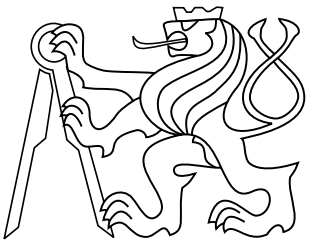
Ph.D. Thesis

Exact and Partial Energy Minimization in Computer Vision

Alexander Shekhovtsov



February 2013

Supervisor: Václav Hlaváč
Supervisor-specialist: Tomáš Werner

Czech Technical University in Prague
Faculty of Electrical Engineering, Department of Cybernetics

Ph.D. programme: Electrical Engineering and Information Technology
Branch of study: Mathematical Engineering, 3901V021

Acknowledgement

I am grateful to my teacher Michail Schlesinger, who once introduced me to the area of pattern recognition and gave me a lot of wise advices since; my supervisor Václav Hlaváč, who has given me the possibility to work in the excellent group, the Center for Machine Perception in Prague; my colleagues Tomáš Werner and Boris Flach, whom I had pleasure to work and discuss with; the EU-funded projects COSPAL, DIPLECS and NIFTi, which were facilitating this research; and all my colleagues for sharing their knowledge and the overall support.

Declaration

I declare that I worked out the presented thesis independently and I quoted all used sources of information in accord with Methodical instructions about ethical principles for writing academic thesis.

Abstract

Velkým úspěchem počítačového vidění v posledních letech je objev efektivních deterministických algoritmů na minimalizaci částečně separabilních funkcí diskrétních proměnných (což je známo také pod názvy 'minimalizace energie', max-plus značkování či problém splňování vážených podmínek). Optimalizční problém v této obecné formě se ukázal užitečným téměř ve všech oblastech. V počítačovém vidění na tento problém vede úloha max-aposteriorní inference v markovských náhodných polích a podmíněných náhodných polích, kterými se modeluje mnoho úloh od husté stereo-korespondence a segmentace obrazů až po detekci a rozpoznávání objektů v obrazech.

Příspěvek této práce lze rozdělit do dvou částí. První příspěvek je sjednocení metod pro výpočet části optimálního řešení. Je-li dána instance úlohy, tyto metody najdou hodnoty podmnožiny proměnných, které jsou částí všech (nebo alespoň některých) optimálních řešení. Ukazujeme, že několik široce užívaných avšak dříve nesouvisejících metod lze získat z jediných počátečních podmínek. Díky tomu lze vidět, že tyto metody mají některé vlastnosti společné, např. zachovávají řešení standardní LP-relaxace. Tyto nové sjednocující postačující podmínky v práci charakterizujeme a navrhujeme systematické metody na hledání části optimálního řešení vzhledem k těmto podmínkám. Konkrétně předkládáme novou netriviální podtřídu úloh, pro které nalezení části optimálního řešení vede na sekvenci úloh na minimální *s-t* řez v grafu.

Druhý příspěvek se týká úlohy minimálního *s-t* řezu v grafu. Mnoho metod na minimalizaci energie je založeno na redukci původního problému (příp. jeho restrikce či relaxované formy) na úlohu minimálního *s-t* řezu. Protože velikost výsledných instancí této úlohy je často obrovská (např. ve 3-D segmentaci), jsou třeba efektivní algoritmy na řešení velkých řídkých úloh minimálního *s-t* řezu. V práci navrhujeme nový distribuovaný algoritmus na tuto úlohu. Sekvenční verze tohoto algoritmu umožňuje řešit velké instance úlohy na jediném počítači s omezenou operační pamětí a externí (diskovou) pamětí. Paralelní verze algoritmu umožňuje urychlit výpočet s pomocí více procesorů či řešit problém paralelně na několika počítačích posílajících si zprávy po síti. Dokazujeme, že horní meze na počet diskových operací resp. poslaných zpráv pro náš algoritmus jsou lepší než pro známé algoritmy. V experimentech s naší efektivní implementaci algoritmu ukazujeme, že dosahuje srovnatelných výsledků jako známé algoritmy, ovšem při významně nižším počtu drahých diskových resp. síťových operací.

Klíčová slova

kombinatorická optimalizace; minimalizace energie; max-aposteriorní inference v Markovských náhodných polích; část optimálního řešení; persistence; distribuovaná úloha minimálního řezu / maximálního toku v grafu

Abstract

One of the major advances in computer vision in the past few years is the development of efficient deterministic algorithms for minimization of a partially separable function of discrete variables, commonly known as energy minimization, max-sum labeling problem, or weighted constraint satisfaction. The optimization model of this general form has proved useful in nearly all areas. In computer vision, it arises in particular as the maximum a *posteriori* inference in Markov random fields and conditional random fields, which are used to model a variety of vision problems ranging from the dense stereo and image segmentation to the use of pictorial structures for object recognition.

This work brings two contributions. The first contribution is a unified study of methods for partial optimality. Such methods, given an instance of the problem, output an assignment for a subset of variables that is guaranteed to be a part of any (or at least some) global optimum. We show that several widely applied but previously unrelated partial optimality methods can be obtained from the same unifying sufficient conditions. It implies that they share several common properties, in particular the property to preserve solutions of the standard LP relaxation. We prove a characterization of the new unifying sufficient condition and propose a systematic way to derive partial optimality guarantees. For several subclasses of problems we derive methods to find the maximum partial optimality w.r.t. the unifying sufficient condition. This includes one new non-trivial subclass, for which the maximum partial optimality is found via solving a series of minimum cut problems.

The second contribution is on the minimum s-t cut problem. It turns out that many energy minimization methods rely on the reduction of the full problem or its restriction or its relaxed form to the minimum s-t cut. Because some problems are of a very large scale (*e.g.*, in 3D segmentation), this poses a challenge of solving the minimum s-t cut problem for large sparse graphs. We develop a novel distributed algorithm for this problem. The sequential version of the algorithm allows solving large instances on a single computer using a limited amount of memory and an external (disk) storage. The parallel version of the algorithm allows to speed-up computations using several processors or to solve the problem in parallel on several computers exchanging messages over the network. We prove the worst case bounds on the number of disk operations and message exchanges between computers, respectively. These bounds are superior to the ones for previously proposed methods. With a dedicated efficient implementation, we are able to demonstrate experimentally that the new algorithms achieve the state-of-the-art performance in terms of used computation time, while significantly lowering the usage of the costly disk access/message exchange operations.

Keywords

combinatorial optimization; energy minimization; maximum a posteriori inference in Markov random fields; partial optimality; persistency; distributed min-cut/maxflow

Contents

A short notation can be very handy,
unless you forget what it denotes.

V. Kushnirevich
my teacher of tensor analysis

Notation

$\mathbb{R}$ ($\mathbb{R}_+$)	the set of (non-negative) real numbers
$\mathbb{N}_0$	natural numbers with zero
$\mathbb{B} = \{0, 1\}$	set of Booleans
$[0,\, 1]$	interval of reals
$0, 1$	vector of zeros (resp. ones) of appropriate size
$\{\dots \mid \dots\}$	set
$(\dots \mid \dots)$	ordered list of elements (tuple)
$\stackrel{\text{def}}{=}$	equality by definition
$[\![\texttt{expression}]\!]$	equals 1 if `expression` is true and 0 if it is false
argmin	the set of all minimizers
conv	convex hull
aff	affine hull
null	null space (kernel) of a linear map
$\prod_{i \in I} A_i$	Cartesian product of sets A_i such that $i \in I$
$\langle \cdot, \cdot \rangle$	scalar product
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	graph of a pairwise energy function
$\mathcal{A}$, $\mathcal{U}$, $\mathcal{W}$	subsets of vertices $\mathcal{V}$
$G = (V, E, s, t, c)$	graph (network) of a MINCUT/MAXFLOW problem
$(A, B)_E$	$(A \times B) \cap E$
f, g, h	parameter vector of the energy function
E_f, E_g, E_h	energy function
x, y, z	labeling
μ, ν	relaxed labeling
$x_{\mathcal{A}}$, $e\vert_R$	restrictions of labeling (function) to a subdomain
$\mathcal{L}$	set of labelings, page 7
$\mathcal{M}$	marginal polytope, page 9
Λ	local polytope, page 10

Abbreviations

subj.	subject to
iff	if and only if
LHS, RHS	left-(right-)hand side of an equation
QPBO	quadratic pseudo-Boolean optimization method by Boros et al. (1991)
MQPBO	multi-label extension of QPBO by (Kohli et al. 2008)
DEE	dead end elimination methods
LP	linear program

1 Introduction

1.1 Problem Formulation

In this work, we study several techniques for minimization of partially separable functions of discrete variables. This problem is commonly known as *energy minimization*, max-sum labeling problem, or weighted constraint satisfaction problem. Mathematically, it is defined as follows. Given a graph $(\mathcal{V}, \mathcal{E})$ and functions $f_s \colon \mathcal{L}_s \to \mathbb{R}$ for all $s \in \mathcal{V}$ and $f_{st} \colon \mathcal{L}_s \times \mathcal{L}_t \to \mathbb{R}$ for all $st \in \mathcal{E}$, where $\mathcal{L}_s$ are finite sets of *labels*, minimize the *energy*

$$E_f(x) = \sum_{s \in \mathcal{V}} f_s(x_s) + \sum_{st \in \mathcal{E}} f_{st}(x_s, x_t)\,, \tag{1}$$

over all assignments $x = (x_s \in \mathcal{L}_s \,|\, s \in \mathcal{V})$. The general energy minimization problem is NP-hard and even APX-hard, meaning that there is no polynomial-time approximation scheme.

Goal 1 We study the linear programing (LP) relaxation approach, in which the problem is formulated as an integer linear programing (ILP) and the integrality constraints are relaxed. For some classes of problems, this relaxation is tight, and the problem can be solved to optimality. In addition, it is known that for some 0-1 ILP problems, their relaxations are persistent: whenever a part of a relaxed solution takes binary values, there exists an integer optimal solution taking the same values. This allows to determine optimally an assignment for the part of variables which are integer in the relaxation. For energy minimization, one can obtain an ILP reformulation possessing the persistency property. However, such an approach depends on the particular reduction applied. There were several methods proposed to find an optimal partial assignment directly for the energy in the form (1). The first goal of this work is to analyze and unify these methods.

Goal 2 Many algorithms for energy minimization exploit solvable subproblems in the form of a minimum s-t cut problem (further on called MINCUT), which is formulated as follows. Given a directed graph (V, E), an edge capacity function $c \colon E \to \mathbb{R}_+$, and a source and sink vertices $s, t \in V$, find a partition $V = (S, V \backslash S)$ (a *cut*) that separates the source and the sink and minimizes the total cost of the cut (of the edges with tail in S and head in $V \backslash S$). Mathematically, this problem is written as

$$\min_{\substack{S \subset V \\ s \in S \\ t \notin S}} \sum_{\substack{(u,v) \in E \\ u \in S \\ v \notin S}} c(u, v)\,. \tag{2}$$

There are many algorithms known which solve this problem in polynomial time. Nevertheless, there is still an active ongoing research on algorithms that would be more efficient for computer vision problems. This includes specialized implementations, development of parallel and massively parallel (GPU) implementations. In some settings,

distributed algorithms are an advantage. Such algorithms divide not only the computation but also the memory between the computational units. One application of such algorithms is to solve the problem by loading and processing only a portion of data at a time. Our second goal was to develop a distributed algorithm for the minimum s-t cut problem that would minimize the number of necessary loads of the problem data. In other words, that would be efficient for solving large problems on a single computer.

1.2 Motivation

Structural Labeling Problems Why to minimize functions of the particular form (1)? Many problems can be formulated as finding the best assignment to a collection of discrete variables. These variables describe hidden states or decisions to be made, depending on the context. The variables are not independent but describe a joint complex structure (consider for example a description needed in text recognition or in hierarchical image segmentation). In this context one speaks of structural pattern recognition (or structural labeling problems) to emphasize the distinction from statistical pattern recognition, where the object of recognition is atomic, *e.g.*, has only several states. The state (decision) in the structural labeling problem is the full collection of many discrete variables. The number of different states is combinatorial. The objective function of interest in structural labeling problems is a function of this joint complex state. It carries the structural information in the sense that it defines which variables are involved and in which way they are coupled.

The pairwise separable objective function, where only dependencies between pairs of variables can be expressed, is the simplest yet already a powerful model. In this work, we restrict to energies of this type, also referred to as energies of the second order. There have been many works recently arguing for more complex models, which include terms of a higher order, terms that are functions of separable functions, *etc.* Nevertheless, the second order model is of principal importance since 1) it is NP-hard, 2) many higher-order models can be re-expressed as a second order model using auxiliary variables and 3) respective algorithms for higher-order models are largely based on a generalization of algorithms for the second order case.

Markov Random Fields and Graphical Models Taking the statistical approach, one has to choose a family of probability distributions p to define a probabilistic model of a discrete variables vector x. Which families of such distributions are tractable and useful in practice? While constructing the model (or a family of models), one can incorporate conditional independences suggested by the intuition or otherwise split the model semantically, *e.g.*, into a prior and a conditional parts. Such splitting is often suggested by the direction of the physical phenomena that are responsible for the image formation process (the forward problem). In low-level vision problems, it is reasonable to restrict the model in such a way that if we fix all variables in a certain neighborhood of a variable x_s then it becomes statistically independent of the description of the rest of the problem (local Markov property). Strictly positive distributions[1] satisfying these conditional independence (Markov properties) are called Markov Random Fields (MRFs).

[1]The positivity is necessary for several reasons, in particular to allow the existence of all conditional probabilities. Moreover, the equivalence between Gibbs random fields and Markov random fields breaks down without this assumption (turns into a one-way implication), and the different types of conditional independence must be distinguished, Lauritzen (1998).

Another, more constructive approach, considers distributions that factorize into a product of factors each depending on a smaller subset of variables. Such distributions are called Gibbs Random Fields (GRF). All possible conditional independences of a MRF can be encoded by a single graph G, such that x_A is independent of x_B given x_S iff S separates A and B in G. The structure of a GRF in general is captured by a hypergraph, where a hyperedge is present in correspondence to every factor. These, and some other models which can be defined by means of a directed or undirected (hyper)graph, form the family of *graphical models* (Wainwright and Jordan 2008).

