma 2.4.1 says the constant term of arbitrary posiform of the given PBF can serve as the lower bound of the optimization problem. Lemma 2.4.2 further claims this lower bound can reach the optimum value. So the pseudo-Boolean optimization problem can be reduced to finding the posiform representation and maximizing the constant term.

¹Suppose $\mathcal{G} = (V, E)$ as the underlying graph of the MRF. So we need to construct an auxiliary flow network with $2|V| + 2$ vertices and $2|V| + 2|E|$ edges. For binary submodular MRFs, the scale of the auxiliary flow network can be reduced to $|V| + 2$ vertices and $2|V| + |E|$ edges due to the symmetricity of the auxiliary flow network. Note that the flow network is the same one used to show the equivalence between the binary submodular MRF inference to the st-MINCUT problem.

However, finding the best posiform representation is still a computational intractable problem, even when f is a quadratic PBF (i.e., $\deg(f) = 2$). However, finding the best quadratic posiform representation and maximizing the constant term is tractable via a linear programming and we can show the characterization of the strong persistency based on the LP results [12, 49].

Given quadratic PBF $f(\mathbf{x}) = \sum_{S \subseteq V} c_S \prod_{i \in S} x_i$, we want to compute $C_2(f) := \max_{\phi \in \mathcal{P}_2(f)} C(\phi)$, where $\mathcal{P}_2(f)$ is the set of all the possible quadratic posiforms of f . We can solve it via the following LP:

$$\begin{aligned}
\max_a \quad & c_0 - \sum_{j=1}^n a_{\bar{x}_j} - \sum_{1 \leq i < j \leq n} a_{\bar{x}_i \bar{x}_j} \\
\text{s.t.} \quad & a_{x_j} - a_{\bar{x}_j} + \sum_{1 \leq i \leq n, i \neq j} (a_{\bar{x}_i x_j} - a_{\bar{x}_i \bar{x}_j}) = c_j, \quad \forall j = 1, \dots, n \\
& a_{x_i x_j} + a_{\bar{x}_i \bar{x}_j} - a_{\bar{x}_i x_j} - a_{x_i \bar{x}_j} = c_{ij}, \quad \forall 1 \leq i < j \leq n, \\
& a_u \geq 0, a_{uv} \geq 0, \quad \forall u, v \in L, u \neq v,
\end{aligned} \tag{2.25}$$

where we want to optimize over the posiform coefficients a_T , the objective here is the constant term of the posiform $C(\phi)$, the first two constraints guarantee what we get is equivalent to f , and the last constraint guarantees it is a posiform.

Theorem 2.4.3 (strong persistency characterization, QPBO [12, 49]). *Given a quadratic pseudo-Boolean function f , let $\phi \in \mathcal{P}_2(f)$ be the optimal quadratic posiform representing f such that $C(\phi) = C_2(f)$. Then we have literal $u = 0$ to be a strong persistent label minimizing f if $a_u > 0$.*

Theorem 2.4.3 provides a polynomial time algorithm to compute strong persistency for quadratic PBFs. There are also more efficient combinatorial approaches to achieve the same goal, especially the network flow based approaches [13].

Given quadratic pseudo-Boolean function f as a quadratic posiform representation $\phi \in \mathcal{P}_2(f)$, we can define a directed flow network $G_\phi(N, A)$, where the vertex set $N = L \cup \{x_0, \bar{x}_0\}$, with $x_0 = 1$ and $\bar{x}_0 = 0$ as the source and sink of the flow network. For every quadratic term $a_{uv}uv$, we let the capacity of arc (u, \bar{v}) and (v, \bar{u}) to be $\frac{1}{2}a_{uv}$. For the linear term $a_u u$, it can be viewed as $a_u u x_0$ now, so we define the capacity of arc (u, \bar{x}_0) and x_0, \bar{u} to be $\frac{1}{2}a_u$.

Conversely, given any directed flow network $G = (N, A)$, where $N = L \cup \{x_0, \bar{x}_0\}$, and with non-negative capacities c_{uv} assigned with arc $(u, v) \in A$, we can define the following quadratic posiform:

$$\phi_G := \sum_{(u,v) \in A} c_{uv} u \bar{v}. \quad (2.26)$$

Note that ϕ_G is a posiform because all the arcs entering x_0 or leaving \bar{x}_0 must vanish, since $u \bar{x}_0 = \bar{x}_0 v = 0$.

Therefore, we have the following lemma.

Lemma 2.4.4 ([12]). *There is a one-to-one correspondence mapping between quadratic posiforms $\phi \in \mathcal{P}_2(f)$ for which $C(\phi) = 0$ and capacitated directed flow network $G = (N, A)$ with vertex set $N = L \cup \{x_0, \bar{x}_0\}$. Furthermore, the involution $G_{\phi_G} = G$ and $\text{phi}_{G_\phi} = \phi$ holds.*

Boros *et al.* [13] further proves the following relationship to compute $C_2(f)$ and strong persistency.

Theorem 2.4.5 ([13]). *Given a quadratic pseudo-Boolean function f , and a posiform representation $\phi \in \mathcal{P}_2(f)$, let's denote by v^* the maximum flow value in the network G_ϕ . Then we have:*

$$C_2(f) = C(\phi) + v^*. \quad (2.27)$$

Theorem 2.4.6 (strong persistency characterization, QPBO (max-flow) [13]). *Let $\phi \in \mathcal{P}_2(f)$ for a quadratic pseudo-Boolean function f , let φ^* denote a maximum flow in G_ϕ , and let $S \subseteq L$ denote the set of vertices of G_ϕ which are reachable from x_0 via a path with positive residual capacities. Then $u = 1$ is strong persistent label minimizing f for $u \in S$.*

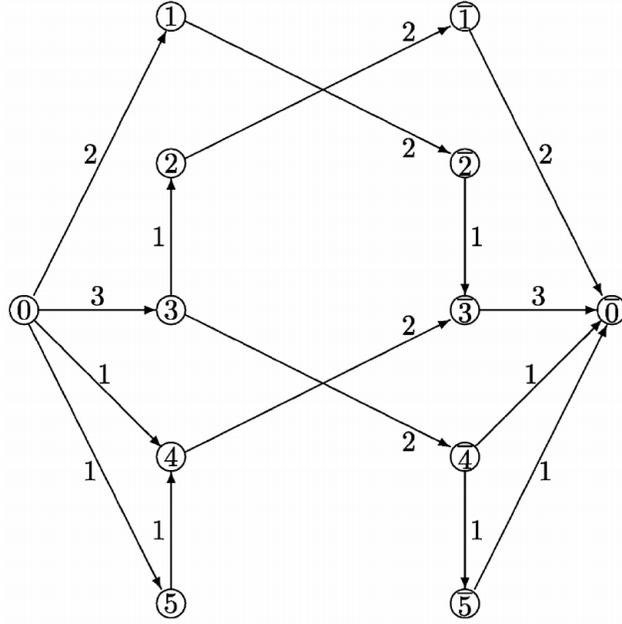


Figure 2.2: The flow network G_ϕ corresponding to the posiform ϕ in Example 2.4.1 [12] © Elsevier.

Example 2.4.1 (strong persistency via max-flow methods, [12]). We will use an example presented in Boros and Hammer [12] to illustrate the max-flow approach to compute strong persistency. Let's say we have the following QPBF f with one quadratic posiform representation ϕ .

$$f(x_1, x_2, x_3, x_4, x_5) = 10 - 4x_1 - 4x_3 - 2x_4 + 4x_1x_2 - 2x_2x_3 + 4x_3x_4 - 2x_4x_5, \quad (2.28)$$

$$\phi = -4 + 4\bar{x}_1 + 6\bar{x}_3 + 2\bar{x}_4 + 2\bar{x}_5 + 4x_1x_2 + 2\bar{x}_2x_3 + 4x_3x_4 + 2\bar{x}_4x_5. \quad (2.29)$$

Then we have construct of the corresponding flow network G_ϕ illustrated in Figure 2.2 and after computing the max-flow on G_ϕ with flow value $v^* = 6$, we

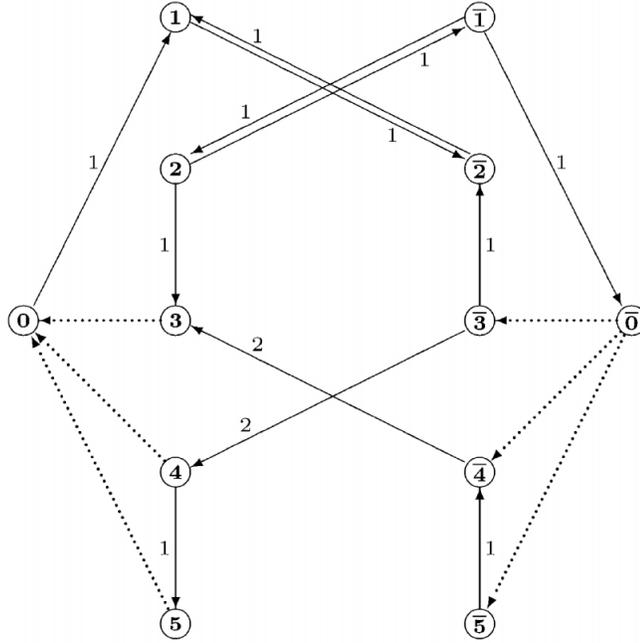


Figure 2.3: Residual network in Example 2.4.1 [12] © Elsevier.

have the residual network illustrated in Figure 2.3, which corresponds to the following quadratic posiform ψ .

$$\psi = 2\bar{x}_1 + 2x_1x_2 + 2\bar{x}_1\bar{x}_2 + 2x_2\bar{x}_3 + 4\bar{x}_3\bar{x}_4 + 2x_4\bar{x}_5. \quad (2.30)$$

Therefore, we have $C_2(f) = C(\phi) + v^* = 2$ and $2 + \psi \in \mathcal{P}_2(f)$. We can see the vertex x_1 and \bar{x}_2 is reachable from the source x_0 , so we have $x_1 = 1$ and $x_2 = 0$ to be strong persistent in our example.

2.4.3 Kovtun's algorithm

Kovtun's algorithm [83, 84, 85] tries to address the persistency problem for multilabel pairwise MRFs. The original Kovtun's algorithm was proposed in Kovtun [83]. Then an iterative version was published in Kovtun's PhD thesis in Ukrainian [84]. Shekhovtsov and Hlavac [128] translated the results from [84]

into English, and [85] is the latest publication on this algorithm. We will review the main result of the original Kovtun algorithm here [83], heavily borrowing the notations from Shekhovtsov and Hlavac [128].

Kovtun's algorithm only handles the multilabel-submodular MRFs. It constructs L ($L := |\cup_{i \in V} \mathcal{L}_i|$ is the number of all labels) auxiliary binary MRFs in the one-versus-others flavor to prove whether one particular label is better than all the others. So the computational cost of Kovtun's algorithm is L times the running time of QPBO.

Definition 2.4.1 (auxiliary binary problems in Kovtun [83, 128]). Given the energy function $f(\mathbf{x}) = \sum_{i \in V} \theta_i(x_i) + \sum_{(i,j) \in E} \theta_{ij}(x_i, x_j)$, and one particular label $\ell \in \mathcal{L}$, we can construct the auxiliary binary MRF $g_\ell(\mathbf{y}) = \sum_{i \in V} \eta_i(y_i) + \sum_{(i,j) \in E} \eta_{ij}(y_i, y_j)$ where

$$\begin{aligned} \eta_i(1) &:= \theta_i(\ell), \\ \eta_i(0) &:= \min_{x_i \neq \ell} \theta_i(x_i), \end{aligned} \tag{2.31}$$

and

$$\begin{aligned} \eta_{ij}(1, 1) &:= \theta_{ij}(\ell, \ell), \\ \eta_{ij}(1, 0) &:= \min_{x_j \neq \ell} \theta_{ij}(\ell, x_j), \\ \eta_{ij}(0, 1) &:= \min_{x_i \neq \ell} \theta_{ij}(x_i, \ell), \\ \eta_{ij}(1, 1) &:= \min \left\{ \eta_{ij}(0, 1) + \eta_{ij}(1, 0) - \eta_{ij}(1, 1), \right. \\ &\quad \left. \min_{x_i \neq \ell, x_j \neq \ell} \left[\theta_{ij}(x_i, x_j) + \min\{\eta_{ij}(1, 0) - \theta_{ij}(\ell, x_j), \eta_{ij}(0, 1) - \theta_{ij}(x_i, \ell)\} \right] \right\}. \end{aligned} \tag{2.32}$$

Then we have the following persistency characterization theorem.

Theorem 2.4.7 (strong persistency characterization, Kovtun [83, 128]). $x_i = \ell$ is

strong persistent for the multilabel-submodular MRF $f(\mathbf{x})$ if $y_i = 1$ is strong persistent for the auxiliary binary problem $g_\ell(\mathbf{y})$ defined in Definition 2.4.1.

2.4.4 Multilabel QPBO (MQPBO)

Multilabel QPBO (MQPBO) [71] computes persistency for multilabel MRFs in a different way than Kovtun's methods. It transforms the multilabel MRFs into a binary MRF with $O(NL)$ variables and $O(NL + NL^2)$ pairwise terms, then solves the persistency for the induced binary MRF by QPBO, and finally proves the persistency of the original multilabel MRF from results of the binary MRF.

The equivalence between the multilabel MRFs and binary MRFs have been studied in the literature for different purposes [12, 55, 123], which has been adopted by MQPBO. The high-level idea of the transformation is for each variables x_i with $|\mathcal{L}_i|$ labels, we introduce $|\mathcal{L}_i|-1$ binary variables $z_{i,\ell}$, and a one-to-one mapping between \mathbf{x} and \mathbf{z} such that $x_i = \sum_\ell z_{i,\ell}$ and the first $|x_i|$ entries in \mathbf{z}_i is 1, and 0 for the remaining entries in \mathbf{z}_i ². Then we can construct the binary energy function $g(\mathbf{z} | \eta)$ such that $g(\mathbf{z}(\mathbf{x}) | \eta) = f(\mathbf{x} | \theta)$ and $g(\mathbf{z} | \eta) < \infty$ if and only if each \mathbf{z}_i has a valid labeling can be maps to a valid $x_i \in \mathcal{L}_i$. In other words, $g(\mathbf{z})$ and $f(\mathbf{x})$ are equivalent in the sense that the function value matches for the corresponding \mathbf{x} and \mathbf{z} and $g(\mathbf{z}) = \infty$ when the given \mathbf{z} is infeasible (i.e., cannot be mapped to a valid \mathbf{x} value). Please refer to [71, 123] for more details about this construction.

Schlesinger *et al.* [123] proves that when the original energy function f is multilabel-submodular, the induced binary MRF g is also submodular, hence we

² $x_i = 0$ maps to $\mathbf{z}_i = (0, 0, 0, \dots, 0)$, $x_i = 1$ maps to $\mathbf{z}_i = (1, 0, 0, \dots, 0)$, $x_i = 2$ maps to $\mathbf{z}_i = (1, 1, 0, \dots, 0)$, \dots , $x_i = |\mathcal{L}| - 1$ maps to $\mathbf{z}_i = (1, 1, 1, \dots, 1)$.

can solve the inference problem exactly via max-flow/min-cut. In the MQPBO method [71], we can apply QPBO over the induced binary MRF g to compute persistency of \mathbf{z} . Then we have the following strong persistency characterization method³.

Theorem 2.4.8 (strong persistency characterization, MQPBO [71]). *When we can prove strong persistency of the $|\mathcal{L}_i| - 1$ binary variables \mathbf{z}_i corresponding to variable x_i in the induced binary MRF $g(\mathbf{z})$, then $x_i = \sum_{\ell} z_{i,\ell}$ is strong persistent for the original multilabel MRF $f(\mathbf{x})$.*

2.4.5 Partial optimality by Pruning (PBP)

Swoboda *et al.* [134] proposed the partial optimality by pruning (PBP) methods to compute persistency for multilabel MRFs. It is known from the pseudo-Boolean optimization research that when we solve the LP relaxation of the MRF inference on the local polytope for the binary MRFs, all the integer variables in the optimum solution will be persistent. However, this claim is not true for multilabel MRFs [49]. Therefore, in this paper, Swoboda *et al.* asked the natural question: what’s the criteria that the optimum integral solution of the LP relaxation will coincide with the persistent labeling of the original multilabel MRF inference problem? Although PBP can handle higher-order MRFs, we will only introduce its idea for pairwise MRFs here for simplicity.

It is known that the multilabel pairwise MRF inference problem can be formalized as an integer linear programming (ILP) problem, and the most widely

³Actually, the original theorem in MQPBO can also be used to eliminate non-persistent labels. The version describes in this thesis is a special version that we can prove persistency when we can eliminate $|\mathcal{L}_i| - 1$ other labels. Please refer to Statement 1 in [71].

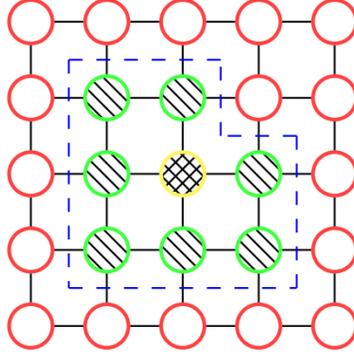


Figure 2.4: Illustration of boundary and interior [134] © IEEE. Set A is given by the nodes in the dashed blue regions. All the green nodes with diagonal pattern are boundary nodes, the yellow nodes with crosshatch pattern is the interior node and all the red nodes are exterior nodes. All the edges crossing the blue dashed line are the boundary edges.

studied LP relaxation is called *local marginal polytope* [142].

$$\begin{aligned}
\min_{\mu} \quad & f^{\wedge}(\mu) := \sum_{i \in V} \sum_{x_i \in \mathcal{L}_i} \theta_i(x_i) \mu_i(x_i) + \sum_{(i,j) \in E} \sum_{x_i \in \mathcal{L}_i, x_j \in \mathcal{L}_j} \theta_{ij}(x_i, x_j) \mu_{ij}(x_i, x_j) \\
\text{s.t.} \quad & \sum_{x_i \in \mathcal{L}_i} \mu_i(x_i) = 1, \forall i \in V \\
& \sum_{x_i \in \mathcal{L}_i} \mu_{ij}(x_i, x_j) = \mu_j(x_j), \forall x_j \in \mathcal{L}_j, (i, j) \in E \\
& \sum_{x_j \in \mathcal{L}_j} \mu_{ij}(x_i, x_j) = \mu_i(x_i), \forall x_i \in \mathcal{L}_i, (i, j) \in E \\
& \mu_i(x_i) \geq 0, \forall i \in V, x_i \in \mathcal{L}_i \\
& \mu_{ij}(x_i, x_j) \geq 0, \forall (i, j) \in E, x_i \in \mathcal{L}_i, x_j \in \mathcal{L}_j,
\end{aligned} \tag{2.33}$$

where we will define Λ_V to be the *local polytope* defined by the constraints in (2.33), and Λ_A for $A \subseteq V$ similarly.

PBP studies the persistency criteria of the boundary variables for the given subset $A \subseteq V$, which is defined as following and illustrated in Figure 2.4.

Definition 2.4.2 (boundary and interior, [134]). For the set $A \subseteq V$, the set $\partial V_A := \{u \in A \mid \exists v \in V \setminus A, s.t. (u, v) \in E\}$ is called the *boundary*. The set $\partial E_A := \{(u, v) \in E \mid$

$u \in A, v \in V \setminus A$ is called the *boundary edges*. The set $A \setminus \partial V_A$ is called the *interior* of A .

PBP defines the boundary energy in the way that we have the worst energy for all labelings conforming to a given boundary labeling \mathbf{y} , and make the energy more favourable for all other labelings not confirming to \mathbf{y} :

Definition 2.4.3 (boundary energy, [134]). For a set $A \subseteq V$ and a boundary labeling $\mathbf{y} \in \mathcal{L}_{\partial V_A}$, we can define the boundary energy term $\hat{\theta}_{ij, y_i} : \mathcal{L}_i \mapsto \mathbb{R}$ as follows:

$$\hat{\theta}_{ij, y_i}(x_i) := \begin{cases} \max_{x_j \in \mathcal{L}_j} \theta_{ij}(x_i, x_j), y_i \neq x_i \\ \min_{x_j \in \mathcal{L}_j} \theta_{ij}(x_i, x_j), y_i = x_i \end{cases}. \quad (2.34)$$

Then we can defined the boundary energy $\hat{E}_{A, \mathbf{y}}(\mathbf{x})$ as:

$$\hat{f}_{A, \mathbf{y}}(\mathbf{x}) := \sum_{i \in A} \theta_i(x_i) + \sum_{(i, j) \in E, i, j \in A} \theta_{ij}(x_i, x_j) + \sum_{(i, j) \in \partial E_A, i \in A} \hat{\theta}_{ij}(x_i). \quad (2.35)$$

Given this extreme definition of the boundary energy, PBP proves the following persistency condition as a generalization of the autarky property that the given partial labeling is persistent if it coincides with the optimum solution of the boundary energy, and the integral optimum solution of its LP relaxation (2.33), which is more tractable to check.

Theorem 2.4.9 (weak persistency characterization, PBP [134]). *A partial labeling \mathbf{x}_A is weak persistent if*

$$\mathbf{x}_A \in \operatorname{argmin}_{\mathbf{x}_A \in \mathcal{L}_A} \hat{f}_{A, \mathbf{x}_A}(\mathbf{x}). \quad (2.36)$$

Corollary 2.4.10 (tractable weak persistency characterization, PBP [134]). *Given partial labeling \mathbf{x}_A and its marginals $\mu_A \in \Lambda_A$ such that $\mu_i(x_i) = 1, \forall i \in A$ (i.e., μ_A is integral). Then we have \mathbf{x}_A is weak persistent if*

$$\mu_A \in \operatorname{argmin}_{\mu \in \Lambda_A} \hat{f}_{A, \mathbf{x}_A}^\Lambda(\mu). \quad (2.37)$$

Finally, PBP adopts the shrinking scheme. It starts with $A = V$, checks the persistency condition in Corollary 2.4.10 by solving the corresponding LP relaxation, removes the variables with non-integer optimum LP solution from A , and then repeats. It can be shown this shrinking scheme gives us the largest persistent partial labeling identifiable by Corollary 2.4.10 [134]. However, we can also see that this algorithm requires to solve the LP-relaxation or using approximate MRF inference technique (which can show optimality from the integrality, e.g., dual decomposition) to solve the auxiliary boundary energies for multiple times. Therefore, although it can effectively find a much larger persistent partial labeling than Kovtun's method or MQPBO, it usually needs to run for extremely long time.

2.4.6 Maximum persistency

Shekhovtsov [126] proposed the maximum persistency idea, which introduced a new idea called *improving mapping*, which generalizes the autarky property. Then it formalizes the problem to find the best improving mapping to get the maximum persistency as a gigantic linear programming. Therefore, although it may find a very large persistent labeling in practice, it is very time consuming.

Definition 2.4.4 (improving mapping, [126]). A mapping $p : \mathcal{L} \mapsto \mathcal{L}$ is called *weakly improving* if

$$f(p(\mathbf{x})) \leq f(\mathbf{x}), \forall \mathbf{x} \in \mathcal{L}, \quad (2.38)$$

and *strictly improving* if

$$f(p(\mathbf{x})) < f(\mathbf{x}), \forall \mathbf{x} \in \mathcal{L}, p(\mathbf{x}) \neq \mathbf{x}. \quad (2.39)$$

It is trivial to see that the autarky property is a special case of the strictly improving. Given \mathbf{x}_A as an autarky, we can set $p(x_i) = \mathbf{x}_A(i)$ for $i \in A$ and $p(x_i) = x_i$ otherwise. Without loss of generality, we also assume p is idempotent, i.e., $p(p(\mathbf{x})) = \mathbf{x}$. We can easily show the following condition to rule out non-persistent labels.

Lemma 2.4.11 ([126]). *When p is a weakly improving pixel-wise idempotent mapping, then there exists an optimum solution \mathbf{x}^* such that:*

$$p_i(x_i) \neq x_i \Rightarrow x_i^* \neq x_i, \forall i \in V. \quad (2.40)$$

In the case p is a strictly improving mapping, all optimum solution \mathbf{x}^ satisfies (2.40).*

We can also take the LP relaxation of the MRF inference problem using the local marginal polytope introduced in (2.33) and define the relaxed improving mapping on the marginal variables μ instead of \mathbf{x} .

Definition 2.4.5 ([126]). A linear extension of p is a linear mapping P satisfies:

$$\mu(p(\mathbf{x})) = P(\mu(\mathbf{x})), \forall \mathbf{x} \in \mathcal{L}. \quad (2.41)$$

In particular, for pixel-wise mapping p , we can define the $|\mathcal{L}_i| \times |\mathcal{L}_i|$ matrix P_i such that $P_i(a, b) := \llbracket a = p_i(b) \rrbracket$, and define the linear extension $P = [p]$ as another large matrix that⁴:

$$\begin{aligned} (P\mu)_i(x_i) &:= P_s \mu_i(x_i), \\ (P\mu)_{ij}(x_i, x_j) &:= P_s \mu_{ij}(x_i, x_j) P_i^\top. \end{aligned} \quad (2.42)$$

Therefore, we can define the relaxed improving mapping similarly for P .

⁴I omitted the constant term in [126] due to we don't include the constant term in our MRF energy definition.

Definition 2.4.6 (relaxed improving mapping, [126]). A mapping P is called *weakly Λ -improving* if

$$f(p(\mathbf{x})) = \langle f^\Lambda, \mu(p(\mathbf{x})) \rangle = \langle f^\Lambda, P(\mu(\mathbf{x})) \rangle \leq \langle f^\Lambda, \mu(\mathbf{x}) \rangle = f(\mathbf{x}), \forall \mathbf{x} \in \mathcal{L}, \quad (2.43)$$

and *strictly Λ -improving* if

$$f(p(\mathbf{x})) = \langle f^\Lambda, \mu(p(\mathbf{x})) \rangle = \langle f^\Lambda, P(\mu(\mathbf{x})) \rangle < \langle f^\Lambda, \mu(\mathbf{x}) \rangle = f(\mathbf{x}), \forall \mathbf{x} \in \mathcal{L}, p(\mathbf{x}) \neq \mathbf{x}, \quad (2.44)$$

where $\langle \cdot \rangle$ is the inner product operator.

Shekhovtsov [126] denoted the space of weakly (resp., strictly) Λ -improving of energy function f defined in Definition 2.4.6 as \mathbb{W}_f (resp., \mathbb{S}_f), which are both convex. Therefore, checking the equations in Definition 2.4.6 can be performed by solving:

$$\min_{\mu \in \Lambda_V} \langle (I - P)^\top f, \mu \rangle, \quad (2.45)$$

and checks whether the result is non-negative (resp., positive) in polynomial time.

Finally, Shekhovtsov [126] formalized the maximum persistency problem as the following programming, which eliminates the most non-persistent labels:

$$\max_p \sum_{i, x_i \in \mathcal{L}_i} \llbracket p_i(x_i) \neq i \rrbracket, \text{ s.t.}, [p] \in \mathbb{W}_f \text{ (or } \mathbb{S}_f \text{)}. \quad (2.46)$$

Shekhovtsov [126] also showed when p is a pixel-wise *subset-to-one* mapping (i.e., some variables maps to one particular label ℓ , while others remain the same label), we can solve (2.46) exactly by introducing slack variables and rewrite (2.46) as an LP, which is guaranteed to give us integral optimum solutions.

2.4.7 Maximum persistency via iterative relaxed inference (IRI)

The maximum persistency via iterative relaxed inference (IRI) [129] is an integration of the PBP method and the maximum persistency method, which takes the advantage of both.

This approach replaces the persistency criteria used in PBP (Corollary 2.4.10) by the more general condition relaxed improving mapping (Definition 2.4.6). Therefore, IRI can find more persistent variables compared to PBP. In the subroutine to check the feasibility of the relaxed improving mapping for subset-to-one pixel-wise mapping, instead of solving the gigantic LP proposed in maximum persistency [126], IRI adopts the idea from PBP to view that LP problem as an auxiliary MRF inference problem again, and solves it approximately via TRWS [73]. This speeds up the whole algorithm a lot compared to solving the LP by simplex algorithm.

This method is the state-of-the-arts persistency algorithm for proving a very large persistent partial labeling for the given MRFs. However, since it still needs to solve the large-scale auxiliary LPs/MRFs multiple times in the algorithm, it is slow for certain large MRF energies.

2.5 Graphcuts algorithm and move-making techniques

Graphcuts algorithm [18] and its extensions are the most successful and most widely used approximate MRF inference algorithms. They have both a nice theoretical guarantee and strong empirical performance compared to other approximate inference algorithms [63, 136]. Therefore, we will use graphcuts algo-

rithm as the approximate inference algorithm in conjunction with our proposed persistency algorithm for evaluation in this thesis. We will introduce the basis of the graphcuts algorithm in this section. We will get started from the most essential binary pairwise submodular MRFs, which can be solved exactly via max-flow/min-cut in Section 2.5.1. Then we will relax the three constraints we have here to handle multilabel MRFs, non-submodular MRFs, and higher-order MRFs in Section 2.5.2, 2.5.3, and 2.5.4 respectively.

2.5.1 Binary pairwise submodular MRFs and st-MIN CUT

It is well known since 1986 that the inference problem of binary pairwise submodular MRFs can be reduced to st-MIN CUT problem, hence it can be solved exactly via max-flow algorithms [8]. However, at the time, this result didn't attract interest from the computer vision community since: 1) binary MRFs were too restrictive for many vision applications, 2) this work was published in a statistics journal. Despite these facts, this reduction is the key subroutine of graphcuts algorithm and its variants. Therefore, we will present this key reduction in this subsection, with the modern notations and reparameterization technique from Kolmogorov and Zabih [79].

Definition 2.5.1 (reparameterization). Given two pairwise MRFs $f(\mathbf{x} \mid \theta) = \sum_{i \in V} \theta_i(x_i) + \sum_{(i,j) \in E} \theta_{ij}(x_i, x_j)$ and $f'(\mathbf{x} \mid \theta') = \sum_{i \in V} \theta'_i(x_i) + \sum_{(i,j) \in E} \theta'_{ij}(x_i, x_j)$. We say f' is a *reparameterization* of f if $f(\mathbf{x}) = f'(\mathbf{x}) + C, \forall \mathbf{x} \in \mathcal{L}$, where C is a fixed constant. We will denote it as $f \sim f'$.

Remark. It is obvious that the MRF inference problem is invariant under reparameterization, i.e., $\operatorname{argmin}_{\mathbf{x}} f(\mathbf{x}) = \operatorname{argmin}_{\mathbf{x}} f'(\mathbf{x})$ when $f \sim f'$.

Theorem 2.5.1 ([79]). *Given any binary pairwise submodular MRF $f(\mathbf{x} \mid \theta)$, there exists $f'(\mathbf{x} \mid \theta') \sim f(\mathbf{x} \mid \theta)$ such that for all $i \in V$, we have*

$$\min(\theta_i(0), \theta_i(1)) = 0, \quad (2.47)$$

$$\max(\theta_i(0), \theta_i(1)) \geq 0, \quad (2.48)$$

and for all $(i, j) \in E$, we have:

$$\theta'_{ij}(0, 0) = \theta'_{ij}(1, 0) = \theta'_{ij}(1, 1) = 0, \quad (2.49)$$

$$\theta'_{ij}(0, 1) \geq 0. \quad (2.50)$$

Proof. We can prove by construction.

For any given submodular term θ_{ij} , we can do the following transformation:

$$\begin{aligned} \theta_{ij} &= \begin{array}{|c|c|} \hline \theta_{ij}(0, 0) & \theta_{ij}(0, 1) \\ \hline \theta_{ij}(1, 0) & \theta_{ij}(1, 1) \\ \hline \end{array} = \begin{array}{|c|c|} \hline a & b \\ \hline c & d \\ \hline \end{array} \\ &= a + \begin{array}{|c|c|} \hline 0 & 0 \\ \hline c-a & c-a \\ \hline \end{array} + \begin{array}{|c|c|} \hline 0 & d-c \\ \hline 0 & d-c \\ \hline \end{array} + \begin{array}{|c|c|} \hline 0 & b+c-a-d \\ \hline 0 & 0 \\ \hline \end{array}, \end{aligned} \quad (2.51)$$

where a is a constant term we can omit, the second and third term could be viewed as a unary term and the last term is the desired new pairwise term.

Therefore, we can define:

$$\begin{aligned} \theta''_i(0) &:= \theta_i(0) \\ \theta''_i(1) &:= \theta_i(1) + \sum_{j:(i,j) \in E} [\theta_{ij}(1, 0) - \theta_{ij}(0, 0)] + \sum_{j:(j,i) \in E} [\theta_{ji}(1, 1) - \theta_{ji}(1, 0)] \\ \theta'_{ij}(0, 0) &:= 0, \\ \theta'_{ij}(0, 1) &:= \theta_{ij}(0, 1) + \theta_{ij}(1, 0) - \theta_{ij}(0, 0) - \theta_{ij}(1, 1), \\ \theta'_{ij}(1, 0) &:= 0, \\ \theta'_{ij}(1, 1) &:= 0. \end{aligned} \quad (2.52)$$

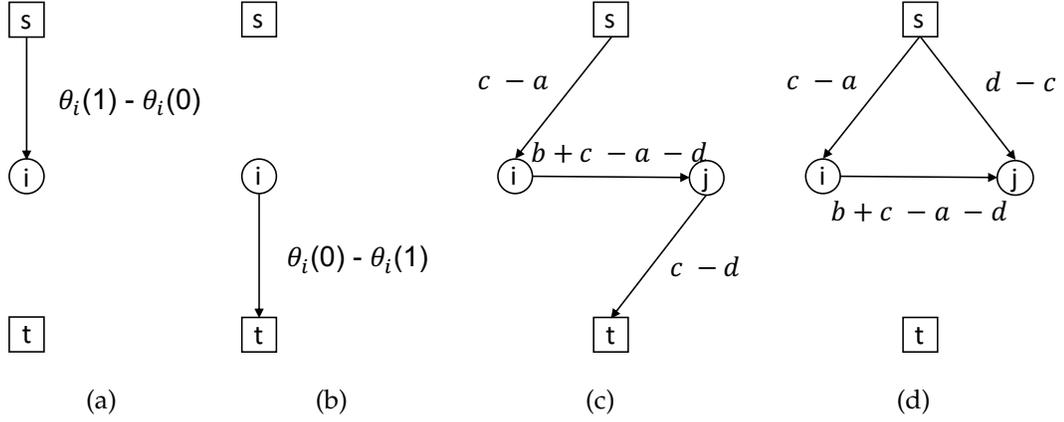


Figure 2.5: Basic building blocks to reduce binary pairwise submodular MRFs to st-MIN CUT. (a) A single unary term θ_i such that $\theta_i(0) \leq \theta_i(1)$. (b) A single unary term θ_i such that $\theta_i(0) > \theta_i(1)$. (c) A single pairwise term θ_{ij} such that $c - a \geq 0$ and $c - d \geq 0$. (d) A single pairwise term θ_{ij} such that $c - a \geq 0$ and $d - c \geq 0$. We define $a := \theta_{ij}(0,0)$, $b := \theta_{ij}(0,1)$, $c := \theta_{ij}(1,0)$, $d := \theta_{ij}(1,1)$.

Now, we can further reparameterize the unary terms θ' as:

$$\begin{aligned} \theta'_i(0) &:= \theta''(0) - \theta''(1), & \theta'_i(1) &:= 0, & \text{when } \theta''(0) \geq \theta''(1), \\ \theta'_i(0) &:= 0, & \theta'_i(1) &:= \theta''(1) - \theta''(0), & \text{when } \theta''(0) < \theta''(1). \end{aligned} \quad (2.53)$$

It is easy to verify that $f(\mathbf{x} | \theta) \sim f'(\mathbf{x} | \theta')$ by our construction and we can see $\theta'_{ij}(0,1) \geq 0$ due to θ_{ij} is submodular. \square

Now with the reparameterized $f'(\mathbf{x} | \theta')$ satisfying the conditions in Theorem 2.5.1, we can construct the following graph $G_{f'}$ in Definition 2.5.2, which is illustrated in Figure 2.5.

Definition 2.5.2 ([79]). Given binary pairwise MRF $f'(\mathbf{x} | \theta')$ satisfying the conditions in Theorem 2.5.1, we can define capacitated directed graph $G_{f'} = (V_{f'}, E_{f'}, c)$ such that

$$V_{f'} := V \cup \{s, t\}, \quad (2.54)$$

$$E_{f'} := \{(i, j) \mid (i, j) \in E, \theta'_{ij}(0,1) > 0\} \cup \{(s, i) \mid i \in V, \theta'_i(1) > 0\} \cup \{(i, t) \mid i \in V, \theta'_i(0) > 0\}, \quad (2.55)$$

$$\begin{aligned}
c_{si} &:= \theta'_i(1), \forall (s, i) \in E_{f'}, \\
c_{it} &:= \theta'_i(0), \forall (i, t) \in E_{f'}, \\
c_{ij} &:= \theta'_{ij}(0, 1), \forall (i, j) \in E_{f'}.
\end{aligned} \tag{2.56}$$

Now we can show the equivalence between the st-MIN CUT of $G_{f'}$ and the minimizer of $f'(\mathbf{x} \mid \theta')$ in Theorem 2.5.2 and Corollary 2.5.3.

Theorem 2.5.2. *Given any st-cut (S, T) on $G_{f'}$, define $x_i := \llbracket i \in T \rrbracket$ for $i \in V$, we have $f'(\mathbf{x} \mid \theta') = \text{cut}(S, T)$.*

Proof. We have the following:

$$\begin{aligned}
\text{cut}(S, T) &= \sum_{i \in S, j \in T} c_{ij} \\
&= \sum_{i \in T \cap V} c_{si} + \sum_{i \in S \cap V} c_{it} + \sum_{i \in S \cap V, j \in T \cap V} c_{ij} \\
&= \sum_{i \in T \cap V} \theta'_i(1) + \sum_{i \in S \cap V} \theta'_i(0) + \sum_{i \in S \cap V, j \in T \cap V} \theta'_{ij}(0, 1) \\
&= \sum_{i \in V, x_i=1} \theta'_i(1) + \sum_{i \in V, x_i=0} \theta'_i(0) + \sum_{(i,j) \in E, x_i=0, x_j=1} \theta'_{ij}(0, 1) \\
&= \sum_{i \in V} \theta'_i(x_i) + \sum_{(i,j) \in E} \theta'_{ij}(x_i, x_j) \\
&= f'(\mathbf{x} \mid \theta'),
\end{aligned} \tag{2.57}$$

where the third line is due to our construction in Definition 2.5.2, the fourth line is due to the definition $x_i := \llbracket i \in T \rrbracket$, the fifth line is due to all the other terms in $f'(\mathbf{x} \mid \theta')$ besides the ones shown on the fourth line are all 0 in our constraints of $f'(\mathbf{x} \mid \theta')$ in Theorem 2.5.1. \square

Corollary 2.5.3. *Finding the minimizer of $f'(\mathbf{x} \mid \theta')$ is equivalent to finding the st-MIN CUT in $G_{f'}$.*

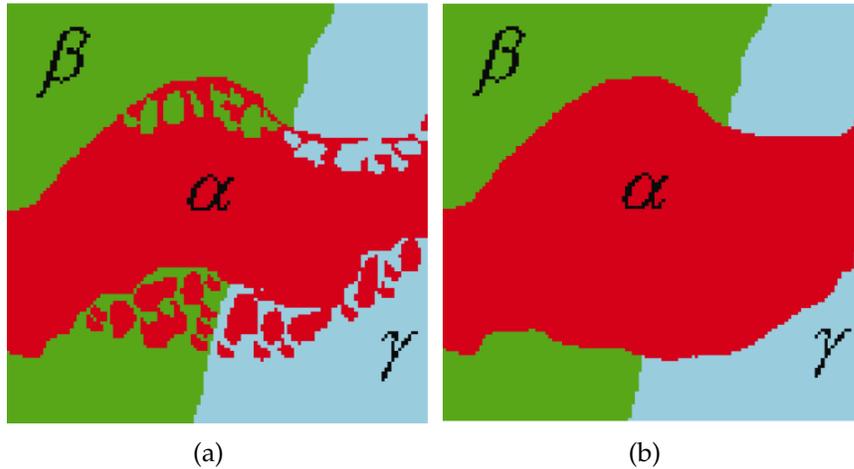


Figure 2.6: Illustration of expansion move algorithm [18] © IEEE. (a) The original labeling. (b) The optimum labeling when we allow some variables from the original labeling to switch to label α (i.e., we expand the α -region.)

In sum, given any binary pairwise submodular function f , we can reparameterize it to f' satisfying certain property described in Theorem 2.5.1, and construct the corresponding graph $G_{f'}$ using Definition 2.5.2. Finally, we can solve the max-flow/min-cut problem on $G_{f'}$, which gives us a minimizer of f' , which is also the minimizer of f .

2.5.2 Expansion move for multilabel MRFs

The expansion move (α -expansion) algorithm [18] is the most successful approximate MRF inference algorithm for multilabel pairwise MRFs. The high-level idea of expansion move algorithm is we can solve the multilabel problem via solving a series of binary subprogram optimally. In each iteration of expansion move algorithm, we will pick a single label α and ask each variable whether it wants to stay with its current label or the new label α , illustrated in Figure 2.6. We can solve this induced binary subproblem optimally via max-

flow / min-cut introduced in Section 2.5.1 (when the induced subproblem is submodular). Then we can loop over all the labels in the label set, and iteratively run the whole algorithm until energy converge or we reach certain number of iterations.