There is the following link between conditional independencies and factorizations. A famous result, the theorem by Hammersley and Clifford (1971), states that p is an MRF w.r.t. graph G if and only if it admits factorization over maximal cliques of graph G. In this way, imposing sufficiently many conditional independencies guarantees that the distribution p factorizes. However, a finer factorization may be possible for p, which is not reflected in its conditional independence properties. For example, a distribution that factorizes over a complete graph will not possess any conditional independencies in general. For this reason, choosing the structure of factorization is a preferable way in practice from the point of view of representation of the distribution and meeting the requirements of the recognition algorithm. In such an approach, conditional independencies are no longer a free choice, but are implied by the factorization[2].

In Conditional Random Fields (CRF) (Lafferty et al. 2001) one models directly the posterior distribution $p(x \mid y)$, where x is the hidden state and y is a vector of observations and features computed from them. The distribution $p(x \mid y)$ is then assumed to possess some conditional independencies and/or factorize in x given y. This form of model is less intuitive because specifying the dependence of hidden states on the observations is an inverse problem. It is not described by natural physical processes. Nevertheless, CRF has proven to be an efficient approach. Indeed, finding the maximum a posteriori (MAP) assignment of x only requires the posterior $p(x \mid y)$. There have been powerful features designed for image analysis. There are machine learning techniques developed for models of this type, too. More on modeling with Markov Random Fields can be found in the book by Li (2009), while Lauritzen (1998) gives a rigorous study of the theoretical properties.

MAP inference in MRF Finding the MAP assignment amounts to maximizing $p(x|y)$ in x. While addressing this problem, the observations y are fixed and may be dropped from the notation. Without loss of generality, we may write that $p(x \mid y) \propto \exp(-E(x))$, where E is the energy function. Such exponential models arise naturally in statistical mechanics, *e.g.*, Boltzmann distribution, where E is proportional to the kinetic energy or a more general one. Clearly, if p factorizes as a product of pairwise factors in x then E is a pairwise-separable function in x, *i.e.*, of the type we consider. In the MAP-MRF/CRF approach, the model is usually initially restricted to have the desired factorization and the most common choice is to require a pairwise factorization. Arguably, finding the (non-unique) MAP assignment may not be the best formulation of the recognition/classification problem. Indeed, posing the problem as the Bayesian decision task with an appropriate loss function generally leads to a computational problem of a more complex form. It yields the MAP problem only in the case of the so-called 0-1 loss function, which is unnatural for many (computer vision) applications. Never-

[2] I have often seen unjustified application of the Hammersley-Clifford theorem in the literature. The authors tried (often incorrectly) to persuade the reader that the factorization of their choice follows from some unspecified conditional independencies, emphasizing the "Markovity" of their model in this way while never actually working with conditional independencies.

theless, MAP assignment was shown to deliver high quality solutions in many applied problems.

Applications In computer vision, energy minimization techniques have been successfully employed to obtain high-quality results. Among them are

- image, video and volumetric data segmentation (Greig et al. 1989; Boykov and Jolly 2000; Boykov et al. 2001; Boykov 2003; Ishikawa 2003; Rother et al. 2004; Kohli and Torr 2005; Boykov and Funka-Lea 2006; Delong and Boykov 2009; Lempitsky et al. 2011),
- image restoration (Boykov et al. 2001),
- optical flow (Roy and Govindu 2000; Boykov et al. 2001; Lempitsky et al. 2008),
- stereo reconstruction (Boykov et al. 1998; Kolmogorov and Zabih 2001),
- 3D reconstruction (Boykov and Lempitsky 2006; Lempitsky et al. 2006; Lempitsky and Boykov 2007),
- stitching, cutting and combining images (Kwatra et al. 2003; Kolmogorov 2005; Rother et al. 2005; Avidan and Shamir 2007),
- visual correspondence (Felzenszwalb and Huttenlocher 2000; Kim et al. 2003),
- multi-model estimation (Delong et al. 2012).

In a wider context, we can mention finding stable states of molecular systems, protein structure prediction (bioinformatics), analysis of spin glasses (statistical mechanics), radio frequency assignment, scheduling, optimal decoding of noisy signals, *etc.*

Algorithms A number of powerful algorithms are present in the literature to deal with the problem. For certain subclasses of the problem, it is possible to compute the exact solution in a polynomial time: MRFs of bounded tree-width, *e.g.* (Lauritzen 1998); with convex pairwise potentials (Ishikawa 2003); with submodular potentials of binary (Hammer 1965; Kolmogorov and Zabih 2004) or with multi-label (Schlesinger and Flach 2000; Kovtun 2004) variables; with permuted submodular potentials (Schlesinger 2007).

A standard approach in optimization for such hard problems is solving a convex relaxation. A recent study by Kumar et al. (2009) showed that among several convex relaxations considered in the literature (including second order cone and semidefinite), the linear relaxation is the tightest one.

Works devoted to solving large scale linear relaxations are discussed in more detail in chapter 2. Other family of methods iteratively improve the current solution by a restricted local search: expansion and swap move (Boykov et al. 2001), fusion-move (Lempitsky et al. 2010), multi-label moves (Kumar et al. 2011; Veksler 2012), tired moves (Vineet et al. 2012). Some of these methods provide approximation ratio guarantee for (semi-)metric energies (Boykov et al. 2001; Komodakis and Tziritas 2005). Finally, there are methods which provide optimality guarantees for a partial variable assignment or variable restriction (partial optimality) for binary (Hammer et al. 1984; Boros et al. 1991, 2006; Rother et al. 2007) and multi-label (Kovtun 2003, 2004; Kohli et al. 2008) problems. Dead-end elimination is another related method for identifying non-optimal assignments based on local sufficient conditions. It was proposed originally by Desmet et al. (1992) for predicting and designing protein structure.

A big variety of algorithms is being developed for more complicated energy functions, that include *e.g.*, higher-order terms and global terms of various forms, attempting to push the applicability of energy minimization even further.

MINCUT in Computer Vision MINCUT problems in computer vision can originate from the energy minimization framework in several ways. Submodular energy minimization problems completely reduce to MINCUT (Ishikawa 2003; Schlesinger and Flach 2006). When the energy minimization is intractable, MINCUT is employed in relaxation and local search methods. The linear relaxation of pairwise energy minimization with 0-1 variables reduces to MINCUT (Boros et al. 1991; Kolmogorov and Rother 2007) as well as the relaxation of problems reformulated in 0-1 variables (Kohli et al. 2008). Expansion-move, swap-move (Boykov et al. 1999) and fusion-move (Lempitsky et al. 2010) algorithms formulate a local improvement step as a MINCUT problem.

Despite of the existence of a number of algorithms for MINCUT with polynomial running time bounds and good practical performance, there is an active ongoing research on such algorithm and their specific implementations. Among sequential algorithms, the augmenting path implementation developed by Boykov and Kolmogorov (2004), denoted BK, can be considered as a baseline for simple vision problems. Several authors proposed sequential implementations which achieve better practical performance. Goldberg et al. (2011) proposed an algorithm, similar to BK, but having a strongly polynomial running time bound and better practical performance on both simple and hard classes of problems. Jamriška et al. (2012) give a cache-efficient implementation of BK, specialized to grid-structured graphs and achieving a significant speed-up. Arora et al. (2010) proposed a variant of preflow-push algorithm achieving better performance on simple vision problems. Verma and Batra (2012) proposed an extended experimental comparison of sequential solvers on a wider set of benchmark problems. Several authors proposed novel parallel implementations (Delong and Boykov 2008; Liu and Sun 2010; Jamriška et al. 2012) and massive parallel implementations on GPU (Vineet and Narayanan 2008; Vineet and Narayanan 2010).

1.3 Contribution and Outline

In Chapter 2, we present the pairwise energy minimization and review the theoretical results that are the most relevant to the subsequent development. Linear programming relaxation and duality relations form the central approach, which is used for analysis of the problem and unification of partial optimality methods in Chapter 4. This chapter also gives more details on the problems and subproblems that can be reduced to MINCUT/MAXFLOW problem.

In Chapters 3 and 4, we develop an unified framework to analyze partial optimality methods. We show that several widely applied but previously unrelated partial optimality methods can be obtained from the same unifying sufficient conditions. These are the roof dual method (Boros et al. 2006), its multi-label extension (Kohli et al. 2008), the method of auxiliary submodular problems (Kovtun 2004) and the family of local methods known as Dead End Elimination (DEE) (Desmet et al. 1992), originally developed in the context of protein structure design. We show that the different sufficient conditions proposed by these methods can be unified into a more general class. We study the common properties of this class, and show the following. All the above mentioned methods can be derived as local sufficient conditions in a specially constructed reparametrization of the problem. All these methods are connected to the standard LP relaxation. In particular, optimal fixation of the part of variables they provide are automatically satisfied by all solutions of LP relaxation. We also show that all fixed points of the expansion move algorithm (with the move step subproblems solved by roof-dual) are preserved by a subclass of the general method. The new framework sug-

gests a way how to derive new partial optimality methods. We study the question of deriving the maximum partial optimality. We prove the solutions providing maximal partial optimality for several subclasses of problems. This includes one new non-trivial subclass: we propose an algorithm which given two test labelings y and z for a multi-label problem, decides whether labels from z can be globally eliminated by replacing them with labels y. The subset of labels eliminated in this way is the maximum one that can be obtained from the unified sufficient conditions for this subclass.

The present work generalizes and extends the results of Shekhovtsov and Hlaváč (2011), where unification of a smaller subset of methods was proposed. Optimality guarantees for the relaxed labellings (restricting the set of optimal relaxed labelings without solving the LP) were first proposed for multilabel QPBO method in the works Shekhovtsov et al. (2008) and Kohli et al. (2008).

In Chapter 5, we develop a novel distributed algorithm for MINCUT. Noting that MINCUT is employed as a subroutine in many places in energy minimization (either allowing to solve the full problem or its relaxation or to find an improvement), a general algorithm suitable for solving large-scale sparse problems is required. The sequential version of the proposed algorithm allows to solve large instances of the mincut/maxflow problem on a single computer using a disk storage. The parallel version of the algorithm allows to speed-up computations using several processors or to solve the problem in parallel on several computers exchanging messages over the network. We prove superior theoretical properties of both proposed algorithms, develop efficient implementations, and show that they achieve a competitive performance on large-scale computer vision instances while greatly improving on the disk operations in the sequential case and message exchange operations in the parallel case. These results were published in the article (Shekhovtsov and Hlaváč 2012).

2 Background on Energy Minimization

This chapter presents the pairwise energy minimization problem and results which are relevant to our contribution. Linear programming relaxation and duality relations are the core of the approach that is used for analysis of the problem and unification of partial optimality methods in Chapters 3 and 4. Duality relations always link quite different descriptions of the problem, thus we attempt to present them whenever possible. We pay special attention to the case of binary variables, where results from pseudo-Boolean optimization apply and review the results which can be obtained via a reduction to binary variables. In §2.7, the expansion-move algorithm is reviewed, an analysis of the truncation technique is presented and its relation to linear majorization is examined. In Chapter 4, guarantees will be proven for fixed points of the expansion move and fusion move algorithms implied by partial optimality methods.

2.1 Energy Minimization Problem

Many image analysis and interpretation problems can be posed as labeling problems, in which the solution to a problem is a set of labels assigned to image pixels or features. Labeling is also a natural representation for the study of MRF's
(Besag 1974)

Let $\mathcal{V}$ be a finite set of *pixels* and $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ is the set of pairwise *interactions*. Notation st denotes the ordered pair (s,t) for $s,t \in \mathcal{V}$. Variable x_s takes its values in a discrete domain $\mathcal{L}_s = \{0,1,\dots,L-1\}$, called *labels* at pixel s. The concatenated vector of all variables $x = (x_s \,|\, s \in \mathcal{V})$ is called a *labeling*. Labeling x takes values in $\mathcal{L} = \prod_s \mathcal{L}_s$, the Cartesian product of all domains $\mathcal{L}_s$[1]. By $x_{\mathcal{A}}$ we denote the restriction of x to $\mathcal{A} \subset \mathcal{V}$ and by x_{st} the pair (x_s, x_t). We consider the *energy* function of the form

$$E_f(x) = f_0 + \sum_{s \in \mathcal{V}} f_s(x_s) + \sum_{st \in \mathcal{E}} f_{st}(x_{st}) \tag{3}$$

and the associated *energy minimization* problem

$$\min_{x \in \mathcal{L}} E_f(x)\,. \tag{4}$$

We require that $st \in \mathcal{E} \Rightarrow ts \notin \mathcal{E}$ to eliminate the redundancy. Let us denote the set $\mathcal{L}_s \times \mathcal{L}_t$ as $\mathcal{L}_{st}$ and the pair of labels $(i,j) \in \mathcal{L}_{st}$ as ij. The following set of indices is associated with the graph $(\mathcal{V}, \mathcal{E})$ and the set of labelings $\mathcal{L}$:

$$\mathcal{I} = \{0\} \cup \{(s,i) \,|\, s \in \mathcal{V},\ i \in \mathcal{L}_s\} \cup \{(st,ij) \,|\, st \in \mathcal{E},\ ij \in \mathcal{L}_{st}\}\,. \tag{5}$$

[1] Thus defined, domains $\mathcal{L}_s$ coincide for all pixels and need not be distinguished. Nevertheless, keeping the subscript helps to distinguish the association of bound variables like $i \in \mathcal{L}_s$.

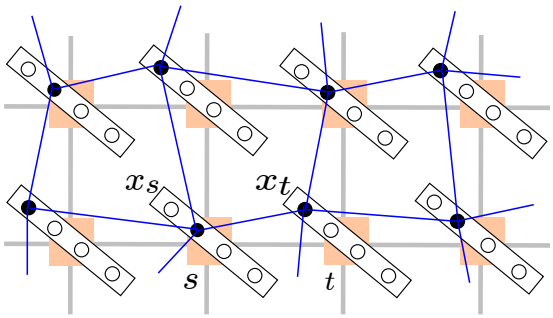


Figure 1 Energy minimization: each discrete domain $\mathcal{L}_s$ at a pixel s is depicted by a box with possible labels. A labeling x is shown by black circles and blue solid lines.

A vector $f \in \mathbb{R}^{\mathcal{I}}$ has the components (coordinates)

$$\begin{array}{lll} f_0\,, & & \textit{(constant term)} \\ f_s(i) & \forall s \in \mathcal{V},\ i \in \mathcal{L}_s\,, & \textit{(unary terms)} \\ f_{st}(i,j) & \forall st \in \mathcal{E},\ ij \in \mathcal{L}_{st}\,. & \textit{(pairwise terms)} \end{array} \tag{6}$$

We further define that $f_{ts}(j,i) = f_{st}(i,j)$, in order to refer to the *same* component irrespectively of the direction of the pair st. The *energy function* E_f is associated with vector f as defined by (3). Let us also denote $\tilde{\mathcal{E}} = \mathcal{E} \cup \{ts \,|\, st \in \mathcal{E}\}$, the symmetric closure of $\mathcal{E}$. The *neighbors* of a pixel s are the vertices in the set $\mathcal{N}(s) = \{t \,|\, st \in \tilde{\mathcal{E}}\}$. Summing the contribution from all neighbors of a given pixel s can be easily written as $\sum_{t \in \mathcal{N}(s)} f_{st}(i,j)$, which expands as $\sum_{t \,|\, st \in \mathcal{E}} f_{st}(i,j) + \sum_{t \,|\, ts \in \mathcal{E}} f_{ts}(j,i)$ due to the convention $f_{st}(i,j) = f_{ts}(j,i)$. The set of labelings restricted to $\mathcal{N}(s)$ will be denoted by $\mathcal{L}_{\mathcal{N}(s)} = \prod_{t \in \mathcal{N}(s)} \mathcal{L}_t$.

The structure of the discrete labeling problem is illustrated in Figure 1. Discrete decision variables x_s are depicted with square boxes and their possible labels with the circles inside. Lines between the discrete variables show the interacting pairs in the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$.

2.1.1 Equivalent Problems

The representation of the energy function $E_f(x)$ via the component vector f is not unique. There is a linear subspace of vectors defining the same energy function.

Definition 1. Two vectors $f, g \in \mathbb{R}^{\mathcal{I}}$ are called *equivalent*, denoted as $f \equiv g$, if their energy functions are equal,

$$(\forall x \in \mathcal{L}) \quad E_f(x) = E_g(x)\,. \tag{7}$$

Clearly, the minimization problems $\min_x E_f(x)$ and $\min_x E_g(x)$ are equivalent for $f \equiv g$. The notion of equivalent transformations appears in (Shlezinger 1976; Wainwright et al. 2003; Kolmogorov and Wainwright 2005) and is very helpful in the design and interpretation of algorithms. Because E_f is linear in f, it is $f \equiv g$ iff $h = f - g$ satisfies $(\forall x \in \mathcal{L})\ E_h(x) = 0$, *i.e.*, E_h is a zero function. All zero energies can be parametrized as follows. Let there be given vectors φ, ψ with components $(\varphi_{st}(i) \in \mathbb{R} \,|\, st \in \tilde{\mathcal{E}}, i \in \mathcal{L}_s)$, $(\psi_s \in \mathbb{R} \,|\, s \in \mathcal{V})$, respectively.

Statement 1. The energy E_h defined by

$$\begin{aligned} h_0 &= \sum_{s\in\mathcal{V}} \psi_s \,, \\ h_s(i) &= \sum_{t\in\mathcal{N}(s)} \varphi_{st}(i) - \psi_s \,, \\ h_{st}(i,j) &= -\varphi_{st}(i) - \varphi_{ts}(j) \end{aligned} \tag{8}$$

is the zero function.

Proof. To prove that $E_h(x) = 0$, we simply substitute its definition,

$$E_h(x) = \sum_{s\in\mathcal{V}} \psi_s + \sum_{s\in\mathcal{V}} \sum_{t\in\mathcal{N}(s)} \Big(\varphi_{st}(x_s) - \psi_s\Big) + \sum_{st\in\mathcal{E}} \Big(-\varphi_{st}(x_s) - \varphi_{ts}(x_t)\Big) \,. \tag{9}$$

We can see that all the terms cancel out. □

The reverse statement holds as well.

Theorem 2 (Schlesinger and Flach 2002, Theorem 1)**.** The energy E_h is a zero function iff it admits representation (8). When the graph $(\mathcal{V},\mathcal{E})$ is connected, any zero energy function E_h with $h_0 = 0$ can be represented in the form (8) with zero ψ.

Proof. See (Werner 2005, Theorems 12, 13) or (Kolmogorov 2004a, Lemma 6.2). □

Constructing $g \equiv f$ is called an *equivalent transformation* of f. It is clear now that the space of equivalent transformations is fully parametrized by vector (φ,ψ). Let us denote by f^φ the equivalent energy vector constructed by adding a zero energy vector parametrized in the form (8) with zero ψ. It has the components

$$\begin{aligned} f^\varphi_0 &= f_0 \,, \\ f^\varphi_s(i) &= f_s(i) + \sum_{t\in\mathcal{N}(s)} \varphi_{st}(i) \,, \\ f^\varphi_{st}(i,j) &= f_{st}(i,j) - \varphi_{st}(i) - \varphi_{ts}(j) \,. \end{aligned} \tag{10}$$

We will not need a notation for the degrees of freedom corresponding to ψ, as these transformations are simple and can be handled explicitly.

2.2 LP relaxation

In this section, we represent energy minimization as an integer linear program and obtain a linear relaxation by dropping the integrality constraints. This construction and its LP dual was considered in the context of computer vision by Shlezinger (1976, p.128), Chekuri et al. (2001); Wainwright et al. (2003) and by Koster et al. (1998). We introduce the following representation in order to make explicit that E_f is linear in f. Let $\delta(x)$ be a vector with components $\delta(x)_0 = 1$, $\delta(x)_s(i) = [\![x_s{=}i]\!]$ and $\delta(x)_{st}(i,j) = [\![x_{st}{=}ij]\!]$. Let $\langle\cdot,\cdot\rangle$ denote the scalar product on $\mathbb{R}^{\mathcal{I}}$. Then we can write the energy as

$$E_f(x) = \langle f, \delta(x)\rangle \tag{11}$$

and the energy minimization as

$$\min_{x\in\mathcal{L}} \langle f,\delta(x)\rangle = \min_{\mu\in\{\delta(x)\,|\,x\in\mathcal{L}\}} \langle f,\mu\rangle \,, \tag{12a}$$

$$= \min_{\mu\in\mathcal{M}} \langle f,\mu\rangle \,. \tag{12b}$$

In the last equality we introduced the convex hull of the integer labelings $\mathcal{M} = \text{conv}(\delta(\mathcal{L}))$, called the *marginal* polytope. Since the objective function is linear and the set $\delta(\mathcal{L})$ is not empty, the minimization over the convex hull attains the same minimal value. Generally, both the number of vertices and the number of facets of polytope $\mathcal{M}$ grows exponentially with the problem size. Hence, (12b) is not a tractable LP. It can be simplified by dropping some inequality constraints as follows. Consider the minimization problem

$$\min_{\mu \in \Lambda} \langle f, \mu \rangle \tag{13}$$

where $\Lambda = \left\{ \mu \in R^{\mathcal{I}} \,|\, \mu \geq 0,\ A\mu = 0,\ B\mu = 1,\ \mu_0 = 1 \right\}$ is called the *local* marginal polytope. The equalities $A\mu = 0$ express *marginalization* constraints

$$\begin{aligned} &(\forall st \in \mathcal{E})\ (\forall i \in \mathcal{L}_s) \qquad \sum_{j' \in \mathcal{L}_t} \mu_{st}(i, j') - \mu_s(i) = 0\,, \\ &(\forall st \in \mathcal{E})\ (\forall j \in \mathcal{L}_t) \qquad \sum_{i' \in \mathcal{L}_s} \mu_{st}(i', j) - \mu_t(j) = 0 \end{aligned} \tag{14}$$

by means of the $m_1 \times |\mathcal{I}|$ matrix A, where $m_1 = \sum_{st \in \mathcal{E}} \left(|\mathcal{L}_s| + |\mathcal{L}_t| \right)$. The equalities $B\mu = 1$ express *normalization* constraints

$$(\forall s \in \mathcal{V}) \qquad \sum_{i \in \mathcal{L}_s} \mu_s(i) = 1 \tag{15}$$

by means of $m_2 \times |\mathcal{I}|$ matrix B, where $m_2 = |\mathcal{V}|$. Polytope Λ is constructed such that its elements having integer coordinates represent the original labelings. It is easy to verify that inequalities (14), (15) hold for all *integer labelings* $\mu = \delta(x)$. And vice-versa, if $\mu \in \Lambda$ has all integer components, there exists a unique labeling x such that $\delta(x) = \mu$. Therefore, we have $\Lambda \cap \{0, 1\}^{\mathcal{I}} = \{\delta(x) \,|\, x \in \mathcal{L}\}$. However, there may be non-integer (*fractional*) labelings in Λ that cannot be obtained by a convex combination of integer ones, and generally we have $\mathcal{M} \subset \Lambda$ and

$$\min_{\mu \in \mathcal{M}} \langle f, \mu \rangle \geq \min_{\mu \in \Lambda} \langle f, \mu \rangle\,. \tag{16}$$

If the equality is achieved in (16) we say that the LP relaxation (13) is *tight*. It has long been known to be tight for tree-structured problems (graph $(\mathcal{V}, \mathcal{E})$ is a tree) and submodular problems (E_f is a submodular function on a distributive lattice[2]). Thapper and Zivny (2012) give sufficient conditions describing a wider class of problems in which the relaxation is tight. This class includes submodular functions on arbitrary lattices and more. When terms f_s and f_{st} take values in the positive rationals (not including ∞), there is a precise characterization (Thapper and Zivny 2012) and a polynomial verification (Kolmogorov 2012b) of when the relaxation is tight for a given set of admissible terms f_s and f_{st}.

The polytope Λ inherits all linear equality constraints of $\mathcal{M}$ but keeps only a small polynomial number of inequality constraints (only the constraints $\mu \geq 0$), therefore it makes an outer approximation to $\mathcal{M}$ (Wainwright et al. 2003). Inequality constraints $\mu \geq 0$ are the facets of Λ. Λ has fewer facets than $\mathcal{M}$, at the same time Λ has more vertices than $\mathcal{M}$ (which are both exponential numbers). A vector $\mu \in \Lambda$ will be called a *relaxed* labeling.

[2]There are also submodular functions on non-distributive lattces

2.2.1 Duality

Looking at the equality constraints of Λ, we can see that the function of μ

$$\langle \varphi, A\mu \rangle + \langle \psi, B\mu \rangle - \langle \psi, 1 \rangle \tag{17}$$

vanishes over Λ for arbitrary $\varphi \in \mathbb{R}^{m_1}$ and $\psi \in \mathbb{R}^{m_2}$. This function can be written in the form of a scalar product as $-\langle h, \mu \rangle$, where $h = -A^{\mathsf{T}}\varphi - B^{\mathsf{T}}\psi + \langle \psi, 1 \rangle \cdot e_0$ and e_0 is the basis vector corresponding to the component g_0. By substituting matrices A and B, we get an expanded component-wise expression for h which coincides identically with the zero problem definition given by (8). It follows from this expression and Theorem 2 that the space of zero problems is the same for discrete and relaxed labelings. A primal counterpart of Theorem 2 would be that the affine hull of Λ coincides with the affine hull of $\mathcal{M}$ (which was intended when constructing the relaxation but it is not directly apparent). Thus we have that $E_f \equiv E_g$ iff

$$(\forall \mu \in \Lambda) \quad \langle f, \mu \rangle = \langle g, \mu \rangle \,. \tag{18}$$

By construction (17), elements of the vectors φ, ψ are nothing else but the Lagrange multipliers for the respective constraints of Λ. They become the dual variables once we re-express the linear relaxation problem as follows:

$$\begin{aligned}
\min_{\substack{A\mu = 0 \\ B\mu = 1 \\ \mu_0 = 1 \\ \mu \geq 0}} \langle f, A\mu \rangle &= \min_{\mu_0 = 1 \;\; \mu \geq 0} \max_{\varphi \in \mathbb{R}^{m_1} \;\; \psi \in \mathbb{R}^{m_2}} \left[\langle f, \mu \rangle - \langle \varphi, A\mu \rangle - \langle \psi, B\mu - 1 \rangle \right] && \text{(19a)} \\
&= \max_{\varphi \in \mathbb{R}^{m_1} \;\; \psi \in \mathbb{R}^{m_2}} \min_{\mu_0 = 1 \;\; \mu \geq 0} \left[\langle f - A^{\mathsf{T}}\varphi - B^{\mathsf{T}}\psi, \mu \rangle + \langle \psi, 1 \rangle \right] && \text{(19b)} \\
&= \max_{\varphi \in \mathbb{R}^{m_1} \;\; \psi \in \mathbb{R}^{m_2}} \left[\langle \psi, 1 \rangle + f_0 \right] \quad \text{subj.} \;\; (f - A^{\mathsf{T}}\varphi - B^{\mathsf{T}}\psi)_{\mathcal{I}\setminus 0} \geq 0 \,. && \text{(19c)}
\end{aligned}$$

In the inequalities $(f - A^{\mathsf{T}}\varphi - B^{\mathsf{T}}\psi)_{\mathcal{I}\setminus 0} \geq 0$ all components except of the constant term are included. Note, we write equalities in (19) since the primal problem is feasible. Clearly, it is also bounded since all components of μ are in the interval [0, 1]. Using the notation of equivalent problems, we can write the dual problem as

$$\begin{aligned}
&\max \;\; g_0 \,, \\
&g \equiv f \\
&g_s \geq 0 \\
&g_{st} \geq 0
\end{aligned} \tag{20}$$

which reads: "*find an equivalent representation that maximizes the constant term under the condition that all other terms are non-negative*". This representation of the dual problem is easily identified with the complementation approach in pseudo-Boolean optimization (Boros and Hammer 2002) (maximization of the constant term of the posiform), equivalent to the roof-dual bound.