Formally, given a multilabel pairwise MRF $f(\mathbf{x} \mid \theta)$, a current labeling $\bar{\mathbf{x}}$ and a label α , we can define the binary MRF $g(\mathbf{y} \mid \eta)$ as following:

$$\begin{aligned}
\eta_i(0) &:= \theta_i(\bar{x}_i), \\
\eta_i(1) &:= \theta_i(\alpha), \\
\eta_{ij}(0, 0) &:= \theta_{ij}(\bar{x}_i, \bar{x}_j), \\
\eta_{ij}(0, 1) &:= \theta_{ij}(\bar{x}_i, \alpha), \\
\eta_{ij}(1, 0) &:= \theta_{ij}(\alpha, \bar{x}_j), \\
\eta_{ij}(1, 1) &:= \theta_{ij}(\alpha, \alpha).
\end{aligned} \tag{2.58}$$

We can easily show from our construction that $f(\mathbf{x} \mid \theta) = g(\mathbf{y} \mid \eta)$ when $x_i = \bar{x}_i \leftrightarrow y_i = 0$ and $x_i = \alpha$ otherwise. Therefore, by solving $g(\mathbf{y} \mid \eta)$ optimally, we find the optimum move to expand the α -region in the current labeling. Unlike the iterative conditional mode (ICM) [40] algorithm can only search for a $\mathcal{O}(|\mathcal{L}_i|)$ space to update a single variable, expansion move algorithm searches a $\mathcal{O}(2^{|\mathcal{V}|})$ space and simultaneously update multiple variables. Therefore, it gives substantially better results than all the previous algorithms for multilabel MRF optimization.

Boykov *et al.* [18] gives the sufficient condition on the multilabel MRF energy f to make sure every induced binary subproblem is submodular, and it is referred as the *regular* property of MRFs [79].

Theorem 2.5.4 ([18]). *Each induced binary subproblem of a given multilabel pairwise*

MRF f is submodular if all the pairwise terms θ_{ij} is defined over a metric space such that:

$$\begin{aligned}\theta_{ij}(\alpha, \beta) &= 0 \leftrightarrow \alpha = \beta, \\ \theta_{ij}(\alpha, \beta) &= \theta_{ij}(\beta, \alpha) \geq 0, \\ \theta_{ij}(\alpha, \beta) &\leq \theta_{ij}(\alpha, \gamma) + \theta_{ij}(\gamma, \beta), \forall \gamma.\end{aligned}\tag{2.59}$$

Boykov *et al.* [18] also proves the approximation error bound for expansion move algorithm on convergence. For Potts energy, we get a 2-approx algorithm, which matches the error bound of the rounding from LP relaxation algorithm for the equivalent uniform metric labeling problem [67]. Despite the worst-case bound here, various empirical study suggests expansion move can get very close to global optimum in practice [63, 82, 136].

Theorem 2.5.5 ([18]). *The multiplicative error bound of expansion move algorithm is*

$$2 \cdot \max_{(i,j) \in E} \frac{\max_{\alpha \neq \beta} \theta_{ij}(\alpha, \beta)}{\min_{\alpha \neq \beta} \theta_{ij}(\alpha, \beta)}.$$

2.5.3 Energy truncation and QPBO for non-submodular MRFs

Despite the fact that minimizing general binary pairwise MRFs is NP-hard, it is still a desired question in vision community to minimize non-submodular MRFs.

Rother *et al.* [116] proposed a technique called energy truncation. The key idea is given any binary non-submodular function f , we can find a submodular upper bound $\bar{f} \geq f$, then we can minimize \bar{f} instead of f . When we apply the reparameterization technique introduced in Theorem 2.5.1 to binary non-submodular function f , the only problem is we may get some $\theta'_{ij}(0, 1) < 0$

due to the original pairwise term θ_{ij} is supermodular. We just need to truncate $\theta'_{ij}(0, 1) := 0$ to get the desired submodular upper bound.

Later, the more principled persistency-based method QPBO was rediscovered by the computer vision community [77, 115]. We can always label the persistent variables and do a random assignment for the remaining variables. Also various heuristics have been proposed to determine the remaining variables in a better way [115]. QPBO then became the standard way to optimize non-submodular MRFs in the vision community.

Both of these techniques can be combined with the expansion move algorithm to get rid of the constraints that the pairwise terms must be defined over a metric space (in Theorem 2.5.4). In this case, there is an even simpler way to handle the variables that cannot be proven to be persistent by QPBO: we never switch to the new label α for those variables. It also motivates the fusion move algorithm [94], which can take arbitrary proposals rather than the constant α and fuse the old labeling and the new proposal labeling together. Fusion move algorithm makes the move making algorithm very flexible and performs well in practice [56, 94].

2.5.4 Sum-of-submodular (SoS) fusion for higher-order MRFs

We will first focus on the binary submodular higher-order MRFs. As a recap, we follow the convention in the vision community to define a MRF to be submodular when it is actually a sum-of-submodular (SoS) function (Definition 2.1.4). But SoS function is still a submodular function overall. It is known from the algorithm community that submodular function minimization is polynomial time

solvable [59, 60, 107]. We can solve it via either a $O(N^5\text{EO} + N^6)$ strongly polynomial time algorithm [107] or a $O((N^4\text{EO} + N^5) \log U)$ weakly polynomial time algorithm [59], where EO is the oracle call to evaluate the function value, U is the maximum absolute value of f . Unfortunately, none of these algorithms are practical for vision problems, and they don't fully utilize the SoS structure. Therefore, we will introduce the SoS-flow and SoS-cut [35, 36, 76], as a straightforward generalization of the conventional flow and cut.

Recall that we can represent a binary higher-order function as a set function such that $x_i = \llbracket i \notin S \rrbracket$ for any $S \subseteq V$. So we can rewrite our binary submodular higher-order MRFs energy as:

$$f(S) := \sum_{i \in S} \theta_i(0) + \sum_{i \notin S} \theta_i(1) + \sum_{C \in \mathcal{C}} \theta_C(S \cap C), \quad S \subseteq V, \quad (2.60)$$

where each θ_C is submodular.

We can define the max-SoS-flow problem with the flow variable $\phi = \{\phi_{s,i}\}_{i \in V} \cup \{\phi_{i,t}\}_{i \in V} \cup \{\phi_{i,C}\}_{C \in \mathcal{C}, i \in C}$.

Definition 2.5.3 (max-SoS-flow, [76]). Given $c_{s,i}, c_{i,t} \geq 0$ and submodular function $g_C(S) \geq 0$ such that $g_C(\emptyset) = g_C(C) = 0$ as the capacity, we can define the max-SoS-flow as following:

$$\begin{aligned} \max_{\phi} \quad & \sum_{i \in V} \phi_{s,i} \\ \text{s.t.} \quad & \phi_{s,i} \leq \theta_i(0), \quad \phi_{i,t} \leq \theta_i(1), \quad \forall i \in V, & \text{(unary capacity)} \\ & \phi_{s,i} - \phi_{i,t} - \sum_{C \ni i} \phi_{i,C} = 0, \quad \forall i \in V, & \text{(flow conservation)} \\ & \sum_{i \in C} \phi_{i,C} = 0, \quad \forall C \in \mathcal{C} & \text{(flow conservation)} \\ & \phi_C(S) := \sum_{i \in S} \phi_{i,C} \leq \theta_C(S), \quad \forall C \in \mathcal{C}, \forall S \subseteq C. & \text{(higher-order capacity)} \end{aligned} \quad (2.61)$$

We require the capacity function $g_C(S)$ to be non-negative submodular function, that satisfies $g_C(\emptyset) = g_C(C) = 0$. Then the higher-order capacity constraint is that our flow value is bounded by our capacity value for all $S \subseteq C$, such that $\phi_C(S) \leq g_C(S)$. As an example, in Figure 2.7(b), we listed all 4 capacity constraints for this single edge in the conventional graph. $0 = \phi_C(\emptyset) \leq g_C(\emptyset) = 0$ and $0 = \phi_C(C) \leq g_C(C) = 0$ are always trivially satisfied. $\phi_C(\{1\}) = \phi_{1,C} \leq g_C(\{1\}) = 2$ and $\phi_C(\{2\}) = \phi_{2,C} \leq g_C(\{2\}) = 0$ represents the capacity constraint we have in the conventional flow, that we can send at most 2 units of flow from vertex 1 to vertex 2, but not flow is allowed from vertex 2 to vertex 1. With this factor graph representation, we can easily generalize our capacity constraint, i.e., a linear flow function is bounded by a submodular capacity function, to cliques more than 2 vertices, as illustrated in Figure 2.7(c).

Now going back to Definition 2.5.3, our s -links and t -links are still conventional edges, so our unary capacity constraints are exactly the same capacity constraints we have in the conventional flow. We just explained the generalized higher-order capacity constraints for all the cliques. The flow conservation constraints here guarantee that both the internal vertex (except source s and sink t) and all the factor nodes don't store flow. We can defined $\phi_{i,s} := -\phi_{s,i}$, $\phi_{t,i} := -\phi_{i,t}$ and $\phi_{C,i} := -\phi_{i,C}$ as the asymmetric constraints in the conventional flow. This concept helps the definition of augmenting path, which we will discuss next. But they are redundant in the problem definition. So we omit them in (2.5.3). The overall goal for max-SoS-flow problem is still the same, i.e., maximize the overall flow from the source s .

Kolmogorov [76] showed that we can mimic the augmenting path algorithm for conventional flow to solve the max-SoS-flow problem. One example aug-

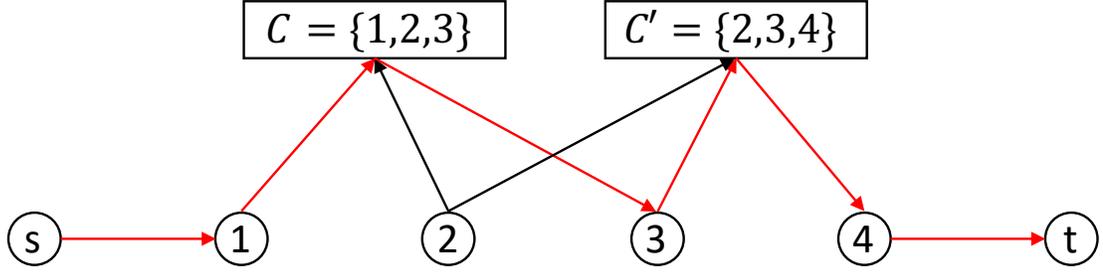


Figure 2.8: An example of augmenting path for SoS-flow. The augmenting path is shown in red.

menting path for SoS-flow is illustrated in Figure 2.8.

Definition 2.5.5 (augmenting path of SoS-flow on factor graph, [76]). It is easier to define the reverse flow $\phi_{C,i} := -\phi_{i,C}$, $\phi_{-C}(S) = -\phi_C(S)$ and reverse capacity $g_{-C}(S) = 0$. Then we can define the capacity in the residual SoS-flow network as:

$$\begin{aligned}
 r_{s,i} &:= c_{s,i} - \phi_{s,i}, \\
 r_{i,t} &:= c_{i,t} - \phi_{i,t}, \\
 r_{i,C} &:= \min_{S \subseteq C: i \in S} g_C(S) - \phi_C(S), \\
 r_{C,i} &:= \min_{S \subseteq C: i \in S} g_{-C}(S) - \phi_{-C}S.
 \end{aligned} \tag{2.62}$$

Then an *augmenting path* of SoS-flow is a $s - t$ path in the residual SoS-flow network with strictly positive flow values.

Theorem 2.5.6 (characterization of max-SoS-flow, [76]). ϕ is a maximum SoS-flow if and only if there is no augmenting path in the residual SoS-flow network.

Therefore, we can employ all the augmenting path-based approaches for conventional max-flow to max-SoS-flow. For example, Fix *et al.* [35] employs IBFS [43]. Note that the computational bottleneck is to compute $r_{i,C}$ and $r_{C,i}$ in (2.62), which is a much smaller scale submodular function minimization problem can be solved by either brute force or the off-the-shelf submodular function

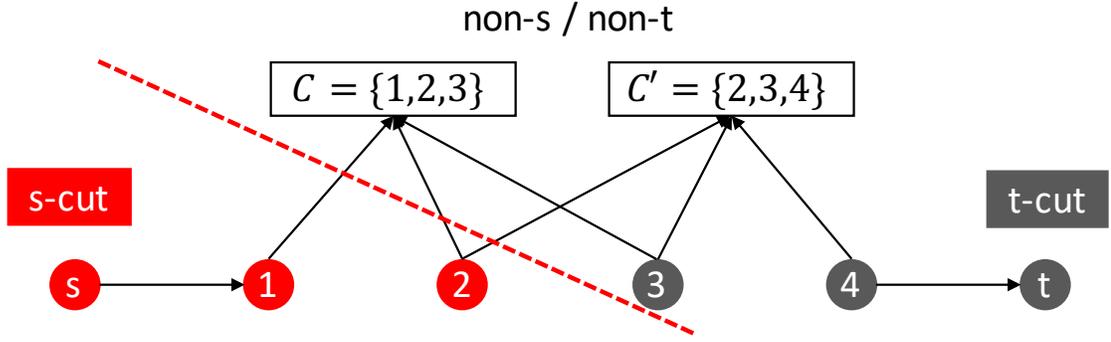


Figure 2.9: An example of st -SoS-cut. The s -cut is denoted as red nodes, while the t -cut is denoted as gray nodes. Clique factor doesn't belong to either s -cut or t -cut. The set S derived from s -cut (exclude the source s) saturates each higher-order capacity. For example, in this example, we have $\phi_C(\{1, 2\}) = g_C(\{1, 2\})$ and $\phi_{C'}(\{2\}) = g_{C'}(\{2\})$.

minimization algorithms. Therefore, we have the overall running time to be either $O(N^2 M 2^K)$ or $O(N^2 M (K^5 \text{EO} + K^6))$. Considering the size of clique K is usually just a small constant for vision applications, brute force usually achieves better performance in practice.

Kolmogorov [76] also showed the generalized max-flow/min-cut theorem for SoS-flow. We have an example of st -SoS-cut in Figure 2.9;

Definition 2.5.6 (st -SoS-cut, [76]). Arbitrary partition on $V \cup \{s, t\}$ to $S \cup \{s\}$ and $T \cup \{t\}$ is called the *SoS-cut*, its *cut value* is defined as:

$$\text{cut}(S, T) := \sum_{i \in S} c_{i,t} + \sum_{i \notin S} c_{s,i} + \sum_{C \in \mathcal{C}} g_C(S \cap C). \quad (2.63)$$

The min-SoS-cut minimizes the cut value.

Theorem 2.5.7 (max-SoS-flow / min-SoS-cut theorem, [76]). *The max-SoS-flow and min-SoS-cut values equal. Furthermore, for any maximum SoS-flow ϕ^* , we can derive a minimum SoS-cut S^* such that S^* is the set of all vertices reachable from the source s in the residual SoS-flow network. Then we have the flow value saturate the corresponding*

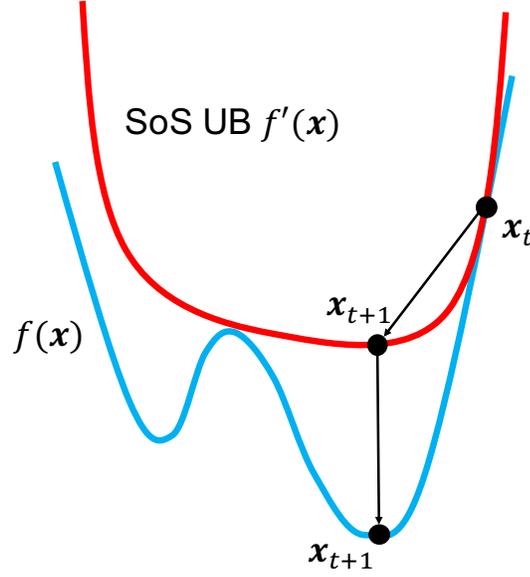


Figure 2.10: Intuition of SoS upper bound approximation. Given any binary higher-order MRF $f(\mathbf{x})$, we can find its SoS upper bound $f'(\mathbf{x}) \geq f(\mathbf{x})$ such that they coincide at the current labeling \mathbf{x}_t . Then the minimizer \mathbf{x}_{t+1} of $f'(\mathbf{x})$ guarantees to decrease the energy value $f(\mathbf{x}_{t+1}) \leq f'(\mathbf{x}_{t+1}) \leq f'(\mathbf{x}_t) = f(\mathbf{x}_t)$.

capacity:

$$\begin{aligned}
\phi_C^*(S^*) &= g_C(S^*), \quad \forall C \in \mathcal{C}, \\
\phi_{s,i} &= c_{s,i}, \quad \forall i \notin S, \\
\phi_{i,t} &= c_{i,t}, \quad \forall i \in S.
\end{aligned} \tag{2.64}$$

With the max-SoS-flow as the generalization of conventional max-flow, we can solve binary submodular higher-order MRFs now. Fix *et al.* [36] showed that given arbitrary binary SoS function $f(S)$, we can reparameterize it to:

$$g(S) = C + \sum_{i \in S} c_{i,t} + \sum_{i \notin S} c_{s,i} + \sum_{C \in \mathcal{C}} g_C(S \cap C), \tag{2.65}$$

where C is a constant we can omit for minimization task, $c_{s,i}, c_{i,t} \geq 0$ for all $i \in S$ and $g_C(S) \geq 0, g_C(\emptyset) = g_C(C)$ for all $C \in \mathcal{C}, S \subseteq C$. Therefore, we can construct a valid SoS-flow network, then solve the min-SoS-cut via max-SoS-flow. The min-SoS-cut S^* minimizes $g(S)$, as well as the original $f(S)$.

In order to solve binary non-submodular higher-order MRFs, we will take the SoS upper bound approximation proposed in [36], which is a generalization of the energy truncation scheme [116] for binary non-submodular pairwise MRFs. Given arbitrary binary higher-order MRF $f(\mathbf{x})$ and a current labeling \mathbf{x}_t , we can construct a SoS upper bound $f'(\mathbf{x}) \geq f(\mathbf{x})$ such that $f(\mathbf{x}_t) = f'(\mathbf{x}_t)$. Now, we can minimize our proxy $f'(\mathbf{x})$ to reach the new labeling \mathbf{x}_{t+1} . We are guaranteed to decrease the energy value since $f(\mathbf{x}_{t+1}) \leq f'(\mathbf{x}_{t+1}) \leq f'(\mathbf{x}_t) = f(\mathbf{x}_t)$. Then we can repeat this procedure until convergence or until we reach the maximum number of iterations. This idea is illustrated in Figure 2.10.

Finally, in order to solve arbitrary multilabel higher-order MRFs, we can further employ the move-making technique and the fusion move framework, which induces the multilabel problem to a series of binary subproblems and fuses the current labeling with the new proposal by solving a binary higher-order MRF. Please refer to [35] for the primal version SoS-Fusion algorithm and [36] for the primal-dual version SoSPD algorithm.

CHAPTER 3

RELATED WORK

There is long history of research on the inference problem of Markov Random Fields and its applications in computer vision. Koller and Friedman [72] and Murphy [105] are two excellent books on Markov Random Fields and probabilistic graphical models. Szeliski [135], Blake *et al.* [9] and Wang *et al.* [143] provided extensive survey and background knowledge about applying MRFs to vision applications. Empirical studies of MRF inference approaches can be found in [63, 136].

We will conduct a literature survey on MRFs for vision applications in Section 3.1. Although the inference problem of MRFs is NP-hard in general, we do have a few sub-classes of MRFs which we can solve optimally. These will be summarized in Section 3.2. We will review persistency algorithms for MRFs in Section 3.3 and approximate inference algorithms for MRFs in Section 3.5 and Section 3.6. Finally, we will mention the existing methods using persistency algorithms as the pre-processing of the approximate inference algorithms in Section 3.7.

3.1 Markov Random Fields in computer vision

Markov Random Fields have achieved significant success for various fundamental computer vision tasks in the past few decades.

Image denoising (image restoration) is one of the most important low-level vision tasks. It is probably the first computer vision application using Markov Random Fields techniques. Geman and Geman [40] did the very first attempt

in this field. They analogized image models to statistical mechanics systems. Each pixel has an underlying grayscale intensity and there exists a dependency between adjacent pixels. Finally, they described the image generation as a sampling from the Markov Random Fields and use the MAP estimation to address the image denoising task. They also discussed the learning problem to estimate the parameters of the generative model. Besag [8] and Greig *et al.* [47] adopted similar MRF formalization as in [40]. They proposed more effective inference algorithms, namely ICM and st-MINCUT, respectively. They achieved very visual appealing results due to the better inference algorithm. Chambolle [22] applies the total variance as the prior in the formalization of denoising tasks. Roth and Black [112] proposed Field of Experts, which involves a learned higher-order patch prior from natural images. This higher-order term significantly improves the denoising quality.

Super resolution recovers a high resolution image from the low resolution input. It is an important low-level vision and image processing task. Freeman *et al.* [38, 39] formalized the super resolution task using MRFs. They treated the generation of the low-res images from the high-res images as a random field. They collect a set of candidate patches for each pixel from the training set, then use the data terms and prior terms to formalize the similarity of the inferred candidate patches to the given low-res input image and the smoothness between the adjacent patches.

Image stereo and its extension to multiview scene reconstruction is the key problem for computer vision to understand the 3D world through images. It is the first task that MRF formalization achieved great success and stimulated the researchers in the vision fields to apply this technology to other vision prob-

lems. Roy and Cox [117] established a mapping between the disparity surface in the stereo problem and a st-cut, and computed the optimum disparity surface via min-cut. In other words, they implicitly formalized the stereo problem via MRFs. Boykov *et al.* [18] proposed the expansion move algorithm, which solves the multilabel MRF inference problem for stereo and achieved significantly more visual appealing results. Kolmogorov and Zabih [78] incorporated the visibility constraint in the 3D multiview scene reconstruction problem other than the fidelity data term and the smoothness prior term in the MRF formalization. Vogiatzis *et al.* [140] adopted a volumetric representation of the multiview stereo/scene reconstruction problem, and formalize the photo consistency constraint and the visibility/occlusion constraints as a MRF. Woodford *et al.* [149] took the second order derivative (curvature) into account and this higher-order MRF setup provides smoother results.

Image stitching composites multiple images with overlap into a single image. Nowadays, every single mobile phone and digital camera is shipped with the panorama creation feature. MRF formalization achieves great success for this vision application as well, and it is still the state-of-the-arts algorithm to find the optimum seam between source images. Kwatra *et al.* [87] is the first work that sets up a MRF to find the optimum transition between given images. They applied this idea to texture synthesis, rather than the image stitching. However, this work motivated Agarwala *et al.* [1], and they adopted the similar MRF setup in the image stitching task. In general, data term provides the hard constraints about the pixel assignments, but most of them are just 0 (i.e., no preference). Meanwhile, the prior term is usually the local patch similarity, which enforces the smoothed, invisible transition between source images. Recently Herrmann *et al.* [53, 54] proposed more complicated data and prior terms

in the MRF formalization to prevent the objects in the stitching results being duplicated or distorted when large motion and parallax presents.

The image segmentation task is the key problem for computer vision to understand the scene. MRFs are widely used to enhance the label smoothness of the segmentation mask to make it aligned with the true boundary of the objects. Boykov and Jolly [15] formalized the foreground/background segmentation problem as a binary MRF, where the data terms describe the likelihood of each pixel label, and the prior terms penalize the object contour at non-boundary regions. Boykov and Kolmogorov [16] connected GraphCuts and the geodesic contours of objects. Rother *et al.* [114] proposed GrabCut, which could dynamically update the foreground/background data terms and perform image segmentation in an interactive way. Shotton *et al.* [131] applies Conditional Random Fields (CRFs) to semantic segmentation tasks, which combines the data terms based on local color, location, texton features and the contrast based smoothness priors. More advanced image segmentation formalization incorporates more sophisticated prior terms and improves quality. Vincete *et al.* [139] considered the connectivity prior to overcome the weakness of GraphCuts-based segmentation preferring short boundary. Jegelka and Bilmes [61] proposed Cooperative Cuts, further improved [139]. They introduced the submodular cooperative terms to preserve the long and thin objects in the segmentation results. Kohli *et al.* [69] proposed the \mathcal{P}^n Potts model, which considers the higher-order patch statistics for semantic segmentation tasks. Ladicky *et al.* [90] considered the co-occurrence statistics of objects in the segmentation tasks.

Optical flow is another fundamental computer vision task to understand the motion. Heitz and Bouthemy [52] is the very first work applying MRFs for

this task. They used MRFs to handle the discontinuity of the flow fields. Roy and Govindu [118] addressed input noise in optical flow problem via MRFs. Glocker *et al.* [42] formalized the angle derivatives and affine motion priors as a higher-order MRF. Liu *et al.* [96] proposed SIFT flow, which contains a local patch similarity term (based on SIFT descriptor), L2 regularizer on the flow vector and L1 discontinuity regularizer. Chen and Koltun [25] proposed Full Flow, which provides an efficient global optimization algorithm to solve the MRF inference of Horn-Schunck-type objective over regular grids.

Even though deep neural network have achieved great successes in almost all vision applications, we can still see Markov Random Fields play an important role to refine the results of pixel labeling problem. Deeplab [24] is the state-of-the-arts deep image segmentation algorithm, which still adopts a fully connected CRF at the end of the pipeline to further improve the segmentation boundary, which is supported by the ablation study. Zheng *et al.* [150] and Arnad *et al.* [4] treated CRFs as the recurrent neural networks and made it joint trainable with the neural network, and achieves better segmentation results both qualitatively and quantitatively. Knobelreiter *et al.* [68] and Colovic *et al.* [27] proposed a structure learning framework for the hybrid CRF-CNN model and improves the quality for both the stereo and segmentation tasks. Meng *et al.* [137] found using MRF loss as a regularizer is performance critical for weakly-supervised semantic segmentation tasks.

3.2 Exact inference for certain sub-class of MRFs

Although the general MRF inference problem is NP-Hard, we can still solve the inference problem for certain sub-classes of MRFs exactly.

Besag [8] reduces the inference for binary pairwise MRFs for Ising model to st-MIN CUT and solves it exactly. Kolmogorov and Zabih [79] further discussed the open-ended question on what energy can be optimized exactly via graphcuts. They showed that all inference problems for binary pairwise submodular MRFs can be reduce to st-MIN CUT, hence can be solved exactly. They also showed that inference for second-order MRFs can be solved exactly when all the pairwise terms are submodular and the higher-order terms' projection to any two variables are still submodular.

Ishikawa [55] proposed a clever transformation to reduce the multilabel pairwise MRFs to binary pairwise submodular MRFs, when all the pairwise terms can be represented as a convex function $g(\cdot)$ over the ordinary difference between labels $g(\ell_i - \ell_j)$. Schlesinger and Flach [122, 123] further relaxed this condition to each pairwise terms satisfying the multilabel-submodularity property $\theta_{ij}(\min(\ell_i^1, \ell_i^2), \min(\ell_j^1, \ell_j^2)) + \theta_{ij}(\max(\ell_i^1, \ell_i^2), \max(\ell_j^1, \ell_j^2)) \leq \theta_{ij}(\ell_i^1, \ell_j^1) + \theta_{ij}(\ell_i^2, \ell_j^2)$ with a total ordering among labels. Ramalingam *et al.* [110] follows this line of research and studied the characterization of multilabel higher-order MRFs which could be solved exactly. Felzenszwalb [33] proposed a graphcuts-based sweep algorithm to solve the multilabel pairwise MRFs with the pairwise terms defined over a tree metric space.

Besides the effort to reduce the MRF inference problem to graphcuts problem, it is also well known that when the underlying graph representation is a

tree or bounded tree width graph that a solution can be found using dynamic programming. When we have a Markov chain, this dynamic programming algorithm is often referred as Viterbi algorithm or forward-backward algorithm in the literature [72, 99, 105]. This dynamic programming idea can be easily generalized to tree structure [72, 99, 105], and it is sometimes referred as the inside-outside algorithm. This is the foundation for all the message passing algorithms for general MRF inference, which just apply the same idea on the loopy graph. When the graph has bounded tree width, we can also apply the junction tree algorithm to do the exact inference efficiently [92].

3.3 Persistency algorithms for MRFs

Since the MRF inference problem is NP-hard in general, rapidly determining the optimal labels for a subset of the variables would obviously be of great utility. We have already reviewed the literature on the persistency algorithms for MRFs and introduced their high-level ideas in Section 2.4 as a mathematical background of this thesis. We will recapitulate the high-level comparison among these methods here.

3.3.1 Local condition-based persistency approaches

Dead-end Elimination (DEE) [31] checks a local sufficient condition which only involves a single variable and its adjacent edges. Given a variable, DEE rules out a label to be persistent when the unary term of this label plus its adjacent pairwise terms in the best situation is still worse than the unary term of another

label plus its adjacent pairwise terms in the worst situation. It was originally used for a non-vision task to predict protein side-chain structure, which could also be formalized as a MRF. This methods significantly reduce the combinatorial search space of that problem.

Goldstein [44] refines the local sufficient condition in the original DEE. It states that a label $x_i = \ell$ cannot be persistent if there exists another label $x_i = \ell'$ that the energy gain of the unary term by switching from ℓ' to ℓ is strictly smaller than the minimum energy loss from the pairwise terms. This condition is tighter than the original DEE condition but still sound. It is similar to the autarky property, but in the rule out favor.

Pierce *et al.* [108] and Looger and Hellinga [97] proposed a even stronger condition than the Goldstein condition. In order to show $x_i = \ell$ cannot be persistent, they partitioned the label space of adjacent variables of x_i into several clusters, and show we can find different labels $x_i = \ell_c$ to be a better choice than $x_i = \ell$ for each cluster of adjacent variables assignment.

Voigt *et al.* [141] generalized DEE condition from eliminating a label for a single variable to a pair of labels for adjacent variables. But the basic idea is still the same that there exists another pair of labels that is always better than our candidates.

3.3.2 Flow-based persistency approaches

Techniques like QPBO [12, 77] find a persistent partial labeling for general binary pairwise MRFs by seeking an even stronger *autarky* condition, namely a

partial labeling which will not increase the energy if it is applied to any complete labeling. QPBO in particular is widely used in computer vision since it often finds the correct label for the vast majority of the variables. It requires to solve a max-flow problem with twice of the scale of the original MRF¹.

Kovtun [83, 84, 85] proposed an approach to handle multilabel MRFs by constructing a series of binary auxiliary problems, which determines whether a particular label is better than the remaining labels, and solving each of them via graphcuts. Therefore, the computational cost of Kovtun’s method is at least solving the flow problem at the scale of the original MRF times L , where L is the number of total labels.

MQPBO [71] generalized QPBO to multilabel cases by introducing the binary encoding of each multilabel variable and transforming the multilabel MRF with N variables and L labels to a binary MRF with NL variables. Then it solves the persistency of the transformed binary problem via QPBO (which is a much larger flow graph compared to the original MRF), and then translates the persistency results back to the multilabel MRF.

Windheuser *et al.* [148] proposed generalized roof duality for multilabel MRFs. It introduces N auxiliary multilabel variables $\bar{x}_i = |\mathcal{L}_i| - x_i + 1$ to mimic the negation of variables in the binary case, and includes a multilabel submodular function which is easier to minimize, with its optimum to be the lower bound of the original problem. Then we can compute the persistency of the original MRF from the minimizer of the auxiliary problem. The computational cost of this method requires the computation of the minimizer of the submodular auxiliary function with $2N$ variables and L labels using Schlesinger’s transforma-

¹If the original MRF is submodular, then we can reduce the scale of the flow network by half due to symmetricity, which is equivalent to reduce the MRF inference problem to st-MINCUT.

tion [123], which involves a max-flow problem with around $2NL$ variables.

3.3.3 MRF/LP-based persistency approaches

Recently, Swoboda *et al.* [134] explored the research question about when the integral optimum solution of the LP relaxation coincides with the persistent labeling of the multilabel MRF. They relaxed the autarky property and proved that the persistency condition for this problem is equivalent to solving another auxiliary MRF inference LP relaxation, which can be either solved by LP techniques exactly, or by some standard MRF inference algorithms approximately. Then they used the shrink scheme to iteratively update the persistent variable set using the proposed persistency condition, starting from the whole set, until convergence.

Shekhovtsov [126] proposed new concepts improving mapping and relaxed improving mapping, which generalizes the autarky property. Then he formalized the problem to maximize the number of optimally labeled variables, which is equivalent to solving a gigantic LP for certain sub-classes of relaxed improving mapping.

They also proposed to combine these two approaches together which can take advantage of both of them [129]. It replaces the persistency criteria used in PBP by the more general condition relaxed improving mapping, which leads to finding more persistent variables compared. In the subroutine to check the feasibility of the relaxed improving mapping, it adopts the shrink idea and approximate inference for MRF, instead of solving the gigantic LP. It speeds up the algorithm significantly.

In general, the number of variables labeled by these approaches are significantly more than Kovtun’s approach and MQPBO. However, the running time of these approaches is significantly longer, since these approaches involve solving complex programming (either via standard MRF inference solver or LP solver) iteratively.

3.4 Persistency for other combinatorial optimization problems

Although the central topic of this thesis is the persistency for MRF inference problem, people also explored persistency for many other important combinatorial optimization problems.

The concept of persistency was first introduced for pseudo-Boolean optimization [49], and the concept of autarky was first introduced for Boolean satisfiability problem (SAT) [104].

However, people used these concepts informally even before we have the definition. For example, Nemhauser and Trotter [106] showed that the optimal LP relaxation solution of vertex packing is half-integral (i.e., $\{0, 0.5, 1\}$), and all the integral variables coincide with the global optimum. Hammer *et al.* [48] explored the criteria that when does a vertex belong to all or none of the maximum stable set.

People also explored persistency and autarky for many other combinatorial optimization problems, e.g., maximum cardinality bipartite matching [29], maximum weighted bipartite matching [20], Boolean satisfiability problem (SAT) [86], traveling salesman problem (TSP) [88], maximum weighted inde-

pendent set of a matroid and maximum cardinality 2-matroids intersection [21], weighted k-matroids intersection [98].

3.5 Approximate inference algorithms for pairwise MRFs

3.5.1 Graphcuts and move-making algorithms

Graphcuts algorithms are strongly motivated by the fact that we can reduce the inference for binary pairwise submodular MRFs to st-MIN CUT, and exactly solve it via max-flow. This connection has been established at least four decades in the history [8]. However, it did not draw the vision community's attention because the binary MRF problem studied in [8] was too restrictive in vision applications. The optimization techniques for multilabel MRFs, including Gibbs sampling, simulated annealing [40] and Iterated Conditional Modes (ICM) [8], all give poor visual results.

Graphcuts methods and move-making techniques [18] were the breakthrough study for the whole MRF inference literature. Move-making techniques generate a new proposal at each iteration and reduce the multilabel problem into a series of binary subproblems (should each pixel stick with the old label or switch to the new label in the proposal). α -expansion move uses the constant α labeling as the new proposal and the derived binary sub-problem is submodular when the pairwise terms in the original multilabel MRF is defined over a metric space. Therefore, each binary sub-problem can be solved exactly via max-flow/min-cut. $\alpha\beta$ -swap move tries to swap the α label and β label in each iteration and the derived binary sub-problem is submodular when

the pairwise terms in the original multilabel MRF is defined over a semi-metric space. Boykov *et al.* [18] also shows the multiplicative bounds of α -expansion move algorithm. But in practice, it usually gets energy very close to global optimum and visually appealing results.

To overcome the limitation of expansion move and $\alpha\beta$ swap can only infer certain sub-class of multilabel MRFs, we need to address the inference problem for arbitrary binary MRFs. Rother *et al.* [116] proposed to truncate the non-submodular terms and perform the inference over a submodular upper bound of the original energy function. Kolmogorov and Rother *et al.* [77] introduced the quadratic pseudo-Boolean optimization (QPBO) and roof duality [12] from the operation research community to the computer vision community. It uses the persistency algorithm to find the partial optimal labeling for arbitrary MRFs, and does not take the move by default for unknown variables. The followed-up work [115] proposed several heuristics (QPBO-P and QPBO-I) to further label the unknown variables from QPBO. This line of research enables move-making algorithms to handle arbitrary proposals, which is the fusion move algorithm introduction by Lempitsky *et al.* [94].

Given the great success achieved by the graphcuts algorithms and move making techniques, researchers also explored how to speedup this algorithm to make it more applicable in practice. Boykov and Kolmogorov [17] proposed a new algorithm (BK algorithm) to compute max-flow. It maintains a search tree from source and sink to find the augmenting path. The novelty of this work is a heuristic to restore the search tree after the augment step which saturates some edges in the flow graph. It works extremely well for vision instances and yields order of magnitude speedup compared to standard max-flow algorithms

like Dinic or Edmonds-Karp. However, the asymptotic worst case running time bound of BK algorithm could be as bad as the Ford-Fulkerson, which depends on the total flow value, hence not polynomial w.r.t. the flow graph size. Goldberg *et al.* [43] proposed an alternative max-flow algorithm for vision instances, which achieves similar speed compared to BK algorithm but with a polynomial asymptotic worst case running time bound $O(N^2M)$, as same as the Dinic algorithm. Komodakis and Tziritas [82] proposed FastPD, which provides a primal-dual view of the move-making algorithms. One big advantage is we can now reuse the dual variables (flow values) between iterations so that we won't need to compute max-flow from scratch each time, and it provides order of magnitude speedup. Alahari *et al.* [3] combines several different ideas to speedup MRF inference: 1) Reduce, nail down variables via persistency algorithm before full inference, 2) Recycle and reuse, reuse the dual variable value/flow values in the previous iteration and the same move in the previous epoch to initialize the flow in the current iteration.

3.5.2 Message passing algorithms

Message passing algorithms are strongly motivated by the fact that we can use dynamic programming to solve the MRF inference problem over tree structure. One advantage of message passing algorithms is it is easy to implement and can be easily parallelized.

Loopy belief propagation (LBP) [105, 146] is probably the most popular message passing methods. It adopts the same message update rule applied on the tree structure to general graph, and it is guaranteed to converges to the global

optimum for acyclic graphs. However, this update rule is purely a heuristic for general graphs and has no theoretical guarantee, although it usually performs well in practice.

Wainwright *et al.* [142] proposed the tree-reweighted max-product message passing algorithm (TRW). It solves the LP relaxation of the MRF inference problem and views the dual problem as a convex combination of trees. The LP relaxation is tight and the global optimum is achieved if the TRW algorithm converge to a fixed point satisfying a *strong tree agreement* condition shown in [142]. Kolmogorov [74] proposed TRWS, using a sequential update instead of the parallel update scheme in TRW. It provides some desired property like the lower bound of energy will never decrease and relax the characterization of global optimum to the *weak tree agreement* condition.

3.5.3 Linear programming relaxation algorithms

We can relax the MRF inference problem to linear programming [82, 142], which is usually referred as the *local polytope relaxation*, so there is a category of algorithm try to solve the LP relaxation, which gives the lower bound of the problem but requires rounding to get the integer solution. Per [82, 142], both the graphcuts algorithms and message passing algorithms could be interpreted as the algorithm based on LP relaxation in some manner. The Max-Product Linear Programming (MPLP) approach and its follow-up work [41, 132, 133] finds the tighter relaxation than the local polytope relaxation and yields better performance.

Dual decomposition is a common framework in this category. Instead of

solving the original complicated primal relaxation, we usually decompose the dual problem into a set of subproblems which are much easier to solve (e.g., the subproblem forms a Markov chain or has the tree structure). Shlezinger [130] adopted a block coordinate descent technique to solve the dual problem, which may get stuck in local sub-optimum fixed point. Komodakis *et al.* [81] and Kappes *et al.* [64] employed the projected subgradient scheme, which is guaranteed to converge to global optimum of the dual problem. But these methods are all sensitive to the step size choice. AD3 [101] and ADSAL [120] provides a different way to address this problem via smoothing the dual problem to avoid local sub-optimum fixed points.

3.5.4 Combinatorial algorithms

Kappes *et al.* [65] studied several exact optimization techniques empirically for MRF inference problems, including the integer linear programming (ILP), breadth-rotating AND/OR branch-and-bound (BRAO-BB), multiway cut, max cut by branch-and-cut (MCBC) and max cut using reweighted perfect matching (RPM) with various model reduction techniques including persistency-based approaches. Savchynskyy *et al.* [119] found the MRFs derived from vision instances usually contains a large easy part and a small hard part. They proposed an algorithm to automatically identify the large easy part by solving LP relaxation and showing tightness to prove optimality, and solve the small hard part by the combinatorial algorithms. Their algorithm is guaranteed to give global optimum solution.

3.6 Approximate inference algorithms for higher-order MRFs

One major category of higher-order MRF inference algorithms is based on *reduction techniques*, which transforms a higher-order MRF energy $f(\mathbf{x})$ to a pairwise MRF $g(\mathbf{x}, \mathbf{y})$ with auxiliary variables \mathbf{y} such that $f(\mathbf{x}) = \min_{\mathbf{y}} g(\mathbf{x}, \mathbf{y})$. Therefore, we can apply any pairwise MRF inference algorithms to $g(\mathbf{x}, \mathbf{y})$.