In the expanded form, we have the following primal-dual pair (where all constraints

are written on the same line as their respective dual variables):

$$\begin{array}{rcl} \min\langle f,\mu\rangle & = & \max \sum_s \psi_s + \sum_{st} \psi_{st} + f_0\,. \\ \sum_j \mu_{st}(i,j) = \mu_s(i) & & \varphi_{st}(i) \in \mathbb{R} \\ \sum_i \mu_s(i) = 1 & & \psi_s \in \mathbb{R} \\ \sum_{i,j} \mu_s(i,j) = 1 & & \psi_{st} \in \mathbb{R} \\ \mu_s(i) \geq 0 & & f_s(i) + \sum_{t\in\mathcal{N}(s)} \varphi_{st}(i) - \psi_s \geq 0 \\ \mu_{st}(i,j) \geq 0 & & f_{st}(i,j) - \varphi_{st}(i) - \varphi_{ts}(j) - \psi_{st} \geq 0 \\ \mu_0 = 1 & & \end{array} \tag{21}$$

where we introduced for convenience additional normalization constraints on pairs (which are implied by other constraints of Λ) and their corresponding dual variables ψ_{st}. The variables ψ in (21) decouple and have the following explicit solution

$$\begin{aligned} \psi_s &= \min_i \big[f_s(i) + \sum_{t\in\mathcal{N}(s)} \varphi_{st}(i)\big] &&= \min_i f_s^\varphi(i)\,, \\ \psi_{st} &= \min_{ij} \big[f_{st}(i,j) - \varphi_{st}(i) - \varphi_{ts}(j)\big] &&= \min_{ij} f_{st}^\varphi(i,j)\,. \end{aligned} \tag{22}$$

Substituting this optimal solutions for ψ allows us to rewrite the dual problem in the form of unconstrained maximization of a piecewise linear concave function:

$$\max_{\varphi\in\mathbb{R}^{m_1}} \Big[\sum_{s\in\mathcal{V}} \min_i f_s^\varphi(i) + \sum_{st\in\mathcal{E}} \min_{i,j} f_{st}^\varphi(i,j)\Big] \stackrel{\text{def}}{=} \max_{\varphi\in\mathbb{R}^{m_1}} LB(f^\varphi)\,. \tag{23}$$

Let us define

$$LB(f) = \sum_{s\in\mathcal{V}} \min_i f_s(i) + \sum_{st\in\mathcal{E}} \min_{i,j} f_{st}(i,j). \tag{24}$$

We have for an arbitrary φ the chain

$$\min_{x\in\mathcal{L}} E_f(x) \geq \min_{\mu\in\Lambda} \langle f,\mu\rangle = \max_{\varphi\in\mathbb{R}^{m_1}} LB(f^\varphi) \geq LB(f^\varphi)\,, \tag{25}$$

that is, $LB(f^\varphi)$ is a lower bound on the energy minimization. Several dedicated algorithms to maximize this and similar lower bounds have been proposed. There are methods which achieve necessary conditions (described below) of optimality (Koval and Schlesinger 1976; Wainwright et al. 2005; Kolmogorov 2006), subgradient methods (Komodakis et al. 2007; Schlesinger and Giginyak 2007; Schlesinger et al. 2011), as well as some other non-differentiable optimization techniques with better convergence rates (Savchynskyy et al. 2011; Kappes et al. 2012).

2.2.2 Decompositional Lower Bounds

The lower bound (23) can be alternatively obtained by the following simple consideration. The energy vector f can be decomposed into a sum of "simpler" vectors by letting

$$f = \sum_{k\in K} f^k\,, \tag{26}$$

where $f^k \in \mathbb{R}^{\mathcal{I}}$ for all $k \in K$. We then have an obvious inequality

$$\min_{x\in\mathcal{L}} \langle f, \delta(x)\rangle = \min_{x\in\mathcal{L}} \Big\langle \sum_{k\in K} f^k, \delta(x)\Big\rangle \geq \sum_{k\in K} \min_{x\in\mathcal{L}} \big\langle f^k, \delta(x)\big\rangle = \sum_{k\in K} \min_{x\in\mathcal{L}} E_{f^k}(x)\,. \tag{27}$$

In this way, we obtain a lower bound which is a sum of individual subproblems, corresponding to energies E_{f^k}. The decomposition is arranged such that the subproblems (*slave* problems) are easier to solve than the original problem (the *master* problem). In the case of the bound (23), the subproblems are of the form $\min_i f_s(i)$ and $\min_{ij} f_{st}(i,j)$, *i.e.*, the function E_f is decomposed into a sum of unary and pairwise terms. There may be many decompositions of the form (26). We want to find among them the one that maximizes the lower bound, *i.e.*, to maximize

$$\sum_k \min_x \langle f^k, \delta(x) \rangle \tag{28}$$

over all $(f^k \,|\, k \in K)$ subject to constraint (26) and constraints of our choice on the form of f^k. The purpose of the latter constraints is to restrict every f^k such that the respective minimization subproblems are tractable (*e.g.*, when f^k are simple unary and pairwise functions). In the case of the bound (23), some components of f^k are forced to be zero. Since these constraints are linear and the objective of (28) is piece-wise linear concave, the maximization (28) is going to be equivalent to a linear program.

Theorem 3 (Wainwright et al. 2005)**.** Let $(T^k \,|\, k \in K)$ be a collection of trees, where each T^k is a subgraph of $\mathcal{G}$. Let each f^k be allowed to have non-zero unary and pairwise terms only on tree T^k. Then the bound (28) is dual to the LP relaxation (13). □

The duality here is understood in a more general sense than LP duality – it includes reduction and substitution of variables. The bound (23), which we derived explicitly, becomes a special case of this theorem when the trees are taken as individual vertices and edges.

For this approach to be practical, it is necessary that the subproblems $\min_x \langle f^k, \delta(x) \rangle$ are tractable. Besides trees, other proposed tractable decompositions include a decomposition into sub- and supermodular subproblems (Shekhovtsov 2006) (dual to the standard LP relaxation of the binarized energy, §2.4), a decomposition into submodular 2-label problems for the Potts model (Shekhovtsov and Hlaváč 2008; Osokin et al. 2011) (dual to the standard LP relaxation), decompositions involving higher order terms (dual to higher order relaxations) (Johnson et al. 2007; Komodakis and Paragios 2008; Werner 2008; Sontag 2010). The decomposition approach is equivalent to splitting the variables in the original (primal) problem and taking the Lagrangian dual w.r.t. equality constraints of the split (Bertsekas 1995, dual decomposition).

2.2.3 Complementary Slackness

Let $\mu \in \Lambda$, $\varphi \in \mathbb{R}^{m_1}$, $\psi \in \mathbb{R}^{m_2}$. The complementary slackness theorem of linear programing states that μ and (φ, ψ) are optimal to their respective problems iff

$$\begin{aligned} (\forall s)\ (\forall i) \quad & \mu_s(i) = 0 \ \text{ or } \ f^\varphi_s(i) = \psi_s \,, \\ (\forall st)\ (\forall ij) \quad & \mu_{st}(i,j) = 0 \ \text{ or } \ f^\varphi_{st}(i,j) = \psi_{st} \,. \end{aligned} \tag{29}$$

As optimal values ψ satisfy (22), conditions (30) are equivalent to

$$\begin{aligned} (\forall s)\ (\forall i) \quad & \mu_s(i) = 0 \ \text{ or } \ f^\varphi_s(i) = \min_i f^\varphi_s(i) \,, \\ (\forall st)\ (\forall ij) \quad & \mu_{st}(i,j) = 0 \ \text{ or } \ f^\varphi_{st}(i,j) = \min_{ij} f^\varphi_{st}(i,j) \,. \end{aligned} \tag{30}$$

Often it is more handy to rewrite these conditions as implications to obtain, *e.g.*, that $\mu \in \Lambda$ and φ are optimal iff

$$\begin{aligned} (\forall s)\ (\forall i) \quad & \mu_s(i) > 0 \ \Rightarrow\ f_s^\varphi(i) = \min_i f_s^\varphi(i)\,, \\ (\forall st)\ (\forall ij) \quad & \mu_{st}(i,j) > 0 \ \Rightarrow\ f_{st}^\varphi(i,j) = \min_{ij} f_{st}^\varphi(i,j)\,. \end{aligned} \tag{31}$$

Therefore, for an optimal (non-unique) primal-dual pair, if $\mu_s(i) > 0$, then it must be $f_s^\varphi(i) = \min_i f_s^\varphi(i)$ (we say $f_s^\varphi(i)$ is *locally minimal*). And vice-versa, if the component $f_s^\varphi(i)$ is not locally minimal, then it must be $\mu_s(i) = 0$. Similarly, we say that $f_{st}^\varphi(i,j)$ is *locally minimal* if $f_{st}^\varphi(i,j) = \min_{ij} f_{st}^\varphi(i,j)$. For the optimality of μ and φ, it is necessary and sufficient that μ is a relaxed labeling assigning non-zero weights only to the locally minimal components of f^φ.

2.2.4 Sufficient Conditions

Pair (s,i) for some $i \in \mathcal{L}_s$ and $s \in \mathcal{V}$ will be called a *node* and a pair of nodes $((s,i),(t,j))$ for $st \in \tilde{\mathcal{E}}$ will be called an *arc*. Let us denote $\bar{f}_{st}(i,j) = [\![f_{st}(i,j) = \min_{i,j} f_{st}(i,j)]\!]$ and $\bar{f}_s(i) = [\![f_s(i) = \min_i f_s(i)]\!]$. The vector $\bar{f}$ encodes which arcs and nodes of f are locally minimal. If the following Boolean formula

$$\bigwedge_{s \in \mathcal{V}} \overline{f^\varphi}_s(x_s) \wedge \bigwedge_{st \in \mathcal{E}} \overline{f^\varphi}_{st}(x_s, x_t) \tag{32}$$

is satisfiable for some $x \in \mathcal{L}$ then it follows that $\mu = \delta(x)$ and φ satisfy complementary slackness (30). Hence, they are optimal to their respective problems. At the same time, we have that $E_f(x) = \langle f, \delta(x) \rangle = \langle f, \mu \rangle = LB(f^\varphi)$, thus the relaxation (resp. the bound) is *tight* and x is a minimizer of E_f.

Suppose φ is optimal and f_s^φ has a unique minimizer for every s. In that case the expression (32) is clearly satisfiable. The global minimum of E_f is unique and is found trivially by minimizing every f_s^φ independently. The idea of problem *trivialization* by Shlezinger (1976) consists in finding an equivalent transformation f^φ such that (32) is satisfiable, which would allow (in some cases easily) to find an optimal labeling. In general, however, verifying satisfiability of (32) is the NP-complete constraint satisfaction problem (CSP). Wainwright et al. (2005) considered a condition (strong tree agreement) equivalent to (32), derived in the context of problem decomposition into trees.

2.2.5 Necessary Conditions

Let μ and φ be a feasible primal-dual pair. It means simply $\mu \in \Lambda$ in our case. The complementary slackness implies the following necessary condition of optimality. For every node (s,i) such that $f_s^\varphi(i)$ is locally minimal, either $\mu_s(i)$ is zero and, by feasibility, also $(\forall t \in \mathcal{N}(s)\ \forall j)\ \Big(\mu_{st}(i,j) = 0$ or $\mu_s(i) > 0\Big)$. In the latter case, again by feasibility, it must be $(\forall t \in \mathcal{N}(s)\ \exists j)\ \mu_{st}(i,j) > 0$ and hence $f_{st}^\varphi(i,j)$ is locally minimal.

We say that locally minimal node (s,i) is *arc-consistent* if

$$(\forall t \in \mathcal{N}(s)\ \exists j)\ f_{st}^\varphi(i,j) \text{ is locally minimal.} \tag{33}$$

Similarly, for every arc $((s,i),(t,j))$ such that $f_{s,t}^\varphi(i,j)$ is locally minimal either $\mu_{st}(i,j) = 0$ or both $f_s^\varphi(i)$ and $f_t^\varphi(j)$ are locally minimal. We say that locally minimal arc $((s,i),(t,j))$ is *arc-consistent* if

$$f_s^\varphi(i) \text{ and } f_t^\varphi(j) \text{ are locally minimal.} \tag{34}$$

The next theorem, following from the construction, states that complementary slackness implies that there is a subset of locally minimal arcs and nodes of f^φ which is arc-consistent.

Theorem 4 (Shlezinger 1976, Theorem 2)**.** For the equivalent f^φ to be dual-optimal it is necessary that there exists an arc-consistent subset of locally minimal nodes and arcs of f^φ. □

The existence of a subset of arc-consistent locally minimal nodes and arcs is equivalent to the weak tree agreement condition considered by (Kolmogorov 2006) in the context of tree decompositions. When *all* locally minimal nodes and arcs of f^φ are arc-consistent, we say that f^φ is *arc-consistent*. It is more than necessary for optimality of f^φ, however an equivalent transformation satisfying this requirement always exists.

Theorem 5. For every energy vector f, there exists an arc-consistent equivalent problem f^φ (called *the arc-consistent equivalent*).

Proof. Let φ be dual-optimal (it exists since the primal problem is feasible and bounded). Then the necessary conditions of optimality are satisfied. It is easy to find an equivalent transformation φ' such that all nodes and arcs of $f^{\varphi'}$ are arc-consistent, see *e.g.* (Werner 2007, Theorem 7). □

We will often give proofs of certain properties like optimality guarantees for the arc-consistent equivalent. By Theorem 5, these properties will hold for the initial problem. To clarify the relations between different conditions, we can draw the following diagram:

$LB(f^\varphi)$ is tight: $\min_x E_f(x) = LB(f^\varphi)$	(a) $\Leftrightarrow$	sufficient condition: $\overline{f^\varphi}$ is satisfiable
$\Downarrow$ (b)		$\Downarrow$ (c)
φ is optimal: $\varphi \in \arg\max_\varphi LB(f^\varphi)$	(d) $\Rightarrow$	necessary condition: subset of $\overline{f^\varphi}$ is AC

Implications (a), (d) and (b) were considered by Shlezinger (1976, Theorems 4 and 2) and Schlesinger and Flach (2002, Theorem 2), respectively. Their approach to solve the energy minimization problem is to maximize LB by linear programming or by an algorithm which achieves the necessary conditions, and check whether $\overline{f^\varphi}$ is satisfiable. This approach is guaranteed to find a solution for all tree-structured and permuted-submodular problems (Schlesinger and Flach 2000).

One important result in the case of two labels is that arc-consistency is not only necessary but also sufficient for the optimality (of LP) and that the primal optimal μ can be recovered by assigning its components to $0, 1/2, 1$ locally as appropriate to satisfy complementary slackness and feasibility.

2.3 Binary Variables

Let $\mathcal{L}_u = \mathbb{B} = \{0, 1\}$ for all $u \in \mathcal{V}$. Variables $x_u \in \mathcal{L}_u$ will be called *binary* or Boolean throughout this text. The unary and pairwise terms of energy E_f can be expanded as

$$\begin{aligned} f_u(x_u) &= f_u(1)x_u + f_u(0)(1 - x_u)\,, \\ f_{uv}(x_u, x_v) &= f_{uv}(1,1)x_u x_v + f_{uv}(0,1)(1 - x_u)x_v \\ &\quad + f_{uv}(1,0)x_u(1 - x_v) + f_{uv}(0,0)(1 - x_u)(1 - x_v)\,. \end{aligned} \tag{35}$$

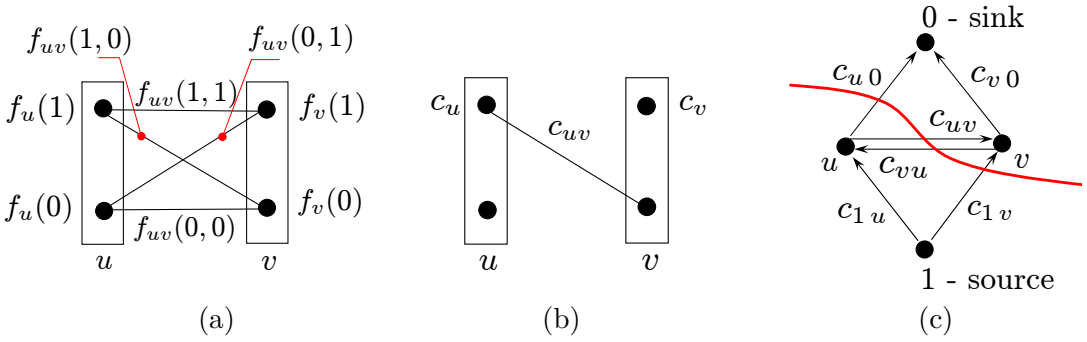


Figure 2 Equivalent MINCUT representation for energy minimization with binary variables. (a) Energy terms for pixels u, v and pair $uv \in \mathcal{E}$. (b) Equivalent transformation of the energy allowing to rewrite it in the form (38). (c) Cut-cost representation of the energy function. The cut shown in red is $(\mathcal{A}, \mathcal{V}\backslash\mathcal{A})$ with $\mathcal{A} = \{1, u\}$. It corresponds to the labeling $x_{uv} = (1, 0)$ and has the cost $c_{u0} + c_{uv} + c_{1v}$ equivalent (up to a constant) to the respective energy cost.

By expanding braces in (35), the energy $E_f(x)$ takes the form

$$E_f(x) = \sum_{u \in \mathcal{V}} a_u x_u + \sum_{uv \in \mathcal{E}} a_{uv} x_u x_v + a_0 \,. \tag{36}$$

Expression (36) is a quadratic polynomial in binary variables x. Functions of the form $\mathbb{B}^{\mathcal{V}} \mapsto \mathbb{R}$ are known as pseudo-Boolean and their minimization (or maximization) are the subject of *pseudo-Boolean* optimization (Boros and Hammer 2002). In this work, we will discuss several methods which are based on the results developed in pseudo-Boolean optimization or are generalizing them in a certain way.

2.3.1 Roof Dual (QPBO)

In the case of two labels, the necessary condition of optimality of the LP relaxation, described in §2.2.5 is also sufficient.

Theorem 6. Let f^φ be an arc-consistent equivalent of energy f with two labels. Then there is an optimal relaxed labeling with components in $\{0, \frac{1}{2}, 1\}$ (*half-integral*) satisfying complementary slackness.

This result was observed independently by Hammer et al. (1984); Schlesinger and Flach (2000); Kolmogorov and Wainwright (2005). See also (Werner 2007, Theorem 8).

In pseudo-Boolean optimization it was shown that several approaches, including the dual in the form (20), lead to the same lower bound, called the *roof dual* (Hammer et al. 1984; Boros and Hammer 2002). This dual problem can be converted to MAXFLOW on a specially constructed graph with a double number of vertices (Boros et al. 1991) and thus can be solved by efficient MAXFLOW algorithms. It was found to be a powerful method for quadratic pseudo-Boolean optimization and was also enhanced by *probing* (Boros et al. 2006; Rother et al. 2007). Kolmogorov and Rother (2007) and Rother et al. (2007) proposed a review, an efficient implementations and further improvements. After them, Quadratic Pseudo-Boolean Optimization, abbreviated as QPBO(-P), refers to this particular efficient method (resp. with probing). Kolmogorov (2010) gives an alternative interpretation of this method via a submodular lower bound.

For our purposes, we will assume that QPBO finds the arc-consistent equivalent f^φ. Let us introduce a function $\texttt{QPBO}(f) = \{O_s \,|\, s \in \mathcal{V}\}$, where $O_s = \operatorname{argmin}_i f^\varphi_s(i)$. It will be proven in §3.2 that any minimizer x of E_f satisfies $x_s \in O_s$ for all $s \in \mathcal{V}$.

For multi-label problems, the QPBO method can be used to *fuse* two given labelings (Lempitsky et al. 2010), restricting thus the search space to a binary choice in every pixel. In §2.7, we review QPBO fusion in the context of the expansion-move algorithm.

2.3.2 MINCUT/MAXFLOW

Let us first introduce the minimum s-t cut problem (MINCUT). It is defined on a capacitated directed graph (V, E, c) with *capacity* function $c\colon E \to \mathbb{R}_+$ and two vertices $s, t \in V$, $s \neq t$, called *source* and *sink*, respectively. The partition $(S, V\backslash S)$ of the vertex set V such that $s \in S$ and $t \notin S$ is called a *cut*. The *cost* of the cut $(S, V\backslash S)$ is the sum of costs of all edges with tail in S and head not in S. The minimum s-t cut problem is to find a cut of the minimum cost,

$$\min_{\substack{S\subset V\\ s\in S\\ t\notin S}} \sum_{\substack{uv\in E\\ u\in S,\ v\notin S}} c_{uv}\,. \tag{37}$$

This problem can be solved in polynomial time, it is dual to the MAXFLOW problem.

2.3.3 Reduction to MINCUT

From expansion (35) it is also clear that $E_f(x)$ can be equivalently written in the form

$$E_f(x) = \sum_{u\in\mathcal{V}} c_u x_u + \sum_{uv\in\tilde{\mathcal{E}}} c_{uv} x_u (1 - x_v) + c_0\,. \tag{38}$$

Let $\mathcal{A}$ be the set of variables with label 1, $\mathcal{A} = \{u \in \mathcal{V} \,|\, x_u = 1\}$. The second sum in (38) sums the costs over edges in $\tilde{\mathcal{E}}$ such that the tail of the edge is in $\mathcal{A}$, where $x_u = 1$, and the head is in $\bar{\mathcal{A}} = \mathcal{V}\backslash\mathcal{A}$, where $x_v = 0$. This is exactly the cost of the cut $(\mathcal{A}, \bar{\mathcal{A}})$ in the capacitated graph $(\mathcal{V}, \tilde{\mathcal{E}}, c)$.

Figuring out the exact coefficients c takes some effort but one can verify that by collecting the terms of (35) w.r.t. $x_u(1 - x_v)$ and x_u the expression (38) is satisfied by the following coefficients (non-uniquely):

$$\begin{aligned}
(\forall uv \in \mathcal{E}) \quad & c_{uv} = f_{uv}(1,0) + f_{uv}(0,1) - f_{uv}(0,0) - f_{uv}(1,1)\,,\\
(\forall uv \in \mathcal{E}) \quad & c_{vu} = 0\,,\\
(\forall u \in \mathcal{V}) \quad & c_u = f_u(1) - f_u(0)\\
& \qquad + \sum_{uv\in\mathcal{E}} \left[f_{uv}(1,1) - f_{uv}(0,1)\right] + \sum_{vu\in\mathcal{E}} \left[f_{uv}(0,1) - f_{uv}(0,0)\right],\\
& c_0 = f_0 + \sum_{u\in\mathcal{V}} f_u(0) + \sum_{uv\in\mathcal{E}} f_{uv}(0,1) + \sum_{vu\in\mathcal{E}} \left[f_{uv}(0,0) - f_{uv}(0,1)\right].
\end{aligned} \tag{39}$$

To represent the cost of the linear terms of (38) as the cost of the cut, let us introduce two auxiliary variables with fixed values, $x_1 = 1$ (source) and $x_0 = 0$ (sink), and rewrite (38) as

$$E_f(x) = \sum_{uv\in\mathcal{E}'} c_{uv} x_u (1 - x_v) + c_{\text{const}}\,, \tag{40}$$

where $\mathcal{E}' = \tilde{\mathcal{E}} \cup \{(1,u) \,|\, u \in \mathcal{V}\} \cup \{(u,0) \,|\, u \in \mathcal{V}\}$ defines the extended graph $\mathcal{G}' = (\mathcal{V} \cup \{0,1\}, \mathcal{E}')$ with extra edges $(1,u)$ and $(u,0)$ having associated costs satisfying $c_{u0} - c_{1u} = c_u$. Then $E_f(x)$ is equivalent up to a constant to the cost of a cut $(\mathcal{A}, \bar{\mathcal{A}})$ in $\mathcal{G}'$ separating the source ($1 \in \mathcal{A}$) and the sink ($0 \notin \mathcal{A}$) and the energy minimization

problem can be equivalently cast as a the (MINCUT) problem, where only the non-negativity constraints on c are not satisfied. Figure 2 illustrates this construction in a more detail.

The costs c_{1u} and c_{u0} can always be selected non-negative and simultaneously satisfy $c_{0u} - c_{1u} = c_u$. However, the costs c_{uv} are non-negative iff f_{uv} is submodular. In the latter case, all the edge weights in the MINCUT problem are non-negative, and the problem can be solved in a polynomial time by combinatorial algorithms.

There is one degree of freedom in the MINCUT representation for every edge $uv \in \mathcal{E}$, which corresponds to the equivalent transformations of the MINCUT (taking residual network w.r.t. a flow, Chapter 5). A tighter connection between energy representation and the MINCUT representation can be established by mapping (a subset of) equivalent transformations of the energy to feasible flows by *e.g.*, defining the following reverse mapping

$$\begin{aligned} &f_u(1) = c_{u0}\,, \quad f_v(1) = c_{v0}\,, \quad f_{uv}(1,0) = c_{uv}\,, \\ &f_u(0) = c_{1u}\,, \quad f_v(0) = c_{1v}\,, \quad f_{uv}(0,1) = c_{vu}\,, \\ &f_0 = c_0\,, \end{aligned} \tag{41}$$

with the rest of the terms of f set to zero. This relation allows us to speak of an equivalent transformation of the energy corresponding to a flow. For multi-label energies, the expansion-move algorithm (discussed in §2.7) optimizes in every iteration a binary labeling crossover problem via a reduction to MINCUT. Applying the maximum flow transformation of the MINCUT problem to the initial multi-label energy function induced by the mapping (41) is the dual counter-part of the expansion-move algorithm. This relation was exploited by Komodakis and Tziritas (2005) to develop a primal-dual algorithm (FastPD) and prove its approximation guarantees (see also the explanation of FastPD by Kolmogorov (2007)).

2.4 Reduction to Binary Variables

Many important results for the energy minimization problem (4) follow from the equivalent representation using binary variables. In particular, results from graph theory and pseudo-Boolean optimization are directly applicable.

In this section, we review the construction (Ishikawa 2003; Kovtun 2004; Schlesinger and Flach 2006) allowing us to represent the energy minimization problem (4) as an equivalent problem $\min_z E_g(z)$ with binary variables[3]. The equivalence is understood in the sense that there is a one-to-one correspondence between the objective functions, and hence solving the binary problem will deliver a solution to the initial problem, see Figure 3. Let $\mathcal{L}_s = \{0, 1, \dots L-1\}$ and $\tilde{\mathcal{L}}_s = \{0, 1, \dots L-2\}$ for all $s \in \mathcal{V}$. For each variable $x_s \in \mathcal{V}$ we introduce a representation with $L-1$ binary variables $(z_{s,i} \,|\, i \in \tilde{\mathcal{L}}_s,\ s \in \mathcal{V})$. The resulting binary labeling is $z\colon V \to \mathbb{B}$, where $V = \prod_{s\in\mathcal{V}} \tilde{\mathcal{L}}_s$. We establish correspondence between multi-label configurations x and binary configurations z by defining the mappings

$$(\forall i' \in \tilde{\mathcal{L}}_s) \qquad z_{s,i'}(x) = [\![x_s > i']\!]\,, \tag{42a}$$

$$x_s(z) = \sum_{i' \in \tilde{\mathcal{L}}} z_{s,i'}\,. \tag{42b}$$

[3] Ishikawa (2003) and Kovtun (2004, §2.3.2) considered a reduction to the MINCUT, while Schlesinger and Flach (2006) reduces to the energy minimization with two labels. It can be seen that these constructions are similar.

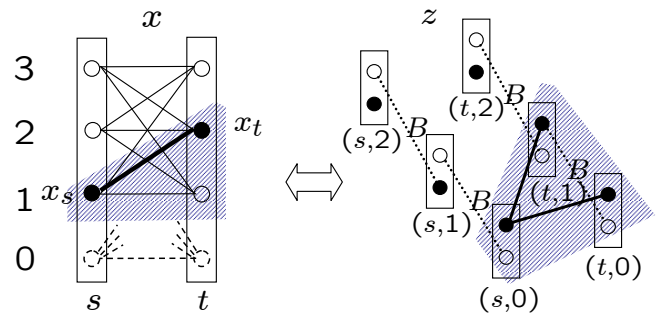


Figure 3 Transformation of multi-label energy to binary energy. Left: an interaction pair $st \in \mathcal{E}$; a labeling x is shown by black circles; the lowest labels are dashed since all weights in $\hat{f}_{st}$ and $\hat{f}_s$ associated with them are 0. Right: binary variables $z_{s,i}, z_{t,j}$; labeling $z(x)$ is shown by black circles; dotted lines marked with B correspond to hard constraints ensuring $z_{s,i+1} \geq z_{s,i}$ in the optimum. Bold nodes and arcs visualize unary and pairwise terms that contribute to $E_f(x)$ and $E_g(z)$ respectively for the given pair of corresponding labelings.