People proposed specialized reduction techniques for certain sub-classes of higher-order MRFs, which includes the multilinear polynomials with only negative coefficients [37, 79], concave priors P^n Potts model [69] and robust P^n Potts model [70], sparse higher-order priors [113] (with only a small number of non-zeros) and curvature-based priors [149].

For general higher-order MRFs, the reduction by substitution method [111] enforces $y_{ij} = x_i x_j$ with auxiliary linear and quadratic terms with very large coefficients, which leads to a binary MRF extremely hard to optimize. Ishikawa [57] is the first applicable reduction technique for general higher-order MRFs. Fix *et al.* [34] improves Ishikawa's reduction technique by reducing multiple terms simultaneously.

A different category of approaches generalizes the roof duality for higher-order MRFs. Kolmogorov [75] proposed a bi-submodular relaxation of an arbitrary higher-order MRF. But there is no algorithm to construct or minimize such a relaxation. Generalized Roof duality [62] proposed a class of submodular relaxation for an arbitrary higher-order MRF up to third-order. It finds the best relaxation in that class via linear programming, then solve the relaxed objective function via graphcuts. But the computational cost of this approach is expensive and cannot solve arbitrary higher-order MRFs.

Message passing-based approaches can handle higher-order MRFs naturally with higher computational costs, including LBP [146], TRW [142], dual decomposition with the slave subproblems solved by message passing [80] and HOP-MAP [138].

There are also flow/cut-based approaches for higher-order MRFs. We have already introduced the generalized sum-of-submodular (SoS) flow and cut for hypergraphs [76, 35] in Section 2.5.4. Fix *et al.* [36] provided a primal-dual interpretation of the primal SoS-flow algorithm, which is a generalization of the FastPD algorithm [82]. Arora *et al.* [5] proposed Generic Cut, which can also solve binary submodular higher-order MRFs exactly. Then there are follow up works on generalizing to binary non-submodular higher-order MRFs [6] and multilabel higher-order MRFs [7].

Shanu *et al.* [125] proposed to use the well studied min-norm point algorithm for binary submodular higher-order MRFs with large cliques. It can solve the higher-order MRFs with clique size larger than 100, which cannot be handled by either SoS-flow or Generic Cut before. In the follow up work, Shanu *et al.* [124] proposed a hybrid algorithm to handle higher-order MRFs with both small cliques and large cliques. It gives significantly better results compared to Generic Cut, which can only handle small cliques, and min-norm point algorithm, which can handle large cliques but doesn't perform very well for small cliques.

3.7 Persistency as a pre-processing for approximate inference

Methods that optimally label a subset of the variables can obviously be used to accelerate MRF inference algorithms such as expansion moves. There are a few studies in the literature explored this idea with different persistency algorithms and approximate inference algorithms. For example, Radhakrishnan and Su [109] used DEE to pre-processing st-MINCUT. Alahari *et al.* [3] applied Kovtun's approach as the pre-processing and used expansion move and FastPD as the approximate inference algorithm. Kappes *et al.* [65] used QPBO and MQPBO to reduce the scale of the MRF inference problem before applying some combinatorial algorithms.

4.1 Motivation

A popular approach to the MRF inference problem is to try to find the persistent partial labeling for a subset of the variables [62, 71, 75, 119, 126, 129, 134, 148]. An optimal labeling for a subset of the variables can be used to reduce the difficulty of the inference problem, or can be the basis for a variety of heuristics such as QPBO-I[115].

Algorithms for finding persistent partial labelings are summarized in Figure 4.1 and discussed in Chapter 3. All the existing persistency algorithms in the literature, except for Dead-end Elimination (DEE) [31], impose significant computational costs, using max flow, linear programming or both.

As a motivating research question for this work, we address the problem of finding a persistent partial labeling as efficiently as possible. We propose a condition to guarantee that a labeling is part of every global minimizer, and represent this condition as a system of linear inequalities. We establish a hierarchy of relaxations of the original system and derive a family of tractable sufficient conditions. We then propose an efficient algorithm to find a set of variables satisfying the corresponding conditions. Using the loosest condition in the relaxation hierarchy, we can find a globally optimal subset of variables by running a small number of iterations, each of which takes $O(N + M)$ time; this is very efficient compared to the $O(N^2M)$ running time of max-flow.¹ The hierarchy of

¹To be precise, $O(N + M)$ is the running time of our inner subroutine, which finds a globally

Table 4.1: Comparison of persistency algorithms. The bottleneck column indicates any subroutine with complexity significantly greater than linear time.

Method	multilabel MRFs	Bottleneck
Our method, PR [145]	Yes	None
DEE [31]	Sometimes	None
QPBO [12]	No	max-flow
Kovtun [83, 84]	Yes	max-flow
Generalized Kovtun [148]	Yes	max-flow
MQPBO [71]	Yes	max-flow
Shekhovtsov [126]	Yes	LP
Swoboda [134]	Yes	LP, MRF inference
Shekhovtsov [129]	Yes	LP, MRF inference

relaxations allow us to trade off between the running time and the number of variables optimally labeled.

Dead-end Elimination (DEE) [31] is the only existing method with cheaper computational costs than max-flow. It checks a local sufficient condition which only involves a single vertex and its adjacent edges. We will show in Section 4.2 that this condition is a special case of the loosest condition of our approach, hence our approach will always label at least as many variables as DEE, with the same running time complexity. Experimental results confirm our approach can label substantially more variables than DEE.

An intuitive comparison of our technique with DEE is provided in Figure 4.1. The most striking difference is that DEE condition considers *one individual variable* at a time and potentially rules out one of its labels; for binary problems, this allows it to determine the globally optimal label for that variable. Our method, as a generalization of DEE, can determine whether a particular label is optimal *for a set of variables*, and is *not restricted to binary labels*.

optimal subset of variables that increases on each iteration. In practice this needs to be run a very limited number of times, as shown in the experimental results that run 5 iterations.

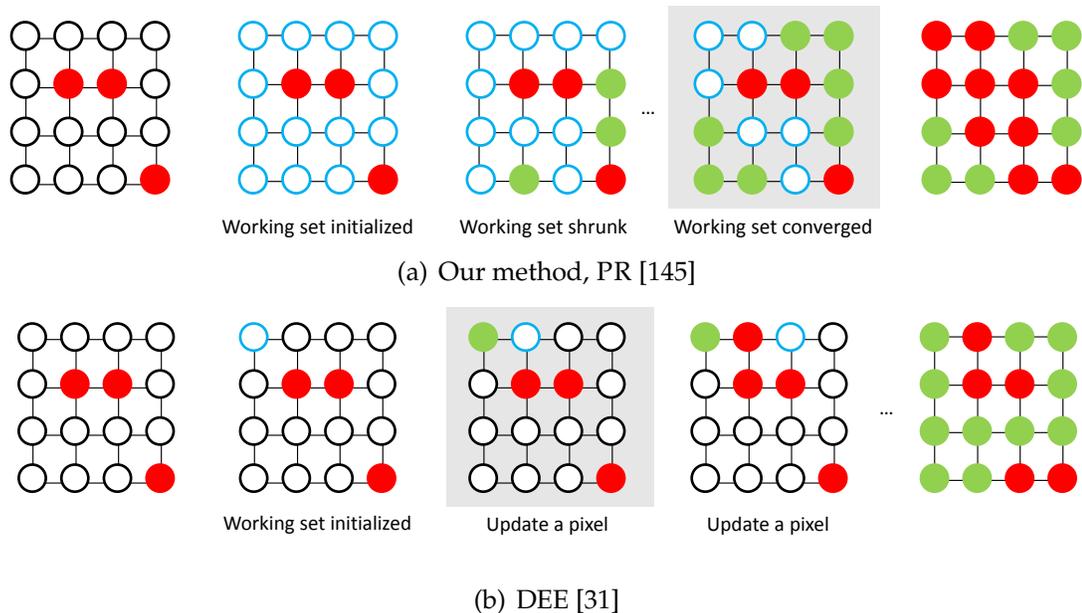


Figure 4.1: High-level comparison between our methods [145] and DEE [31], running on a binary-valued MRF with 16 variables. Optimally labeled variables are shown in red, while the working set is light blue. Green variables fail the sufficient test to be optimally labeled. The key step of each algorithm is highlighted with a grey background.

Due to the nature of DEE condition rule out non-persistent labels, while our condition rule in persistent labels. The DEE algorithm takes the expansion scheme to construct the persistent partial labeling by checking DEE condition for each variable one by one. Meanwhile, our approach uses the shrink scheme. When an entire set of variables fails our sufficient condition for optimality, we shrink the set, as shown in the middle one in Figure 4.1(a), until we converge to a subset of variables we can prove persistency. It is straightforward to see shrinking from the whole set of variables can potentially give us larger persistent partial labeling than expanding from a single variable. Note that the crucial step in our method is the second from the last one in Figure 4.1(a), highlighted with a gray background, where a group of 6 variables is given their optimal labels all at once.

4.1.1 Outline of the chapter

In Section 4.2, we studied the *persistence decision problem*, which determine if the given partial labeling \mathbf{x}_S is persistent for the given energy function given the energy function $f(\mathbf{x})$. We will formalize our hierarchical relaxation of persistence and a fast sound algorithm to check them.

In Section 4.3, we studied the *persistence construction problem*, which finds a persistent partial labeling \mathbf{x}_S as large as possible given the energy function $f(\mathbf{x})$, where the size of partial labeling is defined by $|S|$.

The construction problem is the one which end user cares about. While the decision problem is the center subroutine we used to the construction problem, which should be transparent to the end users.

We will establish the theoretical connection between our methods and the existing methods in the literature in Section 4.4.

We will finally present the empirical study of the proposed method in Section 4.5, which demonstrates that our technique labels a large number of variables with minimal overhead, thus producing a substantial speedup, compared to the baseline methods.

4.2 Persistence decision problem

Since there are exponentially many inequalities in (2.21), it is computationally intractable to examine them one by one. Moreover, the persistence decision problem is NP-complete [12] so that we cannot expect to check it exactly. To

handle it, we will establish a hierarchical relaxation of the autarky inequality system (2.21), which gives us a family of sufficient conditions to check persistency.

4.2.1 Base relaxation of persistency

The center concept in the definition of autarky is the energy change when we perform label substitution. Let's define the energy change Δf when we substitute \mathbf{y}_S for \mathbf{x}_S with the remaining variables $\mathbf{z}_{V \setminus S}$ untouched.

$$\Delta f(\mathbf{y}_S \leftarrow \mathbf{x}_S \mid \mathbf{z}_{V \setminus S}) := f(\mathbf{y}_S \oplus \mathbf{z}_{V \setminus S}) - f(\mathbf{x}_S \oplus \mathbf{z}_{V \setminus S}). \quad (4.1)$$

Let the index set $A := \{i \in S \mid y_i \neq x_i\}$ be the indices of variables we actually changed from \mathbf{x}_S to \mathbf{y}_S . Then we know that

$$\Delta f(\mathbf{y}_A \leftarrow \mathbf{x}_A \mid \mathbf{x}_{S \setminus A}, \mathbf{z}_{V \setminus S}) = \Delta f(\mathbf{y}_S \leftarrow \mathbf{x}_S \mid \mathbf{z}_{V \setminus S}), \quad (4.2)$$

and the autarky property in (2.21) is equivalent to

$$\min_{\mathbf{z}_{V \setminus S} \in \mathcal{L}_{V \setminus S}} \min_{\mathbf{y}_i \neq \mathbf{x}_i, i \in A} \Delta f(\mathbf{y}_A \leftarrow \mathbf{x}_A \mid \mathbf{x}_{S \setminus A}, \mathbf{z}_{V \setminus S}) > 0, \forall A \subseteq S, A \neq \emptyset. \quad (4.3)$$

We can expand the definition of energy using (1.21) and cancel out the unchanged terms. Then a further relaxation could be taken by pushing the min

operators into the summation:

$$\begin{aligned}
& \min_{\mathbf{z}_{V \setminus S} \in \mathcal{L}_{V \setminus S}} \min_{\mathbf{y}_i \neq \mathbf{x}_i, i \in A} \Delta f(\mathbf{y}_A \leftarrow \mathbf{x}_A \mid \mathbf{x}_{S \setminus A}, \mathbf{z}_{V \setminus S}) \\
&= \min_{\mathbf{z}_{V \setminus S} \in \mathcal{L}_{V \setminus S}} \min_{\mathbf{y}_i \neq \mathbf{x}_i, i \in A} \left(\sum_{i \in A} (\theta_i(\mathbf{y}_i) - \theta_i(\mathbf{x}_i)) + \sum_{ij \in (A, S \setminus A)} (\theta_{ij}(\mathbf{y}_i, \mathbf{x}_j) - \theta_{ij}(\mathbf{x}_i, \mathbf{x}_j)) \right. \\
&\quad \left. + \sum_{ij \in (A, V \setminus S)} (\theta_{ij}(\mathbf{y}_i, \mathbf{z}_j) - \theta_{ij}(\mathbf{x}_i, \mathbf{z}_j)) + \sum_{ij \in (A, A)} (\theta_{ij}(\mathbf{y}_i, \mathbf{y}_j) - \theta_{ij}(\mathbf{x}_i, \mathbf{x}_j)) \right) \\
&\geq \sum_{i \in A} \min_{\mathbf{y}_i \neq \mathbf{x}_i} (\theta_i(\mathbf{y}_i) - \theta_i(\mathbf{x}_i)) + \sum_{ij \in (A, S \setminus A)} \min_{\mathbf{y}_i \neq \mathbf{x}_i} (\theta_{ij}(\mathbf{y}_i, \mathbf{x}_j) - \theta_{ij}(\mathbf{x}_i, \mathbf{x}_j)) \\
&\quad + \sum_{ij \in (A, V \setminus S)} \min_{\mathbf{y}_i \neq \mathbf{x}_i, \mathbf{z}_j \in \mathcal{L}_j} (\theta_{ij}(\mathbf{y}_i, \mathbf{z}_j) - \theta_{ij}(\mathbf{x}_i, \mathbf{z}_j)) + \sum_{ij \in (A, A)} \min_{\mathbf{y}_i \neq \mathbf{x}_i, \mathbf{y}_j \neq \mathbf{x}_j} (\theta_{ij}(\mathbf{y}_i, \mathbf{y}_j) - \theta_{ij}(\mathbf{x}_i, \mathbf{x}_j)).
\end{aligned} \tag{4.4}$$

Here $ij \in (A, B)$ is short for $\{(i, j) \in E \mid i \in A, j \in B\}$.

In order to simplify the notation in (4.4), we define a shorthand for each term inside the summation as follows ²:

$$\begin{aligned}
\delta_i(\mathbf{x}_i) &:= \min_{\mathbf{y}_i \neq \mathbf{x}_i} \theta_i(\mathbf{y}_i) - \theta_i(\mathbf{x}_i) \\
\delta_{ij}^i(\mathbf{x}_i, \mathbf{x}_j) &:= \min_{\mathbf{y}_i \neq \mathbf{x}_i} (\theta_{ij}(\mathbf{y}_i, \mathbf{x}_j) - \theta_{ij}(\mathbf{x}_i, \mathbf{x}_j)) \\
\delta_{ij}^i(\mathbf{x}_i, \cdot) &:= \min_{\mathbf{y}_i \neq \mathbf{x}_i, \mathbf{z}_j \in \mathcal{L}_j} (\theta_{ij}(\mathbf{y}_i, \mathbf{z}_j) - \theta_{ij}(\mathbf{x}_i, \mathbf{z}_j)) \\
\delta_{ij}^{ij}(\mathbf{x}_i, \mathbf{x}_j) &:= \min_{\mathbf{y}_i \neq \mathbf{x}_i, \mathbf{y}_j \neq \mathbf{x}_j} (\theta_{ij}(\mathbf{y}_i, \mathbf{y}_j) - \theta_{ij}(\mathbf{x}_i, \mathbf{x}_j))
\end{aligned} \tag{4.5}$$

The relationship between A, S and V and the notation we introduced in (4.5) is illustrated in Figure 4.2. Note that all the unary terms and pairwise terms inside $V \setminus A$ are unchanged. So $\delta_i(\mathbf{x}_i)$, which is the minimum energy change (possibly negative) in unary cost θ_i inside A , and $\delta_{ij}^{ij}(\mathbf{x}_i, \mathbf{x}_j), \delta_{ij}^i(\mathbf{x}_i, \mathbf{x}_j), \delta_{ij}^i(\mathbf{x}_i, \cdot)$, which are the minimum energy change in pairwise cost θ_{ij} inside A and crossing the boundary of A respectively, are the only terms we need to focus on.

²In general, our notation $\delta_A^B(x_A)$ represents the smallest energy change for one term θ_A with initial labeling x_A , minimizing over all different values of all the variables in B . Sometimes we need to consider variables not in B to have arbitrary values, which we represent with a period. For the unary term, the superscript in $\delta_i^i(x_i)$ is redundant, hence omitted.

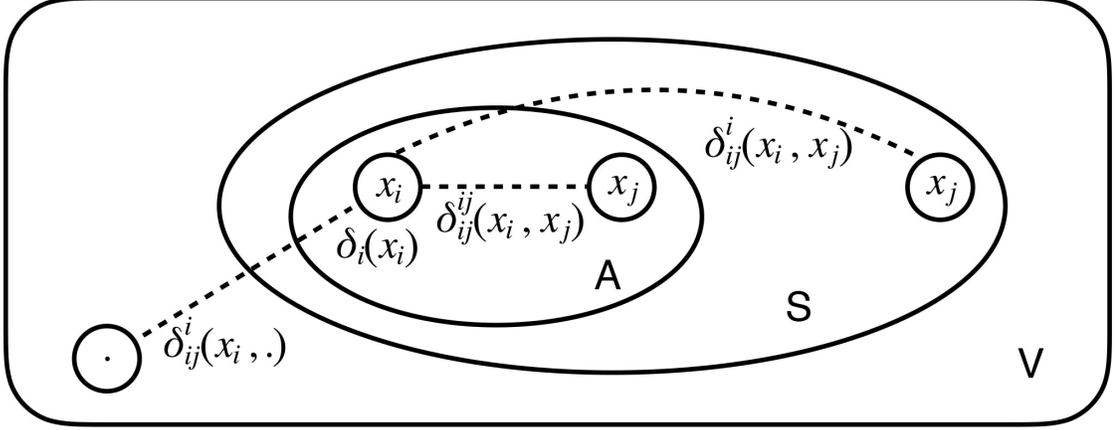


Figure 4.2: Visualization of the terms in our relaxation (4.4). To prove the persistency of the partial labeling \mathbf{x}_S , we show that every subset $A \subseteq S$ meets certain conditions. All the variables in A change from their values in \mathbf{x}_S , the variables in $S \setminus A$ stay the same, and the ones in $V \setminus S$ are arbitrary.

Now we can summarize our analysis above to obtain our first relaxation of the sufficient conditions.

Theorem 4.2.1 (base relaxation of persistency). *The partial labeling x_S is persistent if we satisfy the following condition:*

$$\sum_{i \in A} \delta_i(x_i) + \sum_{ij \in (A, S \setminus A)} \delta_{ij}^i(x_i, x_j) + \sum_{ij \in (A, V \setminus S)} \delta_{ij}^i(x_i, \cdot) + \sum_{ij \in (A, A)} \delta_{ij}^{ij}(x_i, x_j) > 0, \forall A \subseteq S, A \neq \emptyset. \quad (4.6)$$

Proof. For arbitrary $\mathbf{x}_{V \setminus S} \in \mathcal{L}_{V \setminus S}$ and $\mathbf{y}_S \in \mathcal{L}_S$ such that $\mathbf{x}_S \neq \mathbf{y}_S$, let $A = \{i \in S \mid x_i \neq y_i\}$. Since $\mathbf{x}_S \neq \mathbf{y}_S$, we must have $A \neq \emptyset$. Now we have:

$$\begin{aligned} & f(\mathbf{y}_S \oplus \mathbf{x}_{V \setminus S}) - f(\mathbf{x}_S \oplus \mathbf{x}_{V \setminus S}) \\ &= \sum_{i \in A} (\theta_i(y_i) - \theta_i(x_i)) + \sum_{ij \in (A, S \setminus A)} (\theta_{ij}(y_i, x_j) - \theta_{ij}(x_i, x_j)) \\ & \quad + \sum_{ij \in (A, V \setminus S)} (\theta_{ij}(y_i, x_j) - \theta_{ij}(x_i, x_j)) + \sum_{ij \in (A, A)} (\theta_{ij}(y_i, y_j) - \theta_{ij}(x_i, x_j)) \quad (4.7) \\ &\geq \sum_{i \in A} \delta_i(x_i) + \sum_{ij \in (A, S \setminus A)} \delta_{ij}^i(x_i, x_j) + \sum_{ij \in (A, V \setminus S)} \delta_{ij}^i(x_i, \cdot) + \sum_{(i,j) \in (A, A)} \delta_{ij}^{ij}(x_i, x_j) \\ &> 0, \end{aligned}$$

where the first equation above is by expanding the definition of energy function, and the second inequality is due to our definition of $\delta_i(x_i)$, $\delta_{ij}^i(x_i, x_j)$, $\delta_{ij}^i(x_i, \cdot)$ and $\delta_{ij}^{ij}(x_i, x_j)$. \square

4.2.2 Independent local minimum labeling

Note that there are still $O(2^{|S|})$ inequalities in (4.6), and there is no efficient algorithm to check it in general. To further relax it, we will focus on testing the persistency of a particular family of partial labeling, naming the independent local minimum (ILM) labeling defined as follows.

Definition 4.2.1 (independent local minimum labeling). A partial labeling \mathbf{x}_S is called an independent local minimum if it minimizes each pairwise term θ_{ij} such that $i, j \in S$.

Example 4.2.1. Imagine we have $\mathbf{x} = (x_1, x_2, x_3)$, and $\mathcal{L}_1 = \mathcal{L}_2 = \mathcal{L}_3 = \{0, 1\}$. We have two pairwise terms $\theta_{12} = \llbracket x_1 = x_2 \rrbracket$ and $\theta_{23} = \llbracket x_2 \neq x_3 \rrbracket$. Then each single variable with any label label $(x_1 = 0, x_1 = 1, x_2 = 0, x_2 = 1, x_3 = 0, x_3 = 1)$ is an ILM labeling. $\mathbf{x}_{12} = (0, 0)$ and $(1, 1)$ minimize θ_{12} , hence they are ILM labelings, but $\mathbf{x}_{12} = (0, 1)$ and $(1, 0)$ are not. Similarly, $\mathbf{x}_{23} = (0, 1)$ and $(1, 0)$ are ILM labelings while $\mathbf{x}_{23} = (0, 0)$ and $(1, 1)$ are not. Furthermore, $\mathbf{x} = (0, 0, 1)$ and $(1, 1, 0)$ are ILM labelings as well, since they minimize both θ_{12} and θ_{23} .

Remark. ILM labeling might not exist for certain S . For example, if we add one more pairwise term $\theta_{13} = \llbracket x_1 = x_3 \rrbracket$ to Example 4.2.1, then there is no ILM labeling for $S = V = \{1, 2, 3\}$. Since θ_{12} and θ_{13} requires $x_1 = x_2$ and $x_1 = x_3$ but θ_{23} requires $x_2 \neq x_3$.

Restricting to ILM labeling is certainly a limitation to the persistency decision problem itself. However, it is not a concern for the overall problem, due to the following reasons.

- Our algorithm solves the persistency construction problem, and the decision problem is simply an internal subroutine, so this restriction is not imposed on users of our method.
- In the case that the pairwise terms dominates the unary terms, ILM labelings, which minimize each pairwise terms are more likely to be persistent labelings. Therefore, we may only miss a small set of detectable persistent labelings by only looking at ILM labelings. In the other case that unary terms dominates pairwise terms, we may even prove the persistency just by a few local variables, hence we don't worry about we have to find a large ILM labeling for the decision problem.
- ILM labeling contains a single variable with arbitrary label as a special case, therefore, the persistency criteria on ILM labeling still covers the most common cases in the existing persistency algorithm which focus on the persistency of a single variable.
- We can show that for typical vision problems, it is easy to construct a very large ILM labeling. Since the pairwise terms used in vision applications are primarily used to encourage label smoothness, it suggests we prefer to have the same labels for adjacent variables. This is an extremely common assumption in MRFs, since the pairwise costs to enforce the smoothness of the labeling. The most widely used energies, including the Potts energy and truncated convex energy, all fulfill this property, as do all the benchmark MRFs proposed in [136]. Therefore, for those energies, any constant

labeling will be an ILM labeling. More details will be discussed in Section 4.3.1.

- As a practical matter, the restriction to an ILM labeling has very minor impact. We will show that we can find very large ILM labeling in typical vision problems empirically in Section 4.5.5.

4.2.3 k -condition and hierarchical relaxation

The direct benefit from the ILM labeling assumption is we have $\delta_{ij}^i(x_i, x_j) \geq 0$ and $\delta_{ij}^{ij}(x_i, x_j) \geq 0$ by definition that (x_i, x_j) minimizes θ_{ij} . Now we can obtain a hierarchy of relaxations to (4.6) as follows.

Theorem 4.2.2 (k -condition for S). *The ILM partial labeling \mathbf{x}_S containing at least k variables is persistent if $\forall B \subseteq S, |B| = k \geq 1$, the following inequalities hold:*

$$\sum_{i \in C} \delta_i(x_i) + \sum_{ij \in (C, B \setminus C)} \delta_{ij}^i(x_i, x_j) + \sum_{ij \in (C, V \setminus S)} \delta_{ij}^i(x_i, \cdot) > 0, \quad \forall C \subseteq B, C \neq \emptyset \quad (4.8)$$

Proof. We will show for arbitrary non-empty set $A \subseteq S$, (4.6) will be satisfied, hence \mathbf{x}_S is persistent by Theorem 4.2.1. There are two cases.

Case 1: Suppose $0 < |A| \leq k$, find any super set $B \supseteq A$ such that $|B| = k$, we will have:

$$\begin{aligned} & \sum_{i \in A} \delta_i(x_i) + \sum_{ij \in (A, S \setminus A)} \delta_{ij}^i(x_i, x_j) + \sum_{ij \in (A, V \setminus S)} \delta_{ij}^i(x_i, \cdot) + \sum_{ij \in (A, A)} \delta_{ij}^{ij}(x_i, x_j) \\ & \geq \sum_{i \in A} \delta_i(x_i) + \sum_{ij \in (A, S \setminus A)} \delta_{ij}^i(x_i, x_j) + \sum_{ij \in (A, V \setminus S)} \delta_{ij}^i(x_i, \cdot) \\ & \geq \sum_{i \in A} \delta_i(x_i) + \sum_{ij \in (A, B \setminus A)} \delta_{ij}^i(x_i, x_j) + \sum_{ij \in (A, V \setminus S)} \delta_{ij}^i(x_i, \cdot) \\ & > 0, \end{aligned} \quad (4.9)$$

where the first step is due to $\delta_{ij}^{ij}(x_i, x_j) \geq 0$ for ILM labeling, the second step is due to $(A, B \setminus A) \subseteq (A, S \setminus A)$ and $\delta_{ij}^i(x_i, x_j) \geq 0$ and the last step is due to the k -condition for B and $C := A \subseteq B$.

Case 2: Suppose $|A| > k$, find all A 's subset with cardinality k , i.e., $\mathcal{B} = \{B \mid B \subseteq A, |B| = k\}$ and we have:

$$\begin{aligned}
& \sum_{i \in A} \delta_i(x_i) + \sum_{ij \in (A, S \setminus A)} \delta_{ij}^i(x_i, x_j) + \sum_{ij \in (A, V \setminus S)} \delta_{ij}^i(x_i, \cdot) + \sum_{ij \in (A, A)} \delta_{ij}^{ij}(x_i, x_j) \\
& \geq \sum_{i \in A} \delta_i(x_i) + \sum_{ij \in (A, V \setminus S)} \delta_{ij}^i(x_i, \cdot) \\
& = \frac{1}{\binom{|A|-1}{k-1}} \cdot \binom{|A|-1}{k-1} \left(\sum_{i \in A} \delta_i(x_i) + \sum_{ij \in (A, V \setminus S)} \delta_{ij}^i(x_i, \cdot) \right) \tag{4.10} \\
& = \frac{1}{\binom{|A|-1}{k-1}} \cdot \sum_{B \in \mathcal{B}} \left(\sum_{i \in B} \delta_i(x_i) + \sum_{ij \in (B, V \setminus S)} \delta_{ij}^i(x_i, \cdot) \right) \\
& > 0,
\end{aligned}$$

where the first step, again, is due to $\delta_{ij}^i(x_i, x_j) \geq 0$ and $\delta_{ij}^{ij}(x_i, x_j) \geq 0$ for ILM labeling, the third equations is just a re-arrangement of terms in the double summation and the last step is due to the k -condition for B and $C := B$, note that $\sum_{ij \in (B, B \setminus B)} \delta_{ij}^i(x_i, x_j)$ vanishes since $(B, B \setminus B) = \emptyset$. \square

Our k -condition in (4.8) is a hierarchy of relaxations of (4.6) for different k 's. There are $\binom{|S|}{k} (2^k - 1)$ inequalities in the k -condition for S , hence it is computationally efficient to check when k is small. Meanwhile, the larger k is, the more tightly (4.8) approximates (4.6). We thus obtain a trade-off between the complexity and accuracy of the relaxation by varying k .

Example 4.2.2 (a simple case that our 1-condition outperforms DEE condition). Note that DEE is based on such a strong local condition that it may fail even in extremely simple cases. Consider a binary Potts MRF with two variables x_i, x_j

such that $\theta_i(0) = \theta_j(0) = 0, \theta_i(1) = \theta_j(1) = a \geq 0, \theta_{ij}(0, 0) = \theta_{ij}(1, 1) = 0, \theta_{ij}(0, 1) = \theta_{ij}(1, 0) = b > a$. DEE cannot determine any of the variables to be persistent while our approach will easily find that $x_i = x_j = 0$ is a persistent partial labeling. Our experiments demonstrate that our approach indeed finds significantly more persistent variables than DEE.

4.2.4 Approximating the k -condition

Recall our k -condition consists of $\binom{|S|}{k}(2^k - 1)$ inequalities, so checking them one by one will become computationally intractable soon with the growth of k . Therefore, we propose an approximate way to check the k -condition that is very efficient in practice, based on the following lemma. We call these condition an *approximate k -condition*, which is a further relaxation of the k -condition, hence it is still a sound persistency condition.

Lemma 4.2.3. *The ILM partial labeling \mathbf{x}_S is persistent if we can partition S into disjoint subsets $S = \bigcup_t S_t$ and each S_t satisfies the corresponding $|S_t|$ -condition.*

Proof. Again, we will show for arbitrary non-empty set $A \subseteq S$, (4.6) will be satisfied, hence \mathbf{x}_S is persistent by Theorem 4.2.1.

Define $A_t := A \cap S_t$, it is easy to see $A = \bigcup_t A_t$ and A_t are all disjoint. Our goal is to show $\sum_{i \in A} \delta_i(x_i) + \sum_{ij \in (A, S \setminus A)} \delta_{ij}^i(x_i, x_j) + \sum_{ij \in (A, A)} \delta_{ij}^{ij}(x_i, x_j) + \sum_{ij \in (A, V \setminus S)} \delta_{ij}^i(x_i, \cdot) > 0$ is positive. The high-level idea is that we can prove the following by dropping

non-negative terms and rearranging things,

$$\begin{aligned}
& \sum_{i \in A} \delta_i(x_i) + \sum_{ij \in (A, S \setminus A)} \delta_{ij}^i(x_i, x_j) + \sum_{ij \in (A, A)} \delta_{ij}^{ij}(x_i, x_j) + \sum_{ij \in (A, V \setminus S)} \delta_{ij}^i(x_i, \cdot) \\
& \geq \sum_t \left(\sum_{i \in A_t} \delta_i(x_i) + \sum_{ij \in (A_t, S_t \setminus A_t)} \delta_{ij}^i(x_i, x_j) + \sum_{ij \in (A_t, V \setminus S_t)} \delta_{ij}^i(x_i, \cdot) \right).
\end{aligned} \tag{4.11}$$

Then we know the RHS is positive due to each S_t satisfying the $|S_t|$ -condition.

Here's the formal proof of the above idea. We have:

$$\begin{aligned}
& \sum_{i \in A} \delta_i(x_i) + \sum_{ij \in (A, S \setminus A)} \delta_{ij}^i(x_i, x_j) + \sum_{ij \in (A, V \setminus S)} \delta_{ij}^i(x_i, \cdot) + \sum_{ij \in (A, A)} \delta_{ij}^{ij}(x_i, x_j) \\
& \geq \sum_{i \in A} \delta_i(x_i) + \sum_{ij \in (A, S \setminus A)} \delta_{ij}^i(x_i, x_j) + \sum_{ij \in (A, V \setminus S)} \delta_{ij}^i(x_i, \cdot) \\
& \geq \sum_t \sum_{i \in A_t} \delta_i(x_i) + \sum_t \sum_{ij \in (A_t, S_t \setminus A_t)} \delta_{ij}^i(x_i, x_j) + \sum_t \sum_{ij \in (A_t, V \setminus S_t)} \delta_{ij}^i(x_i, \cdot) \\
& = \sum_t \left(\sum_{i \in A_t} \delta_i(x_i) + \sum_{ij \in (A_t, S_t \setminus A_t)} \delta_{ij}^i(x_i, x_j) + \sum_{ij \in (A_t, V \setminus S_t)} \delta_{ij}^i(x_i, \cdot) \right) \\
& > 0
\end{aligned} \tag{4.12}$$

where the last step is due to the $|S_t|$ -condition for $B := S_t$ and $C := A_t \subseteq S_t$.

In the second inequality, it is easy to see $\sum_{i \in A} \delta_i(x_i) = \sum_t \sum_{i \in A_t} \delta_i(x_i)$ since $\{A_t\}$ is a disjoint partition of A , we just use the double summation to re-arrange $\delta_i(x_i)$'s. $\sum_{ij \in (A, S \setminus A)} \delta_{ij}^i(x_i, x_j) \geq \sum_t \sum_{ij \in (A_t, S_t \setminus A_t)} \delta_{ij}^i(x_i, x_j)$ and $\sum_{ij \in (A, V \setminus S)} \delta_{ij}^i(x_i, \cdot) \geq \sum_t \sum_{ij \in (A_t, V \setminus S_t)} \delta_{ij}^i(x_i, \cdot)$ are non-trivial and we will show next.

We want to show $\sum_{ij \in (A, S \setminus A)} \delta_{ij}^i(x_i, x_j) \geq \sum_t \sum_{ij \in (A_t, S_t \setminus A_t)} \delta_{ij}^i(x_i, x_j)$. Firstly, we have $\sum_t \sum_{ij \in (A_t, S_t \setminus A_t)} \delta_{ij}^i(x_i, x_j) = \sum_{ij \in \bigcup_t (A_t, S_t \setminus A_t)} \delta_{ij}^i(x_i, x_j)$, since A_t are all disjoint, hence edge set $(A_t, S_t \setminus A_t)$ are all disjoint as well. Next, we can see $\bigcup_t (A_t, S_t \setminus A_t) \subseteq (A, S \setminus A)$ since we have $A_t \subseteq A, S_t \setminus A_t \subseteq S \setminus A$ by their definition, hence $(A_t, S_t \setminus A_t) \subseteq (A, S \setminus A)$ for each t . Finally, we have $\delta_{ij}^i(x_i, x_j) \geq 0$ for ILM labeling, so putting things together, we have the desired inequality.

Algorithm 4.1: Approximate k -condition Test

Input: ILM partial labeling \mathbf{x}_S

Output: a certificate proves \mathbf{x}_S is persistent or a set of variable U causing we fail to prove persistency of \mathbf{x}_S

```
1  $U \leftarrow S$ ;  
2  $t \leftarrow 0$ ;  
3 for  $i \in U$  s.t.  $\{i\}$  fails 1-condition test do  
4   for  $k' \leftarrow 2$  to  $k$  do  
5     search for  $B \subseteq U$  s.t.  $i \in B$ ,  $|B| = k'$ ,  $B$  satisfies  $k'$ -condition;  
6     if find such a  $B$  then  
7        $t \leftarrow t + 1$ ;  
8        $S_t \leftarrow B$ ;  
9        $U \leftarrow U \setminus B$ ;  
10      break;  
11    end  
12  end  
13 end  
14 for  $i \in U$  s.t.  $\{i\}$  satisfies 1-condition do  
15    $t \leftarrow t + 1$ ;  
16    $S_t \leftarrow \{i\}$ ;  
17    $U \leftarrow U \setminus \{i\}$ ;  
18 end  
19 if  $U = \emptyset$  then  
20   return  $\mathbf{x}_S$  is persistent;  
21 else  
22   return  $U$  as the cause that  $\mathbf{x}_S$  fails the test;  
23 end
```

We also want to show $\sum_{ij \in (A, V \setminus S)} \delta_{ij}^i(x_i, \cdot) \geq \sum_t \sum_{ij \in (A_t, V \setminus S_t)} \delta_{ij}^i(x_i, \cdot)$. Again, we have $\sum_t \sum_{ij \in (A_t, V \setminus S_t)} \delta_{ij}^i(x_i, \cdot) = \sum_{ij \in \bigcup_t (A_t, V \setminus S_t)} \delta_{ij}^i(x_i, \cdot)$ since $(A_t, V \setminus S_t)$ are all disjoint. Next, we can show $\bigcup_t (A_t, V \setminus S_t) \supseteq (A, V \setminus S)$. Since $\forall (i, j) \in (A, V \setminus S)$, we must be able to find one A_t such that $i \in A_t$ since $A = \bigcup_t A_t$. Now it is easy to show $(i, j) \in (A_t, V \setminus S_t)$ since $V \setminus S \subseteq V \setminus S_t$. Finally, we have $\delta_{ij}^i(x_i, \cdot) \leq 0$ by its definition (given x_i , we can pick any $y_i \neq x_i$ and let $z_j \in \arg \min_z \theta_{ij}(y_i, z)$), so putting things together, we have the desired inequality. \square

In practice, we can approximately test the k -condition for $k > 1$ by doing

an incremental breadth-first search (BFS) style greedy partition of $S = \cup_t S_t$, such that S_t are all disjoint, $|S_t| \leq k$, and S_t satisfies the $|S_t|$ -condition, which is described in Algorithm 4.1. The idea is for each single variable i not satisfying the 1-condition, we will search for a subset B containing i that can satisfy the 2-condition, 3-condition, etc. The first found B will be added to our partition. Finally, we add all the left-over single variables satisfying the 1-condition into our partition and claim the remaining variables (i.e., U at the end of Algorithm 4.1) cannot be proved to be persistent. Note this approximation is still a sound condition guaranteed by Lemma 4.2.3, i.e., when $U = \emptyset$ at the end, we know that \mathbf{x}_S is persistent.

4.3 Persistency construction problem

In Section 4.2, we described a hierarchy of sufficient conditions and their sound approximations to check persistency of an ILM partial labeling. Now, we will use these conditions as a subroutine to construct a persistent partial labeling for a given energy $f(\mathbf{x})$.

The method is shown in Algorithm 4.2. Assume we can find a set of ILM partial labelings \mathcal{X} as candidates (line 4, we defer a discussion of how to do this until Section 4.3.1). For each $\mathbf{x}_S \in \mathcal{X}$, we adopt a shrinking scheme (line 5-16). We will apply the k -condition test³ or its approximation to check the persistency of \mathbf{x}_S (line 7). The test will either proves \mathbf{x}_S is persistent or reports some sets B 's violating (4.8); we will shrink S by removing $i \in B$ with the minimum $\delta_i(x_i) + \sum_{(i,j) \in E} \delta_{ij}^i(x_i, \cdot)$ value for each violated B (line 9-10). If we apply our ap-

³To be completely precise, for the corner case of a tiny labeling with $|S| < k$, we would test the $|S|$ -condition instead of the k -condition.

Algorithm 4.2: Persistency Construction

Input: energy function $f(\mathbf{x})$
Output: persistent partial labeling \mathbf{x}_W of $f(\mathbf{x})$

- 1 $W \leftarrow \emptyset;$
- 2 $\mathbf{x}_W \leftarrow \emptyset;$
- 3 **repeat**
- 4 construct a set of ILM partial labeling \mathcal{X} described in Section 4.3.1;
- 5 **for** $\mathbf{x}_S \in \mathcal{X}$ **do**
- 6 **repeat**
- 7 test \mathbf{x}_S using k -condition or approximated k -condition;
- 8 **if** \mathbf{x}_S *fails the test* **then**
- 9 find x_i causing violation;
- 10 $S \leftarrow S \setminus \{i\};$
- 11 **end**
- 12 **until** \mathbf{x}_S *passes the test* or $S = \emptyset;$
- 13 $\mathbf{x}_W \leftarrow \mathbf{x}_W \oplus \mathbf{x}_S;$
- 14 $W \leftarrow W \cup S;$
- 15 $\mathcal{L}_i \leftarrow \{x_i\}, \forall i \in S;$
- 16 **end**
- 17 **until** *converge* or *after τ iterations*;
- 18 **return** \mathbf{x}_W as the persistent partial labeling of $f(\mathbf{x})$;

proximation to the k -condition, we will remove the remaining variables in U from S . We repeat this procedure until \mathbf{x}_S satisfies the k -condition. Now we can composite all the persistent partial labelings we found together (line 13-15). It is easy to see the composition of persistent partial labelings is still persistent by definition, which proves that our algorithm is sound.

Finally, similar to the iterative idea in DEE, after determining \mathbf{x}_S to be persistent, we can update $f(\mathbf{x})$ by fixing \mathbf{x}_S without changing the minimizer. This in turn can potentially find additional persistent variables. We iteratively run the procedure described above until it converges or reaches the pre-defined stopping parameter τ .

4.3.1 ILM labeling construction

In this section, we will show how to construct a candidate set \mathcal{X} of ILM partial labelings. Ideally, we want to start Algorithm 4.2 with as large a partial labeling as possible. We can show that for a wide family of energy functions used in typical vision problems, we can efficiently find the maximum ILM partial labeling, and even for an arbitrary MRF we can guarantee an ILM partial labeling of at least size 2. We consider three special cases that are widely used in vision: weakly associative energies, binary submodular, and binary non-submodular. Finally, we discuss the case of an arbitrary multilabel MRF.

Definition 4.3.1 (weakly associative energy). $f(\mathbf{x})$ is called weakly associative if all of its pairwise costs satisfy $\theta_{ij}(x_i, x_j) \geq 0$ and $\theta_{ij}(x_i, x_j) = 0$ when $x_i = x_j$.

Weakly associative: It is easy to see that any constant labeling (i.e., all the variables take the same value) is ILM. So for each label ℓ , we can let $S := \{i \mid \ell \in \mathcal{L}_i\}$ and $\mathbf{x}_S := \vec{\ell}$ then put it into \mathcal{X} .

Binary submodular: We use the reparameterization scheme introduced in [77] to equivalently transform the energy function into a weakly associative one. Therefore, we can put the maximum constant partial labeling with label 0 and 1 into our \mathcal{X} .

Binary non-submodular: Again, we use the reparameterization scheme introduced in [77]. Now, all the submodular terms will be transformed to be weakly associative. Meanwhile, all the supermodular terms will be transformed as $\theta_{ij}(0, 0) = \theta_{ij}(1, 1) = 0$ and $\theta_{ij}(0, 1) = \theta_{ij}(1, 0) = c < 0$ with $(0, 1)$ and $(1, 0)$ as the local minimizer. Therefore, in the ILM partial labeling, we want x_i and x_j to take the same value when θ_{ij} is submodular and to take different values other-

wise. We use a greedy approach⁴ to find a large enough ILM partial labeling. After we find the first ILM partial labeling \mathbf{x}_S , we can add \mathbf{x}_S and its complementary $\bar{\mathbf{x}}_S$ into \mathcal{X} ⁵ and iteratively run the greedy algorithm on the remaining variables in $V \setminus S$ to find more ILM partial labelings to be added into \mathcal{X} .

Arbitrary multilabel: The Potts model and truncated L_p prior, the two most widely used multilabel pairwise terms in vision, are weakly associative. Therefore, we can just construct the maximum constant labeling for each label and add them into \mathcal{X} . For an arbitrary multilabel energy, it is hard to find the maximum ILM partial labeling. However, we can still use the greedy algorithm we used in the binary non-submodular case as a good heuristics in practice. This will return multiple ILM partial labelings with size at least 2, which is still better than checking persistency on a single variable as DEE does.

4.4 Theoretical connection to previous works

4.4.1 Connection to DEE

Now we can claim the sufficient condition to check persistency in DEE [31] is a special case of our 1-condition (i.e., $k = 1$). Note that our 1-condition says the constant partial labeling \mathbf{x}_S is persistent if

$$\delta_i(x_i) + \sum_{j \in S, (i,j) \in E} \delta_{ij}^i(x_i, x_j) + \sum_{j \notin S, (i,j) \in E} \delta_{ij}^i(x_i, \cdot) > 0, \quad \forall i \in S, \quad (4.13)$$

⁴Starting from $\mathbf{x}_S = x_1 \in \mathcal{L}_1$, then for $i = 2, 3, \dots, n$, when we can find $x_i \in \mathcal{L}_i$ such that compositing x_i into \mathbf{x}_S is still ILM, then do so.

⁵It is easy to show that after the reparameterization scheme introduced in [77], each pairwise term can be written as $\theta_{ij}(0, 0) = \theta_{ij}(1, 1) = 0$ and $\theta_{ij}(0, 1) = \theta_{ij}(1, 0) = c$. Therefore, either $(0, 0)$ and $(1, 1)$ minimizes θ_{ij} or $(0, 1)$ and $(1, 0)$ minimizes θ_{ij} . Furthermore, when we have \mathbf{x}_S to be ILM, $\bar{\mathbf{x}}_S$ must be ILM as well.

while the Goldstein condition used in DEE says⁶ that variable x_i is persistent if

$$\delta_i(x_i) + \sum_{(i,j) \in E} \delta_{ij}^i(x_i, \cdot) > 0. \quad (4.14)$$

This is a special case of our 1-condition when $S = \{i\}$. Thus, our 1-condition generalized DEE’s Goldstein condition from a single variable to an ILM partial labeling.

Recall that, as shown in Figure 4.1, our method considers an input labeling and then shrinks it, while DEE considers a single pixel at a time. Our input must be an ILM labeling, and the larger the better. Fortunately we will show in Section 4.3.1 that we can easily construct large ILM labelings for the vast majority of MRFs used in vision, and can also guarantee an ILM labeling of at least size 2 for an arbitrary MRF. So even in the worst case we retain our advantage over DEE. Formally, we show prove our method must outperform DEE in term of finding persistent partial labeling. Let P_{DEE} be the set of persistent variables DEE found, and P_{PR} the set of persistent variables our algorithm found at convergence. Our algorithm always finds at least as many persistent variables as DEE does.

Theorem 4.4.1. $P_{\text{DEE}} \subseteq P_{\text{PR}}$ for binary MRFs.

Proof. We prove this by contradiction. Suppose $\exists x_i \in P_{\text{DEE}}, x_i \notin P_{\text{PR}}$, then x_i satisfies the 1-condition at convergence of PR. This contradicts our assumption that PR has converged, since we should have added x_i into P_{PR} .

Assume $x_i = \alpha$ is the first variable which is proved to be persistent by running DEE that is not in P_{PR} . Denote the minimum energy change for unary and

⁶The original Goldstein condition is to claim one label cannot be persistent, which is equivalent to say its opposite is persistent for the binary case. For the multilabel case, we need to check $|\mathcal{L}_i| - 1$ labels cannot be persistent, so that the remaining one is persistent.

pairwise costs as $\delta_i(x_i), \delta_{ij}^i(x_i, x_j)$, label set as \mathcal{L}_i at the time when we test if x_i is persistent. It satisfies $\delta_i(x_i) + \sum_{(i,j) \in E} \delta_{ij}^i(x_i, \cdot) > 0$. Next, considering running PR for one more iteration from its converging status to construct a constant persistent partial labeling using α . Denote the minimum energy change for this iteration as $\bar{\delta}_i(x_i), \bar{\delta}_{ij}^i(x_i, x_j), \bar{\delta}_{ij}^i(x_i, \cdot)$ and label set as $\bar{\mathcal{L}}_i$.

It is easy to see $\delta_i(x_i) = \bar{\delta}_i(x_i)$, which also depends on x_i , since both of them are checking $x_i = \alpha$. We also have $\bar{\delta}_{ij}^i(x_i, \cdot) \geq \delta_{ij}^i(x_i, \cdot)$ since before x_i , DEE only finds a subset of P_{PR} to be persistent, i.e., $\mathcal{L}_j \supseteq \bar{\mathcal{L}}_j, \forall j$ for the binary MRFs, which makes $\bar{\delta}_{ij}^i(x_i, \cdot) = \min_{y_i \neq \alpha, y_j \in \bar{\mathcal{L}}_j} (\theta_{ij}(y_i, y_j) - \theta_{ij}(\alpha, y_j)) \geq \min_{y_i \neq \alpha, y_j \in \mathcal{L}_j} (\theta_{ij}(y_i, y_j) - \theta_{ij}(\alpha, y_j)) = \delta_{ij}^i(x_i, \cdot)$. Therefore, we must have $\bar{\delta}_i(x_i) + \sum_{(i,j) \in E} \bar{\delta}_{ij}^i(x_i, \cdot) > 0$. So when we use the approximation of k -condition to test persistency, x_i will never be removed from S_α since $\{i\}$ satisfies the 1-condition, so it will never be shown in the unpartitioned variable set U . When we use k -condition to test persistency, x_i will also never be removed from S_α during the shrinking procedure. Otherwise, suppose we find $B \subseteq S_\alpha$ violating k -condition and we decided to remove x_i . Recall we choose the knocked out variable with the minimum $\bar{\delta}_i(x_i) + \sum_{(i,j) \in E} \bar{\delta}_{ij}^i(x_i, \cdot)$ value, it means we have $\bar{\delta}_i(x_i) + \sum_{(i,j) \in E} \bar{\delta}_{ij}^i(x_i, \cdot) > 0$ for all $i \in B$, which is a contradiction with B violates the k -condition.

In sum, no matter what variable of PR algorithm we run, we will never remove x_i from S_α . Therefore, x_i will be proved as persistent at the end of the new iteration, which is a contraction to PR has converged. \square

Remark. The nice thing about DEE for binary MRFs is ruling out one label is equivalent to nailing down one variable. That's the key fact for us to claim $\mathcal{L}_j \supseteq \bar{\mathcal{L}}_j, \forall j$ in the proof above. For the multilabel MRFs, we cannot guarantee it. It is possible that DEE have ruled out some labeling from optimal labeling

in \mathcal{L}_j but cannot prove x_j to be persistent since we still have $|\mathcal{L}_j| > 1$. However, in our experiment, we never observed that DEE can prove one variable to be persistent but PR cannot. In general, PR can find substantially more persistent variables than DEE.

Remark. Although in our hierarchical relaxation and decision problem, k -condition is always stronger than $(k+1)$ -condition, we do not necessarily have $P_{\text{PR-}k} \subseteq P_{\text{PR-}(k+1)}$, due to it depends on how to choose the knocked out variable from a violated set B . However, our experiments do indicate PR- $(k+1)$ can usually find significantly more persistent variables than PR- k .

4.4.2 Connection to MRF/LP-based persistency approaches

The autarky property in (2.19) is a special case of the improving mapping described in IRI [126]. Our sufficient conditions in (4.6) is a special case of the partial optimality criteria described in PBP [134], and the shrink scheme used in our construction algorithm is also the same shrink scheme adopted in PBP. However, checking the sufficient conditions in [126, 129, 134] require a general MRF inference solver/LP solver as a subroutine, which is computational expensive. We proposed a set of computational tractable sufficient conditions and approximation algorithm in (4.8) and Section 4.2.4. Therefore, while the sufficient conditions in [126, 129, 134] are tighter, our conditions can be checked more efficiently.

4.5 Experimental results

We will present the empirical study of the proposed persistency relaxation methods in this Section. We conduct experiments on a variety of vision applications and obtain promising experimental results. In particular, when integrated into expansion moves [18] as the MRF inference algorithm our technique labels a large number of variables with minimal overhead, thus producing a substantial speedup.

We organize this section as following. We will first introduce the datasets and the experimental environment in Section 4.5.1. Then we will introduce all the baseline methods and the variant of our proposed methods and its shorthand notations in Section 4.5.2. We will define the quantitative measurements used in our experiments in Section 4.5.3. The overall experimental results will be provided in Section 4.5.4, which will demonstrated the effectiveness and efficiency of the proposed methods. Then we explored the questions on the effectiveness of the greedy ILM labeling construction algorithm (described in Section 4.3.1), the sensitivity analysis to the energy type, graph structure, iteration number τ in Section 4.5.5, 4.5.6, 4.5.7 respectively. We further break down the overall running time into persistency algorithm time and approximate inference algorithm time, and study it in Section 4.5.8. We conducted the preliminary experiments comparing the proposed method on multilabel MRFs and the induced binary MRFs in Section 4.5.9. Finally, we provided a detailed experimental results in Section 4.5.10.

Table 4.2: Datasets description.

Dataset	$ V $	$ \mathcal{L} $	# Instances	Energy
Brain MRI	785540–1413972	5	8	Potts
Color Seg	65536–86400	3–12	18	Potts
Inpainting	14400	4	2	Potts
Middlebury	21838–514080	5–256	7	non-submodular
Scene Decomp	150–208	8	715	non-submodular

4.5.1 Datasets and experimental environment

We conducted experiments on a variety of computer vision benchmarks for MRF inference, including brain-MRI [26], color segmentation [93], inpainting [23], Middlebury MRF dataset (including stereo, image inpainting and photomontage tasks) [136] and scene decomposition [46]. All these datasets are wrapped in OpenGM2 [63] and are available online.

Table 4.2 briefly summarizes the scale of each dataset. Note that the first three datasets use the Potts model, so the binary subproblem is submodular, while the last two have non-submodular (non-weakly associative) subproblems. We have all the datasets label each single pixel using MRFs, except Scene decomposition dataset labels superpixels. We have \mathcal{N}_6 3D grid structure for Brain MRI. We have both \mathcal{N}_4 and \mathcal{N}_8 structure for the Color Segmentation dataset. We have \mathcal{N}_8 structure for the Inpainting dataset, and \mathcal{N}_4 for the Middlebury dataset. The graph structure for Scene Decomposition is explicit, since it is based on superpixels. More details can be found in [63].

All the experiments were executed on a single machine with dual 3GHz Intel i7 Core and 16GB 1600 MHz DDR3 memory.

4.5.2 Approaches

We will focus on three partial optimality based pre-processing techniques in the experiment section, namely DEE [31], Kovtun’s approach [84, 83] and our approach [145]. They will be referred as DEE, KOVTUN and PR (Persistence Relaxation) respectively. We will use PR- k to refer our approach using k -conditions and PR- k -APX when we use its approximation. We experimented with MQPBO [71], but it was too slow to be competitive. LP-based approaches [126, 129, 134] are also not considered due to their computational overhead, which is documented in [129]. We will apply α -expansion as the approximate inference algorithm for MRF [18], using the max-flow algorithm of [17]. We will refer to α -expansion algorithm without any pre-processing technique as α -EXP, which is the baseline against which we compare all other approaches.

The α -expansion algorithm, like most move-making techniques, reduces the multilabel MRF inference problem to a series of binary MRF inference problems. Therefore, we can either (1) apply partial optimality based pre-processing techniques to the multilabel MRF directly and use α -EXP to infer the remaining variables, or (2) in each iteration of α -EXP, apply the pre-processing technique to the induced binary MRF. We will refer to approach (1) as mDEE and mPR, and to (2) as iDEE and iPR. As a small optimization, for iDEE and iPR we only determined which variables do not switch to the new label, except for on the first iteration through the label set. We can also combine these two approaches, i.e., applying pre-processing for both the multilabel MRF and each induced binary MRF. Our experiments shows that it will only provide marginal improvement over only applying pre-processing to each induced binary MRF, so we don’t re-

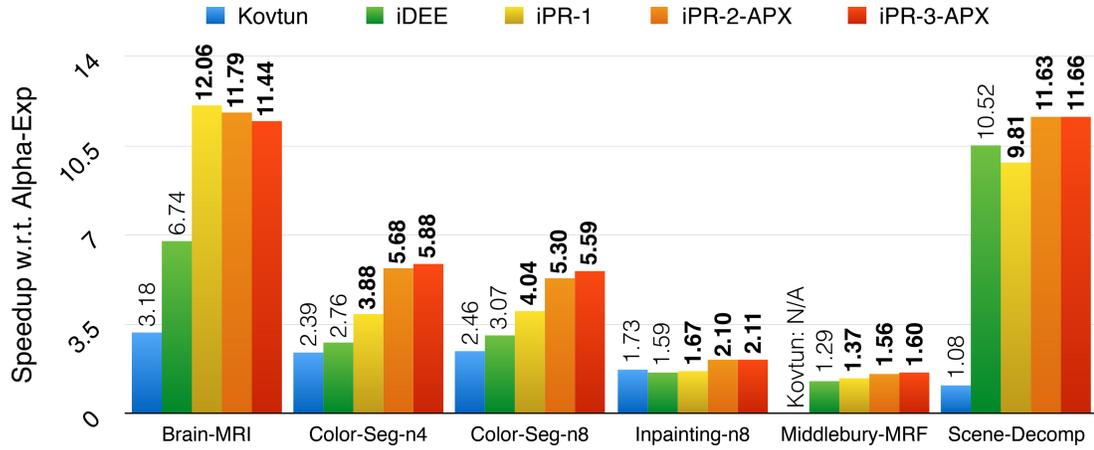
port the results. Note that KOVTUN can only be used in approach (1) since it degenerates to QPBO for the induced binary problem, which is equivalent to the max-flow problem α -EXP needs to solve in each iteration.

4.5.3 Measurement

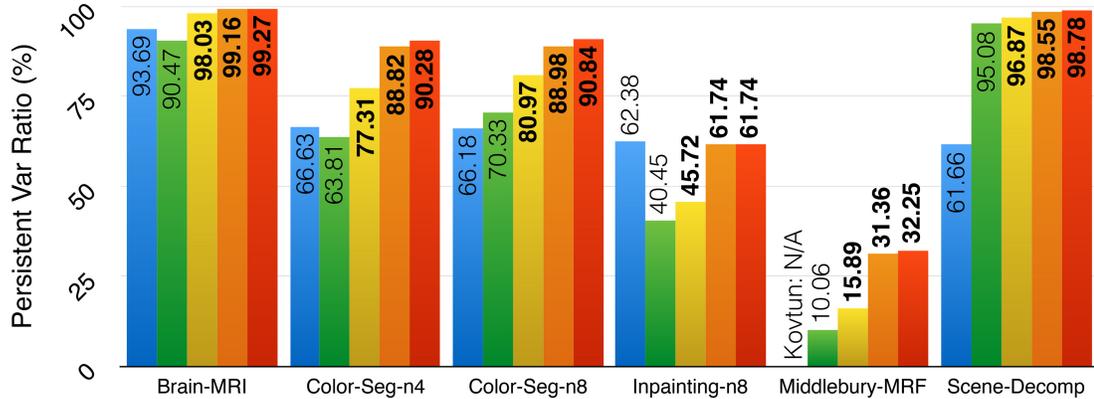
We will evaluate the different approaches in two respects.

First, we report the improvement in overall running time for both the persistency-based pre-processing step and the approximate inference on the remaining undetermined variables. We use α -EXP as the baseline and report the speedup for other methods compared to α -EXP. The reported numbers are first computed for each instance in the dataset, then averaged over all instances in the same dataset. We further break the running time into pre-processing time and flow computation time (for approximate inference) and report them in Section 4.5.10. Note that the time spent in overhead like loading and decompressing the model from input files, writing the log and output are excluded from the total running time.

Second, we report the size of the partial optimal labeling found by the various pre-processing methods as the ratio of the persistent variables found. The reported numbers are first averaged for each iteration of α -EXP in each instance, then averaged over all instances in one dataset.



(a) Speedup with respect to α -EXP



(b) Persistent variables ratio (%)

Figure 4.3: Performance of various methods in terms of speedup and percentage of persistent variables. Higher numbers indicate better performance. Our three methods are at right, with numbers on the chart in bold. KOVTUN was too slow on the Middlebury-MRF dataset.

4.5.4 Experimental results on speedup and persistency ratio

We summarize the experimental results on several benchmarks in Figure 4.3; the detailed numbers are deferred to Section 4.5.10. Besides the baseline technique α -EXP we also show results from iDEE and KOVTUN. Our approaches obtain a 1.5x-12x speedup compared to α -EXP and label significantly more variables than all other methods. For some specific instances, the speedup can be up to 40x; our speedup numbers, of course, include the cost of pre-processing. Note that

KOVTUN was too slow on the Middlebury dataset to be competitive, running at least 5x slower than the baseline algorithm α -EXP. This suggests that when we have a large label set, it is very hard to compute partial optimality on the multilabel MRF directly, which is a major limitation of KOVTUN. We can also see PR-based methods find significantly more persistent variables than all the baseline methods. Per instance analysis (in Section 4.5.10) indicates that our methods are superior on almost all instances.

Figure 4.3 also illustrate the power of the relaxation hierarchy we proposed. For example, on Color-Seg- \mathcal{N}_4 dataset, iPR-1 finds 14% more partial persistent variables than iDEE, iPR-2-APX finds additional 10% more than iPR-1. The gap between iPR-3-APX and iPR-2-APX are less significant, but still exists. It indicates that PR-based approaches significantly outperform the baseline DEE. The further we utilize the hierarchy, the more variables we can label, although the marginal gain is diminishing.

It is also easy to see from the experimental result that the more persistent variables we can prove during the pre-processing step, the more speedup we can gain. Since the flow computation in α -EXP is a relatively computational expensive subroutine, meanwhile we can compute persistency for pre-processing very efficiently. Brain-MRI and Scene Decomposition datasets are the two easier one. Both PR-based methods and the baseline methods prove more than 90% persistent variables during pre-processing and gets a 10x speedup, although PR-based methods perform slightly better. The PR-based methods proves significantly more variables than the baseline methods on the moderate challenging Color Segmentation dataset, so we also got a significant speedup gain compared to the baseline methods. The Inpainting dataset and the Middlebury

dataset are the most challenging dataset. The proposed PR-based methods still proves more persistent variables and have a small speedup gain as well.

4.5.5 Effectiveness on the greedy ILM labeling construction

We covered two common energy types in our dataset, the Potts priors, which gives us submodular binary subproblems, and truncated convex priors, which gives us non-submodular binary subproblem. The experimental results presented in Figure 4.3 has already illustrated that the proposed method is robust on both of these two types of energy functions.

For the submodular binary subproblems, we have shown in Section 4.3.1 that we can just use any constant labeling as a proposed ILM labeling for persistency decision making. However, for the non-submodular binary subproblems, we can only apply the greedy approach to construct the ILM candidate partial labelings used for our construction algorithm. Note that we achieved good performance on Middlebury MRF dataset and the Scene Decomposition dataset (with non-submodular binary subproblems). It indicates that our greedy approach can find substantially large ILM partial labeling in practice, and can compute persistent partial labeling effectively on non-submodular datasets.

4.5.6 Sensitivity analysis on the graph structure

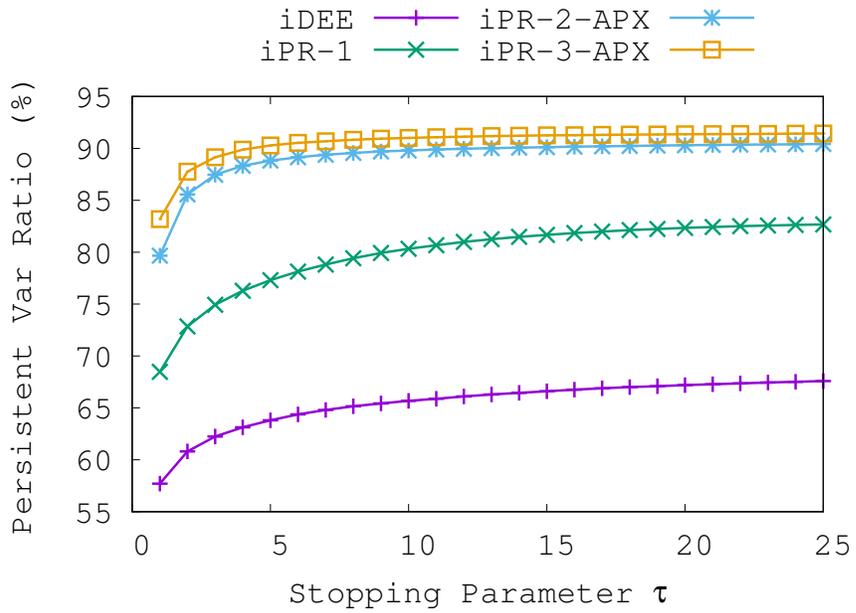
Our dataset also contains a rich collections of common graph structure used in vision problems. We covered \mathcal{N}_6 3D grid structure (Brain MRI), \mathcal{N}_4 (Color Segmentation, Middlebury), \mathcal{N}_8 (Color Segmentation, Inpainting) and the explicit

graph structure (Scene Decomposition).

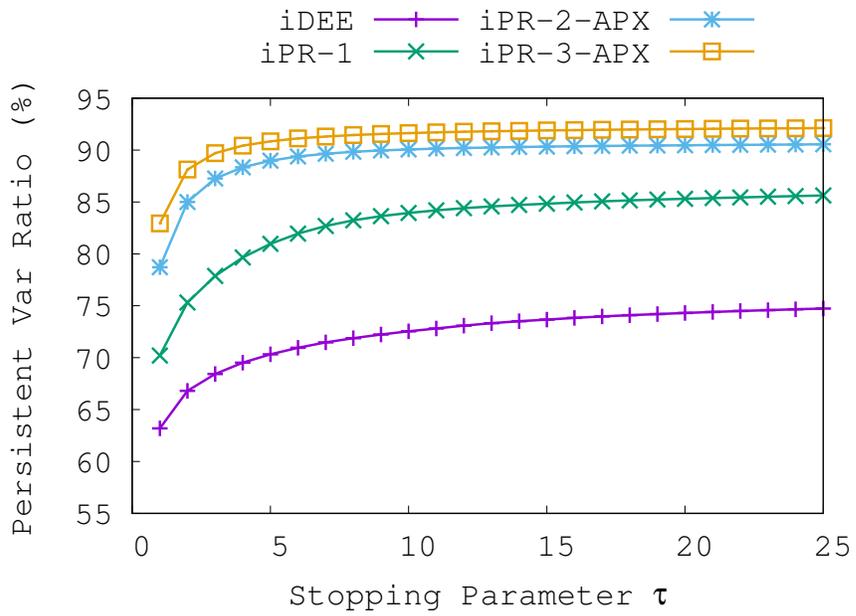
According to the experimental results in Figure 4.3, the PR-based approaches achieve good performance regardless of the grid structure we used in the MRFs. In addition, we have both the \mathcal{N}_4 and \mathcal{N}_8 setup for the same set of images in the Color Segmentation dataset. The PR-based approaches achieve almost identical good performance on both of the MRF setups. It demonstrates that the proposed method is robust to the graph structure in practice.

4.5.7 Sensitivity analysis on number of iterations τ

Our algorithms have one parameter, the maximum number of iterations τ . Figure 4.4 and 4.5 illustrate running DEE and PR-based methods with different τ 's on the Color Segmentation dataset. Similar trend are observed on other datasets. We can see the persistent var ratio converges very quickly with the growth of τ . For all these four approaches listed here, they can find the persistent variables when $\tau = 25$ no less than 1.5% compared to their converging values in our experiments. In general, the overall running time decreases first and then increases due to it is a trade-off between the speed and the quality of the pre-processing step. The proposed approach is not very sensitive to the choice of this stopping parameter, low total running time can be achieved in a very broad range. PR-based approaches significantly outperform DEE and other baseline methods no matter which τ we choose. Figure 4.3 was computed with $\tau = 5$, but other choices produce similar results.



(a) Color-Seg- \mathcal{N}_4 dataset

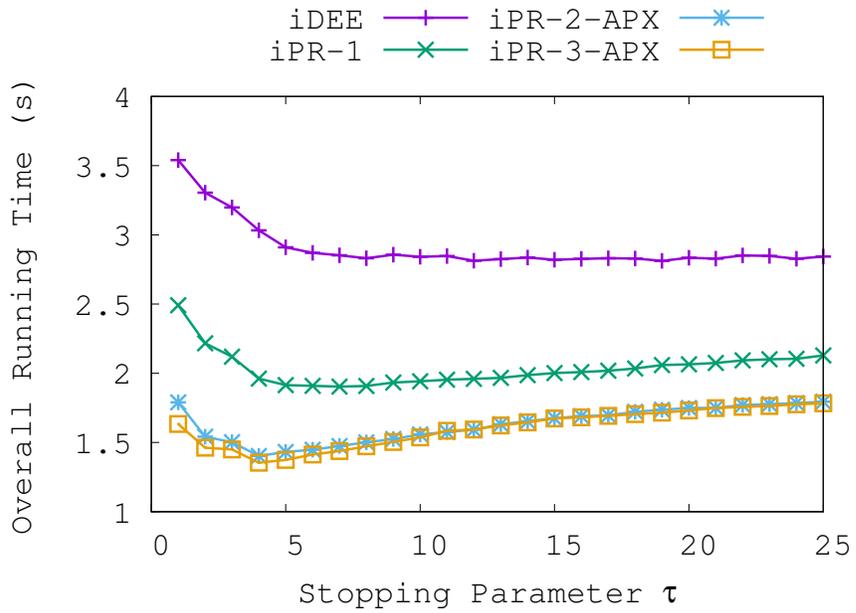


(b) Color-Seg- \mathcal{N}_8 dataset

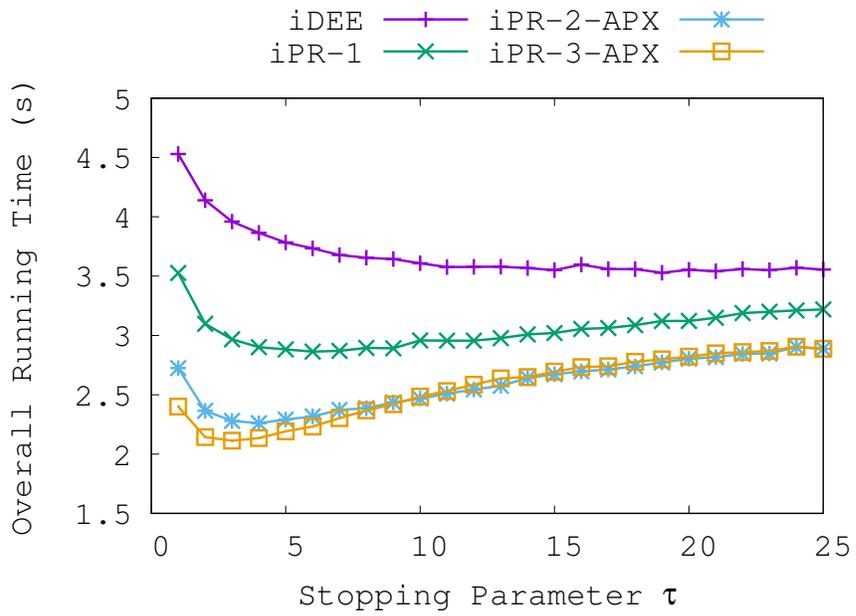
Figure 4.4: Persistent variables ratio vs. stopping parameter τ .

4.5.8 Overall running time break down

We can further break down the overall running time into the persistency-based pre-processing time and the flow computation time for using α -EXP to do the



(a) Color-Seg- \mathcal{N}_4 dataset



(b) Color-Seg- \mathcal{N}_8 dataset

Figure 4.5: Overall running time vs. stopping parameter τ .

approximate inference on the remaining variables. We illustrated the experimental results in Figure 4.6 for iDEE, iPR-1, iPR-2-APX, iPR-3-APX on the Color Segmentation dataset (\mathcal{N}_4 version).

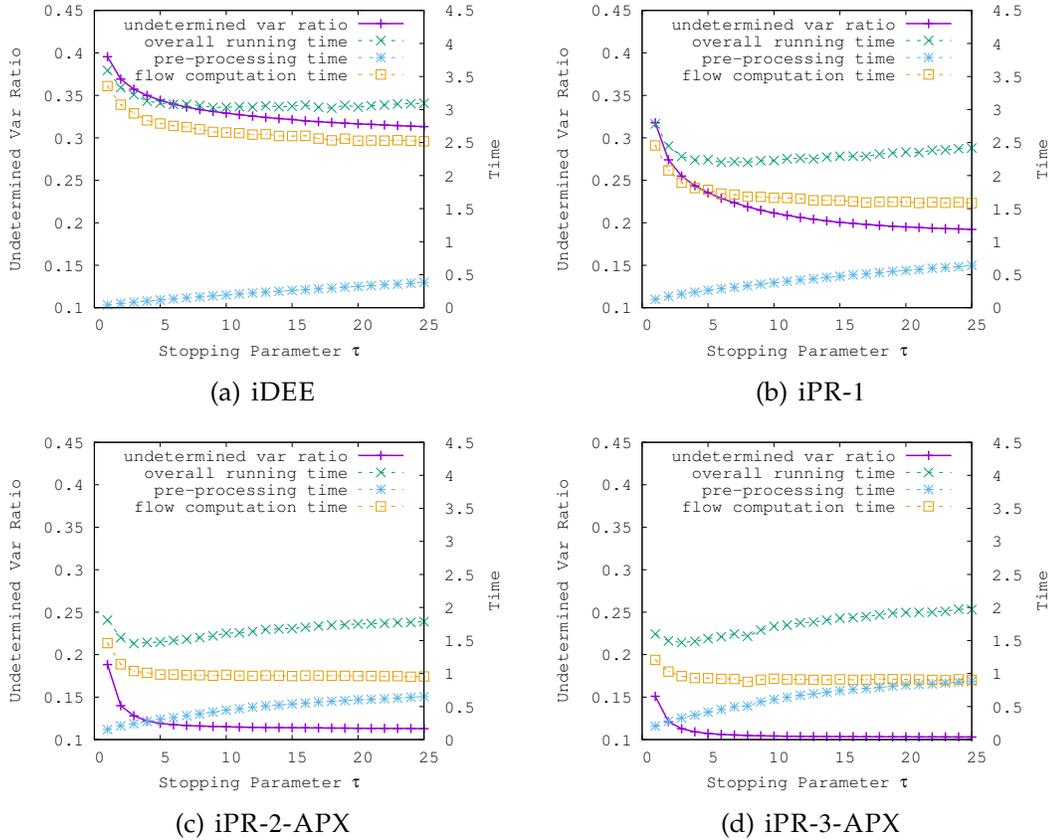


Figure 4.6: Overall running time break down on Color-Seg- \mathcal{N}_4 .

The blue dash line is the pre-processing time of persistency-based algorithm. We can say it increases monotonically with the growth of τ for all the methods. It grows almost linearly for iDEE and grows sub-linearly for the PR-based approaches due to an efficient implementation ignoring the variables which have been declared persistent.

The orange dash line is the flow time of α -EXP. It decreases monotonically with the growth of τ due to the persistency algorithm can find a larger persistent partial labeling with more iterations, which is illustrated as the purple solid line in the figure.

Finally, the trend of the overall running time is complicated (the green dash

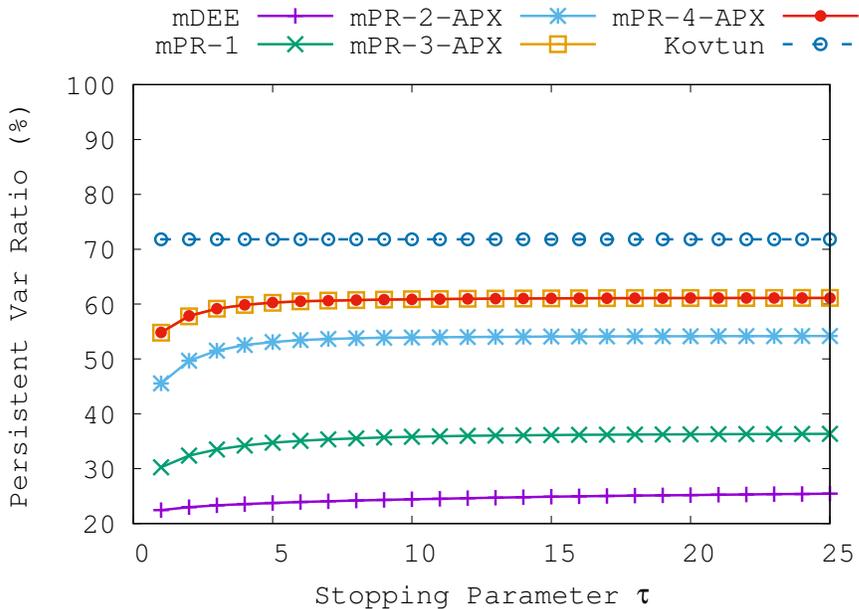


Figure 4.7: Experimental results of PR for multilabel MRFs.

line) due to it is a trade-off between the pre-processing time and flow time. In general it decreases initially then increases.

4.5.9 Persistency relaxation for multilabel MRFs

We showed the results on applying pre-processing to the multilabel MRFs directly in Figure 4.7. It is conducted on Color Segmentation dataset (\mathcal{N}_4 version) which contains up to 12 labels. It is amazing that the mPR-based approach can achieve comparable result to KOVTUN (although still 10% less). Recalling it treats all the labels independently and purely rely on the local condition, which is much simpler than the flow-based condition KOVTUN used. mDEE is struggling in the multilabel setting, it finds very few partial optimal variables.

We showed the results on applying pre-processing to each induced binary MRF in Figure 4.8. Comparing Figure 4.7 and Figure 4.8. We will find the

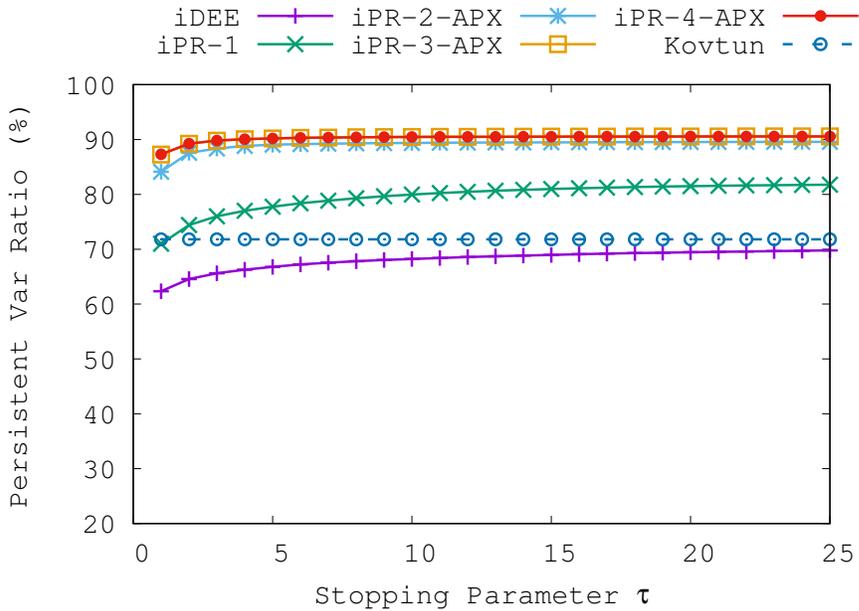


Figure 4.8: Experimental results of PR for induced binary MRFs.

advantage of applying pre-processing technique to each induced binary MRF. Note that all the iPR-based approaches all easily beat KOVTUN in this case. It indicates that although it is hard to find some global partial optimal variables, the induced binary problem are much easier. Before our approach, iDEE is the only choice suits for this per iteration pre-processing in α -expansion. Now, we provide a more general and powerful alternative.

4.5.10 Detailed experimental results

The followed up tables provide a more detailed per-instance results with pre-processing time, flow computation time, overall running time and persistent variables ratio. (We only report average number on Scene Decomposition dataset due to there are 715 instances.) We can see that PR-based approaches almost win on every instance. Note that we just pick $\tau = 5$ as an example to

report the results. We conducted experiments on a variety of choices from 1 to 25 and ∞ (i.e., waiting until converge). For only very few instances, waiting until converge will spend a long time. But in general, they all achieve similar performance for $k \geq 5$. Just as we said in the sensitivity analysis section, the proposed methods are pretty robust to the choice of τ .

The PR-based approaches achieved both the best speedup and the best persistent variables ratio on all the 8 instances in the Brain-MRI dataset.

Table 4.3: Experimental results on Brain-MRI dataset.