Mapping (42a) is one-to-one. Mapping (42b) becomes one-to-one if we constrain z to satisfy $z_{s,i+1} \geq z_{s,i}$ for $i = 0, 1, \ldots, L-3$. We will make sure that $E_g(z')$ is sufficiently big for any z' that does not satisfy these constraints. Now we construct a binary energy E_g such that $E_g(z) = E_f(x)$ for all x and z satisfying (42). It is achieved as follows. First, we construct an equivalent problem $\hat{f} \equiv f$ such that the following properties hold:

$$\begin{aligned} (\forall st \in \mathcal{E})\,(\forall ij \in \mathcal{L}_{st}) \;\; \hat{f}_{st}(0,j) = \hat{f}_{st}(i,0) &= 0\,, \\ (\forall s \in \mathcal{V}) \qquad\qquad\qquad\quad \hat{f}_s(0) &= 0\,. \end{aligned} \tag{43}$$

In this representation, the correspondence of E_f and E_g can be established easier. However, because representation (43) is unique, the full equivalence class of f will be mapped to the same g. In other words, equivalent transformations of f will not be mapped onto those of the binary problem g. Should such extended mapping be needed, it can be established directly, or dually by defining the mapping of relaxed labelings as in §4.2.4, where it is used to analyze corresponding LP relaxations. The equivalent problem $\hat{f}$ satisfying (43) is constructed as follows:

$$\begin{aligned} (\forall st \in \mathcal{E})\,(\forall ij \in \mathcal{L}_{st}) \qquad \hat{f}_{st}(i,j) &= f_{st}(i,j) - f_{st}(i,0) - f_{st}(0,j) + f_{st}(0,0)\,, \\ (\forall s \in \mathcal{V})\,(\forall i \in \mathcal{L}_s) \qquad \hat{f}_s(i) &= f_s(i) + \sum_{t \in \mathcal{N}(s)} f_{st}(i,0) - f_s(0) - \sum_{t \in \mathcal{N}(s)} f_{st}(0,0)\,, \\ \hat{f}_0 &= f_0 + \sum_{st \in \mathcal{E}} f_{st}(0,0) + \sum_{s \in \mathcal{V}} f_s(0)\,. \end{aligned} \tag{44}$$

It is easy to verify that conditions (43) hold for $\hat{f}$ and that $E_{\hat{f}} = E_f$. Note, in the case of two labels, the non-zero terms of $\hat{f}$ are only $\hat{f}_s(1)$, $\hat{f}_{st}(1,1)$ and $\hat{f}_0$, thus $\hat{f}$ is the polynomial representation (36). For a multi-label problem, representation $E_{\hat{f}}$ is the minimal (and hence unique) encoding of the energy function.

We can represent $\hat{f}_s$ as the following cumulative sum

$$(\forall i \in \mathcal{L}_s) \qquad \hat{f}_s(i) = \sum_{0 \leq i' < i} D_{i'} \hat{f} = \sum_{i' \in \tilde{\mathcal{L}}_s} [\![i' < i]\!] D_{i'} \hat{f}\,, \tag{45}$$

where $D_i \hat{f}_s = \hat{f}_s(i+1) - \hat{f}_s(i)$ is the *discrete derivative* (finite difference) of f_s at $i \in \tilde{\mathcal{L}}_s$ (where we adopt that the sum over empty set is zero, *i.e.*, when $i = 0$).

Analogously, the function $\hat{f}_{st}$ can be represented as

$$(\forall ij \in \mathcal{L}_{st}) \quad \hat{f}_{st}(i,j) = \sum_{\substack{0 \le i' < i \\ 0 \le j' < j}} D_{i'j'} \hat{f}_{st} = \sum_{i'j' \in \tilde{\mathcal{L}}_{st}} [\![i' < i]\!] [\![j' < j]\!] D_{i'j'} \hat{f}_{st} \,, \tag{46}$$

where $D_{ij}\hat{f}_{st} = \hat{f}_{st}(i,j) + \hat{f}_{st}(i+1,j+1) - \hat{f}_{st}(i,j+1) - \hat{f}_{st}(i+1,j)$ is the mixed second difference of $\hat{f}_{st}$ at $(i,j) \in \tilde{\mathcal{L}}_{st}$.

It follows from these representations that using variables $z_{s,i} = [\![x_s \ge i]\!]$, the energy $\hat{E}_f(x)$ can be expressed as a quadratic polynomial in z. We can summarize now the components of the equivalent binary energy minimization problem:

- Graph (V, E), where $E = E^1 \cup E^2$. $E^1 = \big\{((s,i),(t,j)) \,|\, st \in \mathcal{E},\, ij \in \tilde{\mathcal{L}}_{st}\big\}$. $E^2 = \big\{((s,i),(s,i+1)) \,|\, s \in \mathcal{V},\, i = 0,1,\dots L-3\big\}$.
- Binary configuration $z \in \mathbb{B}^V$. For a configuration $x \in \mathcal{L}^{\mathcal{V}}$ the corresponding binary configuration z is defined via mapping $z(x)$ (42a).
- Binary energy function $E_g(z)$ is constructed as

$$\begin{aligned} & & g_0 &= \hat{f}_0 \,, \\ (\forall st \in \mathcal{E})\ (\forall ij \in \tilde{\mathcal{L}}_{st}) & & g_{(s,i)(t,j)}(1,1) &= D_{ij}\hat{f}_{st} = D_{ij} f_{st} \,, \\ (\forall s \in \mathcal{V})\ (\forall i \in \tilde{\mathcal{L}}_s) & & g_{(s,i)}(1) &= D_i \hat{f}_s \,, \\ (\forall s \in \mathcal{V})\ (\forall i \in \{0,1,\dots L-3\}) & & g_{(s,i)(s,i+1)}(0,1) &= B \end{aligned} \tag{47}$$

  and the remaining components of g are set to zero. In (47), B is a sufficiently big number, ensuring that inequality $z_{s,i} \ge z_{s,i+1}$, for $i = 0,1,\dots L-3$ holds for any minimizer of g.

Statement 7. Constructed binary energy is equivalent to the original multi-label energy: For all $x \in \mathcal{L}$ it is $E_f(x) = E_g(z(x))$.

Proof. Let $z = z(x)$. From representations of unary and pairwise terms by (45) and (46) we have that

$$\begin{aligned} \hat{f}_0 &= g_0 \,, \\ \hat{f}_s(x_s) &= \sum_{i' \in \tilde{L}} g_{(s,i')}(z_{s,i'}) \,, \\ \hat{f}_{st}(x_{st}) &= \sum_{i',j' \in \tilde{L}} g_{(s,i')(t,j')}(z_{s,i'}, z_{t,j'}) \,. \end{aligned} \tag{48}$$

Therefore $E_g(z) = E_{\hat{f}}(x) = E_f(x)$. □

For binary configurations z, for which the constraint terms $g_{(s,i)(s,i+1)}(0,1)$ are not active, it holds $z_{s,i} \ge z_{s,i+1}$ for $i = 0,1,\dots L-3$ and there holds $z(x(z)) = z$. Whenever one of the constraint terms is active, the energy E_g gets an increase of B, and by construction the corresponding configuration is not a minimizer. Therefore, we have a one-to-one correspondence between minimizers of E_f and E_g.

Other reductions to binary variables are possible. A reduction which is independent of the order of labels (Shekhovtsov et al. 2008) was shown to have degenerate LP relaxation. The specific reduction we considered is interconnected with the notion of submodular multi-label energies.

Induced MINCUT Representation By reduction of a multi-label E_f to binary E_g and the subsequent reduction to MINCUT we can reduce arbitrary pairwise energy minimization to MINCUT where the non-negativity capacity constraint may be violated. Recalling the construction of pairwise costs (47) we see that the capacities in the MINCUT representation will be non-negative iff $g_{(s,i)(t,j)}(1,1) \leq 0$, or, equivalently, $D_{ij} f_{st} \leq 0$ $\forall ij \in \tilde{\mathcal{L}}_{st}$. This condition coincides with the common characterization of multi-label submodular (pairwise) energies.

2.5 Submodular Energy Minimization

Let us define the component-wise minimum and maximum of two labellings $x, y \in \mathcal{L}$:

$$(\forall s \in \mathcal{V}) \quad (x \wedge y)_s = \min(x_s, y_s)\,, \tag{49a}$$

$$(\forall s \in \mathcal{V}) \quad (x \vee y)_s = \max(x_s, y_s)\,. \tag{49b}$$

These two operations on the set of labelings define a distributive lattice[4] $(\mathcal{L}, \vee, \wedge)$. Function $E\colon \mathcal{L} \to \mathbb{R}$ is called *submodular* if

$$(\forall x, y \in \mathcal{L}) \quad E(x \vee y) + E(x \wedge y) \leq E(x) + E(y)\,. \tag{50}$$

Submodular function minimization on a distributive lattice is solvable in a polynomial time. In the case $E = E_f$, it is submodular iff (see *e.g.* (Werner 2007)) for all $st \in \mathcal{E}$

$$(\forall x_{st}, y_{st} \in \mathcal{L}_{st}) \quad f_{st}(x_{st} \wedge y_{st}) + f_{st}(x_{st} \vee y_{st}) \leq f_{st}(x_{st}) + f_{st}(y_{st})\,, \tag{51}$$

that is, all pairwise terms f_{st} are submodular functions.

It can be seen (*e.g.*, (Fisher et al. 1978)) that f_{st} satisfies (51) iff

$$(\forall ij \in \tilde{\mathcal{L}}_{st}) \quad D_{ij} f_{st} \leq 0\,. \tag{52}$$

We can see that when E_f is submodular, the pairwise coefficients of the binary-reduced energy E_g calculated by (47) are non-positive, which is precisely the condition when the binary energy is submodular and can be minimized by solving the MINCUT problem.

2.6 Relaxation of Binarized Problem (MQPBO)

Kohli et al. (2008) and Shekhovtsov et al. (2008) proposed reducing the problem to the binary energy minimization and applying QPBO method to the latter. This method is abbreviated as MQPBO (M for multi-label). Since LP relaxation of the binary problem can be solved via MAXFLOW, it is a computationally plausible approach. However, it was established that LP relaxation of the binarized problem is generally weaker than the multi-label LP relaxation. Nevertheless, for the following class of problems the two relaxations coincide:

Theorem 8 (Shekhovtsov et al. 2008, Theorem 1)**.** Let for every $st \in \mathcal{E}$ the term f_{st} be either sub- or supermodular. Then the LP relaxation of the binarized problem is equivalent to the LP relaxation of the multi-label problem (13). □

It was also shown that the LP relaxation of the binarized problem is dual to the problem of the tightest decomposition into sub- and supermodular problems (Shekhovtsov et al. 2008, Theorem 3). In §3.3 we will review how the optimal partial assignments derived by QPBO are transferred to the original multi-label problem for integer as well as relaxed labelings, simplifying the construction of Shekhovtsov et al. (2008, Theorem 2).

[4] Distributivity follows from distributivity of min and max, *e.g.*, $\max(a, \min(b, c)) = \min(\max(a, b), \max(a, c))$.

2.7 Expansion Move

The expansion move algorithm (Boykov et al. 2001) seeks to improve a current solution x by considering a *move* that for every $s \in \mathcal{V}$ either keeps the current label x_s or changes it to the candidate label y_s. While x and y are two labelings in the multi-label problem E_f, the choice between x and y can be encoded as a binary labeling z. We let $z_s = 0$ correspond to x_s and $z_s = 1$ correspond to y_s. The restriction of E_f to $\prod_{s\in\mathcal{V}}\{x_s, y_s\}$ becomes equivalent to a binary energy. The *move energy* function $E_g \colon \mathbb{B}^{\mathcal{V}} \to \mathbb{R}$ (the crossover problem) corresponding to this choice is defined as follows:

$$\begin{aligned} &g_0 = f_0\,, \quad g_s(0) = f_s(x_s)\,, \quad g_s(1) = f_s(y_s)\,, \\ &g_{st}(1,1) = f_{st}(y_s, y_t)\,, \quad g_{st}(1,0) = f_{st}(y_s, x_t)\,, \\ &g_{st}(0,1) = f_{st}(x_s, y_t)\,, \quad g_{st}(0,0) = f_{st}(x_s, x_t)\,. \end{aligned} \tag{53}$$

Here, the state 0 or 1 corresponds to choosing x_s or y_s respectively and $E_g(z)$ equals E_f of the corresponding crossover between labelings x and y. The expansion-move algorithm (Algorithm 1) iteratively improves the current labeling x by finding an optimal crossover with constant proposals y such that $(\forall s)\ y_s = k$.

Algorithm 1: Expansion Move (Boykov et al. 2001)

1 Input: f, x;
2 **while** *not converged in* $E_f(x)$ **do** /* */
3 **for** $k = 0 \dots L-1$ **do**
4 Create the crossover problem g using (53) between labelings x and y, where $(\forall s)\ y_s = k$;
5 Find the optimal switch $z = \operatorname{argmin}_z E_g(z)$;
6 $(\forall s \in \mathcal{V})$ assign $x_s \leftarrow \begin{cases} x_s & \text{if } z_s = 0\,, \\ y_s & \text{if } z_s = 1\,. \end{cases}$

In the case E_f is a metric energy (Boykov et al. 2001), the move energy E_g is submodular for arbitrary x and step 5 reduces to MINCUT.

2.7.1 Truncated Expansion Move

In the case of non-submodular move energy, it can be "truncated" to make it submodular while still preserving the property that the move does not increase $E_f(x)$ (Rother et al. 2005). We propose here an analysis of the truncation heuristic and show that the family of useful truncations can be identified with submodular majorants. We are not aware of such an analysis present in the literature.

Let $\Delta_{st} = g_{st}(0,0) + g_{st}(1,1) - g_{st}(0,1) - g_{st}(1,0) = \mathcal{D}_{00} g_{st}$. The pairwise term g_{st} is submodular iff $\Delta_{st} \le 0$.

Definition 2. The *truncation* $g^{\alpha,\beta}$ of g is defined as follows. It is different from g only in non-submodular pairwise components, which are set as

$$\begin{aligned} g^{\alpha,\beta}_{st}(0,0) &= g_{st}(0,0) - \beta_{st}\Delta_{st}\,, \\ g^{\alpha,\beta}_{st}(0,1) &= g_{st}(0,1) + \alpha_{st}\Delta_{st}\,, \\ g^{\alpha,\beta}_{st}(1,0) &= g_{st}(1,0) + (1 - \alpha_{st} - \beta_{st})\Delta_{st}\,, \\ g^{\alpha,\beta}_{st}(1,1) &= g_{st}(1,1)\,, \end{aligned} \tag{54}$$

where α_{st} and β_{st} are free parameters satisfying $\alpha_{st} \geq 0$, $\beta_{st} \geq 0$, $\alpha_{st} + \beta_{st} \leq 1$.

Statement 9. Energy function $E_{g^{\alpha,\beta}}$ is submodular and

$$(\forall z \in \mathbb{B}^{\mathcal{V}}) \quad E_g(z) - E_g(0) \leq E_{g^{\alpha,\beta}}(z) - E_{g^{\alpha,\beta}}(0)\,. \tag{55}$$

Inequality (55) says that the decrease of E_g is at least as big as the decrease of $E_{g^{\alpha,\beta}}$ when changing from 0 to z. Assuming $E_{g^{\alpha,\beta}}(z) < E_{g^{\alpha,\beta}}(0)$, the inequality (55) implies that $E_g(z) < E_g(0)$. Because g is the move energy vector for f, the value of $E_f(x)$ will decrease after step (6).