Approach	Persistency Time (s)	Flow Time (s)	Overall Time (s)	Speedup	Persistent Var Ratio (%)
Instance Brain-0-9mm (5 labels)					
α -EXP	0.0005	29.5179	30.1123	1.00	0.00
KOVTUN	5.7239	2.0030	8.2565	3.65	92.15
iDEE	0.7012	3.5980	4.8680	6.19	89.28
iPR-1	1.1698	1.0659	2.8513	10.56	97.65
iPR-2-APX	1.3974	0.6506	2.5919	11.62	99.01
iPR-3-APX	1.6068	0.6801	2.9005	10.38	99.15
iPR-4-APX	1.5891	0.6089	2.7399	10.99	99.16
Instance Brain-1-9mm (5 labels)					
α -EXP	0.0006	28.9777	29.5599	1.00	0.00
KOVTUN	5.5585	2.5094	8.7403	3.38	92.11
iDEE	0.5971	3.3291	4.4639	6.62	89.05
iPR-1	1.0582	0.9848	2.6253	11.26	97.64
iPR-2-APX	1.4479	0.6336	2.6252	11.26	98.99
iPR-3-APX	1.5777	0.6223	2.7479	10.76	99.12
iPR-4-APX	1.5129	0.5853	2.6180	11.29	99.13
Instance Brain-2-9mm (5 labels)					
α -EXP	0.0006	34.5505	35.2329	1.00	0.00
KOVTUN	8.8595	3.3385	13.0484	2.70	91.86
iDEE	0.7438	4.2960	5.7512	6.13	89.71
Continued on next page					

Table 4.3 – continued from previous page

Approach	Persistency Time (s)	Flow Time (s)	Overall Time (s)	Speedup	Persistent Var Ratio (%)
iPR-1	1.2384	1.1436	3.0590	11.52	97.77
iPR-2-APX	1.5239	0.7936	2.9844	11.81	99.05
iPR-3-APX	1.6548	0.6915	2.9911	11.78	99.18
iPR-4-APX	1.6875	0.6852	2.9992	11.75	99.19
Instance Brain-3-9mm (5 labels)					
α -EXP	0.0007	7.4870	7.6523	1.00	0.00
KOVTUN	5.3640	0.1280	5.6350	1.36	100.00
iDEE	0.0930	0.1834	0.4386	17.45	100.00
iPR-1	0.1432	0.1287	0.4196	18.24	100.00
iPR-2-APX	0.2118	0.1383	0.5037	15.19	100.00
iPR-3-APX	0.2250	0.1276	0.4943	15.48	100.00
iPR-4-APX	0.2254	0.1309	0.4998	15.31	100.00
Instance Brain-0-5mm (5 labels)					
α -EXP	0.0009	88.9107	90.7756	1.00	0.00
KOVTUN	12.6237	6.1703	20.6173	4.40	93.35
iDEE	2.3417	12.2558	16.5301	5.49	88.98
iPR-1	3.0442	3.0347	7.8581	11.55	97.79
iPR-2-APX	3.8545	2.0903	7.5635	12.00	99.04
iPR-3-APX	4.3486	2.0382	8.1504	11.14	99.17
iPR-4-APX	4.9834	2.1079	8.8157	10.30	99.18
Instance Brain-1-5mm (5 labels)					
α -EXP	0.0029	91.4609	93.3824	1.00	0.00
KOVTUN	13.0218	5.4269	20.1310	4.64	93.39
iDEE	1.9013	10.7959	14.3866	6.49	88.92
iPR-1	3.3861	3.2456	8.3935	11.13	97.80
iPR-2-APX	4.5024	2.6153	9.2742	10.07	99.04
iPR-3-APX	4.5294	2.0818	8.2837	11.27	99.17
iPR-4-APX	4.6707	2.1165	8.5005	10.99	99.18
Instance Brain-2-5mm (5 labels)					
α -EXP	0.0012	93.4645	95.3155	1.00	0.00
Continued on next page					

Table 4.3 – continued from previous page

Approach	Persistency Time (s)	Flow Time (s)	Overall Time (s)	Speedup	Persistent Var Ratio (%)
KOVTUN	12.7966	4.7755	19.0363	5.01	93.52
iDEE	1.9529	10.5398	14.2952	6.67	89.12
iPR-1	3.1350	3.0258	7.7878	12.24	97.73
iPR-2-APX	4.4257	2.3652	8.7054	10.95	99.03
iPR-3-APX	4.5862	2.1204	8.5141	11.20	99.17
iPR-4-APX	4.4486	1.9875	8.1466	11.70	99.19
Instance Brain-3-5mm (5 labels)					
α -EXP	0.0013	95.7588	97.6432	1.00	0.00
KOVTUN	12.6239	5.3238	19.5473	5.00	93.16
iDEE	2.1083	12.1846	16.4071	5.95	88.73
iPR-1	3.1185	2.9804	7.7995	12.52	97.85
iPR-2-APX	3.9139	2.0976	7.7210	12.65	99.08
iPR-3-APX	4.9082	2.2410	9.0263	10.82	99.20
iPR-4-APX	4.1823	1.8582	7.6376	12.78	99.21

The PR-based approaches achieved the best speedup on all the 9 instances in the Color Segmentation dataset (\mathcal{N}_4 version), and the best persistent variables ratio on 7 among 9 instances. KOVTUN wins the remaining instances but the gap between the proposed PR-based approaches and KOVTUN are very minor (90.36% vs. 91.71%, 96.73% vs. 97.55%).

Table 4.4: Experimental results on Color Segmentation dataset (\mathcal{N}_4 version).

Approach	Persistency Time (s)	Flow Time (s)	Overall Time (s)	Speedup	Persistent Var Ratio (%)
Instance clownfish (12 labels)					
α -EXP	0.0001	11.0117	11.2568	1.00	0.00
KOVTUN	0.7680	2.0686	3.0606	3.68	73.77
iDEE	0.1386	1.2712	1.6311	6.90	86.38
iPR-1	0.2445	0.4459	0.9135	12.32	95.49
iPR-2-APX	0.3068	0.2574	0.7887	14.27	98.15
Continued on next page					

Table 4.4 – continued from previous page

Approach	Persistency Time (s)	Flow Time (s)	Overall Time (s)	Speedup	Persistent Var Ratio (%)
iPR-3-APX	0.3497	0.2458	0.8158	13.80	98.30
iPR-4-APX	0.3528	0.2411	0.8063	13.96	98.31
Instance <i>crops</i> (12 labels)					
α -EXP	0.0001	14.0721	14.4047	1.00	0.00
KOVTUN	0.7777	4.7165	5.8195	2.48	64.44
iDEE	0.1684	2.3073	2.7720	5.20	82.52
iPR-1	0.3215	0.8315	1.4374	10.02	93.73
iPR-2-APX	0.3803	0.3885	1.0617	13.57	97.62
iPR-3-APX	0.4347	0.3488	1.0742	13.41	97.91
iPR-4-APX	0.4409	0.3494	1.0861	13.26	97.92
Instance <i>fourcolors</i> (4 labels)					
α -EXP	0.0001	1.7075	1.7472	1.00	0.00
KOVTUN	0.2423	0.4117	0.6915	2.53	69.16
iDEE	0.0155	1.5649	1.6145	1.08	0.23
iPR-1	0.1046	1.2016	1.3422	1.30	24.17
iPR-2-APX	0.1426	0.4985	0.6756	2.59	73.28
iPR-3-APX	0.1716	0.4621	0.6665	2.62	76.24
iPR-4-APX	0.1761	0.4511	0.6605	2.65	76.42
Instance <i>lake</i> (12 labels)					
α -EXP	0.0001	9.1911	9.3905	1.00	0.00
KOVTUN	0.7707	2.0279	3.0037	3.13	74.41
iDEE	0.1104	0.9078	1.2074	7.78	88.95
iPR-1	0.2127	0.4008	0.8039	11.68	95.37
iPR-2-APX	0.2769	0.2257	0.6974	13.47	98.02
iPR-3-APX	0.3198	0.2112	0.7234	12.98	98.19
iPR-4-APX	0.3226	0.2123	0.7234	12.98	98.20
Instance <i>palm</i> (12 labels)					
α -EXP	0.0001	11.6085	11.8688	1.00	0.00
KOVTUN	0.7842	3.4438	4.4894	2.64	68.44
iDEE	0.1527	4.8633	5.2478	2.26	54.86
Continued on next page					

Table 4.4 – continued from previous page

Approach	Persistency Time (s)	Flow Time (s)	Overall Time (s)	Speedup	Persistent Var Ratio (%)
iPR-1	0.3215	2.7908	3.3506	3.54	74.29
iPR-2-APX	0.4940	1.5906	2.3290	5.10	85.71
iPR-3-APX	0.5313	1.3276	2.0944	5.67	88.16
iPR-4-APX	0.5546	1.3395	2.1239	5.59	88.19
Instance <i>penguin</i> (8 labels)					
α -EXP	0.0001	4.7494	4.8516	1.00	0.00
KOVTUN	0.4280	0.3012	0.8263	5.87	91.71
iDEE	0.0501	1.0353	1.1810	4.11	75.61
iPR-1	0.1004	0.7650	0.9635	5.04	82.37
iPR-2-APX	0.1864	0.5120	0.7919	6.13	88.07
iPR-3-APX	0.1919	0.4410	0.7326	6.62	90.36
iPR-4-APX	0.1926	0.4329	0.7213	6.73	90.36
Instance <i>pfau</i> (12 labels)					
α -EXP	0.0001	12.4477	12.7087	1.00	0.00
KOVTUN	0.6869	13.1554	14.1369	0.90	5.55
iDEE	0.1860	6.8432	7.2792	1.75	43.33
iPR-1	0.4169	4.4423	5.1083	2.49	63.63
iPR-2-APX	0.6941	3.1020	4.0396	3.15	75.86
iPR-3-APX	0.8554	2.8141	3.9155	3.25	78.26
iPR-4-APX	0.8814	2.8139	3.9503	3.22	78.29
Instance <i>snail</i> (3 labels)					
α -EXP	0.0001	1.1219	1.1490	1.00	0.00
KOVTUN	0.1589	0.0330	0.2202	5.22	97.55
iDEE	0.0211	0.1637	0.2179	5.27	83.11
iPR-1	0.0381	0.0983	0.1632	7.04	89.24
iPR-2-APX	0.0484	0.0431	0.1180	9.74	96.24
iPR-3-APX	0.0557	0.0400	0.1217	9.44	96.74
iPR-4-APX	0.0563	0.0394	0.1220	9.42	96.73
Instance <i>strawberry</i> (12 labels)					
α -EXP	0.0001	11.7457	12.0142	1.00	0.00
Continued on next page					

Table 4.4 – continued from previous page

Approach	Persistency Time (s)	Flow Time (s)	Overall Time (s)	Speedup	Persistent Var Ratio (%)
KOVTUN	0.6986	5.3305	6.2930	1.91	54.64
iDEE	0.1395	4.6473	5.0390	2.38	59.32
iPR-1	0.3003	2.5991	3.1464	3.82	77.54
iPR-2-APX	0.5163	1.6148	2.3899	5.03	86.39
iPR-3-APX	0.5810	1.3912	2.2202	5.41	88.32
iPR-4-APX	0.5908	1.4092	2.2461	5.35	88.41

The PR-based approaches achieved the best speedup on all the 9 instances in the Color Segmentation dataset (\mathcal{N}_8 version), and the best persistent variables ratio on 7 among 9 instances. KOVTUN wins the remaining instances but the gap between the proposed PR-based approaches and KOVTUN are very minor (91.07% vs. 91.76%, 97.36% vs. 97.45%).

Table 4.5: Experimental results on Color Segmentation dataset (\mathcal{N}_8 version).

Approach	Persistency Time (s)	Flow Time (s)	Overall Time (s)	Speedup	Persistent Var Ratio (%)
Instance <i>clownfish</i> (12 labels)					
α -EXP	0.0001	15.7031	16.0336	1.00	0.00
KOVTUN	1.3051	2.7580	4.3214	3.71	73.28
iDEE	0.1992	1.0167	1.5301	10.48	93.15
iPR-1	0.3505	0.5900	1.2578	12.75	96.55
iPR-2-APX	0.4731	0.3858	1.1771	13.62	98.26
iPR-3-APX	0.5272	0.3480	1.1971	13.39	98.59
iPR-4-APX	0.5286	0.3489	1.1935	13.43	98.62
Instance <i>crops</i> (12 labels)					
α -EXP	0.0001	20.0284	20.4477	1.00	0.00
KOVTUN	1.3156	6.8882	8.6202	2.37	64.40
iDEE	0.2449	2.2717	2.9275	6.98	89.16
iPR-1	0.4669	1.0430	1.9139	10.68	95.31
iPR-2-APX	0.5811	0.5690	1.5582	13.12	97.80
Continued on next page					

Table 4.5 – continued from previous page

Approach	Persistency Time (s)	Flow Time (s)	Overall Time (s)	Speedup	Persistent Var Ratio (%)
iPR-3-APX	0.6512	0.5196	1.5801	12.94	98.16
iPR-4-APX	0.6644	0.4984	1.5657	13.06	98.23
Instance <code>fourcolors</code> (4 labels)					
α -EXP	0.0001	2.6387	2.6982	1.00	0.00
KOVTUN	0.4013	0.7271	1.1909	2.27	67.48
iDEE	0.0464	2.5394	2.6462	1.02	3.26
iPR-1	0.2163	1.5642	1.8404	1.47	43.08
iPR-2-APX	0.2779	0.8137	1.1507	2.34	73.61
iPR-3-APX	0.3351	0.7340	1.1294	2.39	77.14
iPR-4-APX	0.3563	0.7275	1.1436	2.36	77.77
Instance <code>lake</code> (12 labels)					
α -EXP	0.0001	13.4400	13.7211	1.00	0.00
KOVTUN	1.3099	3.1433	4.7301	2.90	73.86
iDEE	0.1689	0.9119	1.3642	10.06	93.05
iPR-1	0.3076	0.5178	1.0967	12.51	96.45
iPR-2-APX	0.4256	0.3429	1.0407	13.18	98.20
iPR-3-APX	0.4847	0.3144	1.0663	12.87	98.46
iPR-4-APX	0.4928	0.2992	1.0628	12.91	98.54
Instance <code>palm</code> (12 labels)					
α -EXP	0.0001	17.1379	17.4925	1.00	0.00
KOVTUN	1.3670	5.4137	7.1379	2.45	67.95
iDEE	0.2456	5.7760	6.3707	2.75	66.62
iPR-1	0.5631	4.2506	5.1738	3.38	75.80
iPR-2-APX	0.7890	2.8098	3.9386	4.44	84.61
iPR-3-APX	0.8457	2.1140	3.3120	5.28	88.76
iPR-4-APX	0.8739	2.0500	3.2671	5.35	88.96
Instance <code>penguin</code> (8 labels)					
α -EXP	0.0001	7.3409	7.4918	1.00	0.00
KOVTUN	0.7543	0.4608	1.3577	5.52	91.76
iDEE	0.0839	1.4313	1.6677	4.49	80.11
Continued on next page					

Table 4.5 – continued from previous page

Approach	Persistency Time (s)	Flow Time (s)	Overall Time (s)	Speedup	Persistent Var Ratio (%)
iPR-1	0.1854	1.2173	1.5574	4.81	83.52
iPR-2-APX	0.3123	0.9932	1.4572	5.14	86.87
iPR-3-APX	0.3105	0.6887	1.1465	6.53	90.95
iPR-4-APX	0.3239	0.6726	1.1451	6.54	91.07
Instance <i>pfau</i> (12 labels)					
α -EXP	0.0001	20.1059	20.5130	1.00	0.00
KOVTUN	1.1149	16.2915	17.7468	1.16	5.64
iDEE	0.3387	9.5948	10.3246	1.99	53.47
iPR-1	0.7873	7.0523	8.2420	2.49	66.57
iPR-2-APX	1.2240	4.9666	6.5963	3.11	77.38
iPR-3-APX	1.6829	4.6373	6.7180	3.05	78.93
iPR-4-APX	1.7893	4.5348	6.7227	3.05	79.21
Instance <i>snail</i> (3 labels)					
α -EXP	0.0001	1.9664	2.0148	1.00	0.00
KOVTUN	0.2941	0.0445	0.3721	5.41	97.45
iDEE	0.0324	0.1875	0.2682	7.51	89.02
iPR-1	0.0661	0.1455	0.2615	7.70	91.79
iPR-2-APX	0.0892	0.0817	0.2217	9.09	96.44
iPR-3-APX	0.0931	0.0666	0.2074	9.71	97.26
iPR-4-APX	0.0950	0.0656	0.2083	9.67	97.36
Instance <i>strawberry</i> (12 labels)					
α -EXP	0.0001	17.4269	17.7993	1.00	0.00
KOVTUN	1.1757	8.0612	9.6044	1.85	53.82
iDEE	0.2346	6.3368	6.9496	2.56	65.13
iPR-1	0.4871	3.7356	4.5943	3.87	79.66
iPR-2-APX	0.7770	2.3417	3.4844	5.11	87.67
iPR-3-APX	0.9608	2.0517	3.3714	5.28	89.32
iPR-4-APX	0.9965	2.0572	3.4192	5.21	89.59

The PR-based approaches achieved the best speedup on all the 2 instances

in the Inpainting dataset (\mathcal{N}_8 version), and the best persistent variables ratio on 1 of them, while KOVTUN wins the other one.

Table 4.6: Experimental results on Inpainting dataset (\mathcal{N}_8 version).

Approach	Persistency Time (s)	Flow Time (s)	Overall Time (s)	Speedup	Persistent Var Ratio (%)
Instance triplepoint4-plain-ring-inverse (4 labels)					
α -EXP	0.0000	0.4400	0.4469	1.00	0.00
KOVTUN	0.1287	0.1628	0.2959	1.51	37.39
iDEE	0.0025	0.2701	0.2814	1.59	40.89
iPR-1	0.0059	0.2640	0.2765	1.62	43.99
iPR-2-APX	0.0113	0.2231	0.2424	1.84	56.02
iPR-3-APX	0.0167	0.2159	0.2407	1.86	56.02
iPR-4-APX	0.0144	0.2204	0.2448	1.83	56.02
Instance triplepoint4-plain-ring (4 labels)					
α -EXP	0.0000	0.3949	0.4018	1.00	0.00
KOVTUN	0.1606	0.0330	0.1984	2.03	87.38
iDEE	0.0040	0.2393	0.2510	1.60	40.00
iPR-1	0.0061	0.2188	0.2335	1.72	47.46
iPR-2-APX	0.0151	0.1422	0.1648	2.44	67.47
iPR-3-APX	0.0133	0.1423	0.1639	2.45	67.47
iPR-4-APX	0.0138	0.1476	0.1682	2.39	67.47

The PR-based approaches achieved the best speedup on 6 among 7 instances in the Middlebury dataset, and the best persistent variables ratio on all the instances. DEE is slightly faster than the proposed PR-based approaches on 1 instance (1.45 vs. 1.43). Note that in this dataset, we have a large label set so KOVTUN timesout (at least 5x slower than α -EXP) on the entire dataset.

Table 4.7: Experimental results on Middlebury dataset.

Approach	Persistency Time (s)	Flow Time (s)	Overall Time (s)	Speedup	Persistent Var Ratio (%)
Instance T_{ed} (60 labels)					
α -EXP	0.0001	401.6440	407.3978	1.00	0.00
KOVTUN	N/A	N/A	N/A	N/A	N/A
iDEE	2.7050	238.0053	244.7326	1.66	18.31
iPR-1	11.1137	178.5905	193.5722	2.10	40.96
iPR-2-APX	14.6403	125.7790	144.2356	2.82	60.24
iPR-3-APX	17.6393	121.4567	142.8571	2.85	61.47
iPR-4-APX	17.8660	121.4136	143.0377	2.85	61.51
Instance T_{su} (16 labels)					
α -EXP	0.0001	43.4328	43.8352	1.00	0.00
KOVTUN	N/A	N/A	N/A	N/A	N/A
iDEE	0.3030	30.8764	31.4713	1.39	7.36
iPR-1	1.1738	28.7990	30.2676	1.45	14.31
iPR-2-APX	2.0666	25.8126	28.1741	1.56	24.74
iPR-3-APX	2.6680	25.4327	28.3969	1.54	25.81
iPR-4-APX	2.6253	25.3058	28.2289	1.55	25.81
Instance V_{en} (20 labels)					
α -EXP	0.0002	125.0321	126.2441	1.00	0.00
KOVTUN	N/A	N/A	N/A	N/A	N/A
iDEE	0.7714	85.6125	87.2133	1.45	2.43
iPR-1	2.7314	85.0040	88.5553	1.43	2.98
iPR-2-APX	5.3439	83.8337	90.0001	1.40	4.23
iPR-3-APX	7.3890	83.6677	91.8756	1.37	4.46
iPR-4-APX	7.6478	83.7015	92.1813	1.37	4.46
Instance F_{amily} (5 labels)					
α -EXP	0.0004	97.8842	98.6301	1.00	0.00
KOVTUN	N/A	N/A	N/A	N/A	N/A
iDEE	0.4242	95.6904	96.7831	1.02	4.27
iPR-1	1.3475	96.6937	98.7191	1.00	4.34
Continued on next page					

Table 4.7 – continued from previous page

Approach	Persistency Time (s)	Flow Time (s)	Overall Time (s)	Speedup	Persistent Var Ratio (%)
iPR-2-APX	2.1232	94.8641	97.6514	1.01	4.61
iPR-3-APX	2.2898	93.2351	96.1786	1.03	4.63
iPR-4-APX	2.2855	89.4289	92.3023	1.07	4.63
Instance Pano (7 labels)					
α -EXP	0.0004	106.8310	108.2290	1.00	0.00
KOVTUN	N/A	N/A	N/A	N/A	N/A
iDEE	0.1902	107.1636	108.7092	1.00	0.00
iPR-1	0.9723	105.7037	108.1483	1.00	0.00
iPR-2-APX	2.6996	67.4241	71.5048	1.51	64.63
iPR-3-APX	2.2728	55.2445	58.7441	1.84	67.10
iPR-4-APX	2.2762	53.0225	56.4861	1.92	67.10
Instance House (256 labels)					
α -EXP	0.0001	921.9603	934.8721	1.00	0.00
KOVTUN	N/A	N/A	N/A	N/A	N/A
iDEE	4.9624	807.0047	823.6554	1.14	5.29
iPR-1	14.7405	672.4787	697.0348	1.34	5.71
iPR-2-APX	24.2710	638.4153	672.0552	1.39	7.23
iPR-3-APX	29.9763	638.7680	678.0096	1.38	7.57
iPR-4-APX	30.3008	631.6456	671.1801	1.39	7.74
Instance Penguin (256 labels)					
α -EXP	0.0000	262.8118	267.1431	1.00	0.00
KOVTUN	N/A	N/A	N/A	N/A	N/A
iDEE	1.3234	139.3348	143.6865	1.86	32.79
iPR-1	5.1941	120.3164	128.6085	2.08	42.92
iPR-2-APX	7.4991	98.4549	109.0183	2.45	53.85
iPR-3-APX	8.9226	96.6814	108.6613	2.46	54.70
iPR-4-APX	9.1422	96.4438	108.6417	2.46	54.72

Both DEE and PR-based approaches proves persistency for almost every variables of the induced binary subproblems on Scene Decomposition dataset.

Therefore, they all achieve an order of magnitude speedup improvement.

Table 4.8: Experimental results on Scene Decomposition dataset.

Approach	Persistency Time (s)	Flow Time (s)	Overall Time (s)	Speedup	Persistent Var Ratio (%)
Average over all 715 instances (8 labels)					
α -EXP	0.0000	0.0111	0.0113	1.00	0.00
KOVTUN	0.0061	0.0040	0.0103	1.10	61.66
iDEE	0.0002	0.0007	0.0011	10.27	95.08
iPR-1	0.0004	0.0005	0.0011	10.27	96.87
iPR-2-APX	0.0005	0.0003	0.0010	11.30	98.55
iPR-3-APX	0.0005	0.0003	0.0010	11.30	98.78
iPR-4-APX	0.0005	0.0003	0.0010	11.30	98.82

5.1 Motivation

We will explore another direction to compute persistency for the first-order Markov Random Fields (MRFs) in this chapter. We argue that all the existing persistency algorithms are too conservative, in the sense that these techniques are guaranteed to never wrongly label a variable but they often leave a large number of variables unlabeled. We will address this shortcoming by interpreting the persistency problem as a classification problem, which allows us to trade off false positives (i.e., giving a variable an incorrect label) versus false negatives (i.e., failing to label a variable). We will describe an efficient discriminative rule that finds optimal solutions for a subset of variables.

Persistency algorithms for MRF inference seek to determine the optimal labeling of a subset of variables as a pre-processing step, thus reducing the complexity of the remaining combinatorial search problem. The best known persistency methods are Dead-end Elimination (DEE) [31] and QPBO [12, 77], but there are a number of others [71, 84, 83, 126, 129, 134, 145]. (Similar approaches are used for other NP-hard problems, a prominent example is Davis-Putnam's pure literal rule for SAT [30].)

The key weakness of such methods is that they are inherently conservative, since they only label variables whose value can be determined in every global minimum. Yet the MRFs that occur in computer vision are so large that in prac-

Table 5.1: An example to show the restriction of the conservative persistency conditions. The left labeling is an unlikely labeling for the neighborhood around the center pixel in the global minimum, compared to the labeling at center and right. Existing pre-processing methods treat all neighbor labelings equally, and as a result fail to label many variables.

1	0	1
0	?	0
1	0	1

1	1	1
1	?	1
1	1	1

0	0	0
0	?	0
0	0	0

tice we almost never compute the actual global minimum.¹ As a result, a pre-processing step that is carefully designed to never prune the global minimum is followed by a search step that almost never finds the global minimum. Our fundamental observation is that the pre-processing step can be viewed as a classification problem, and that existing pre-processing methods are designed to avoid false positives (i.e., to never label a variable incorrectly), at the cost of many false negatives (i.e., variables that are left unlabeled). By revisiting this trade-off we can design techniques where the combination of the pre-processing step and the search step leads to better overall performance, especially on the most difficult problems.

As an example, consider a tiny 8-connected binary MRF with 9 variables (pixels), and suppose we wish to determine by persistency condition that the center pixel should be labeled with 0. In order to soundly compute this by DEE or QPBO, we need to establish the autarky property that switching the center pixel from 1 to 0 will always decrease the energy, no matter what the labeling of the surrounding pixels. Yet as demonstrated in Table 5.1, there are local neighbor labelings that are quite unlikely shown in the global minimum. However, as the cost of a sound persistency condition, we still need to verify the autarky condition holds for all the unlikely local neighbor labelings, thus

¹See [55, 102] for rare counterexamples.

we may fail to prove consistency. However, if we can sacrifice the soundness of the persistency condition, we may just show the autarky condition is hold on the middle and right labeling in Table 5.1, which are more likely local neighbor labelings present in the global minimum. Therefore, we can conclude with a high probability in practice, the center pixel should be labeled as 0.

5.1.1 Outline of the chapter

In Section 5.2, we will review persistency and its sufficient relaxation autarky, and argue autarky is too conservative to check persistency in practice. An equivalent definition of persistency and autarky will be proposed to show the connection between these two concepts and further illustrate how conservative autarky is. This will also be used to motivated the key discriminative criterion proposed in Section 5.2.2.

We will present our discriminative persistency algorithm in Section 5.3, based on the proposed discriminative criterion. It provides several different approaches to approximate the discriminative criterion and make it computational tractable.

Section 5.4 gives the theoretical analysis of the algorithm. We analyze the running time, as well as the per-instance and worst-case performance of our algorithm when followed by an approximate inference algorithm that produces a solution with performance bounds.

Finally, we present the experimental results in Section 5.5. This demonstrates that by compromising the accuracy of persistency condition by a tiny bit, we can

boost the percentage of labeled variables significantly. Therefore, our discriminative persistency algorithm provides way better speed energy trade-off compared to the existing persistency algorithms. It achieves substantial speedup with similar or even lower energy. We also study the parameter sensitivity and how does the theoretical error bounds help the performance in practice.

5.2 Discriminative view of persistency

5.2.1 Comparison between persistency and autarky

Let's recapitulate the important concepts *persistency* and *autarky* again, and rethink the theoretical differences between them.

Definition 5.2.1 (Recap Definition 2.3.1). A partial labeling \mathbf{x}_S is (strong) persistent if

$$\mathbf{x}_S = \mathbf{x}_S^*, \quad \forall \mathbf{x}^* \in \operatorname{argmin}_{\mathbf{x}} f(\mathbf{x}). \quad (5.1)$$

Definition 5.2.2 (Recap Definition 2.3.3, rewrite slightly in an equivalent way). A partial labeling \mathbf{x}_S is an autarky if

$$f(\mathbf{x}_S \oplus \mathbf{z}_{V \setminus S}) < f(\mathbf{y}_S \oplus \mathbf{z}_{V \setminus S}), \quad \forall \mathbf{y}_S \in \mathcal{L}_S, s.t., \mathbf{y}_S \neq \mathbf{x}_S, \forall \mathbf{z}_{V \setminus S} \in \mathcal{L}_{V \setminus S}. \quad (5.2)$$

Persistency is the key property we are looking for, since it determines the optimal value of a subset of the variables and thus reduces the remaining combinatorial search problem. In general, though, checking for persistency is intractable [12]. All existing persistency algorithms appear to check the autarky property as a sufficient condition, which states that overwriting an arbitrary labeling with this partial labeling will reduce the energy.

We will reuse our definition on the energy change when we substitute \mathbf{x}_S by \mathbf{y}_S given the partial labeling $\mathbf{z}_{V \setminus S}$ for the variables not in S .

$$\Delta f(\mathbf{y}_S \leftarrow \mathbf{x}_S \mid \mathbf{z}_{V \setminus S}) := f(\mathbf{y}_S \oplus \mathbf{z}_{V \setminus S}) - f(\mathbf{x}_S \oplus \mathbf{z}_{V \setminus S}). \quad (5.3)$$

After expanding the RHS by the definition of $f(\mathbf{x})$ and cancelling terms, the Markov property of MRFs gives us a sum over terms only depending on x_i, y_i for $i \in S$ and z_j for $j \in V \setminus S$ with some $i \in S$ such that $(i, j) \in E$ (i.e., z_j is adjacent to S).

$$\begin{aligned} \Delta f(\mathbf{y}_S \leftarrow \mathbf{x}_S \mid \mathbf{z}_{V \setminus S}) &= \sum_{i \in S} (\theta_i(y_i) - \theta_i(x_i)) + \sum_{ij \in (S, S)} (\theta_{ij}(y_i, y_j) - \theta_{ij}(x_i, x_j)) \\ &+ \sum_{ij \in (S, V \setminus S)} (\theta_{ij}(y_i, z_j) - \theta_{ij}(x_i, z_j)). \end{aligned} \quad (5.4)$$

Let $\mathcal{N}(S) := \{j \in V \setminus S \mid \exists i \in S, (i, j) \in E\}$, and we can rewrite

$$\Delta f(\mathbf{y}_S \leftarrow \mathbf{x}_S \mid \mathbf{z}_{V \setminus S}) = \Delta f(\mathbf{y}_S \leftarrow \mathbf{x}_S \mid \mathbf{z}_{\mathcal{N}(S)}). \quad (5.5)$$

This allows us to rewrite the autarky property (5.2) as:

$$\min_{\mathbf{y}_S \neq \mathbf{x}_S} \Delta f(\mathbf{y}_S \leftarrow \mathbf{x}_S \mid \mathbf{z}_{\mathcal{N}(S)}) > 0, \forall \mathbf{z}_{\mathcal{N}(S)} \in \mathcal{L}_{\mathcal{N}(S)}. \quad (5.6)$$

The key issue is the universal quantification in (5.6). To ensure that a partial labeling \mathbf{x}_S presents in all global minimizer, we look at all possible values that the neighbors might have. For each of these, we check that any other assignment \mathbf{y}_S would increase the energy.

Yet this is obviously quite conservative. We now show the desired persistency property can be rewritten by only looking at assignments to the neighboring variables that occur in a global minimizer. Define $\mathcal{L}_{\mathcal{N}(S)}^* := \{\mathbf{z}_{\mathcal{N}(S)}^* \mid \mathbf{z}^* \in \arg \min f(\mathbf{z})\}$ be all possible labelings of $\mathcal{N}(S)$ in a global minimizer.

Lemma 5.2.1. \mathbf{x}_S is persistent if and only if

$$\min_{\mathbf{y}_S \neq \mathbf{x}_S} \Delta f(\mathbf{y}_S \leftarrow \mathbf{x}_S \mid \mathbf{z}_{N(S)}) > 0, \forall \mathbf{z}_{N(S)} \in \mathcal{L}_{N(S)}^*. \quad (5.7)$$

Proof. The if direction is trivial: consider an arbitrary global minimizer \mathbf{z}^* , we have $\mathbf{z}_{N(S)}^* \in \mathcal{L}_{N(S)}^*$ by definition. Suppose $\mathbf{x}_S \neq \mathbf{z}_S^*$, we will have $f(\mathbf{x}_S \oplus \mathbf{z}_{V \setminus S}^*) < f(\mathbf{z}^*)$, which contradicts the assumption that \mathbf{z}^* is a minimizer. Therefore, we have $\mathbf{x}_S = \mathbf{z}_S^*, \forall \mathbf{z}^*$, so it is persistent.

For the only if direction, suppose (5.7) is not true, then $\exists \mathbf{z}_{N(S)} \in \mathcal{L}_{N(S)}^*, \exists \mathbf{y}_S \neq \mathbf{x}_S$ such that $\Delta f(\mathbf{y}_S \leftarrow \mathbf{x}_S \mid \mathbf{z}_{N(S)}) \leq 0$. We can expand $\mathbf{z}_{N(S)}$ to one minimizer \mathbf{z}^* such that $\mathbf{z}_{N(S)}^* = \mathbf{z}_{N(S)}$. Since \mathbf{x}_S is persistent, we also know $\mathbf{z}_S^* = \mathbf{x}_S$. Therefore, $f(\mathbf{y}_S \oplus \mathbf{z}_{V \setminus S}^*) \leq f(\mathbf{x}_S \oplus \mathbf{z}_{V \setminus S}^*) = f(\mathbf{z}^*)$. Since \mathbf{z}^* is a minimum this inequality is an equality, hence $\mathbf{y}_S \oplus \mathbf{z}_{V \setminus S}^*$ is also a global minimum. This contradicts the assumption that \mathbf{x}_S is persistent, since $\mathbf{y}_S \neq \mathbf{x}_S$. \square

Comparing (5.6) and (5.7), we immediately observe that the universal quantifier makes autarky a sound but stronger condition than persistency, since $\mathcal{L}_{N(S)}^* \subseteq \mathcal{L}_{N(S)}$. We can also understand the gap between the persistency and autarky. Usually $\mathcal{L}_{N(S)}^*$ is a relative small set. However, due to we don't know what is $\mathcal{L}_{N(S)}^*$ exactly beforehand, so autarky property checks all the possible local neighbor labelings to ensure soundness.

5.2.2 Discriminative criterion

We have compared the persistency and autarky property in the previous section. Now we understand the major theoretical gap between them is persistency only

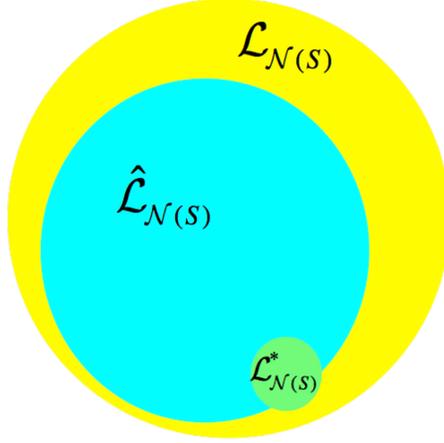


Figure 5.1: Diagram to illustrate the discriminative criterion for persistency. The yellow region is the set of all possible local neighbor labelings $\mathcal{L}_{N(S)}$. The autarky property asks to check \mathbf{x}_S is a better choice for all of them. The green region is the set of local neighbor labelings present in global minimum $\mathcal{L}_{N(S)}^*$. It is sufficient to check \mathbf{x}_S is better only for this region to prove persistency. However, we don't know it exactly beforehand. Therefore, our discriminative criterion views the persistency decision problem as a classification problem, and computes the cyan region $\hat{\mathcal{L}}_{N(S)}(\mathbf{x}_S)$, which is the set of all the local neighbor labelings where \mathbf{x}_S is better. We will claim \mathbf{x}_S is persistent if the cyan region is large enough or it covers significant important local neighbor labelings more likely to show in the global minimum (i.e., we have a high confidence that the cyan region can cover the green region).

requires us to check \mathbf{x}_S can help us to reduce energy for local neighbor labelings in the minimum labeling $\mathcal{L}_{N(S)}^*$, while autarky checks that for all the possible local neighbor labelings $\mathcal{L}_{N(S)}$. Crucially, this suggests a discriminative criterion to trade off false positives against false negatives.

The high level idea is the following. We will view the persistency decision problem as a classification problem. Let

$$\hat{\mathcal{L}}_{N(S)}(\mathbf{x}_S) := \{\mathbf{z}_{N(S)} \in \mathcal{L}_{N(S)} \mid \min_{\mathbf{y}_S \neq \mathbf{x}_S} \Delta f(\mathbf{y}_S \leftarrow \mathbf{x}_S \mid \mathbf{z}_{N(S)}) > 0\} \quad (5.8)$$

be the set of local neighbor labelings $\mathbf{z}_{N(S)}$ such that given them \mathbf{x}_S is always a better choice. When $\hat{\mathcal{L}}_{N(S)}(\mathbf{x}_S)$ is large enough or covers the most important local neighbor labelings, it is very likely that we will have $\mathcal{L}_{N(S)}^* \subseteq \hat{\mathcal{L}}_{N(S)}(\mathbf{x}_S)$.

This in turn implies \mathbf{x}_S is persistent, even though $\hat{\mathcal{L}}_{N(S)}(\mathbf{x}_S) \neq \mathcal{L}_{N(S)}$ and we do not precisely know $\mathcal{L}_{N(S)}^*$. This idea is also illustrated in Figure 5.1.

By taking this discriminative approach, we cannot guarantee soundness anymore. However, we can now capture more persistent partial labelings which cannot be proved by the autarky property. It is trade-off between the false negatives (persistent labeling cannot be found) and false positives (mislabeling some variables) from the classification point of view. As we will see in the experimental section, usually we only need to pay for the cost of very few false positives and we can reduce the number of false negatives significantly.

Formally, assume we have a ground truth distribution $p(\mathbf{z}_{N(S)})$ which is uniform over $\mathcal{L}_{N(S)}^*$ and 0 otherwise. Then a sound condition to check persistency is $\sum_{\mathbf{z}_{N(S)} \in \hat{\mathcal{L}}_{N(S)}(\mathbf{x}_S)} p(\mathbf{z}_{N(S)}) = 1$. Of course, computing $\mathcal{L}_{N(S)}^*$ and $p(\mathbf{z}_{N(S)})$ is computationally intractable. So we use an estimated distribution $q(\mathbf{z}_{N(S)})$ that approximates $p(\mathbf{z}_{N(S)})$. Looking back to Table 5.1, one would assume that the left neighbor labeling would not appear in $Z_{N(S)}^*$, while the other two quite plausibly could; there should be a lower q value for the left one but a higher q value for the others.

Definition 5.2.3 (discriminative criterion for persistency). Our *discriminative criterion for persistency* is:

$$\sum_{\mathbf{z}_{N(S)} \in \hat{\mathcal{L}}_{N(S)}(\mathbf{x}_S)} q(\mathbf{z}_{N(S)}) \geq \kappa. \quad (5.9)$$

Here $\kappa \in [0, 1]$ is the key parameter that controls the trade-off between false positives and false negatives, as shown by the following (obvious) lemma.

Lemma 5.2.2. *For the same set of decision problems for persistency, we will never increase the number of false positives by increasing κ .*

Proof. This one is trivial. Consider any non-persistent \mathbf{x}_S , it will be a false positive with parameter κ_2 if and only if it meets our discriminative criterion, i.e., $\sum_{\mathbf{z}_{N(S)} \in \hat{\mathcal{L}}_{N(S)}} q(\mathbf{z}_{N(S)}) \geq \kappa_2$. Now for the algorithm using parameter $\kappa_2 > \kappa_1$, our discriminative criterion still holds, hence it is still a false positive for our algorithm with parameter κ_1 . \square

5.3 Discriminative persistency algorithm

We now address the two crucial issues: how to choose q to effectively approximate p , and how to efficiently check (5.9) in Section 5.3.1 and Section 5.3.2 respectively. Finally, we will present our overall algorithm in Section 5.3.3.

5.3.1 Approximating underlying probability p

Uniform distribution approximation

A trivial baseline is to treat each $\mathbf{z}_{N(S)}$ as equally important and set our approximation $q(\mathbf{z}_{N(S)})$ to be the uniform distribution over $\mathcal{L}_{N(S)}$. In this special case, (5.9) is equivalent to count the number of local neighbor labelings $\mathbf{z}_{N(S)}$ that satisfy $\min_{\mathbf{y}_S \neq \mathbf{x}_S} \Delta f(\mathbf{y}_S \leftarrow \mathbf{x}_S \mid \mathbf{z}_{N(S)}) > 0$. We expect $\hat{\mathcal{L}}_{N(S)}(\mathbf{x}_S)$ to cover the unknown $\mathcal{L}_{N(S)}^*$ with high probability when $|\hat{\mathcal{L}}_{N(S)}(\mathbf{x}_S)|$ is large enough.

Max-marginal probability approximation

A more elegant approach is to estimate the marginal probability of a particular assignment $\mathbf{z}_{N(S)}$ via the generative MRF model, and use this as our approximation for p . This problem is well studied in the message passing literature, and is often solved by max-product loopy belief propagation (LBP) [105, 146].

An important special case is if we only use the initialization of LBP, $q_i(z_i) \propto e^{-\theta_i(z_i)}$. This makes a certain amount of intuitive sense: in the MRF energy functions that occur in computer vision it is well known that most of the weight comes from the unary terms [105], which provide a strong signal as to the optimal label for each variable.

More generally, we can define $q(\mathbf{z}_{N(S)})$ to be a fully independent distribution $q(\mathbf{z}_{N(S)}) = \prod_{i \in N(S)} q_i(z_i)$ with $q_i(z_i) \propto e^{-\theta_i(z_i)} \prod_{j \in N(i)} m_{j \rightarrow i}(z_i)$, where $m_{j \rightarrow i}(z_i)$ is the message we have from the belief propagation algorithm. Since this is just an approximation, we would not need to pay the cost of running LBP to convergence. In our experiments, the more general approach does not seem to pay dividends, but other ways of estimating the marginals are worth investigating.

5.3.2 Efficiently checking our discriminative criterion

Checking (5.9) is generally computational intractable, due to the size of $\mathcal{L}_{N(S)}(\mathbf{x}_S)$ and $\{\mathbf{y}_S \in \mathcal{L}_S \mid \mathbf{y}_S \neq \mathbf{x}_S\}$. We now propose a polynomial time algorithm to compute a lower bound for $\sum_{\mathbf{z}_{N(S)} \in \hat{\mathcal{L}}_{N(S)}(\mathbf{x}_S)} q(\mathbf{z}_{N(S)})$.

We will focus on the persistency of a single variable x_i from this point forward. However, our methods can handle an arbitrary \mathbf{x}_S for $|S| > 1$; the details

are deferred to Section 5.3.4, but are similar to the single variable case. This subroutine is used by our construction algorithm (which will be described in Section 5.3.3) to construct a partial labeling for the given energy function $f(\mathbf{x})$.

Our general strategy is to find a subset of $\mathcal{L}_{N(i)}$ which we know is inside $\hat{\mathcal{L}}_{N(i)}(x_i)$ and can be easily factorized. We start by considering each node $j \in N(i)$ independently. For each j , define \mathcal{A}_j to be the set of labels ℓ where the autarky condition holds if $z_j = \ell$. Since autarky is a stronger condition than persistency, we know that all $\mathbf{z}_{N(i)}$ values where $z_j \in \mathcal{A}_j$ are inside $\hat{\mathcal{L}}_{N(i)}(x_i)$. The union of these sets across different $j \in N(i)$ will still be a subset of $\hat{\mathcal{L}}_{N(i)}(x_i)$.

Formally, define $\mathcal{L}_{N(i)}^{z_j=\ell} := \{\mathbf{z}_{N(i)} \mid z_j = \ell\}$. Then $\mathcal{A}_j = \{\ell \mid \min_{y_i \neq x_i} \Delta f(y_i \leftarrow x_i \mid \mathbf{z}_{N(i)}) > 0, \forall \mathbf{z}_{N(i)} \in \mathcal{L}_{N(i)}^{z_j=\ell}\}$. Let $\mathcal{L}_{N(i)}^{z_j \in \mathcal{A}_j} := \cup_{\ell \in \mathcal{A}_j} \mathcal{L}_{N(i)}^{z_j=\ell}$. Then, we know that $\mathcal{L}_{N(i)}^{z_j \in \mathcal{A}_j} \subseteq \hat{\mathcal{L}}_{N(i)}(x_i)$ and $\cup_{j \in N(i)} \mathcal{L}_{N(i)}^{z_j \in \mathcal{A}_j} \subseteq \hat{\mathcal{L}}_{N(i)}(x_i)$.

We establish a computationally tractable lower bound for $\sum_{\mathbf{z}_{N(i)} \in \hat{\mathcal{L}}_{N(i)}(x_i)} q(\mathbf{z}_{N(i)})$ by the following lemma, which we can check instead.

Lemma 5.3.1. *We have the following lower bound:*

$$\sum_{j \in N(i)} Q_j \prod_{k \in N(i), k < j} (1 - Q_k) \leq \sum_{\mathbf{z}_{N(i)} \in \hat{\mathcal{L}}_{N(i)}(x_i)} q(\mathbf{z}_{N(i)}), \quad (5.10)$$

where $Q_i = \sum_{\ell \in \mathcal{A}_i} q_i(z_i = \ell)$.

Proof. We can view $\sum_{\mathbf{z}_{N(i)} \in \hat{\mathcal{L}}_{N(i)}} q(\mathbf{z}_{N(i)})$ as the probability $Pr(\mathbf{z}_{N(i)} \in \hat{\mathcal{L}}_{N(i)})$ given distribution q .

Because our $q(\mathbf{z}_{N(i)})$ can be factorized independently, we can integrate over the variables other than z_j to get $Pr(\mathbf{z}_{N(i)} \in \mathcal{L}_{N(i)}^{z_j=\ell}) = Pr(z_j = \ell) = q_j(z_j = \ell)$.

We also have $Pr(\mathbf{z}_{N(i)} \in \mathcal{L}_{N(i)}^{z_j \in \mathcal{A}_j}) = Pr(z_j \in \mathcal{A}_j) = \sum_{\ell \in \mathcal{A}_j} q_j(z_j = \ell) = Q_j$ since $\mathcal{L}_{N(i)}^{z_j=\ell}$

are all disjoint. Then, using independence again, we have

$$\begin{aligned}
& \sum_{\mathbf{z}_{N(i)} \in \bigcup_{j \in N(i)} \mathcal{L}_{N(i)}^{z_j \in \mathcal{A}_j}} q(\mathbf{z}_{N(i)}) \\
&= \Pr\left(\mathbf{z}_{N(i)} \in \bigcup_{j \in N(i)} \mathcal{L}_{N(i)}^{z_j \in \mathcal{A}_j}\right) \\
&= \Pr\left(\bigcup_{j \in N(i)} (\mathbf{z}_{N(i)} \in \mathcal{L}_{N(i)}^{z_j \in \mathcal{A}_j})\right) \\
&= \Pr\left(\bigcup_{j \in N(i)} (z_j \in \mathcal{A}_j)\right) \\
&= \Pr(z_{j_1} \in \mathcal{A}_{j_1}) + \Pr(z_{j_2} \in \mathcal{A}_{j_2}) \Pr(z_{j_1} \notin \mathcal{A}_{j_1}) \cdots \\
&= \sum_{j \in N(i)} Q_j \prod_{k \in N(i), k < j} (1 - Q_k)
\end{aligned} \tag{5.11}$$

Finally, note that we argued $\bigcup_{j \in N(i)} \mathcal{L}_{N(i)}^{z_j \in \mathcal{A}_j} \subseteq \hat{\mathcal{L}}_{N(i)}(x_i)$ before, which concludes the proof. \square

Constructing \mathcal{A}_j requires us to be able to efficiently check $\min_{y_i \neq x_i} \Delta f(y_i \leftarrow x_i \mid \mathbf{z}_{N(i)}) > 0, \forall \mathbf{z}_{N(i)} \in \mathcal{L}_{N(i)}^{z_j = \ell}$. We expand it by the definition of $f(\mathbf{x})$ then swap the min and sum operators. This gives the following lower bound, which we check for being strictly positive:

$$\begin{aligned}
& \min_{y_i \neq x_i} (\theta_i(y_i) - \theta_i(x_i)) + \min_{y_i \neq x_i} (\theta_{ij}(y_i, \ell) - \theta_{ij}(x_i, \ell)) \\
& + \sum_{k \in N(i), k \neq j} \min_{z_k, y_i \neq x_i} (\theta_{ij}(y_i, z_k) - \theta_{ij}(x_i, z_k)) > 0
\end{aligned} \tag{5.12}$$

5.3.3 Overall algorithm

We have presented our discriminative criterion to decide if a given partial labeling $x_i = \ell$ is persistent. Now we will use it as a key subroutine to compute partial labeling for MRF pre-processing, as shown in Algorithm 5.1. We first loop over the unlabeled variables and its label set (line 5). For each given $x_i = \ell$, use our discriminative rule to judge whether it is persistent (line 9-14). We will

Algorithm 5.1: Discriminative persistency algorithm

Input: Energy function $f(\mathbf{x})$
Output: Partial labeling $\hat{\mathbf{x}}_S$

- 1 $\hat{\mathbf{x}} \leftarrow \emptyset;$
- 2 $S \leftarrow \emptyset;$
- 3 Compute $q(\mathbf{z}_{N(i)});$
- 4 **for** $t \leftarrow 1$ **to** τ **do**
- 5 **for** $i \in V \setminus S, \ell \in \mathcal{L}_i$ **do**
- 6 **if** $|\mathcal{L}_i| = 1$ **then**
- 7 **continue;**
- 8 **end**
- 9 Compute $LB \leq \sum_{\mathbf{z}_{N(i)} \in \hat{\mathcal{L}}_{N(i)}(x_i = \ell)} q(\mathbf{z}_{N(i)})$ described in Section 5.3.2;
- 10 **if** $LB \geq \kappa$ **then**
- 11 $\hat{\mathbf{x}} \leftarrow \hat{\mathbf{x}} \oplus \{x_i = \ell\};$
- 12 $\mathcal{L}_i \leftarrow \{\ell\};$
- 13 $S \leftarrow S \cup \{i\};$
- 14 **end**
- 15 **end**
- 16 **end**
- 17 **return** $\hat{\mathbf{x}}_S;$

fix its value if it satisfies our criterion by setting $\mathcal{L}_i = \{\ell\}$, and concatenate it with our inference result $\hat{\mathbf{x}}$ (line 11-13). Note that fixing $x_i = \ell$ will also provide additional information as to the unlabeled variables which were checked before x_i , so we repeat the whole procedure for τ iterations (line 4).

An interesting detail is that computing q using LBP is time consuming, yet we experimentally observe that our algorithm is robust against an imperfect distribution q . So we simply estimate q globally by LBP and never update it even when we fix the values of some variables (line 3).

After our discriminative persistency algorithm has terminated and labeled the variables in the set S , we fix the variables $\hat{\mathbf{x}}_S$ and use any MRF inference algorithms to solve the remaining energy minimization problem, which gives us a labeling $\hat{\mathbf{x}}_{V \setminus S}$ on the remaining variables. Finally, we obtain our inference

result by concatenating them together.

5.3.4 Generalized efficient check of our discriminative criterion

Now let's generalize our algorithm described in Section 5.3.2 to compute the lower bound $\sum_{\mathbf{z}_{N(S)} \in \hat{\mathcal{L}}_{N(S)}} q(\mathbf{z}_{N(S)})$ efficiently for given partial labeling \mathbf{x}_S . Basically, the only big difference is that we need a subroutine to efficiently check $\min_{\mathbf{y}_S \neq \mathbf{x}_S} \Delta f(\mathbf{y}_S \leftarrow \mathbf{x}_S \mid \mathbf{z}_{N(S)}) > 0$ for $\mathbf{z}_{N(S)} \in \mathcal{L}_{N(S)}$ with $z_j = \ell$. Persistency relaxation (PR) [145], presented in Chapter 4, generalizes dead-end elimination (DEE) [31] from checking persistency of a single variable x_i to an independent local minimum (ILM) partial labeling \mathbf{x}_S . The subproblem in PR is to decide if $\min_{\mathbf{y}_S \neq \mathbf{x}_S} \Delta f(\mathbf{y}_S \leftarrow \mathbf{x}_S \mid \mathbf{z}_{N(S)}) > 0$ for $\mathbf{z}_{N(S)} \in \mathcal{L}_{N(S)}$, without the additional constraint that $z_j = \ell$. Actually, it's trivial to enforce the additional constraint $z_j = \ell$ in PR. We just need to remove z_j from the free variables and force it takes value ℓ in the subroutine proposed in PR. Note that those subroutines are sound so we can still apply Lemma 5.3.1 to partial labeling \mathbf{x}_S and get the lower bound of $\sum_{\mathbf{z}_{N(S)} \in \hat{\mathcal{L}}_{N(S)}} q(\mathbf{z}_{N(S)})$. Once we have our discriminative criteria as the decision subroutine, we can follow the construction algorithm in PR (Algorithm 4.2) as the generalization of Algorithm 5.1.

5.4 Theoretical analysis

We will analyze the running time of Algorithm 5.1 in Section 5.4.1. We can analyze the per-instance and worst-case performance of our discriminative persistency methods when followed by an approximate inference algorithm that

produces a solution with performance bounds. We will present the results in Section 5.4.2 and Section 5.4.3 respectively.

5.4.1 Running time analysis

We will give a asymptotic analysis on the running time of our discriminative persistency algorithm here. Assuming we have an oracle to give us data term $\theta_i(x_i)$ and prior term value $\theta_{ij}(x_i, x_j)$ in $O(1)$ time. Let $N = |V|$, $M = |E|$ and $L = \max_i |\mathcal{L}_i|$ to be the number of variables, edges and maximum possible labels, $d = \max_i |\mathcal{N}(i)|$ is the maximum degree of the graph. For a typical vision problem, we usually have a sparse graph like grid, meaning $M = O(N)$ and d is also usually a small constant like 4 or 8.

Estimating the probability q on line 3 will take $O(NL)$ time if we apply the uniform distribution or just initialize it from the unary terms $q_i(x_i) = e^{-\theta_i(x_i)}$. If we apply the max-product belief propagation, it will takes $O(ML^2)$ time (we will just run it for a constant number of iterations).

Computation time of the for loop from line 4 to 16 needs some thinking. τ is usually a small constant, so we can omit it in the asymptotic analysis. For the given $x_i = \ell$, a naive implementation of brute force algorithm to compute $\sum_{\mathbf{z}_{N(i)} \in \hat{\mathcal{L}}_{N(i)}} q(\mathbf{z}_{N(i)})$ needs to enumerate all the possible local neighboring labelings $\mathbf{z}_{N(i)}$, and it takes $O(dL)$ to compute $\min_{y_i \neq x_i} \Delta f(y_i \leftarrow x_i \mid \mathbf{z}_{N(i)})$, so it takes $O(dL^{d+1})$ time. Therefore, the overall running time is $O(dNL^{d+2})$ for brute force so it is still feasible when both d and L are small constant.

When we use the approximated way to compute the lower bound using

Lemma 5.3.1, we need an faster way to compute (5.12). We can pre-compute all the terms we may used here in $O(NL + EL^2)$ time globally and then query it in $O(d)$ time without solving the min operator each time. Then it takes $O(d^2L)$ time to compute \mathcal{A}_j , $O(dL)$ time to compute Q_i and $O(d)$ to compute the sum each iteration. Also note that once we fix a variable, it also takes $O(L + dL^2)$ to update our pre-computations result. But each variable will only be fixed at most once during the pre-processing, so the amortized running time to update the pre-computations result is $O(NL + EL^2)$. So in sum, we have the overall running time $O(d^2NL^2 + EL^2)$ for approximated calculation.

5.4.2 Per-instance performance bounds

There are a number of MRF approximate inference algorithms that produce per-instance guarantees (i.e., they produce a certificate after execution that their solution is close to the global minimum). These methods, which are typically based on linear programming, include [74, 82, 142], and they provide a per-instance additive error bound by computing the duality gap.

Our algorithm has a natural way to bound additive errors. Recall our notation $\Delta f(y_i \leftarrow x_i \mid \mathbf{z}_{N(i)})$ describing the energy change when we flip x_i to y_i with the local neighboring labeling $\mathbf{z}_{N(i)}$. Therefore, $\min_{\mathbf{z}_{N(i)}} \min_{y_i} \Delta f(y_i \leftarrow x_i \mid \mathbf{z}_{N(i)}) \leq 0$ is the worst case energy decrement when we flip x_i to arbitrary y_i with arbitrary neighbor labelings $\mathbf{z}_{N(i)}$. It is non-positive since we can always set $y_i = x_i$. Now we can negate it and define $\delta_i := -\min_{\mathbf{z}_{N(i)}} \min_{y_i} \Delta f(y_i \leftarrow x_i \mid \mathbf{z}_{N(i)})$ to be the maximum potential energy loss when we use our discriminative criterion to decide x_i is persistent. Then we have the following two lemmas.

Lemma 5.4.1. *Let $\hat{\mathbf{x}}_S$ be the persistent variables found by our Algorithm 5.1. For arbitrary $\hat{\mathbf{x}}_{V \setminus S}$, and arbitrary \mathbf{x}'_S , we have $f(\hat{\mathbf{x}}_S \oplus \hat{\mathbf{x}}_{V \setminus S}) \leq f(\mathbf{x}'_S \oplus \hat{\mathbf{x}}_{V \setminus S}) + \sum_{i \in S} \delta_i$.*

Proof. With $\hat{\mathbf{x}}_{V \setminus S}$ fixed, we flip \hat{x}_i to x'_i in the reverse order of them being added to S by our algorithm. Due to the analysis before, we will lose at most δ_i at each step. \square

Theorem 5.4.2. *Suppose the inference algorithm has per-instance ζ -additive bound, then $f(\hat{\mathbf{x}}) \leq f(\mathbf{x}^*) + \zeta + \sum_{i \in S} \delta_i$.*

Proof. Let $\bar{\mathbf{x}}_{V \setminus S}$ be the minimizer of $f(\mathbf{x})$ with $\hat{\mathbf{x}}_S$ fixed, which might be different than the global minimizer $\mathbf{x}^*_{V \setminus S}$. Then we will have

$$\begin{aligned} f(\hat{\mathbf{x}}_S \oplus \hat{\mathbf{x}}_{V \setminus S}) &\leq f(\hat{\mathbf{x}}_S \oplus \bar{\mathbf{x}}_{V \setminus S}) + \zeta \\ &\leq f(\hat{\mathbf{x}}_S \oplus \mathbf{x}^*_{V \setminus S}) + \zeta \\ &\leq f(\mathbf{x}^*_S \oplus \mathbf{x}^*_{V \setminus S}) + \zeta + \sum_{i \in S} \delta_i \end{aligned} \tag{5.13}$$

The first step is because we use an inference algorithm with ζ -additive errors to solve the problem with $\hat{\mathbf{x}}_S$ fixed. The second step follows because $\bar{\mathbf{x}}_{V \setminus S}$ is the minimizer w.r.t. $\hat{\mathbf{x}}_S$. \square

As a special case, any sound condition like (5.6) guarantees $\delta_i = 0$, i.e., we don't make mistakes. In practice it is computationally intractable to compute δ_i , so just as in Section 5.3.2 we swap the min and sum operators, and compute the upper bound $\bar{\delta}_i \geq \delta_i$ efficiently:

$$\begin{aligned} \bar{\delta}_i &:= - \min_{y_i \neq x_i} (\theta_i(y_i) - \theta_i(x_i)) - \sum_{k \in \mathcal{N}(i)} \min_{z_k, y_i \neq x_i} (\theta_{ij}(y_i, z_k) - \theta_{ij}(x_i, z_k)) \\ &\geq - \min_{\mathbf{z}_{\mathcal{N}(i)}} \min_{y_i} \Delta f(y_i \leftarrow x_i \mid \mathbf{z}_{\mathcal{N}(i)}) \\ &= \delta_i. \end{aligned} \tag{5.14}$$

Then we use $\sum_{i \in S} \bar{\delta}_i$ as our per-instance additive bound.

5.4.3 Worst case performance bounds

Some MRF inference algorithms produce a solution that is guaranteed to lie within a known factor of the global minimum. The best known such technique is the expansion move algorithm [18] but there are others [45, 67, 82].

We can easily turn our per-instance bounds into the worst case bounds by introducing a pre-defined error tolerance ϵ . Then we define our new discriminative criteria to be:

$$\sum_{\mathbf{z}_{N(S)} \in \hat{\mathcal{L}}_{N(S)}(\mathbf{x}_S)} q(\mathbf{z}_{N(S)}) \geq \kappa \text{ and } \bar{\delta}_i \leq \epsilon, \quad (5.15)$$

where the first part is our previous discriminative criterion in (5.9), and we enforce that we don't lose the energy by at most ϵ in the second part of the criteria.

We can easily extend our analysis in the previous section to get the worst case bound with an approximate inference with additive error bound in the following corollary.

Corollary 5.4.3. *Suppose the inference algorithm has worst case ζ -additive bound, then $f(\hat{\mathbf{x}}) \leq f(\mathbf{x}^*) + \zeta + |S|\epsilon$ is our worst case additive bound.*

Inference algorithm with worst case guarantees are usually multiplicative bounds other than additive bounds, but we can modify our proof of Theorem 5.4.2 to get the following bounds.

Theorem 5.4.4. *Suppose the inference algorithm has a worst case β -multiplicative bound, then we will have $f(\hat{\mathbf{x}}) \leq \beta \cdot f(\mathbf{x}^*) + \beta \cdot |S|\epsilon$.*

Proof. Following the proof of Theorem 5.4.2, we have:

$$\begin{aligned}
f(\hat{\mathbf{x}}_S \oplus \hat{\mathbf{x}}_{V \setminus S}) &\leq \beta \cdot f(\hat{\mathbf{x}}_S \oplus \bar{\mathbf{x}}_{V \setminus S}) \\
&\leq \beta \cdot f(\hat{\mathbf{x}}_S \oplus \mathbf{x}_{V \setminus S}^*) \\
&\leq \beta \cdot (f(\mathbf{x}_S^* \oplus \mathbf{x}_{V \setminus S}^*) + |S|\epsilon).
\end{aligned} \tag{5.16}$$

□

A more careful analysis can give us a tighter bound (dropping the coefficient β before $|S|\epsilon$), for the important special case where we use the expansion move algorithm [18] for inference.

Theorem 5.4.5. *Suppose we use expansion moves as the inference algorithm, with the β -multiplicative bound, then we will have $f(\hat{\mathbf{x}}) \leq \beta \cdot f(\mathbf{x}^*) + |S|\epsilon$.*

Proof. Following the proof of the multiplicative bound of expansion moves algorithm [18] (Theorem 6.1), we will see actually the multiplicative factor β will not be applied to unary terms. In other words, $f'(\hat{\mathbf{x}}) = \sum_i \theta'_i(\hat{x}_i) + \sum_{ij} \theta'_{ij}(\hat{x}_i, \hat{x}_j) \leq \sum_i \theta'_i(x_i^*) + \beta \sum_{ij} \theta'_{ij}(x_i^*, x_j^*) \leq \beta f'(\mathbf{x}^*)$.

Note that in our algorithm, the energy function $f'(x)$ of expansion moves is induced by fixing $\hat{\mathbf{x}}_S$ in $f(\mathbf{x})$, all the pairwise terms θ_{ij} crossing S and $V \setminus S$ could be viewed as the unary terms in $f'(\mathbf{x})$ since one variable will be fixed. Therefore,

we will have following:

$$\begin{aligned}
& f(\hat{\mathbf{x}}_S \oplus \hat{\mathbf{x}}_{V \setminus S}) \\
&= \sum_{i \in S} \theta_i(\hat{x}_i) + \sum_{i,j \in S, (i,j) \in E} \theta_{ij}(\hat{x}_i, \hat{x}_j) + \sum_{i \in S, j \in V \setminus S, (i,j) \in E} \theta_{ij}(\hat{x}_i, \hat{x}_j) + \sum_{i \in V \setminus S} \theta_i(\hat{x}_i) + \sum_{i,j \in V \setminus S, (i,j) \in E} \theta_{ij}(\hat{x}_i, \hat{x}_j) \\
&\leq \sum_{i \in S} \theta_i(\hat{x}_i) + \sum_{i,j \in S, (i,j) \in E} \theta_{ij}(\hat{x}_i, \hat{x}_j) + \sum_{i \in S, j \in V \setminus S, (i,j) \in E} \theta_{ij}(\hat{x}_i, x_j^*) + \sum_{i \in V \setminus S} \theta_i(x_i^*) + \beta \sum_{i,j \in V \setminus S, (i,j) \in E} \theta_{ij}(x_i^*, x_j^*) \\
&\leq \sum_{i \in S} \theta_i(x_i^*) + \sum_{i,j \in S, (i,j) \in E} \theta_{ij}(x_i^*, x_j^*) + \sum_{i \in S, j \in V \setminus S, (i,j) \in E} \theta_{ij}(x_i^*, x_j^*) + \sum_{i \in V \setminus S} \theta_i(x_i^*) + \beta \sum_{i,j \in V \setminus S, (i,j) \in E} \theta_{ij}(x_i^*, x_j^*) + |S|\epsilon \\
&\leq \beta \cdot f(\mathbf{x}^*) + |S|\epsilon,
\end{aligned} \tag{5.17}$$

where the first step is the expansion of the definition of energy function. The second step is due to the approximate inference algorithm has the β -multiplicative bound. Note that with fixed $\hat{\mathbf{x}}_S$, the second term $\sum_{i,j \in S, (i,j) \in E} \theta_{ij}(\hat{x}_i, \hat{x}_j)$ is a constant here, while the third term $\sum_{i \in S, j \in V \setminus S, (i,j) \in E} \theta_{ij}(\hat{x}_i, \hat{x}_j^*)$ should be viewed as a unary term. The third step is due to we won't lose more than ϵ when we substitute \hat{x}_i by x_i^* in the reversed order when we add i to S in our persistency algorithm. \square

5.5 Experimental results

We conducted the experiments to evaluate our proposed discriminative persistency algorithm to pre-process the expansion move algorithm, from both the speed and the quality (energy) aspects. We also studied the robustness of the proposed method to various parameters in the algorithm and how do they affect the performance.

The whole section is organized as following. We first describe the dataset and experimental environment in Section 5.5.1. Then we introduce the baseline methods and measurements in Section 5.5.2 and 5.5.3 respectively. Our

leave-one-out principle to choose parameters is introduced in Section 5.5.4. We present the overall performance of our discriminative persistency algorithm in Section 5.5.5, with more detailed discussion on the speedup and recall values in Section 5.5.6, energy and precision values in Section 5.5.7 and the precision/recall trade-off in Section 5.5.8. We investigate the experiments on parameter sensitivity and conduct experiments with a typical parameter setup without cross-validation in Section 5.5.9 and 5.5.10 respectively. Then we compare our algorithm with other state-of-the-arts approximate inference algorithms in Section 5.5.11. We provide some visualization of the results using our approach in Section 5.5.12. Preliminary experimental results on applying our algorithm to multilabel MRFs are given in Section 5.5.13. Finally, we showed that how does the parameters controlling the worst-case bound of the algorithm help the performance of our algorithm in practice in Section 5.5.14.

5.5.1 Datasets and experimental environment

We conducted experiments on a large collection of MRF inference benchmarks, where the MRF inference problems come from different vision applications, including color segmentation [93], stereo, image inpainting, denoising [136] and optical flow [25]. Datasets for the first three tasks are wrapped in OpenGM2 [63] and are available online. We use the BSDS300 [100] for the denoising task with the MRF setup following [136]². We use the MPI Sintel dataset [19] for the optical flow task with the MRF setup following [25]³.

We briefly summarize the basic statistics of each dataset in Table 5.2. Our

²We chose 21077, 37073, 86000, 210088 and 376043 in our experiments.

³We chose `bamboo_3`, `cave_3`, `mountain_2` and `wall` in our experiments.

Table 5.2: Datasets description.

Dataset	$ V $	$ \mathcal{L} $	# Instances	Energy
Color Seg	65536–86400	3–12	18	Potts
Stereo	110592–168750	16–60	3	truncated L1/L2
Inpainting	21838–130560	256	2	truncated L2
Denoising	38400	256	10	L2/truncated L2
Optical Flow	27178	225	4	L1

focus, of course, is on the difficult inference problems where the induced binary subproblem is non-submodular, which includes the stereo, inpainting, denoising and optical flow dataset. For comparison, we also included some experiments on relatively easy color segmentation problems where the induced binary subproblem is submodular. We have all the datasets label each single pixel using MRFs. We have both \mathcal{N}_4 and \mathcal{N}_8 structure for the color segmentation dataset, and \mathcal{N}_4 for all the other datasets. For the denoising dataset, we have both the denoising-sq version with the L2 prior and denoising-ts version with the truncated L2 prior.

All the experiments were executed on a single machine with Hexa 3.5GHz Intel Xeon Core and 32GB 1866 MHz DDR3 memory.

5.5.2 Approaches

The most natural baselines for us to compare against include inference without pre-processing, and inference using the sound (but conservative) DEE [31] and PR [145] techniques. We employ expansion moves for MRF inference [18]. In order to achieve better speedup, we apply pre-processing to each induced binary subproblem of expansion moves as the input of DEE, PR or Algorithm 5.1,

and then run QPBO [77] with BK [17] algorithm to solve max-flow. For the unlabeled variables from QPBO, we follow the standard convention in the graphcuts literature, which is that they retain their previous label.

At the other end of the spectrum are high overhead techniques such as Kovtun’s approach [83, 84, 85], MQPBO [71], and MRF/LP-based approaches [126, 129, 134]. These algorithms require more running time than max-flow on each induced binary subproblem. Therefore, we apply them to the multilabel problem, and then use expansion move to infer the remaining part. We choose the IRI method [129] as the representative among [126, 129, 134] since it is significantly faster and can prove more persistent variables. Note that the R^3 [3] method also uses Kovtun’s method as their pre-processing (reduce) step in order to speed up MRF inference. The reuse and recycle parts attempt to speed up the inference algorithm itself, which is orthogonal to what we propose to do, so we do not compare against this method.

We also compared against other widely used MRF inference algorithms besides expansion moves, including loopy belief propagation (LBP) [105, 146], dual decomposition (DD) [64], TRWS [74] and MPLP [41, 132, 133]. The comparison among these inference algorithms are provided in survey papers [63, 136]. In our experiments, expansion moves is usually significantly faster than other methods, and gives comparable or better energy.

5.5.3 Measurement

We report the improvement in overall running time (including both pre-processing and the approximate inference for the remaining unlabeled vari-

ables) and relative energy change. The baseline is expansion moves with no pre-processing. Let T_i^{ALG} and E_i^{ALG} be the running time and energy for algorithm ALG on the i -th instance. We define the *speedup* as $T_i^{\alpha\text{-EXP}}/T_i^{\text{ALG}}$ and *energy change* $(E_i^{\text{ALG}} - E_i^{\alpha\text{-EXP}})/E_i^{\alpha\text{-EXP}}$ for each instance, and then report the average speedup and energy change for the whole dataset.

We also report the percentage of labeled variables during the pre-processing. Since we view the decision problem (whether a given partial labeling is persistent) as a classification problem, we interchangeably use the term *percentage of labeled variables* and *recall* value. Getting the *precision* value is tricky. Since it is a NP-hard problem so we cannot have the ground truth label for every variable. However, we apply our pre-processing technique to the binary subproblems induced from expansion moves. We know that either max-flow solves the subproblem exactly for the submodular cases or QPBO can find a sufficiently large subset of partial persistent labeling for the non-submodular cases (in our experiments, it labels almost all the variables). Therefore, we report the *precision* value of our method on the subset of the variables where we know the ground truth labeling. Formally, let \mathbf{x}^{ALG} be the persistent partial labeling our algorithm claims on V^{ALG} and \mathbf{x}^{OPT} be the correct persistent partial labeling claimed by max-flow or QPBO on V' . Then we define precision as $\frac{|\{i \in V^{\text{ALG}} \cap V' \mid \mathbf{x}_i^{\text{ALG}} = \mathbf{x}_i^{\text{OPT}}\}|}{|V^{\text{ALG}} \cap V'|}$, recall as $\frac{|\{i \in V^{\text{ALG}} \cap V' \mid \mathbf{x}_i^{\text{ALG}} = \mathbf{x}_i^{\text{OPT}}\}|}{|V'|}$. Finally, we define F1 score being their harmonic mean.

5.5.4 Parameter setup

The discriminative rule in our approach has a few parameters. In order to achieve a fair comparison, we employed leave-one-out cross-validation (see,

e.g. [105]) to use all but one instances in the same dataset as the validation set to choose the best parameter⁴ and test on the remaining instance. We explored all the combinations of 1) threshold $\kappa \in \{0.7, 0.8, 0.9\}$, 2) using a uniform distribution or the distribution derived from the unary term for our $q(\mathbf{x})$, 3) using Section 5.3.2 to compute LB on line 3 of Algorithm 5.1 or using brute force to compute $\sum_{\mathbf{z}_{N(i)} \in \hat{\mathcal{L}}_{N(i)}(x_i=\ell)} q(\mathbf{z}_{N(i)})$ exactly, 4) number of iterations $\tau \in \{1, 3, 5, 7, 9\}$. We run expansion moves until convergence or after 5 iterations through of the whole label set. We only apply our discriminative persistency criterion on a single variable, described in Section 5.3.2. We set our worst case bound $\epsilon = \infty$ by default, in order to investigate how good our discriminative rule is even without the worst case guarantee. We further study the role of ϵ in Section 5.5.14.

Experimental results demonstrate that our approach achieves good performance over a wide range of parameters. We observed that cross validation picked nearly identical parameters for every instance in the same dataset. Using nearby parameters also produced good results. We will defer more discussions on parameter sensitivity to Section 5.5.9 and 5.5.10.

5.5.5 Overall performance of discriminative persistency

We summarize our main experimental results in Table 5.3. Our primary goal is to speedup MRF inference on hard problems, and there is evidence that our benchmarks are challenging. The state-of-the-art IRI method [129], which delivers impressive performance on the easier problems in our benchmarks, strug-

⁴Based on the criterion that we choose the fastest overall running time when the false positive rate is less than 1%.

Table 5.3: Comparison between discriminative persistency (DisPer) and other persistency algorithms (N/A: not applicable, TO: time out, MEM: out of memory).

	Dataset	Measurement	DisPer	DEE	PR	Kovtun	MQPBO	IRI
Challenging Datasets (non-Potts energy, large $ L $)	Stereo	Speedup	1.78x	1.06x	1.13x	N/A	MEM	0.51x
	~170000 vars	Energy Change	-0.06%	0.00%	0.00%	N/A	MEM	-0.15%
	16–60 labels	Labeled Vars	44.76%	10.07%	18.06%	N/A	MEM	56.45%
	Trunc. L1/L2	Precision	99.74%	100.00%	100.00%	N/A	MEM	100.00%
	Inpainting	Speedup	3.40x	1.28x	1.32x	N/A	MEM	0.12x
	~130000 vars	Energy Change	-1.71%	0.00%	0.00%	N/A	MEM	0.00%
	256 labels	Labeled Vars	74.29%	21.05%	23.75%	N/A	MEM	0.36%
	Trunc. L2	Precision	96.16%	100.00%	100.00%	N/A	MEM	100.00%
	Denoising-sq	Speedup	11.83x	1.20x	1.37x	N/A	MEM	0.29x
	38400 vars	Energy Change	-0.02%	0.00%	0.00%	N/A	MEM	0.00%
	256 labels	Labeled Vars	97.91%	16.54%	29.83%	N/A	MEM	0.39%
	L2	Precision	99.95%	100.00%	100.00%	N/A	MEM	100.00%
	Denoising-ts	Speedup	11.91x	10.53x	10.64x	N/A	MEM	0.18x
	38400 vars	Energy Change	0.00%	0.00%	0.00%	N/A	MEM	-0.03%
	256 labels	Labeled Vars	98.32%	95.65%	97.69%	N/A	MEM	5.85%
	Trunc. L2	Precision	99.79%	100.00%	100.00%	N/A	MEM	100.00%
Optical Flow	Speedup	4.69x	2.63	3.40x	N/A	MEM	TO	
27178 vars	Energy Change	-0.04%	0.00%	0.00%	N/A	MEM	TO	
225 labels	Labeled Vars	77.25%	54.34%	65.51%	N/A	MEM	TO	
L1	Precision	99.88%	100.00%	100.00%	N/A	MEM	TO	
Easy Datasets (Potts, small $ L $)	Color-Seg-N_4	Speedup	7.02x	4.55x	6.34x	2.43x	0.37x	3.67x
	~86000 vars	Energy Change	0.00%	0.00%	0.00%	0.00%	0.00%	-0.12%
	3–12 labels	Labeled Vars	85.74%	65.38%	77.50%	70.32%	17.27%	98.44%
	Potts	Precision	99.79%	100.00%	100.00%	100.00%	100.00%	100.00%
	Color-Seg-N_8	Speedup	8.33x	5.61x	6.37x	2.33x	0.32x	1.45x
	~86000 vars	Energy Change	+0.04%	0.00%	0.00%	0.00%	0.00%	-0.10%
3–12 labels	Labeled Vars	90.39%	71.62%	82.05%	70.05%	17.87%	99.35%	
Potts	Precision	99.77%	100.00%	100.00%	100.00%	100.00%	100.00%	

gles with the harder problems⁵. The only source code for Kovtun [83, 84, 85] we could find is restricted to the Potts model, while MPQBO [71] runs out of memory for all these challenging problems.

In summary, our proposed discriminative persistency approach (DisPer) achieves a high quality trade-off between running time and energy among all the methods, particularly on challenging datasets. It runs significantly faster than its competitors, makes expansion move algorithm without pre-processing 2x to 12x faster, and achieves an energy that is similar and sometimes even

⁵In the table, time out means we do not obtain results after running for 10x the overall running time that expansion moves take.

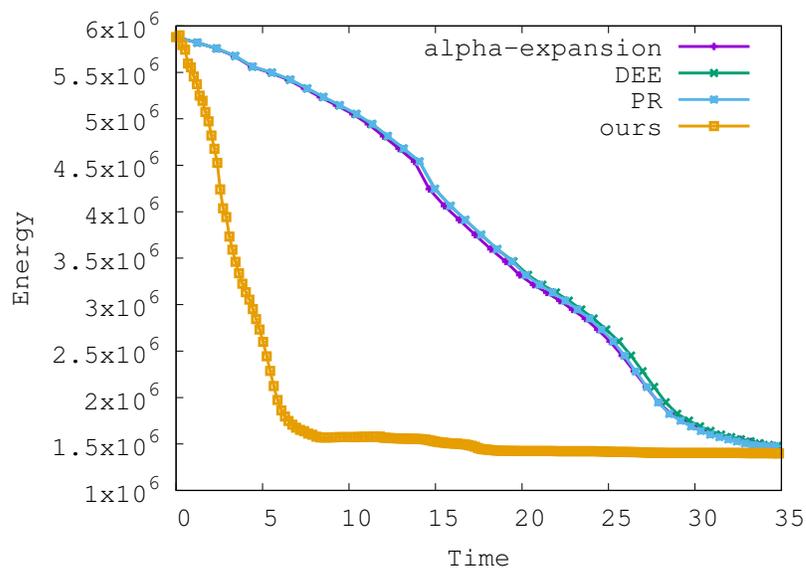


Figure 5.2: Energy vs. time curves for instance T_{ed} on stereo dataset.

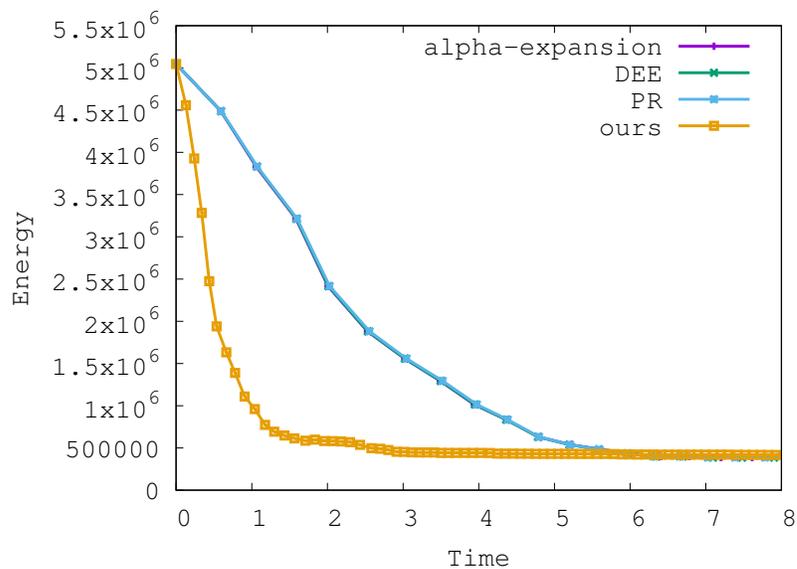


Figure 5.3: Energy vs. time curves for instance T_{su} on stereo dataset.

lower. We will discuss these two aspects in depth separately. A more detailed discussion on the speedup and recall values (the percentage of labeled variables) will be provided in Section 5.5.6. Then we will talk about energy and precision values in Section 5.5.7.

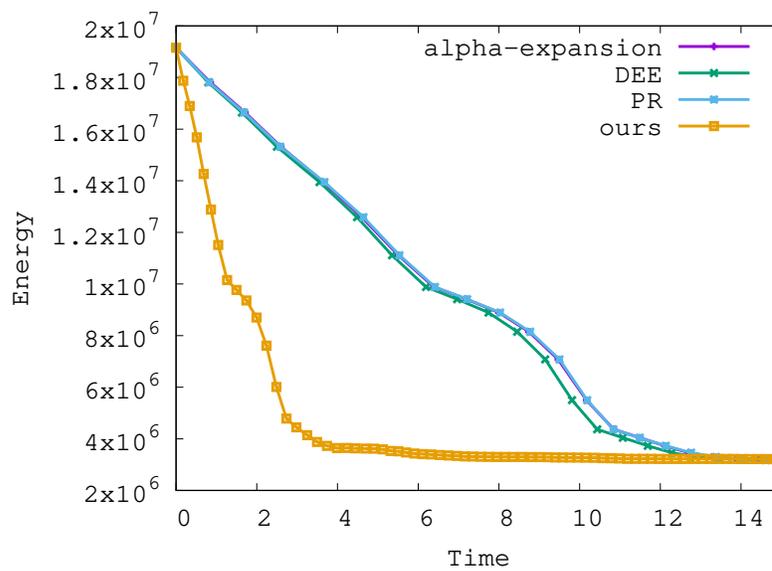


Figure 5.4: Energy vs. time curves for instance Ven on stereo dataset.

5.5.6 Experimental results on speedup and recall values

We will focus on speedups and recall values (percentage of labeled variables) in this section. It corresponds to the first line and third line for each dataset in Table 5.3.