Proof. Submodularity of $E_{g^{\alpha,\beta}}$ is verified easily as follows. For pairs st, where g_{st} is non-submodular we have: $Dg_{st}^{\alpha,\beta} = 0$, therefore these pairs become modular (decoupled). Submodular pairs of g are preserved in $g^{\alpha,\beta}$. Let us verify inequality (55). By construction of $g^{\alpha,\beta}$, for all $st \in \mathcal{E}$ such that $\Delta_{st} > 0$,

$$\begin{aligned}
&g_{st}^{\alpha,\beta}(0,0) - g_{st}^{\alpha,\beta}(0,0) = 0\,,\\
&g_{st}^{\alpha,\beta}(0,1) - g_{st}^{\alpha,\beta}(0,0) = g_{st}(0,1) - g_{st}(0,0) + (\alpha_{st} + \beta_{st})\Delta_{st}\,,\\
&g_{st}^{\alpha,\beta}(1,0) - g_{st}^{\alpha,\beta}(0,0) = g_{st}(1,0) - g_{st}(0,0) + (1 - \alpha_{st})\Delta_{st}\,,\\
&g_{st}^{\alpha,\beta}(1,1) - g_{st}^{\alpha,\beta}(0,0) = g_{st}(1,1) - g_{st}(0,0) + \beta_{st}\Delta_{st}\,,
\end{aligned} \tag{56}$$

we see that only positive values are added on RHS. □

Statement 10. The truncation with α and $\beta_{st} > 0$ is *never better* than the truncation with α and $\beta = 0$:

$$E_{g^{\alpha,0}}(z) - E_{g^{\alpha,0}}(0) \leq E_{g^{\alpha,\beta}}(z) - E_{g^{\alpha,\beta}}(0)\,. \tag{57}$$

Proof. It can be seen that the values which are added in the RHS of (56) increase with the increase of β_{st}. This implies inequality (57). □

If an improving move z was found for a truncation $E_{g^{\alpha,\beta}}$ it will surely be an improving move also for truncation $E_{g^{\alpha,0}}$.

In the construction of auxiliary submodular problem by (Kovtun 2004) reviewed in §3.5, a truncation of the form $g^{0,1}$ occurs. Clearly, this truncation is looser than any other.

Statement 11. The truncation $g^{0,1}$ is not better than any truncation $g^{\alpha,\beta}$:

$$(\forall z)\ (\forall \alpha, \beta) \quad E_{g^{\alpha,\beta}}(z) - E_{g^{\alpha,\beta}}(0) \leq E_{g^{0,1}}(z) - E_{g^{0,1}}(0)\,. \tag{58}$$

Proof. The statement is verified by collecting all summands of inequality (58) on one side and examining each pairwise component:

$$g_{st}^{0,1}(z_{st}) - g_{st}^{0,1}(0) - g_{st}^{\alpha,\beta}(z_{st}) + g_{st}^{\alpha,\beta}(0) = \begin{cases} 0 & \text{if } \ z_{st} = (0,0)\,,\\ \Delta_{st}(1 - (\alpha + \beta)) & \text{if } \ z_{st} = (0,1)\,,\\ \Delta_{st}(1 - (1 - \alpha)) & \text{if } \ z_{st} = (1,0)\,,\\ \Delta_{st}(1 - \beta) & \text{if } \ z_{st} = (1,1)\,. \end{cases} \tag{59}$$

It is seen that the RHS is non-negative in all the cases. □

If z is an improving move for $E_{g^{0,1}}$ then it is also an improving move for any other truncation.

By Statement 10, one degree of freedom is excluded from the construction of truncations. It can be verified that truncated energies $E_{g^{\alpha,0}}$ are submodular *majorants* of E_g, *i.e.*,

$$\forall z \quad E_{g^{\alpha,0}}(z) \geq E_g(z)\,. \tag{60}$$

Several algorithms, such as submodular-supermodular procedure of Narasimhan and Bilmes (2005), can be formulated as constructing a submodular majorant that is exact at the current solution, and then minimizing this majorant. Clearly, by choosing α_{st} appropriately as 0 or 1 one can make sure that the truncated energy $E_{g^{\alpha,0}}$ is *exact* at the current solution z (it satisfies $E_{g^{\alpha,0}}(z) = E_g(z)$). Therefore, the expansion move with truncation can be understood as application of the majorize-minimize algorithm to solve the crossover problem. The truncation is exact at the current labeling 0. Either an improving move is found and Algorithm 1 proceeds, or $z = 0$ is a fixed point of the majorize-minimize for each of the crossover sub-problems. One can verify that the family of submodular majorants $\{E_{g^{\alpha,0}} \,|\, \alpha\}$ is tight and complete for E_g, *i.e.*, it describes all submodular majorants which are not dominated by other submodular majorants.

2.7.2 Fusion Move

Having two candidate solutions x and y, the fusion move of Lempitsky et al. (2010) applies the QPBO method to optimize the crossover problem E_g without truncation. QPBO can be used to find a solution z satisfying $E_f(z) \leq E_f(x)$, guaranteeing monotonous behavior. The algorithm is presented in Algorithm 2. We will show in Chapter 4 that fixed points of fusion move algorithm with constant proposals ($\forall k\; y_s = k$) satisfy multi-label partial optimality constraints.

Algorithm 2: Fusion-Move (Lempitsky et al. 2010)

1 Input: f, x, y;
2 Construct the crossover problem g using (53) between labelings x and y;
3 $O = \texttt{QPBO}(g)$; `/* all minimizers` z `of` g `satisfy` $z_s \in O_s$ `*/`
4 $(\forall s \in \mathcal{V})$ assign $x_s \leftarrow \begin{cases} x_s & \text{if } 0 \in O_s\,, \\ y_s & \text{otherwise}\,. \end{cases}$ `/* improve` x `*/`

3 Review of Partial Optimality Methods

In this chapter, we review in more detail several methods for optimal partial assignment. Such methods recover a "part of the optimal labeling" even in the case when finding the complete optimal labeling is not tractable.

Definition 3. Consider a subset of variables $\mathcal{A} \subset \mathcal{V}$ and the assignment of labels over this subset $y = (y_s \,|\, s \in \mathcal{A})$. The pair $(\mathcal{A}, y)$ is called a *strong optimal partial assignment* if for any minimizer x^* it holds $x^*_{\mathcal{A}} = y$. The pair $(\mathcal{A}, y)$ is called a *weak optimal partial assignment* if *there exists* a minimizer x^* such that $x^*_{\mathcal{A}} = y$.

Two or more strong optimal partial assignments can be combined together because each of them preserves all optimal solutions. This is not true for weak assignments because they may not share any globally optimal solutions. However, a weak optimal partial assignment could be more helpful – the best one assigns all variables. Therefore we are interested in recovering both: strong and weak partial assignments.

Several fundamental results identifying optimal partial assignments are obtained from the properties of linear relaxations of some discrete problems. An optimal solution to the continuous relaxation of a mixed-integer 0-1 programming problem is defined to be *persistent* if the set of $[0, 1]$ relaxed variables realizing binary values retains the same binary values in at least one integer optimum (Adams et al. 1998). A mixed-integer program is said to be *persistent* (or possess the *persistency* property) if *every* solution to its continuous relaxation is persistent. Nemhauser and Trotter (1975) proved that the vertex packing problem is persistent. This result was later generalized to optimization of quadratic pseudo-Boolean functions by Hammer et al. (1984). *Strong persistency* was also proved, stating that if a variable takes the same binary value in *all* optimal solutions to the relaxation, then *all* optimal solutions to the original 0-1 problem take this value.

Several works considered generalization of persistency to higher-order pseudo-Boolean functions. Adams et al. (1998) considered a hierarchy of continuous relaxations of 0-1 polynomial programming problems. Given an optimal relaxed solution, they derive sufficient conditions on the dual multipliers which ensure that the solution is persistent. This result generalizes the roof duality approach, coinciding with it in the case of quadratic polynomials in binary variables.

Kolmogorov (2010, 2012a) showed that bisubmodular relaxations provide a natural generalization of the roof duality approach to higher-order terms and possess the persistency property. He also considered submodular relaxations which form a special case of bisubmodular and showed the following. The roof duality relaxation for quadratic pseudo-Boolean functions is a submodular relaxation, and it dominates all other bisubmodular relaxations. For non-quadratic pseudo-Boolean functions, bisubmodular relaxations can be tighter than submodular ones. Kahl and Strandmark (2011, 2012) proposed a polynomial time algorithm to find the tightest submodular relaxation and evaluated it on problems in computer vision.

Lu and Williams (1987), Ishikawa (2011) and Fix et al. (2011) obtained partial optimalities via different reductions to quadratic problems and subsequent application of

the roof dual. We present here the specific methods which we are going to unify in Chapter 4. We thus follow two purposes: to formally introduce the respective methods and to make certain preparations to ease the subsequent unification. The following methods are considered:

- The family of local methods known as the dead end elimination (DEE), originally proposed by Desmet et al. (1992). DEE methods were developed in the context of protein structure design and are not widely known in the machine learning and computer vision communities. They formulate simple sufficient conditions allowing to exclude a label in a given pixel based on its unary and adjacent pairwise terms.
- The roof dual method (QPBO). We give a proof of the strong persistency theorem (§3.2) based on the component-wise inequalities satisfied by the arc-consistent equivalent. This proof is essentially in the form of the unified sufficient conditions.
- The MQPBO method (Kohli et al. 2008) extends partial optimality properties of QPBO to multi-label problems via the reduction of the problem to binary variables.
- Auxiliary submodular problems by Kovtun (2004).

In all these methods there is a mapping introduced (implicitly or explicitly) that is shown to improve (not necessarily strictly) any given labeling.

Definition 4. A mapping $p\colon \mathcal{L} \to \mathcal{L}$ is called *improving* if

$$(\forall x \in \mathcal{L}) \quad E_f(p(x)) \leq E_f(x) \tag{61}$$

and *strictly improving* if

$$(p(x) \neq x) \quad \Rightarrow \quad E_f(p(x)) < E_f(x)\,. \tag{62}$$

If every labeling x with $x_s = \alpha$ satisfies the strict inequality $E_f(p(x)) < E_f(x)$, then (s, α) cannot be a part of any optimal assignment and can be eliminated. The idea of improving mappings will be generalized in Chapter 4 to improving mappings of relaxed labelings, *i.e.*, of the from $\Lambda \to \Lambda$. In QPBO, an improving mapping is defined in terms of an *autarky* (Boros et al. 2006). In MQPBO and the method of (Kovtun 2004), an improving mapping is defined in terms of a generalized autarky, which we introduce as follows.

Definition 5. A pair $(x^{\min}, x^{\max})$ is a *weak autarky* for E_f if the inequality

$$E_f((x \vee x^{\min}) \wedge x^{\max}) \leq E_f(x) \tag{63a}$$

holds for all $x \in \mathcal{L}$. If, in addition, for every $x \neq (x \vee x^{\min}) \wedge x^{\max}$ inequality (63a) is strict, the autarky $(x^{\min}, x^{\max})$ is called *strong*.

It is clear that an autarky $(x^{\min}, x^{\max})$ defines an improving mapping $x \mapsto (x \vee x^{\min}) \wedge x^{\max}$. We have then for any labeling x that the labeling $x^* = (x \vee x^{\min}) \wedge x^{\max}$ is not worse than x. Moreover, if $x^* \neq x$ we have $E_f(x^*) < E_f(x)$. Therefore, any optimal labeling must satisfy $x = (x \vee x^{\min}) \wedge x^{\max}$, which translate to the inequalities

$$x^{\min} \leq x \leq x^{\max}\,. \tag{64}$$

For a strong (resp. weak) autarky $(x^{\min}, x^{\max})$ the inequality (64) is satisfied for all (resp. some) minimizers x, providing an optimal partial assignment in pixels s where $x^{\min}_s = x^{\max}_s$. In the case where $x^{\min}_s < x^{\max}_s$ we speak of domain constraints.

Definition 6. Let $\mathcal{A} \subset \mathcal{V}$, let $(\forall s \in \mathcal{A})$ $K_s \subset \mathcal{L}_s$. Let K be the Cartesian product of K_s over $s \in \mathcal{A}$. We say that a pair $(\mathcal{A}, K)$ is a *strong (resp. weak) domain constraint* if for all (resp. at least one) minimizer x^* there holds $x^*_{\mathcal{A}} \in K$.

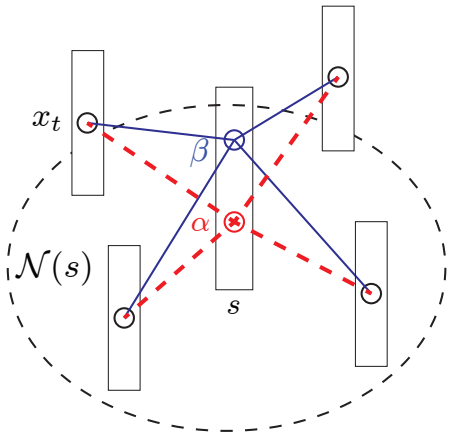


Figure 4 Singles dead end elimination. If, for any configuration of the neighborhood, there is a labeling through label β in s (blue) that dominates the labeling through a fixed label α in s (red) then (s, α) can be eliminated as non-optimal.

Obviously, domain constraints include partial assignment as a special case, or can be veiewd as a partial assignment in some binary encoding of the problem. A weak (resp. strong) autarky $(x^{\min}, x^{\max})$ provides weak (resp. strong) domain constraints with $K_s = \{i \in \mathcal{L}_s \,|\, x_s^{\min} \leq i \leq x_s^{\max}\}$.

Even in the case of binary variables determining whether a given pair $(z^{\min}, z^{\max})$ is a strong autarky is NP-hard (Boros et al. 2006). The methods we review can be viewed as proposing autarkies based on simpler sufficient conditions which are polynomially verifiable. In the special case of submodular energies, autarkies *are* polynomially verifiable, and this case is also covered by our unified sufficient condition.