Our approach achieved a significant speed improvement, making expansion moves 2x to 12x faster on various datasets. Our pre-processing method beats its natural competitor DEE by around 2x, and outperforms all the baseline methods. Figure 5.2, 5.3 and 5.4 shows typical energy vs. time curves. We can see our approach drives the energy curve down much faster than expansion move without pre-processing, DEE and PR. Note that this is a very challenge dataset that the conservative persistency algorithms like DEE and PR cannot label many variables, hence don't help speed.

The key factor for the speedup is the percentage of labeled variables. The values of these variables are fixed during the pre-processing step, resulting in a

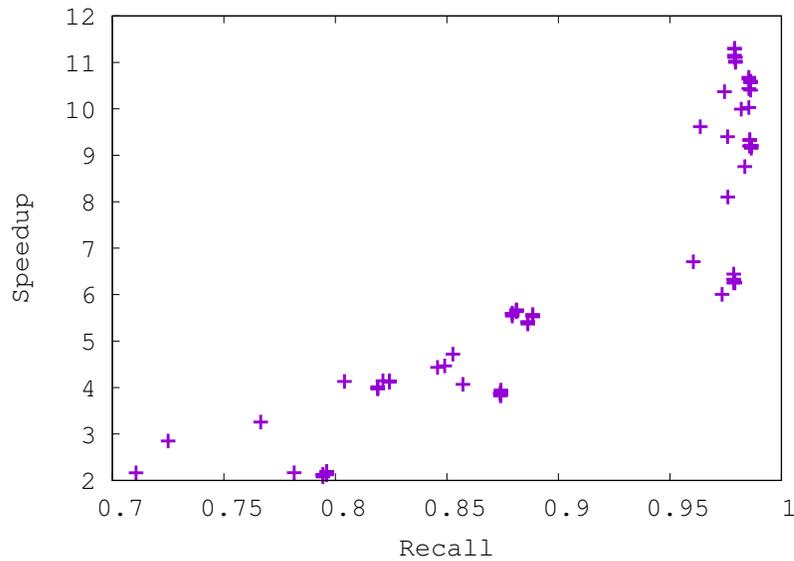


Figure 5.5: Speedup-recall scatter on Color-Seg- \mathcal{N}_8 dataset. We plot all the 9 instances in the dataset with all the different parameter combinations described Section 5.5.4. We can see a strong correlation between recall values (percentage of labeled variables) and speedup.

smaller problem for max-flow/QPBO to solve. We observed a strong positive correlation between the percentage of labeled variables and the speedup we can get, illustrated in Figure 5.5.

Table 5.3 shows our approach labels significantly more variables than DEE and PR, especially on the inpainting and denoising-sq datasets. Kovtun, MQPBO and IRI have very expensive overhead as the pre-processing step. While it is impressive that IRI labels almost every variable on the easy dataset, it is still 2x-6x slower than our proposed method. Furthermore, Kovtun, MQPBO and IRI do not perform well on our challenging datasets. When the size of the label set is large (which is common in many vision problems such as inpainting, denoising or optical flow), even IRI only proves a few variables to be persistent after spending 3x-70x as much time as our method. This demonstrates the advantage of performing pre-processing on the binary subproblem, which is

consistent with the observation in [145].

5.5.7 Experimental results on energy and precision values

We will focus the energy change and precision metric on the second line of fourth line for each dataset in Table 5.3.

Our method also performs well in terms of energy, especially on the hard benchmarks. Because we can label some variables incorrectly during pre-processing, there is a risk of producing a larger energy. However, the experimental results are reassuring: on the hard problems we actually produce slightly lower energy, while on the easier problems we can produce slightly higher energy.

While it is somewhat counter-intuitive, occasionally labeling variables incorrectly can plausibly lead to a better overall energy by getting out of a local minimum. Expansion moves can be viewed as a local search algorithm although its search space has an exponential size [2]. Therefore, a random walk going uphill occasionally may help us escape from the local minimizer, as in the Metropolis algorithm [103] or simulated annealing [66]. At one iteration of the expansion move algorithm, our method may label some variables incorrectly and solve the binary subproblem suboptimally (i.e., our pre-processing may cause the energy to increase during expansion move framework). It is plausible that this suboptimal move for the binary subproblem may also help us escape from the local minimizer. To verify this hypothesis, we experimented with a variant of our method where we reject an expansion move if it makes the energy worse. In experiments, this change led to a worse final energy. This suggests that allowing

Table 5.4: Precision/recall value vs. κ (P: Precision, R: Recall).

κ		0.7	0.8	0.9	1.0
Stereo	P	90.40%	99.71%	99.41%	100.00%
	R	91.31%	56.77%	11.35%	9.26%
Inpainting	P	95.11%	99.88%	99.96%	100.00%
	R	90.51%	47.06%	25.97%	21.93%
Denoising-sq	P	99.66%	99.95%	99.95%	100.00%
	R	99.47%	97.52%	19.11%	15.15%
Denoising-ts	P	99.75%	99.95%	99.99%	100.00%
	R	98.61%	96.65%	94.99%	94.62%
Optical Flow	P	94.01%	99.50%	99.98%	100.00%
	R	99.27%	93.74%	60.85%	56.79%
Color-Seg-\mathcal{N}_4	P	94.77%	99.50%	99.86%	100.00%
	R	98.52%	90.80%	77.20%	66.65%
Color-Seg-\mathcal{N}_8	P	99.48%	99.76%	99.87%	100.00%
	R	92.84%	90.43%	86.92%	71.66%

suboptimal moves is beneficial.

We believe that our method achieves competitive energy due to the very high precision, shown in Table 5.3. Although our discriminative criterion is not a sufficient condition for persistency anymore (unlike all the other baseline algorithms with 100% precision), we still achieved $> 99.7\%$ for all the datasets except the inpainting dataset (with 96.16% precision). It demonstrates that our discriminative rule described in (5.9) is effective and powerful despite being simple and intuitive.

5.5.8 Experimental results on precision/recall trade-off

The main contribution of our discriminative persistency algorithm is to view the decision problem for persistency as a classification problem to allow us make a trade-off between precision (having labeled variables to be actually persistent) and recall (getting more variables to be labeled). In general, we demonstrated

that by compromising precision a little bit, we can significantly boost the recall value for vision applications. The experimental results are illustrated in Table 5.4. We vary the κ value in our discriminative rule (5.9), which controls the aggressiveness of our approach. Note that $\kappa = 1$ makes our approach degenerate to conservative baselines. However, by setting $\kappa = 0.8$, we make the recall value significantly larger than before, especially on the challenging dataset, while only losing less than 0.5% precision. We can get recall value greater than 90% for all the datasets by let $\kappa = 0.7$. However, we will lose precision too much and hurt the energy. Therefore, it is crucial to achieve a balance point between balance and recall, hence provides a great trade-off between speedup and energy.

5.5.9 Investigation on parameter sensitivity

We claimed in Section 5.5.4 that the parameters chosen by the leave-one-out procedure are very similar for the same dataset. We summarized the parameters chosen by cross validation in Table 5.5. The exception column shows that, out of the 37 instances in 7 datasets we tested, the leave-one-out procedure only results in 5 instances where the parameters are different from the majority of the dataset. We also observed that the exception instance achieves good performance when applied to the majority parameter setup of the whole dataset. Therefore, we conclude the best parameter suit for one dataset is quite stable, and the parameters chosen from a set of energy can still be applicable to other energy functions derived from the same vision task.

In addition, we also observed that the proposed method achieves good performance across all the datasets we tested when the parameters are chosen

Table 5.5: Parameters chosen from the leave-one-out procedure.

Dataset	κ	Choice of q	Criterion check	Exception
Stereo	0.8	uniform	approximate	none
Inpainting	0.7	uniform	approximate	1 instance with unary
Denoise-sq	0.8	uniform	approximate	1 instance with $\kappa = 0.9$
Denoise-ts	0.7	uniform	approximate	1 instance with $\kappa = 0.8$
Optical Flow	0.9	unary	exact	1 instance with $\kappa = 0.8$, approximate check check
Color-Seg-\mathcal{N}_4	0.9	unary	exact	1 instance with $\kappa = 0.8$, uniform, approximate check
Color-Seg-\mathcal{N}_8	0.9	unary	exact	1 instance with $\kappa = 0.8$

from a wide range, including the typical parameter setup we reported in Section 5.5.10 next. Therefore, the proposed method is robust to its parameters.

5.5.10 Experimental results with a typical parameter setup

We also experimented with the following fixed parameters, to avoid the expense of cross-validation. Our experiments suggest that the proposed method can achieve good performance with the parameters in a wide range. We report the experimental results in Table 5.6 with the following fixed parameters to avoid the expense of cross-validation: $\kappa = 0.8$, $\tau = 3$, using the uniform distribution for $q(\mathbf{x})$ and checking with our efficient subroutine described in Section 5.3.2.

Even though this is a fairly conservative assumption (we use the exact same parameters for very different energy functions), we still obtain good results. We achieve a 2x-12x speedup on different datasets with the energy increasing 0.1% on the worst case. In addition, we still get lower energy on 4 of the 5 challenging dataset.

We also listed the performance of our method with the parameters selected with the leave-one-out cross validation procedure as a reference. We see that the performance of our method is very similar no matter whether we use fixed pa-

Table 5.6: Performance of discriminative persistency with typical parameters.

Dataset	Measurement	Leave one out	Typical parameters
Stereo	Speedup	1.78x	2.14x
	Energy Change	-0.06%	-0.04%
	Labeled Vars	44.76%	56.77%
	Precision	99.74%	99.71%
Inpainting	Speedup	3.40x	2.10x
	Energy Change	-1.71%	-0.53%
	Labeled Vars	74.29%	47.06%
	Precision	96.16%	99.88%
Denoising-sq	Speedup	11.83x	11.71x
	Energy Change	-0.02%	-0.03%
	Labeled Vars	97.91%	97.39%
	Precision	99.95%	99.95%
Denoising-ts	Speedup	11.91x	10.61x
	Energy Change	0.00%	-0.09%
	Labeled Vars	98.32%	96.64%
	Precision	99.79%	99.95%
Optical Flow	Speedup	4.69x	8.92
	Energy Change	-0.04%	+0.11%
	Labeled Vars	77.25%	93.74%
	Precision	99.88%	99.50%
Color-Seg-\mathcal{N}_4	Speedup	7.02x	9.31x
	Energy Change	0.00%	+0.01%
	Labeled Vars	85.74%	90.80%
	Precision	99.79%	99.50%
Color-Seg-\mathcal{N}_8	Speedup	8.33x	8.45x
	Energy Change	+0.04%	+0.05%
	Labeled Vars	90.39%	90.43%
	Precision	99.77%	99.76%

rameters or use cross validation to choose the parameters. The key observation we got from leave-one-out parameter selection still holds even with this fixed typical parameter setup, i.e., our method achieves significant speedup against baseline methods with very minor compromise on the accuracy of the partial optical labelings (usually lose $< 0.5\%$ precision). We also achieve comparable or smaller energy even though we compromise the accuracy of the partial optical labelings in the pre-processing step.

Table 5.7: Comparison between discriminative persistency and other approximate inference algorithms (TO: time out, MEM: out of memory).

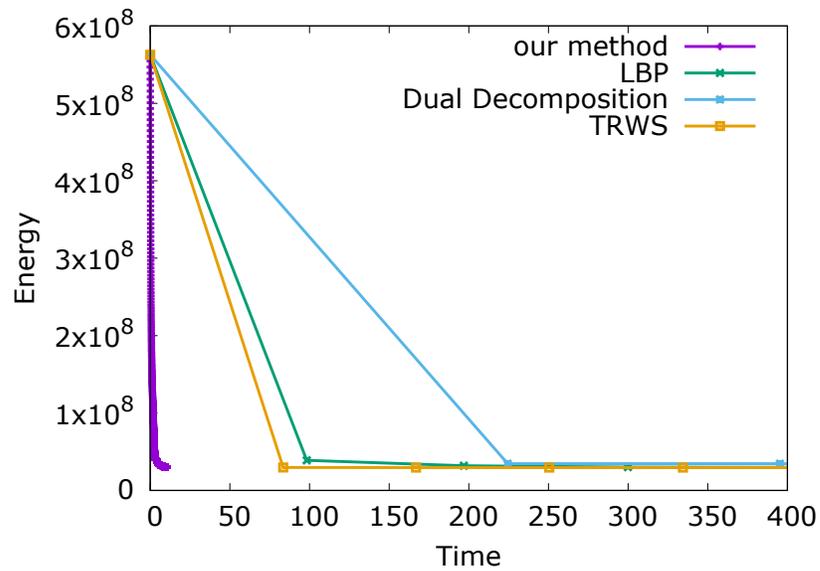
Dataset	Measurement	DisPer	LBP	DD	TRWS	MPLP
Stereo	Speedup	1.78x	0.17x	0.10x	0.10x	MEM
	Energy Change	-0.06%	+86.55%	+92.25%	-0.63%	MEM
Inpainting	Speedup	3.40x	0.10x	0.10x	0.10x	MEM
	Energy Change	-1.71%	+25.94%	+51.39%	-9.71%	MEM
Denoising-sq	Speedup	12.76x	0.10x	0.09x	0.10x	MEM
	Energy Change	-0.02%	0.00%	+17.14%	-0.65%	MEM
Denoising-ts	Speedup	13.08x	0.10x	0.09x	0.10x	MEM
	Energy Change	0.00%	-0.78%	+13.29%	-0.99%	MEM
Optical Flow	Speedup	4.69x	0.10x	0.09x	0.10x	MEM
	Energy Change	-0.04%	+9.63%	+16.07%	-0.58%	MEM
Color-Seg-\mathcal{N}_4	Speedup	7.02x	0.14x	0.10x	0.36x	0.10x
	Energy Change	0.00%	+1.72%	+3.17%	-0.13%	+0.25%
Color-Seg-\mathcal{N}_8	Speedup	8.33x	0.10x	0.10x	0.12x	0.10x
	Energy Change	+0.04%	+0.39%	+4.49%	-0.11%	+0.22%

Therefore, these experiments demonstrate that it is sufficient to use the typical parameter setup of our method in practice. We can achieve very good performance without using the expensive cross validation parameter selection procedure.

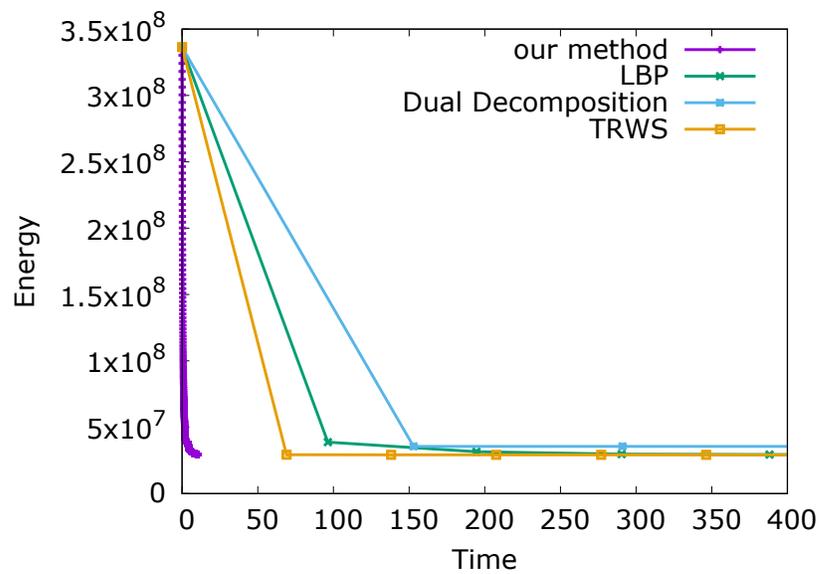
5.5.11 Comparison to other MRF inference algorithm

The main focus of this section of the thesis is to demonstrate that the proposed decision criterion is efficient and effective in finding a partial optimal labeling of MRFs. We achieve a very good trade-off between the running time and the final energy by employing our proposed method as the pre-processing for the expansion moves algorithm.

The comparison among different inference algorithms are provided in survey papers [63, 136], which demonstrates that expansion moves is the state-of-the-art MRF inference algorithm. However, for the completeness of the the-



(a) Instance 21077



(b) Instance 86000

Figure 5.6: Energy vs. time curves compared with approximate inference algorithms on denoising-sq dataset.

sis, we still perform the experiments comparing against other widely used MRF inference algorithms besides expansion move, including loopy belief propagation (LBP) [105, 146], dual decomposition (DD) [64], TRWS [74] and MPLP [41, 132, 133].

The experimental results are reported in Table 5.7. We set the time budget for the baseline methods as the 10x of the running time used by expansion moves. In our experiments, expansion move is usually significantly faster than other methods, and results in comparable or even better energy. This observation is consistent with the survey papers [63, 136]. We can see that LBP, DD, and MPLP usually will get higher energy compared to expansion moves even with 10x of time budget. TRWS is promising since it can provide (slightly) lower energy than expansion moves, although it is much slower. On the datasets we tested, TRWS will spend 3-10x longer time to get energy comparable to our proposed method, through its final energy might be slightly smaller. Typical energy-time curves are presented in Fig. 5.6. We can see that LBP, DD, TRWS are usually much slower than our method with comparable converging energy.

5.5.12 Visualization results

We present the visualization results on the stereo task in Fig. 5.7. We can see there is no significant visual difference between the expansion moves results and our results, even in the case that our method has slightly higher energy. Therefore, it is appealing to apply our method in practice, since it has almost the same visual quality but makes the inference much faster. When we set up a limited time budget in real applications, see the middle column of Fig. 5.7, our approach can generate much better visual result than regular expansion moves algorithm without pre-processing. In this case, regular expansion moves even doesn't finish its first epoch and has a very poor disparity map.

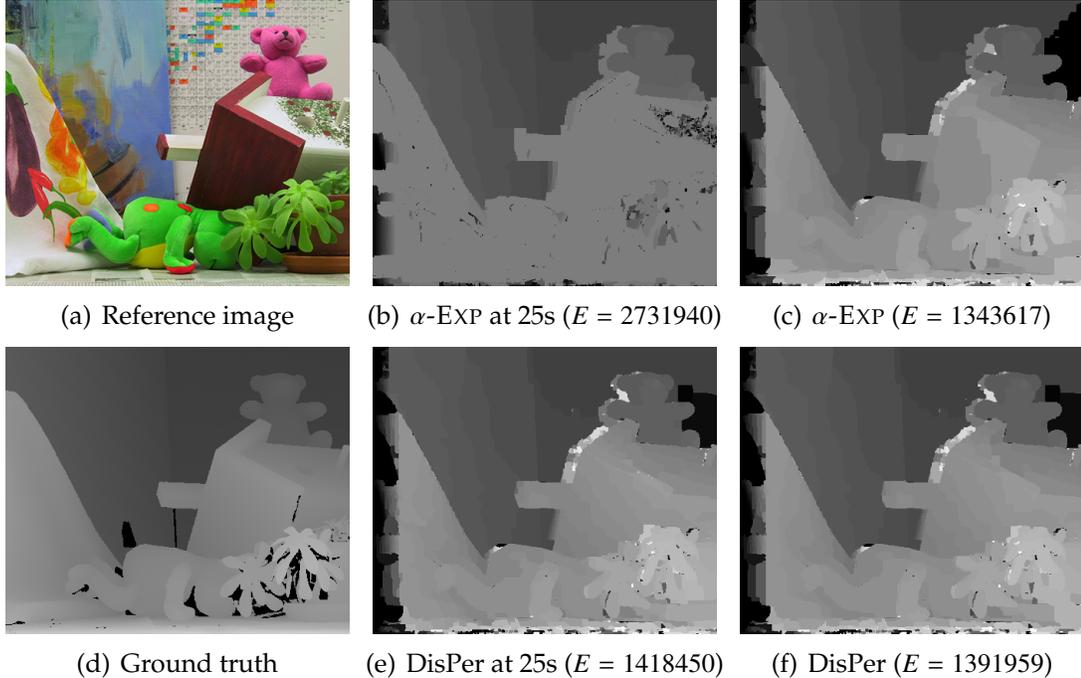


Figure 5.7: Visualization of stereo instance Teddy.

5.5.13 Discriminative persistency for multilabel MRFs

In the previous experiments, we mainly focused on applying the proposed pre-processing technique to each induced binary subproblem from the expansion moves algorithm. We can also apply the proposed pre-processing technique to the multilabel MRFs directly. We give preliminary results in Table 5.8. Since the multilabel MRFs are NP-hard, it is very challenging to get the ground truth persistent labeling for each variable. Therefore, we don't report the precision/recall values. We just use the energy change as an indirect measurement to evaluate the quality of the persistent labeling we found. We also reported the percentage of labeled variables. Both metrics are computed in the per-dataset fashion. We set our distribution $q_i(x_i) = e^{-\theta_i(x_i)}$, which is only from the unary terms and used the fast approximation subroutine to check our discriminative criterion. Then we vary the κ value in $\{0.6, 0.7, 0.8, 0.9\}$. We can see from Table 5.8 that the

Table 5.8: Preliminary experimental results on multilabel MRFs.

Algorithm	Color-Seg- \mathcal{N}_4	
	% of labeled variables	Energy Change
α -EXP	0.00%	0.0000%
DEE	15.62%	0.0000%
Ours with $\kappa = 0.9$	25.55%	+2.3589%
Ours with $\kappa = 0.8$	30.77%	+3.3480%
Ours with $\kappa = 0.7$	34.01%	+4.4572%
Ours with $\kappa = 0.6$	44.62%	+5.7841%
	Color-Seg- \mathcal{N}_8	
α -EXP	0.00%	0.0000%
DEE	19.67%	0.0000%
Ours with $\kappa = 0.9$	29.80%	+0.1049%
Ours with $\kappa = 0.8$	30.62%	+0.1161%
Ours with $\kappa = 0.7$	31.05%	+0.1243%
Ours with $\kappa = 0.6$	31.27%	+0.1297%

proposed method labels significantly more variables than the baseline method DEE, while increases the energy by a couple of percents. It is still the case that the proposed method can achieve a better trade-off between the number of labeled variables and the energy we can get for the multilabel MRFs. In practice, it is more effective to apply our proposed method to each induced binary sub-problems.

5.5.14 Experimental results on worst case bound parameter ϵ

In all the other experiments presented in the thesis, we set the parameter $\epsilon = \infty$, and investigated how our algorithm performed without the worst case bound. We have demonstrated the proposed discriminative rule (5.9) itself is empirically effective. All the post-running per-instance bounds we proved in Section 5.4.2 are still sound, although in this variant of our method there is no

Table 5.9: Experimental results with different ϵ on Color-Seg- \mathcal{N}_4 dataset.

$\kappa = 0.8$				
ϵ	Speedup	Energy Change	Labeled Vars	Precision
0	4.16x	0.00%	37.82%	100.00%
0.01	4.31x	0.00%	67.73%	99.99%
0.1	6.05x	0.00%	72.07%	99.93%
0.2	6.68x	0.00%	74.67%	99.86%
0.3	6.97x	0.00%	76.32%	99.81%
0.4	7.07x	0.00%	77.86%	99.80%
0.5	7.59x	0.00%	81.16%	99.74%
0.6	7.79x	+0.01%	82.49%	99.67%
0.7	7.84x	+0.01%	82.67%	99.67%
0.8	7.83x	+0.01%	84.87%	99.69%
0.9	7.87x	+0.01%	86.43%	99.69%
1.0	7.92x	+0.01%	88.48%	99.69%
10.0	8.12x	+0.01%	90.80%	99.50%
$\kappa = 0.6$				
0	4.16x	0.00%	37.82%	100.00%
0.01	4.46x	0.00%	68.25%	99.97%
0.1	6.47x	+0.01%	73.69%	99.70%
0.2	7.93x	+0.20%	78.32%	99.34%
0.3	8.38x	+0.34%	81.43%	99.12%
0.4	9.62x	+1.29%	85.91%	98.68%
0.5	11.73x	+3.01%	88.41%	97.59%
0.6	12.06x	+6.88%	89.29%	96.54%
0.7	12.14x	+6.88%	89.82%	96.54%
0.8	12.29x	+6.88%	95.67%	96.54%
0.9	12.23x	+6.88%	96.01%	96.53%
1.0	12.38x	+6.88%	96.25%	96.51%
10.0	15.02x	+7.83%	98.52%	94.77%

worst case theoretical guarantee.

However, if we combine (5.9) and $\bar{\delta}_i \leq \epsilon$ as our decision rule, as described in Section 5.4.3, we will have the worst case bounds. We also conducted experiments with different ϵ 's. The experimental results on Color-Seg- \mathcal{N}_4 dataset are summarized in Table 5.9.

We first applied the typical parameter setup we used in Section 5.5.10. The results are reported on the top part of Table 5.9. Note that $\epsilon = 0$ is the special case where our method only uses the sound condition to check the partial optimal labeling, hence the proposed algorithm degenerates to the DEE algorithm. Therefore, in this special case, we have a 100% precision and label around 38% variables, hence we get a moderate speedup without affecting the energy. We also know that $\epsilon = \infty$ is another special case where we don't try to bound the worst case. These results is reported in the Table 5.3 and Table 5.6. We already know that the fixed parameters we choose here are reasonable, so even in this extreme case, we still get good performance without the theoretical guarantee. As ϵ decreases, we know that the criterion used becomes more strict. Therefore we will have higher precision and less labeled variables. Due to that, we label fewer variables, and the speedup we achieve decreases. In this setup, since we always maintain the precision value at a extremely high level, ϵ 's impact on energy change is not that obvious.

To test this, we conducted the experiments under another set of purposely bad parameters, i.e., changed $\kappa = 0.6$. We summarized our results on the bottom part of Table 5.9. We see that with $\kappa = 0.6$ and large ϵ value (e.g., $\epsilon = 10$), our criterion is loose enough to hurt the precision and result in 8% higher energy than before. In our experiments, we observed that as ϵ decreases from 10 down to 0, the precision increases dramatically and the energy increment becomes smaller. Therefore, ϵ values not only give us the theoretical worst case guarantee, but also make real impact in practice (make the criterion we used close to the sound condition and makes the energy smaller).

6.1 Motivation

In this chapter, we will discuss the persistency for higher-order MRFS, which coincides with the minimizer of the following energy function:

$$f(\mathbf{x}) = \sum_{i \in V} \theta_i(x_i) + \sum_{C \in \mathcal{C}} \theta_C(\mathbf{x}_C) \quad (6.1)$$

where $\mathcal{G} = (V, C)$ is the hypergraph representation of the higher-order MRF, and $\theta_i : \mathcal{L}_i \mapsto \mathbb{R}$ are the unary costs and $\theta_C : \prod_{i \in C} \mathcal{L}_i \mapsto \mathbb{R}$ are the higher-order costs.

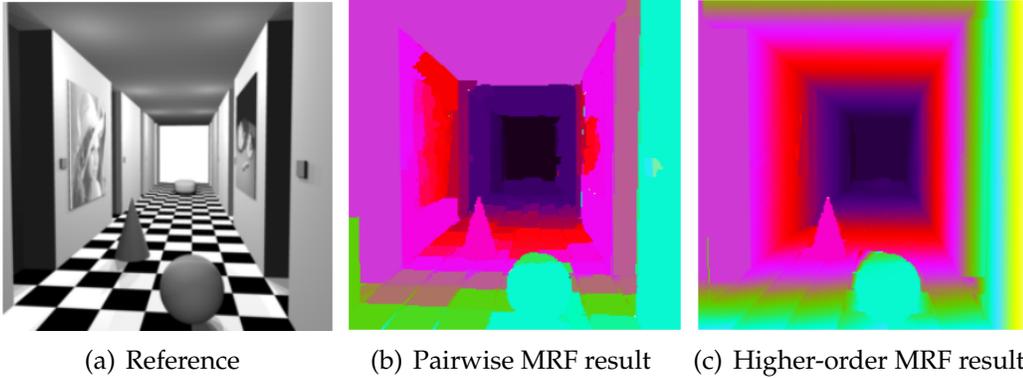


Figure 6.1: Higher-order stereo [149] © IEEE. Higher-order MRF formalization achieves smoother result compared to pairwise MRF formalization.

Higher-order MRFS provide a much more flexible way to model prior terms in vision applications. Therefore, employing higher-order MRF formalization can significantly improve the quality of vision application. We have shown a couple of examples in Figure 6.1, 6.2 and 6.3.

Woodford *et al.* [149] proposed a second-order curvature-based prior for stereo problem. As we can see from Figure 6.1, the higher-order MRF result is much smoother compared to the first-order MRF formalization.



(a) Ground truth

(b) Noisy input

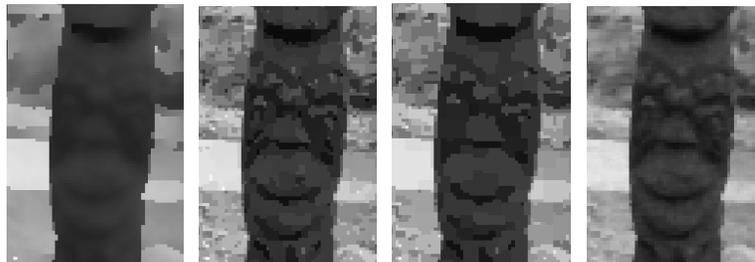
(c) Pairwise MRF result



(d) Pairwise MRF result

(e) Pairwise MRF result

(f) Higher-order MRF result



(g) Detail of (c) (h) Detail of (d) (i) Detail of (e) (j) Detail of (f)

Figure 6.2: Higher-order image denoising [91] © Springer. (c) - (e) Denoising results from various pairwise MRF formalizations. (f) Denoising results from higher-order MRF. (g) - (j) Details of (c) - (f). Higher-order MRF achieves better quality.

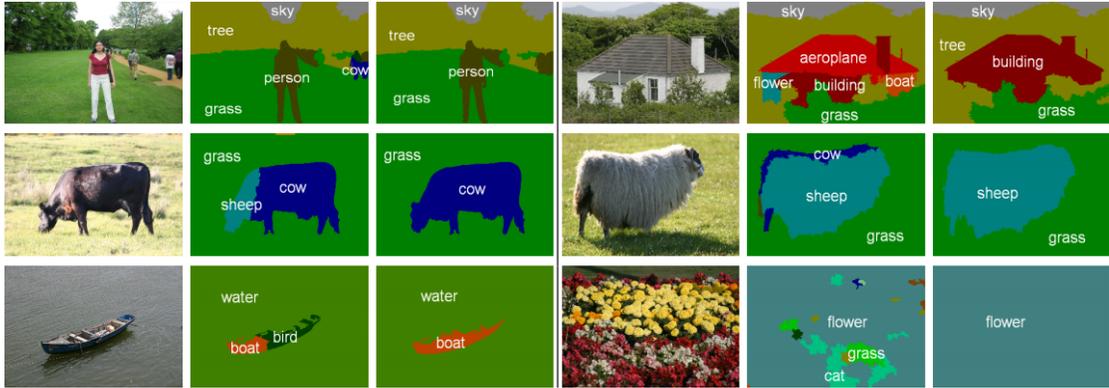


Figure 6.3: Higher-order semantic segmentation [89] © Springer. (Left) Input image. (Middle) Pairwise MRF result. (Right) Higher-order MRF result. Higher-order MRF formalization achieves more accurate results due to the co-occurrence prior.

Lan *et al.* [91] learned the 2×2 patch statistics of natural images, and applied it for image denoising task. Figure 6.2 shows that this higher-order prior leads to cleaner denoising results compared to various different pairwise MRF formalizations.

Ladicky *et al.* [89, 90] proposed a higher-order co-occurrence prior for semantic segmentation. It improves the label coherence in the segmentation results, as we can see from Figure 6.3.

Higher-order MRF inference problem is more challenging compared to the pairwise MRFs. However, its promising performance on vision applications still motivates researchers to explore the inference algorithms for them. We have reviewed many of these algorithms for higher-order MRF in our related work Section 3.6. There are a few existing work on the persistency of higher-order MRFs. However, generalized roof duality [62] is limited to MRFs up to third-order. The bi-submodular relaxation [75] is a pure theoretical work and hard to applied to practice. MQPBO [71], PBP [134] and IRI [129] can be generalized to higher-order MRFs, but they are all extremely computationally expensive. Note

that Shekhovtsov [127] conducted an experiment to compare these methods, but only with synthetic MRFs with 100 variables and second-order and third-order priors.

Therefore, in this chapter, we will explore on the computational tractable persistency approaches, e.g., dead-end elimination (DEE) [31]. We will also explore our discriminative view of MRF persistency in the higher-order case. We wonder whether it is still true that the autarky property is too restrictive as a relaxation of persistency for higher-order MRFs.

6.2 Generalized persistency conditions for higher-order MRFs

6.2.1 Generalized Dead-end Elimination

We have already presented the original Dead-end Elimination (DEE) [31] and its straightforward improvement with Goldstein condition [44] in Section 2.4.1. Actually, both of these two conditions can be naturally generalized to higher-order cases, although the original publications didn't discuss this generalization.

Theorem 6.2.1 (higher-order DEE condition). *We can rule out $x_i = \ell$ to be persistent for a single variable when $\exists \ell' \in \mathcal{L}_i, \ell' \neq \ell$ such that:*

$$\theta_i(\ell) + \sum_{C \in \mathcal{C}: i \in C} \min_{\mathbf{x}_C \in \mathcal{L}_C: (\mathbf{x}_C)_i = \ell} \theta_C(\mathbf{x}_C) > \theta_i(\ell') + \sum_{C \in \mathcal{C}: i \in C} \max_{\mathbf{x}_C \in \mathcal{L}_C: (\mathbf{x}_C)_i = \ell'} \theta_C(\mathbf{x}_C). \quad (6.2)$$

Proof. By contradiction, assuming $x_i = \ell$ is persistent and $\mathbf{x}^* \in \operatorname{argmin}_{\mathbf{x}} f(\mathbf{x})$. We know $x_i^* = \ell$ due to persistency. Now, let's consider the energy change when we

substitute $x_i = \ell'$ into \mathbf{x}^* to get $\mathbf{x}_{-i}^* \oplus (x_i = \ell')$:

$$\begin{aligned}
& f(\mathbf{x}^*) - f(\mathbf{x}_{-i}^* \oplus (x_i = \ell')) \\
&= \theta_i(\ell) - \theta_i(\ell') + \sum_{C \in \mathcal{C}: i \in C} \left(\theta_C(\mathbf{x}_C^*) - \theta_C(\mathbf{x}_{C-i}^* \oplus (x_i = \ell')) \right) \\
&= \theta_i(\ell) - \theta_i(\ell') + \sum_{C \in \mathcal{C}: i \in C} \theta_C(\mathbf{x}_C^*) - \sum_{C \in \mathcal{C}: i \in C} \theta_C(\mathbf{x}_{C-i}^* \oplus (x_i = \ell')) \quad (6.3) \\
&\geq \theta_i(\ell) - \theta_i(\ell') + \sum_{C \in \mathcal{C}: i \in C} \min_{\mathbf{x}_C \in \mathcal{L}_C: (\mathbf{x}_C)_i = \ell} \theta_C(\mathbf{x}_C) - \sum_{C \in \mathcal{C}: i \in C} \max_{\mathbf{x}_C \in \mathcal{L}_C: (\mathbf{x}_C)_i = \ell'} \theta_C(\mathbf{x}_C) \\
&> 0,
\end{aligned}$$

where \mathbf{x}_{-i} is a subvector of \mathbf{x} except the i -th entry, \mathbf{x}_{C-i} is a subvector of \mathbf{x} on $C \setminus \{i\}$. The first equation is from expanding the definition of higher-order MRF energy, and cancelling the same terms. The third step is from a standard min/max relaxation. The last step is from our condition (6.2). Therefore, we have $f(\mathbf{x}_{-i}^* \oplus (x_i = \ell')) < f(\mathbf{x}^*)$, which contradicts \mathbf{x}^* to be the minimizer of $f(\mathbf{x})$. So we can exclude the possibility that $x_i = \ell$ to be persistent if (6.2) holds. \square

Theorem 6.2.2 (higher-order Goldstein condition). *We can rule out $x_i = \ell$ to be persistent for a single variable when $\exists \ell' \in \mathcal{L}_i, \ell' \neq \ell$ such that:*

$$\theta_i(\ell) - \theta_i(\ell') + \sum_{C \in \mathcal{C}: i \in C} \min_{\mathbf{x}_{C-i} \in \mathcal{L}_{C-i}} \left(\theta_C(\mathbf{x}_{C-i} \oplus (x_i = \ell)) - \theta_C(\mathbf{x}_{C-i} \oplus (x_i = \ell')) \right) > 0. \quad (6.4)$$

Proof. It's similar to the proof in Theorem 6.2.1. Let's assume $x_i = \ell$ is persistent and $\mathbf{x}^* \in \operatorname{argmin}_{\mathbf{x}} f(\mathbf{x})$ and consider the energy change when we substitute $x_i = \ell'$ into \mathbf{x}^* :

$$\begin{aligned}
& f(\mathbf{x}^*) - f(\mathbf{x}_{-i}^* \oplus (x_i = \ell')) \\
&= \theta_i(\ell) - \theta_i(\ell') + \sum_{C \in \mathcal{C}: i \in C} \left(\theta_C(\mathbf{x}_C^*) - \theta_C(\mathbf{x}_{C-i}^* \oplus (x_i = \ell')) \right) \\
&\geq \theta_i(\ell) - \theta_i(\ell') + \sum_{C \in \mathcal{C}: i \in C} \min_{\mathbf{x}_{C-i} \in \mathcal{L}_{C-i}} \left(\theta_C(\mathbf{x}_{C-i} \oplus (x_i = \ell)) - \theta_C(\mathbf{x}_{C-i} \oplus (x_i = \ell')) \right) \quad (6.5) \\
&> 0,
\end{aligned}$$

where \mathbf{x}_{-i} is a subvector of \mathbf{x} except the i -th entry, \mathbf{x}_{C-i} is a subvector of \mathbf{x} on $C \setminus \{i\}$. Therefore, we have $f(\mathbf{x}_{-i}^* \oplus (x_i = \ell')) < f(\mathbf{x}^*)$, which contradicts \mathbf{x}^* to be the minimizer of $f(\mathbf{x})$. So we can exclude the possibility that $x_i = \ell$ to be persistent if (6.4) holds. \square

Note that the higher-order Goldstein condition (6.4) is still a straightforward improvement of the higher-order DEE condition (6.2). Therefore, given any higher-order MRFs, we can take the same iterative scheme used in the pairwise DEE algorithm to iterative over all the variables and its remaining labels, and apply the higher-order Goldstein condition to test whether we can eliminate $x_i = \ell$ to be persistent. Similarly, we prove persistency once we can eliminate all but one label for a given variable.

6.2.2 Generalized discriminative persistency

It's easy to see our equivalent definition of persistency and autarky introduced in Section 5.2.1 is still valid for higher-order MRFs. As a recapitulation, we have \mathbf{x}_S to be persistent if and only if

$$\min_{\mathbf{y}_S \neq \mathbf{x}_S} \Delta f(\mathbf{y}_S \leftarrow \mathbf{x}_S \mid \mathbf{z}_{N(S)}) > 0, \forall \mathbf{z}_{N(S)} \in \mathcal{L}_{N(S)}^*, \quad (6.6)$$

and \mathbf{x}_S is an autarky if and only if

$$\min_{\mathbf{y}_S \neq \mathbf{x}_S} \Delta f(\mathbf{y}_S \leftarrow \mathbf{x}_S \mid \mathbf{z}_{N(S)}) > 0, \forall \mathbf{z}_{N(S)} \in \mathcal{L}_{N(S)}. \quad (6.7)$$

Then we can still apply our discriminative criterion to claim \mathbf{x}_S to be persistent when

$$\sum_{\mathbf{z}_{N(S)} \in \hat{\mathcal{L}}_{N(S)}(\mathbf{x}_S)} q(\mathbf{z}_{N(S)}) \geq \kappa. \quad (6.8)$$

where $\kappa \in [0, 1]$ is still our confidence threshold, $\hat{\mathcal{L}}_{\mathcal{N}(S)}(\mathbf{x}_S)$ is the set of $\mathbf{z}_{\mathcal{N}(S)}$ such that $\Delta f(\mathbf{y}_S \leftarrow \mathbf{x}_S \mid \mathbf{z}_{\mathcal{N}(S)}) > 0$ and $q(\mathbf{z}_{\mathcal{N}(S)})$ is our importance score of $\mathbf{z}_{\mathcal{N}(S)}$.

The nature of higher-order MRFs requires us to have a even more efficient subroutine to compute the LHS in (6.8) compared to the first-order MRFs. To simplify the problem, we will only consider the discriminative rule for a single variable x_i with $q(\cdot)$ to be a uniform distribution.

We can still compute the exact sum in (6.8), with the cost of $\mathcal{O}(L^{|\mathcal{N}(i)|+1})$ oracle calls to evaluate the MRF energy change $\Delta f(y_i \leftarrow x_i \mid \mathbf{z}_{\mathcal{N}(i)})$. Given the fact that each variables is usually associated with a larger neighbor $\mathcal{N}(i)$ for higher-order MRFs compared to the most widely used \mathcal{N}_4 structure for pairwise MRFs, it is computationally inefficient to compute the exact sum in general.

We can also use our approximation algorithm proposed in Section 5.3.2 to computes a lower bound of the sum in (6.8) more efficiently. The computational cost of this approach is $\mathcal{O}(KML^{K+1} + dNL^{K+1})$ oracle calls to evaluate prior terms¹, where K is the maximum size of a clique and d is the maximum degree of a variable. This approach is more applicable in practice since it doesn't depend on the size of all variables adjacent to x_i , but the maximum clique size adjacent to x_i .

We also propose an alternative rank-based heuristics for higher-order MRFs, which is faster than the approximated sum above. Recalling the Goldstein condition for DEE is the following:

$$\theta_i(\ell) - \theta_i(\ell') + \sum_{C \in \mathcal{C}: i \in C} \min_{\mathbf{x}_{C-i} \in \mathcal{L}_{C-i}} (\theta_C(\mathbf{x}_{C-i} \oplus (x_i = \ell)) - \theta_C(\mathbf{x}_{C-i} \oplus (x_i = \ell'))) > 0. \quad (6.9)$$

We can sort all the $(\theta_C(\mathbf{x}_{C-i} \oplus (x_i = \ell)) - \theta_C(\mathbf{x}_{C-i} \oplus (x_i = \ell')))$ values for $\mathbf{x}_{C-i} \in \mathcal{L}_{C-i}$

¹Assuming we only run the whole algorithm for constant iterations.

in a rank list A , and define the value

$$\delta_C^{x_i:\ell \rightarrow \ell'}(\rho) := \max \left\{ v \mid |\{v' \in A \mid v' \geq v\}| \geq \rho|A| \right\} \quad (6.10)$$

to be the maximum value that ρ percentage of values in the rank list are not less than it. Then we can write our discriminative criterion as:

$$\theta_i(\ell) - \theta_i(\ell') + \sum_{C \in \mathcal{C}: i \in C} \delta_C^{x_i:\ell \rightarrow \ell'}(\rho) > 0. \quad (6.11)$$

When $\rho = 1.0$, we have $\delta_C^{x_i:\ell \rightarrow \ell'}(\rho)$ to be the minimum value in the rank list, i.e., (6.11) degenerates to Goldstein condition in DEE. This approach takes $\mathcal{O}(KML^K \log L^K + dL)$ oracle calls to evaluate prior terms².

We can also show that our per-instance error bound and worst-case error bound in Theorem 5.4.2 and 5.4.4 is also applicable to higher-order MRFs, when we use either the exact or approximate way to compute (6.8). The rank-based criterion in (6.11) is a pure heuristics, so we don't have any theoretical guarantee on it.

6.3 Experimental results

6.3.1 Experiments setup

The overall experiments setup for higher-order MRFs is very similar to the experiments setup we introduced in Section 5.5.

We will use the Field of Experts (FoE) image denoising [112] as the benchmark to evaluate our discriminative persistency for higher-order MRFs. FoE is

²Assuming we only run the whole algorithm for constant iterations.

a patch-based prior learned from natural images. Therefore, we can formalize the denoising task as a higher-order MRF with $\mathcal{N}_{2 \times 2}$ structure. We follow the setup in [36] to generate the dataset. The FoE model is trained on the training set of BSDS300 [100]. Then we choose 10 instances from the testing set³. We downsample each image to 161×241 , convert them to grayscale and add random Gaussian white noise to generate the noisy input can constructed 10 higher-order MRF instances for evaluation. Therefore, we have 38801 variables, 256 labels and 38400 2×2 patch-based prior terms.

We will use the fusion move algorithm with the sum-of-submodular flow (SoS-flow) introduced in Section 2.5.4 as the approximate inference algorithm (denoted as SoS-Fusion). Following [36] again, we will use the gradient-based proposal generation described in [56] and the sum-of-submodular upper bound proposed in [36] to handle to non-SoS induced binary subproblems.

For the persistency algorithms, we will only consider the DEE and our discriminative persistency (DisPer) for each induced binary subproblem generated in the fusion move algorithm, since all the other potential methods are too slow at the scale of real problems we have in the FoE dataset [127]. For discriminative persistency, our preliminary experimental results showed that when we use the exact check of the criterion, it will be around 3x slower than the fusion move without pre-processing, due to the large adjacent neighborhood we need to enumerate for each variable. Therefore, we will only consider the approximate check (denoted as DisPer-ApproxSum) and the new rank-based heuristics proposed in Section 6.2.2 (denoted as DisPer-Rank).

We will still evaluate the proposed method from the speed and energy point

³We chose 3096, 38082, 58060, 86000, 106024, 134035, 163085, 208001, 241048 and 299086 in our experiments.

Table 6.1: Experimental results for higher-order discriminative persistency.

Algorithm	Speedup	Energy Change	Labeled Vars
SoS-Fusion	1x	0.0000%	0.00%
DEE	0.96x	-0.3150%	26.25%
DisPer-ApproxSum	1.10x	-0.6013%	65.00%
DisPer-Rank	1.84x	-0.4918%	81.30%

of view. So we will reuse the Speedup measurement and the energy change measurement defined in Section 5.5.3, with the fusion move as the baseline. We will also report the percentage of labeled variables. But due to the extreme hardness of get the global minimizer for the higher-order MRFs, even for the binary higher-order MRFs, we won't report precision value.

All the experiments were executed on a single machine with dual 3GHz Intel i7 Core and 16GB 1600 MHz DDR3 memory.

6.3.2 Overall performance of discriminative persistency

We summarize the overall experimental results in Table 6.1, with a typical parameter setup that we have number of iterations $\tau = 1$ for DEE, DisPer-ApproxSum and DisPer-Rank, the confidence threshold $\kappa = 0.6$ for DisPer-ApproxSum and the rank threshold $\rho = 0.5$ for DisPer-Rank.

We can see that all the persistency algorithms achieve lower energy compared to the fusion move algorithm with pre-processing. This result is consistent for all the 10 instances among these three persistency algorithms on FoE dataset. The major reason make our persistency algorithm achieve better energy is due to in the FoE denoising, our prior term is learned from natural im-

age. Therefore, the induced binary subproblem for fusion move is not sum-of-submodular. Unlike we employ QPBO to solve the non-submodular induced subproblem for pairwise MRFs, we minimize the sum-of-submodular upper bound as an approximation (similar to the truncation scheme [116] for pairwise MRFs). This approximation can guarantee that we won't have a higher energy, but it cannot guarantee that the new labeling we have is persistent for any variables. Therefore, any persistency information can help us not only reduce the scale of the problem to speedup inference, but also nail down several variables to simplify the inference. Therefore, all the persistency algorithms get lower energy.

Furthermore, DisPer-ApproxSum gets the lowest energy since it labels significantly more variables than DEE. Analogue to the discussion of discriminative persistency for pairwise MRFs, DisPer-ApproxSum will label more variables correctly compared to DEE, due to its (hypothetical) high precision. The gain we get from these additional correctly label variables pay off the loss from the small number of wrongly labeled variables. DisPer-ApproxSum also achieves lower energy than DisPer-Rank since the approximate sum scheme is a better and more principled way to estimate our confidence, compared to the rank-based heuristics used in DisPer-Rank.

We can also see from Table 6.1 that both DisPer-ApproxSum and DisPer-Rank is faster than the original fusion move algorithm, while DEE is slower. DEE only labels around 26.25% variables, so that the speed gain from SoS-flow doesn't pay off the computational overhead introduced by DEE. In contrast, both DisPer-ApproxSum and DisPer-Rank labels significantly more variables than DEE, hence they speedup the overall inference. But the speedup gain we

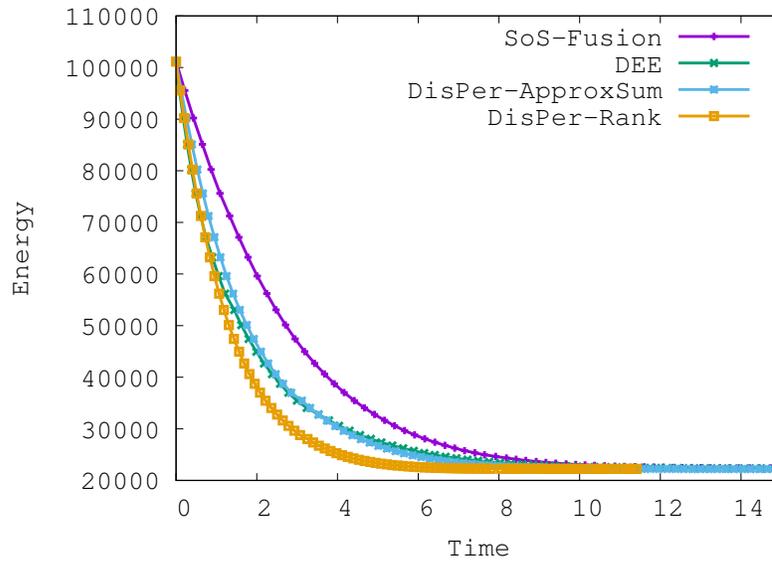


Figure 6.4: Energy vs. time curves for instance 106024 on FoE dataset.

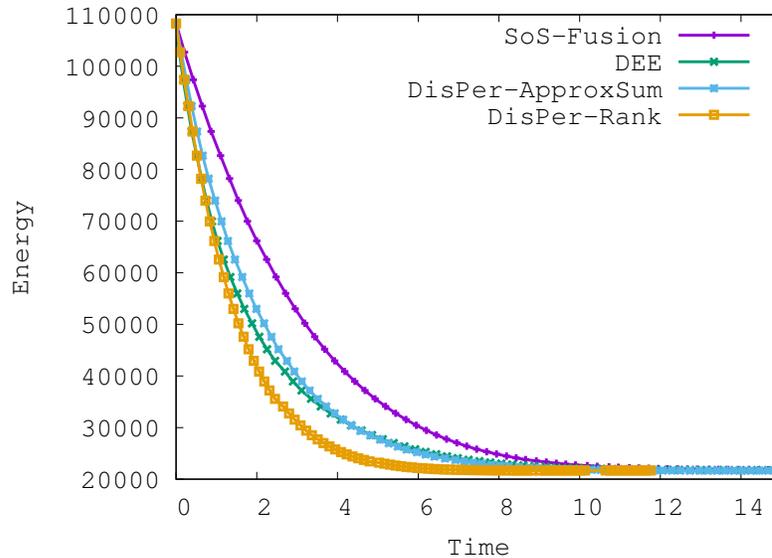


Figure 6.5: Energy vs. time curves for instance 299086 on FoE dataset.

have for higher-order MRFs is not as significantly as the ones in pairwise MRFs. The major reason is due to the overhead of computing discriminative persistency for higher-order MRFs is much larger than pairwise MRFs.

A typical speed vs. energy curve is illustrated in Figure 6.4 and 6.5. We can

Table 6.2: Experimental results for DisPer-ApproxSum with different κ .

κ	Speedup	Energy Change	Labeled Vars
1.0 (DEE)	0.96x	-0.3150%	26.25%
0.9	0.88x	-0.4920%	37.65%
0.8	1.01x	-0.5439%	44.91%
0.7	1.07x	-0.5953%	57.00%
0.6	1.10x	-0.6013%	65.00%
0.5	1.25x	-0.5890%	69.50%
0.4	1.32x	-0.5805%	71.10%

see that DisPer-Rank drives the energy decreasing faster than all the other methods, which is consistent with its overall running time is the fastest. The curve for DEE and DisPer-ApproxSum are similar. We can see that even DEE makes the energy decreasing faster than SoS-Fusion, although its overall running time is longer.

In sum, the conclusion we drawn for pairwise MRFs still hold in the higher-order case: discriminative persistency can provide a better speed energy trade-off than DEE. It makes the fusion move algorithm both faster and better.

6.3.3 Experimental results on parameter sensitivity

We also studied the sensitivity of the parameters for our discriminative persistency algorithm and how does these parameters affect algorithm's performance.

We first vary our confidence threshold of our discriminative criterion $\kappa \in \{0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$ for DisPer-ApproxSum. DisPer-ApproxSum degenerates to DEE when $\kappa = 1.0$. We summarize the experimental results in Table 6.2 and illustrate the time vs. energy curve for different κ values on instance 106024 in Figure 6.6. Our discriminative criterion is looser with smaller

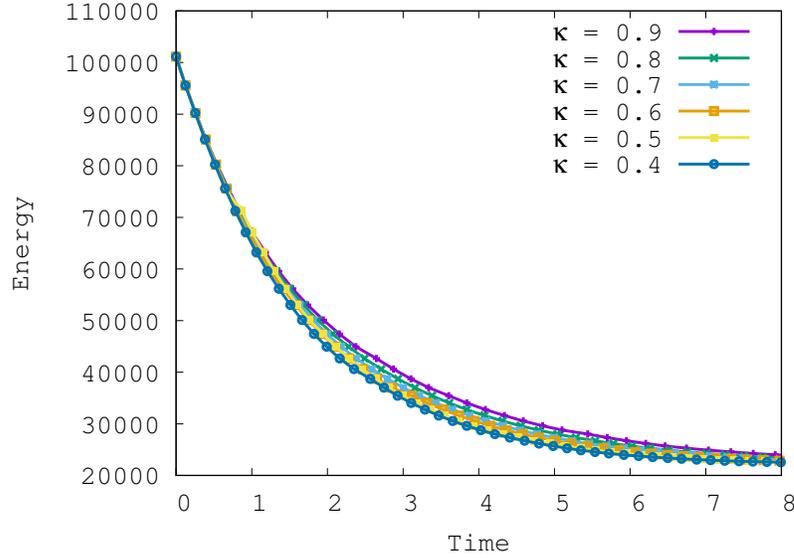


Figure 6.6: Energy vs. time curves for DisPer-ApproxSum with different κ .

Table 6.3: Experimental results for DisPer-Rank with different ρ .

ρ	Speedup	Energy Change	Labeled Vars
1.0 (DEE)	0.96x	-0.3150%	26.25%
0.875	0.89x	-0.3150%	26.25%
0.75	0.93x	-0.5080%	40.22%
0.625	1.27x	-0.5992%	59.41%
0.5	1.84x	-0.4918%	81.30%
0.375	2.47x	+0.6630%	100.00%

κ values. So it is obvious that the speedup and percentage of labeled variables increase monotonically when we decrease κ (except $\kappa = 1.0$ to $\kappa = 0.9$, the algorithm is slower due to the extra overhead introduced by DisPer-ApproxSum). On the other hand, the trade-off between precision and recall of our discriminative rule makes then energy decreases initially due to we label more variables correctly, then increase due to our persistency condition becomes too loose that the wrongly labeled variables start to dominate.

We also vary our rank threshold $\rho \in \{0.375, 0.5, 0.625, 0.75, 0.875, 1.0\}$ for DisPer-Rank. DisPer-Rank degenerates to DEE when $\rho = 1.0$. We choose this

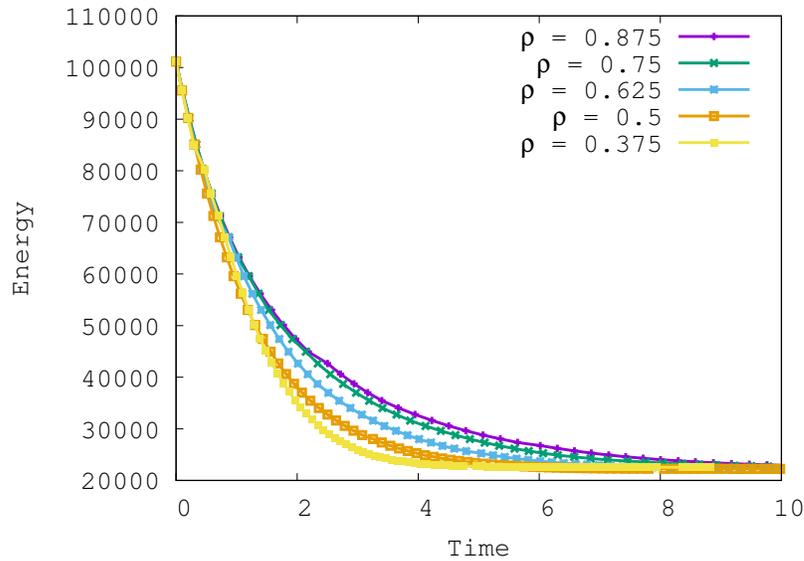


Figure 6.7: Energy vs. time curves for DisPer-Rank with different ρ .

set of ρ values since we have 4-clique priors on FoE dataset. So we need to rank $2^{(4-1)} = 8$ energy differences in DisPer-Rank. We summarize the experimental results in Table 6.3 and illustrate the time vs. energy curve for different ρ values on instance 106024 in Figure 6.7. We can achieve the same conclusion on parameter ρ for DisPer-Rank as the parameter κ for DisPer-ApproxSum. Our discriminative criterion is looser with smaller ρ values. So the speedup and percentage of labeled variables increase monotonically with the decrements of ρ (except $\rho = 1.0$ to $\rho = 0.9$ again, due to the extra overhead introduced by DisPer-Rank). The trade-off between precision and recall also makes then energy decreases initially, then increase.

We can see from Table 6.2 and 6.3 that our proposed discriminative persistency algorithm can achieve good performance with a wide range of parameter setups. Therefore, we can also see the discriminative persistency algorithm is robust to these parameters.

BIBLIOGRAPHY

- [1] Aseem Agarwala, Mira Dontcheva, Maneesh Agrawala, Steven Drucker, Alex Colburn, Brian Curless, David Salesin, and Michael Cohen. Interactive digital photomontage. *ACM Transactions on Graphics (ToG)*, 23(3):294–302, 2004.
- [2] Ravindra K. Ahuja, Özlem Ergun, James B. Orlin, and Abraham P. Punnen. A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics*, 123(1):75–102, 2002.
- [3] Karteek Alahari, Pushmeet Kohli, and Philip H.S. Torr. Dynamic hybrid algorithms for MAP inference in discrete MRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 32(10):1846–1857, 2010.
- [4] Anurag Arnab, Sadeep Jayasumana, Shuai Zheng, and Philip H.S. Torr. Higher order conditional random fields in deep neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 524–540, 2016.
- [5] Chetan Arora, Subhashis Banerjee, Prem Kalra, and S.N. Maheshwari. Generic cuts: An efficient optimal algorithm for submodular MRF-MAP problems with higher order cliques. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 17–30, 2012.
- [6] Chetan Arora, Subhashis Banerjee, Prem Kalra, and S.N. Maheshwari. Fast approximate inference in higher order MRF-MAP labeling problems. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1338–1345, 2014.
- [7] Chetan Arora and S.N. Maheshwari. Multi label generic cuts: Optimal inference in multi label multi clique MRF-MAP problems. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1346–1353, 2014.
- [8] Julian Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society. Series B (Methodological)*, 48(3):259–302, 1986.
- [9] Andrew Blake, Pushmeet Kohli, and Carsten Rother. *Markov Random Fields for vision and image processing*. Mit Press, 2011.

- [10] Endre Boros. Pseudo-boolean optimization - in memory of P.L. Hammer. Talk at Boolean Seminar Liblice 2013, Czech Republic, 2013.
- [11] Endre Boros and Peter L. Hammer. A generalization of the pure literal rule for satisfiability problems. Technical report, RRR 20-1992, RUTCOR, 1992.
- [12] Endre Boros and Peter L. Hammer. Pseudo-boolean optimization. *Discrete Applied Mathematics*, 123(1):155–225, 2002.
- [13] Endre Boros, Peter L. Hammer, and Xiaorong Sun. Network flows and minimization of quadratic pseudo-Boolean functions. Technical report, RRR 17-1991, RUTCOR, 1991.
- [14] Endre Boros, Peter L. Hammer, and Gabriel Tavares. Preprocessing of unconstrained quadratic binary optimization. Technical report, RRR 10-2006, RUTCOR, 2006.
- [15] Yuri Boykov and Marie-Pierre Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in ND images. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 105–112, 2001.
- [16] Yuri Boykov and Vladimir Kolmogorov. Computing geodesics and minimal surfaces via graph cuts. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 26–33, 2003.
- [17] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 26(9):1124–1137, 2004.
- [18] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 23(11):1222–1239, 2001.
- [19] Daniel J. Butler, Jonas Wulff, Garrett B. Stanley, and Michael J. Black. A naturalistic open source movie for optical flow evaluation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 611–625, 2012.
- [20] Katarina Cechlárová. Persistency in the assignment and transportation

- problems. *Mathematical Methods of Operations Research*, 47(2):243–254, 1998.
- [21] Katarina Ceclárová and Vladimír Lacko. Persistency in combinatorial optimization problems on matroids. *Discrete Applied Mathematics*, 110(2-3):121–132, 2001.
- [22] Antonin Chambolle. Total variation minimization and a class of binary MRF models. In *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 136–152, 2005.
- [23] Antonin Chambolle, Daniel Cremers, and Thomas Pock. A convex approach to minimal partitions. *SIAM Journal on Imaging Sciences*, 5(4):1113–1158, 2012.
- [24] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 40(4):834–848, 2018.
- [25] Qifeng Chen and Vladlen Koltun. Full flow: Optical flow estimation by global optimization over regular grids. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4706–4714, 2016.
- [26] Chris A. Cocosco, Vasken Kollokian, Remi K.-S. Kwan, G. Bruce Pike, and Alan C. Evans. Brainweb: Online interface to a 3D MRI simulated brain database. *NeuroImage*, 5:425, 1997.
- [27] Aleksander Colovic, Patrick Knöbelreiter, Alexander Shekhovtsov, and Thomas Pock. End-to-end training of hybrid CNN-CRF models for semantic segmentation using structured learning. In *Computer Vision Winter Workshop*, 2017.
- [28] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. McGraw-Hill, 2001.
- [29] Marie-Christine Costa. Persistency in maximum cardinality bipartite matchings. *Operations Research Letters*, 15(3):143–149, 1994.

- [30] Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *Journal of the ACM (JACM)*, 7(3):201–215, 1960.
- [31] Johan Desmet, Marc De Maeyer, Bart Hazes, and Ignace Lasters. The dead-end elimination theorem and its use in protein side-chain positioning. *Nature*, 356(6369):539–542, 1992.
- [32] Mark Everingham, Luc Van Gool, Christopher K.I. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (VOC) challenge. *International Journal of Computer Vision (IJCV)*, 88(2):303–338, 2010.
- [33] Pedro F Felzenszwalb, Gyula Pap, Eva Tardos, and Ramin Zabih. Globally optimal pixel labeling algorithms for tree metrics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3153–3160, 2010.
- [34] Alexander Fix, Aritanan Gruber, Endre Boros, and Ramin Zabih. A hypergraph-based reduction for higher-order binary Markov Random Fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 37(7):1387–1395, 2015.
- [35] Alexander Fix, Thorsten Joachims, Sung Min Park, and Ramin Zabih. Structured learning of sum-of-submodular higher order energy functions. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 3104–3111, 2013.
- [36] Alexander Fix, Chen Wang, and Ramin Zabih. A primal-dual algorithm for higher-order multilabel Markov Random Fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1138–1145, 2014.
- [37] Daniel Freedman and Petros Drineas. Energy minimization via graph cuts: Settling what is possible. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 939–946, 2005.
- [38] William T. Freeman, Thouis R. Jones, and Egon C. Pasztor. Example-based super-resolution. *IEEE Computer Graphics and Applications*, 22(2):56–65, 2002.
- [39] William T. Freeman, Egon C. Pasztor, and Owen T. Carmichael. Learning low-level vision. *International Journal of Computer Vision (IJCV)*, 40(1):25–47, 2000.

- [40] Stuart Geman and Donald Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 6(6):721–741, 1984.
- [41] Amir Globerson and Tommi S. Jaakkola. Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS)*, pages 553–560, 2008.
- [42] Ben Glocker, T. Hauke Heibel, Nassir Navab, Pushmeet Kohli, and Carsten Rother. Triangleflow: Optical flow with triangulation-based higher-order likelihoods. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 272–285, 2010.
- [43] Andrew V. Goldberg, Sagi Hed, Haim Kaplan, Robert E. Tarjan, and Renato F. Werneck. Maximum flows by incremental breadth-first search. In *European Symposium on Algorithms*, pages 457–468, 2011.
- [44] Robert F. Goldstein. Efficient rotamer elimination applied to protein side-chains and related spin glasses. *Biophysical Journal*, 66(5):1335–1340, 1994.
- [45] Stephen Gould, Fernando Amat, and Daphne Koller. Alphabet soup: A framework for approximate energy minimization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 903–910, 2009.
- [46] Stephen Gould, Richard Fulton, and Daphne Koller. Decomposing a scene into geometric and semantically consistent regions. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1–8, 2009.
- [47] Dorothy M. Greig, Bruce T. Porteous, and Allan H. Seheult. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society. Series B (Methodological)*, 51(2):271–279, 1989.
- [48] Peter L. Hammer, Pierre Hansen, and Bruno Simeone. Vertices belonging to all or to no maximum stable sets of a graph. *SIAM Journal on Algebraic Discrete Methods*, 3(4):511–522, 1982.
- [49] Peter L. Hammer, Pierre Hansen, and Bruno Simeone. Roof duality, complementation and persistency in quadratic 0–1 optimization. *Mathematical Programming*, 28(2):121–155, 1984.

- [50] Peter L. Hammer and Sergiu Rudeanu. *Boolean methods in operations research and related areas*. Springer-Verlag, 1968.
- [51] John M. Hammersley and Peter E. Clifford. Markov Random Fields on finite graphs and lattices. *Unpublished manuscript*, 1971.
- [52] Fabrice Heitz and Patrick Bouthemy. Multimodal estimation of discontinuous optical flow using Markov Random Fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 15(12):1217–1232, 1993.
- [53] Charles Herrmann, Chen Wang, Richard S. Bowen, Emil Keyder, Michael Krainin, Ce Liu, and Ramin Zabih. Robust image stitching using multiple registrations. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.
- [54] Charles Herrmann, Chen Wang, Richard S. Bowen, Emil Keyder, and Ramin Zabih. Object-centered image stitching. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.
- [55] Hiroshi Ishikawa. Exact optimization for Markov Random Fields with convex priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 25(10):1333–1336, 2003.
- [56] Hiroshi Ishikawa. Higher-order gradient descent by fusion-move graph cut. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 568–574, 2009.
- [57] Hiroshi Ishikawa. Transformation of general binary MRF minimization to the first-order case. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 33(6):1234–1249, 2011.
- [58] Ernst Ising. Beitrag zur theorie des ferromagnetismus. *Zeitschrift für Physik*, 31(1):253–258, 1925.
- [59] Satoru Iwata. A faster scaling algorithm for minimizing submodular functions. *SIAM Journal on Computing*, 32(4):833–840, 2003.
- [60] Satoru Iwata and James B. Orlin. A simple combinatorial algorithm for submodular function minimization. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, pages 1230–1237, 2009.

- [61] Stefanie Jegelka and Jeff Bilmes. Cooperative cuts for image segmentation. Technical report, Technical Report 2010-0003, University of Washington, 2010.
- [62] Fredrik Kahl and Petter Strandmark. Generalized roof duality. *Discrete Applied Mathematics*, 160(16):2419–2434, 2012.
- [63] Jörg H. Kappes, Bjoern Andres, Fred A. Hamprecht, Christoph Schnörr, Sebastian Nowozin, Dhruv Batra, Sungwoong Kim, Bernhard X. Kausler, Thorben Kröger, Jan Lellmann, Nikos Komodakis, Bogdan Savchynskyy, and Carsten Rother. A comparative study of modern inference techniques for discrete energy minimization problems. *International Journal of Computer Vision (IJCV)*, 115(2):155–184, 2015.
- [64] Jörg H. Kappes, Bogdan Savchynskyy, and Christoph Schnörr. A bundle approach to efficient MAP-inference by Lagrangian relaxation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1688–1695, 2012.
- [65] Jörg H. Kappes, Markus Speth, Gerhard Reinelt, and Christoph Schnörr. Towards efficient and exact MAP-inference for large scale discrete computer vision problems via combinatorial optimization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1752–1758, 2013.
- [66] Scott Kirkpatrick, C. Daniel Gelatt, and Mario P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.
- [67] Jon Kleinberg and Eva Tardos. Approximation algorithms for classification problems with pairwise relationships: metric labeling and Markov Random Fields. *Journal of the ACM (JACM)*, 49(5):616–639, 2002.
- [68] Patrick Knöbelreiter, Christian Reinbacher, Alexander Shekhovtsov, and Thomas Pock. End-to-end training of hybrid CNN-CRF models for stereo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [69] Pushmeet Kohli, M. Pawan Kumar, and Philip H.S. Torr. P^3 & beyond: Move making algorithms for solving higher order functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 31(9):1645–1656, 2009.

- [70] Pushmeet Kohli, L'ubor Ladický, and Philip H.S. Torr. Robust higher order potentials for enforcing label consistency. *International Journal of Computer Vision (IJCV)*, 82(3):302–324, 2009.
- [71] Pushmeet Kohli, Alexander Shekhovtsov, Carsten Rother, Vladimir Kolmogorov, and Philip H.S. Torr. On partial optimality in multi-label MRFs. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 480–487, 2008.
- [72] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.
- [73] Vladimir Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 28(10):1568–1583, 2006.
- [74] Vladimir Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 28(10):1568–1583, 2006.
- [75] Vladimir Kolmogorov. Generalized roof duality and bisubmodular functions. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS)*, pages 1144–1152, 2010.
- [76] Vladimir Kolmogorov. Minimizing a sum of submodular functions. *Discrete Applied Mathematics*, 160(15):2246–2258, 2012.
- [77] Vladimir Kolmogorov and Carsten Rother. Minimizing nonsubmodular functions with graph cuts—a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 29(7):1274–1279, 2007.
- [78] Vladimir Kolmogorov and Ramin Zabih. Multi-camera scene reconstruction via graph cuts. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 82–96, 2002.
- [79] Vladimir Kolmogorov and Ramin Zabih. What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 26(2):147–159, 2004.
- [80] Nikos Komodakis and Nikos Paragios. Beyond pairwise energies: Efficient optimization for higher-order MRFs. In *Proceedings of the IEEE*

Conference on Computer Vision and Pattern Recognition (CVPR), pages 2985–2992, 2009.

- [81] Nikos Komodakis, Nikos Paragios, and Georgios Tziritas. MRF energy minimization and beyond via dual decomposition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 33(3):531–552, 2011.
- [82] Nikos Komodakis and Georgios Tziritas. Approximate labeling via graph cuts based on linear programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 29(8):1436–1453, 2007.
- [83] Ivan Kovtun. Partial optimal labeling search for a NP-hard subclass of (max,+) problems. *Pattern Recognition*, pages 402–409, 2003.
- [84] Ivan Kovtun. Image segmentation based on sufficient conditions of optimality in NP-complete classes of structural labelling problem. *Ukrainian. PhD thesis. IRTC ITS National Academy of Sciences, Ukraine*, 2004.
- [85] Ivan Kovtun. Sufficient condition for partial optimality for (max,+)-labeling problems and its usage. *Control Systems and Computers*, 3(2):35–42, 2011.
- [86] Oliver Kullmann. Investigations on autark assignments. *Discrete Applied Mathematics*, 107(1-3):99–137, 2000.
- [87] Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron Bobick. Graphcut textures: image and video synthesis using graph cuts. *ACM Transactions on Graphics (ToG)*, 22(3):277–286, 2003.
- [88] Vladimír Lacko. Persistency in the traveling salesman problem on Halin graphs. *Discussiones Mathematicae Graph Theory*, 20(2):231–242, 2000.
- [89] L’ubor Ladický, Chris Russell, Pushmeet Kohli, and Philip H.S. Torr. Graph cut based inference with co-occurrence statistics. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 239–253, 2010.
- [90] L’ubor Ladický, Chris Russell, Pushmeet Kohli, and Philip H.S. Torr. Inference methods for CRFs with co-occurrence statistics. *International Journal of Computer Vision (IJCV)*, 103(2):213–225, 2013.
- [91] Xiangyang Lan, Stefan Roth, Daniel Huttenlocher, and Michael J. Black. Efficient belief propagation with learned higher-order Markov Random

- Fields. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 269–282, 2006.
- [92] Steffen L. Lauritzen and David J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 157–224, 1988.
- [93] Jan Lellmann and Christoph Schnörr. Continuous multiclass labeling approaches and algorithms. *SIAM Journal on Imaging Sciences*, 4(4):1049–1096, 2011.
- [94] Victor Lempitsky, Carsten Rother, Stefan Roth, and Andrew Blake. Fusion moves for Markov Random Field optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 32(8):1392–1405, 2010.
- [95] Mengtian Li, Alexander Shekhovtsov, and Daniel Huber. Complexity of discrete energy minimization problems. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 834–852, 2016.
- [96] Ce Liu, Jenny Yuen, and Antonio Torralba. SIFT flow: Dense correspondence across scenes and its applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 33(5):978–994, 2011.
- [97] Loren L. Looger and Homme W. Hellinga. Generalized dead-end elimination algorithms make large-scale protein side-chain structure prediction tractable: implications for protein design and structural genomics. *Journal of Molecular Biology*, 307(1):429–445, 2001.
- [98] Dimitris Magos, Ioannis Mourtos, and Leonidas Pitsoulis. Persistency and matroid intersection. *Computational Management Science*, 6(4):435–445, 2009.
- [99] Christopher D. Manning and Hinrich Schütze. *Foundations of statistical natural language processing*. MIT press, 1999.
- [100] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 416–423, 2001.

- [101] Andre F.T. Martins, Noah A. Smith, Eric P. Xing, Pedro M.Q. Aguiar, and Mario A.T. Figueiredo. Augmented dual decomposition for MAP inference. In *Neural Information Processing Systems: Workshop in Optimization for Machine Learning*, 2010.
- [102] Talya Meltzer, Chen Yanover, and Yair Weiss. Globally optimal solutions for energy minimization in stereo vision using reweighted belief propagation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 428–435, 2005.
- [103] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21(6):1087–1091, 1953.
- [104] Burkhard Monien and Ewald Speckenmeyer. Solving satisfiability in less than 2^n steps. *Discrete Applied Mathematics*, 10(3):287–295, 1985.
- [105] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.
- [106] George L. Nemhauser and Leslie E. Trotter. Vertex packings: structural properties and algorithms. *Mathematical Programming*, 8(1):232–248, 1975.
- [107] James B. Orlin. A faster strongly polynomial time algorithm for submodular function minimization. *Mathematical Programming*, 118(2):237–251, 2009.
- [108] Niles A. Pierce, Jan A. Spriet, Johan Desmet, and Stephen L. Mayo. Conformational splitting: A more powerful criterion for dead-end elimination. *Journal of Computational Chemistry*, 21(11):999–1009, 2000.
- [109] Mala L. Radhakrishnan and Sara L. Su. Dead-end elimination as a heuristic for min-cut image segmentation. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, pages 2429–2432, 2006.
- [110] Srikumar Ramalingam, Pushmeet Kohli, Karteek Alahari, and Philip H.S. Torr. Exact inference in multi-label CRFs with higher order cliques. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.

- [111] Ivo G. Rosenberg. Reduction of bivalent maximization to the quadratic case. *Cahiers du Centre d'etudes de Recherche Operationnelle*, 17:71–74, 1975.
- [112] Stefan Roth and Michael J. Black. Fields of experts. *International Journal of Computer Vision (IJCV)*, 82(2):205–229, 2009.
- [113] Carsten Rother, Pushmeet Kohli, Wei Feng, and Jiaya Jia. Minimizing sparse higher order energy functions of discrete variables. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1382–1389, 2009.
- [114] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics (TOG)*, 23(3):309–314, 2004.
- [115] Carsten Rother, Vladimir Kolmogorov, Victor Lempitsky, and Martin Szummer. Optimizing binary MRFs via extended roof duality. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2007.
- [116] Carsten Rother, Sanjiv Kumar, Vladimir Kolmogorov, and Andrew Blake. Digital tapestry [automatic image synthesis]. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 589–596, 2005.
- [117] Sébastien Roy and Ingemar J. Cox. A maximum-flow formulation of the n-camera stereo correspondence problem. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 492–499, 1998.
- [118] Sébastien Roy and Venu Govindu. MRF solutions for probabilistic optical flow formulations. In *Proceedings of the IEEE International Conference on Pattern Recognition (ICPR)*, pages 1041–1047, 2000.
- [119] Bogdan Savchynskyy, Jörg H. Kappes, Paul Swoboda, and Christoph Schnörr. Global MAP-optimality by shrinking the combinatorial search area with convex relaxation. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS)*, pages 1950–1958, 2013.
- [120] Bogdan Savchynskyy, Stefan Schmidt, Jörg H. Kappes, and Christoph Schnörr. Efficient MRF energy minimization via adaptive diminishing smoothing. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 746–755, 2012.

- [121] Daniel Scharstein and Richard Szeliski. High-accuracy stereo depth maps using structured light. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 195–202, 2003.
- [122] Dmitriy Schlesinger. Exact solution of permuted submodular minsum problems. In *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 28–38, 2007.
- [123] Dmitriy Schlesinger and Boris Flach. Transforming an arbitrary MinSum problem into a binary one. *Unpublished manuscript*, 2006.
- [124] Ishant Shanu, Chetan Arora, and S.N. Maheshwari. Inference in higher order MRF-MAP problems with small and large cliques. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7883–7891, 2018.
- [125] Ishant Shanu, Chetan Arora, and Parag Singla. Min norm point algorithm for higher order MRF-MAP inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5365–5374, 2016.
- [126] Alexander Shekhovtsov. Maximum persistency in energy minimization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1162–1169, 2014.
- [127] Alexander Shekhovtsov. Higher order maximum persistency and comparison theorems. *Computer Vision and Image Understanding (CVIU)*, 143:54–79, 2016.
- [128] Alexander Shekhovtsov and Václav Hlaváč. On partial optimality by auxiliary submodular problems. *arXiv preprint arXiv:1109.1149*, 2011.
- [129] Alexander Shekhovtsov, Paul Swoboda, and Bogdan Savchynskyy. Maximum persistency via iterative relaxed inference with graphical models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 521–529, 2015.
- [130] Michail I. Shlezinger. Syntactic analysis of two-dimensional visual signals in the presence of noise. *Cybernetics*, 12(4):612–628, 1976.
- [131] Jamie Shotton, John Winn, Carsten Rother, and Antonio Criminisi. Tex-tonboost: Joint appearance, shape and context modeling for multi-class

- object recognition and segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 1–15, 2006.
- [132] David Sontag, Do Kook Choe, and Yitao Li. Efficiently searching for frustrated cycles in MAP inference. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 795–804, 2012.
- [133] David Sontag, Talya Meltzer, Amir Globerson, Tommi Jaakkola, and Yair Weiss. Tightening LP relaxations for MAP using message passing. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 503–510, 2008.
- [134] Paul Swoboda, Bogdan Savchynskyy, Jörg H. Kappes, and Christoph Schnörr. Partial optimality by pruning for MAP-inference with general graphical models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1170–1177, 2014.
- [135] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.
- [136] Rick Szeliski, Ramin Zabih, Daniel Scharstein, Olga Veksler, Vladimir Kolmogorov, Aseem Agarwala, Marshall Tappen, and Carsten Rother. A comparative study of energy minimization methods for Markov Random Fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 30(6):1068–1080, 2008.
- [137] Meng Tang, Federico Perazzi, Abdelaziz Djelouah, Ismail Ben Ayed, Christopher Schroers, and Yuri Boykov. On regularized losses for weakly-supervised CNN segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.
- [138] Daniel Tarlow, Inmar Givoni, and Richard Zemel. HOP-MAP: Efficient message passing with high order potentials. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 812–819, 2010.
- [139] Sara Vicente, Vladimir Kolmogorov, and Carsten Rother. Graph cut based image segmentation with connectivity priors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.
- [140] George Vogiatzis, Carlos H. Esteban, Philip H.S. Torr, and Roberto Cipolla. Multiview stereo via volumetric graph-cuts and occlusion ro-

- bust photo-consistency. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 29(12):2241–2246, 2007.
- [141] Christopher A. Voigt, D. Benjamin Gordon, and Stephen L. Mayo. Trading accuracy for speed: a quantitative comparison of search algorithms in protein sequence design. *Journal of Molecular Biology*, 299(3):789–803, 2000.
- [142] Martin Wainwright, Tommi Jaakkola, and Alan Willsky. MAP estimation via agreement on (hyper)trees: Message-passing and linear-programming approaches. *IEEE Transactions on Information Theory*, 51(11):3697–3717, 2005.
- [143] Chaohui Wang, Nikos Komodakis, and Nikos Paragios. Markov Random Field modeling, inference & learning in computer vision & image understanding: A survey. *Computer Vision and Image Understanding (CVIU)*, 117(11):1610–1627, 2013.
- [144] Chen Wang, Charles Herrmann, and Ramin Zabih. A discriminative view of MRF pre-processing algorithms. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 5504–5513, 2017.
- [145] Chen Wang and Ramin Zabih. Relaxation-based preprocessing techniques for Markov Random Field inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5830–5838, 2016.
- [146] Yair Weiss and William T. Freeman. On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs. *IEEE Transactions on Information Theory*, 47(2):736–744, 2001.
- [147] David P. Williamson and David B. Shmoys. *The design of approximation algorithms*. Cambridge university press, 2011.
- [148] Thomas Windheuser, Hiroshi Ishikawa, and Daniel Cremers. Generalized roof duality for multi-label optimization: optimal lower bounds and persistency. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 400–413, 2012.
- [149] Oliver Woodford, Philip H.S. Torr, Ian Reid, and Andrew Fitzgibbon. Global stereo reconstruction under second order smoothness priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 31(12):2115–2128, 2009.

- [150] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip H.S. Torr. Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1529–1537, 2015.