3.1 Dead End Elimination

A series of works related to energy minimization was published in the context of protein design and protein structure prediction. Several applied problems in this area are formulated as a pairwise energy minimization[1]. These works try to simplify the problem as much as possible by removing states that cannot be a part of any optimal configuration. There is a number of sufficient conditions proposed, which are generally referred to as *dead end elimination* (DEE), see (Desmet et al. 1992; Goldstein 1994; Lasters et al. 1995; Pierce et al. 2000; Georgiev et al. 2006) and others. We will review the so-called singles elimination techniques. The goal is to determine for a fixed $s \in \mathcal{V}$ and $\alpha \in \mathcal{L}_s$ whether the condition $x_s \neq \alpha$ is true for any optimal assignment x. In that case, (s, α) can be eliminated without affecting any of the optimal assignments. In the literature these different but related criteria are proven separately. It will be instructive to start with the strongest criterion and then gradually relax it, ordering the criteria proposed by different authors in a chain of inequalities. In this way we will simultaneously prove the respective criteria and demonstrate how they compare to each other in a concise way.

[1] For example, the assignment x_s can define a selection and a spatial conformation of side chains of amino-acids onto residue positions of the protein scaffold and the energy is the sum of self-energy of residues plus pairwise interaction energies. See *e.g.* (Looger and Hellinga 2001) and references therein.

The pair (s, α) can be eliminated if

$$(\forall x \in \mathcal{L}_{\mathcal{N}(s)})\ (\exists \beta)\quad f_s(\alpha) + \sum_{t \in \mathcal{N}(s)} f_{st}(\alpha, x_t) > f_s(\beta) + \sum_{t \in \mathcal{N}(s)} f_{st}(\beta, x_t)\,. \tag{65}$$

For each labeling of the neighbors of s, changing label α to label β in s (not necessarily the same β for different x) gives an improved energy. This condition is called *bottom line* DEE (Pierce et al. 2000). It is illustrated in Figure 4. It is rather of theoretical interest as it may be computationally too expensive to verify when the neighborhood of s is large. Let us rewrite it equivalently as

$$(\forall x \in \mathcal{L}_{\mathcal{N}(s)})\ (\exists \beta)\quad f_s(\alpha) - f_s(\beta) + \sum_{t \in \mathcal{N}(s)} [f_{st}(\alpha, x_t) - f_{st}(\beta, x_t)] > 0\,, \tag{66}$$

$$\min_{x \in \mathcal{L}_{\mathcal{N}(s)}} \max_{\beta \in \mathcal{L}_s} \Big(f_s(\alpha) - f_s(\beta) + \sum_{t \in \mathcal{N}(s)} [f_{st}(\alpha, x_t) - f_{st}(\beta, x_t)] \Big) > 0\,. \tag{67}$$

Next, we represent β with a vector of indicator variables $(\bar\beta_i \in \{0,1\} \,|\, i \in \mathcal{L}_s)$ such that $\sum_i \bar\beta_i = 1$. In this representation we can rewrite LHS of (67) as

$$\min_{x \in \mathcal{L}_{\mathcal{N}(s)}} \max_{\substack{\bar\beta \in \{0,1\}^{\mathcal{L}_s} \\ \sum_i \bar\beta_i = 1}} \Big(f_s(\alpha) - \sum_i f_s(i)\bar\beta(i) + \sum_{t \in \mathcal{N}(s)} [f_{st}(\alpha, x_t) - \sum_i f_{st}(i, x_t)\bar\beta(i)] \Big) = \text{DEE1}\,. \tag{68}$$

Because the objective becomes linear in $\bar\beta$, we can allow $\bar\beta \in [0,\,1]^{\mathcal{L}_s}$ and write DEE1 as

$$\min_{x \in \mathcal{L}_{\mathcal{N}(s)}} \max_{\substack{\bar\beta \in [0,1]^{\mathcal{L}_s} \\ \sum_i \bar\beta_i = 1}} \Big(f_s(\alpha) - \sum_i f_s(i)\bar\beta(i) + \sum_{t \in \mathcal{N}(s)} [f_{st}(\alpha, x_t) - \sum_i f_{st}(i, x_t)\bar\beta(i)] \Big)\,. \tag{69}$$

By switching min and max we obtain the first simplification,

$$\begin{aligned}
\text{DEE1} &\geq \max_{\substack{\bar\beta \in [0,1]^{\mathcal{L}_s} \\ \sum_i \bar\beta_i = 1}} \min_{x \in \mathcal{L}_{\mathcal{N}(s)}} \Big(f_s(\alpha) - \sum_i f_s(i)\bar\beta(i) + \sum_{t \in \mathcal{N}(s)} [f_{st}(\alpha, x_t) - \sum_i f_{st}(i, x_t)\bar\beta(i)] \Big) \\
&= \max_{\substack{\bar\beta \in [0,1]^{\mathcal{L}_s} \\ \sum_i \bar\beta_i = 1}} \Big(f_s(\alpha) - \sum_i f_s(i)\bar\beta(i) + \sum_{t \in \mathcal{N}(s)} \min_{x_t \in \mathcal{L}_t} [f_{st}(\alpha, x_t) - \sum_i f_{st}(i, x_t)\bar\beta(i)] \Big) \qquad (70) \\
&= \text{DEE2}\,,
\end{aligned}$$

where the equality is because minimization over $x_{\mathcal{N}(s)}$ decouples. If for any $\bar\beta \in [0,1]^{\mathcal{L}_s}$ such that $\sum_i \bar\beta_i = 1$, the inner bracket of (70) is positive, then condition (67) is satisfied. Therefore (s, α) can be eliminated. This sufficient condition is known as Goldstein's *general* DEE (Goldstein 1994). See example in Figure 10(a). The optimal choice of $\bar\beta$, namely solving the maximization in (70) was addressed by Lasters et al. (1995). It can be seen that the objective in (70) is a piecewise linear concave function in $\bar\beta$ and (70) can be written as a linear program[2]. We make the next simplifying step by constraining $\bar\beta$ to be 0-1 again and obtaining

$$\text{DEE2} \geq \max_{\beta \in \mathcal{L}_s} \Big(f_s(\alpha) - f_s(\beta) + \sum_{t \in \mathcal{N}(s)} \min_{x_t \in \mathcal{L}_t} [f_{st}(\alpha, x_t) - f_{st}(\beta, x_t)] \Big) = \text{DEE3}\,. \tag{71}$$

[2]Lasters et al. (1995) applied a cutting-plane method to this problem to gain additional speed-up. However, their experiments demonstrated that this criterion provided a relatively small improvement over a simpler singles criterion (71).

The condition DEE3 > 0 is known as Goldstein's *simple* DEE (Goldstein 1994). It is easily interpreted as

$$(\exists \beta)\ (\forall x \in \mathcal{L}_{\mathcal{N}(s)}) \quad f_s(\alpha) - f_s(\beta) + \sum_{t \in \mathcal{N}(s)} [f_{st}(\alpha, x_t) - f_{st}(\beta, x_t)] > 0\,, \tag{72}$$

which means that a single improving switch from α to β exists for an arbitrary labelling of the neighbors of s. Thus, it is a very intuitive condition, which can be proposed directly. Let us continue with the chain of inequalities. Let us choose a fixed $\beta \in \mathcal{L}_s$, then

$$\text{DEE3} \geq f_s(\alpha) - f_s(\beta) + \sum_{t \in \mathcal{N}(s)} \big[\min_{x_t \in \mathcal{L}_t} f_{st}(\alpha, x_t) + \min_{x_t \in \mathcal{L}_t} (-f_{st}(\beta, x_t))\big] \tag{73}$$

$$= f_s(\alpha) - f_s(\beta) + \sum_{t \in \mathcal{N}(s)} \big[\min_{x_t \in \mathcal{L}_t} f_{st}(\alpha, x_t) - \max_{x_t \in \mathcal{L}_t} f_{st}(\beta, x_t)\big] = \text{DEE4}\,. \tag{74}$$

The sufficient condition DEE4 > 0 allowing to eliminate (s, α) is the original DEE theorem by Desmet et al. (1992). Though it is the weakest one of the above conditions, it has the lowest computational complexity.

Extensions of the DEE method include sufficient conditions to eliminate pairs of labels (Desmet et al. 1992), fixing a part of variables and evaluating the sufficient conditions on the restricted energy (*split* DEE (Pierce et al. 2000)), incorporating the DEE method into a branch and bound search (Georgiev et al. 2006) and other.

The unified class of sufficient conditions we analyze will include the general Goldstein DEE for single labels (equation (70)) and thus also all weaker DEE criteria for single labels. We will not cover the extensions of DEE, only remark that they could be potentially related to higher-order relaxations.

3.2 QPBO

In this section we review weak and strong persistency of QPBO. We give a simple proof of strong persistency relying on the properties of the arc-consistent equivalent.

Theorem 12 (*Weak persistency*, Hammer et al. 1984)**.** Let μ be any optimal relaxed labeling for binary energy E_f. Let $O_s = \{i \in \mathbb{B} \,|\, \mu_s(i) > 0\}$. Then

$$(\exists x \in \operatorname*{argmin}_x E_f(x))\ (\forall s \in \mathcal{V})\ x_s \in O_s\,. \tag{75}$$

Clearly, if μ_s is integer then O_s contains only one label and there exists an optimal labeling taking that label. If, on the other hand, μ_s is not integer, O_s is necessarily $\{0, 1\}$ and x_s can be any.

The following property, called *strong persistency*, was also shown for the roof dual relaxation (Hammer et al. 1984). If a variable takes the same binary value in *all* optimal solutions to the relaxation then it realizes that binary value in *all* optimal integer solutions. We reformulate this result in terms of the dual optimal solution.

Theorem 13 (*Strong persistency*, Hammer et al. 1984)**.** Let f^φ be a (non-unique) arc-consistent equivalent of binary energy E_f. Let $O_s = \operatorname{argmin}_i f^\varphi_s(i)$. Then

$$(\forall x \in \operatorname*{argmin}_x E_f(x))\ (\forall s \in \mathcal{V})\ x_s \in O_s\,. \tag{76}$$

Proof. Assume for contradiction that x is an optimal labeling and $x_p \notin O_p$ for some $p \in \mathcal{V}$. We will show that the labeling y defined by

$$y_s = \begin{cases} x_s & \text{if } x_s \in O_s\,, \\ \neg x_s & \text{if } x_s \notin O_s \end{cases} \tag{77}$$

attains a strictly lower energy, which contradicts optimality of x. By construction, we have $(\forall s)\ y_s \in O_s$, therefore for every unary term of f^φ the inequality holds

$$f^\varphi_s(y_s) \leq f^\varphi_s(x_s) \tag{78}$$

and at least one of these inequalities is strict, namely the one for pixel p. Let us show that similar component-wise inequalities hold for pairwise terms as well. Let $O_{st} = \operatorname{argmin} ij f^\varphi_{st}(i,j)$. Consider the following cases:

- $|O_s| = 1$, $|O_t| = 1$. By arc-consistency, there is only one locally minimal arc in O_{st}, the arc (y_s, y_t). Therefore, $f^\varphi_{st}(y_{st}) \leq f^\varphi_{st}(x_{st})$.
- $|O_s| = 1$, $|O_t| = 2$. By arc-consistency, both arcs $(y_s, 0)$ and $(y_s, 1)$ are locally minimal (both belong to O_{st}). Therefore, $y_{st} \in O_{st}$ and $f^\varphi_{st}(y_{st}) \leq f^\varphi_{st}(x_{st})$.
- $|O_s| = 2$, $|O_t| = 2$. In this case $y_{st} = x_{st}$ and hence $f^\varphi_{st}(y_{st}) = f^\varphi_{st}(x_{st})$.

Therefore, for every $st \in \mathcal{E}$ we have

$$f^\varphi_{st}(y_{st}) \leq f^\varphi_{st}(x_{st})\,. \tag{79}$$

Summing unary inequalities (78) over $s \in \mathcal{V}$ and pairwise inequalities (79) over $st \in \mathcal{E}$ and adding f^φ_0 to both sides, we obtain

$$E_f(y) = E_{f^\varphi}(y) < E_{f^\varphi}(x) = E_f(x)\,, \tag{80}$$

which contradicts the optimality of x. □

Clearly, if $|O_s| = 1$, by complementary slackness, it must be that $\{i \,|\, \mu_s(i) > 0\} = O_s$ for any optimal μ. On the other hand, a primal solution μ can be constructed for φ such that $\{i \,|\, \mu_s(i) > 0\} = O_s$. By this argument, sets O_s in Theorem 13 can be alternatively defined as $O_s = \{i \,|\, (\exists \mu)\ \mu_s(i) > 0,\ \mu \text{ is optimal}\}$. It follows that if μ_s is the same integer assignment in *all* optimal relaxed solutions, then this assignment is taken by *all* optimal labelings x.

It is easy to see that constraints of the form $z_u \in O_u$ may be expressed as:

$$z^{\min} \leq z \leq z^{\max} \tag{81}$$

by letting $(\forall u \in V)\ z^{\min}_u = \min O_u$ and $z^{\max}_u = \max O_u$. For example, $0 \leq z_u \leq 1$ does not restrict z_u, while $0 \leq z_u \leq 0$ asserts that $z_u = 0$ for any minimizer z. It is also clear that the explicitly constructed improved labeling (77) can be written in the form $y = (z \vee z^{\min}) \wedge z^{\max}$. Thus we in fact proved that the pair $(z^{\min}, z^{\max})$ computed by QPBO is a strong autarky.

3.3 MQPBO

Consider a multi-label energy E_f and its binary reduction E_g according to the construction in §2.4. We can obtain partial optimality guarantees for the binary problem E_g via QPBO and reinterpret them for the initial multi-label problem E_f (Kohli et al. 2008). The next theorem shows how a strong autarky $(z^{\min}, z^{\max})$ for the binary energy E_g is interpreted as a strong autarky $(x^{\min}, x^{\max})$ for the original multi-label energy E_f. This interpretation is illustrated in Figure 5.

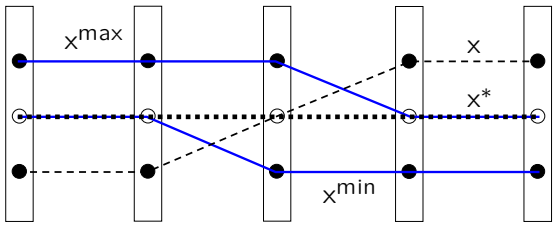


Figure 5 A strong autarky $(x^{\min}, x^{\max})$: if a labeling x (thin dashed) does not satisfy constraints $x^{\min} \leq x \leq x^{\max}$ then labeling $x^* = (x \vee x^{\min}) \wedge x^{\max}$ (thick dashed) does satisfy them and it has a lower energy.

Theorem 14. Let E_g be the binarized problem for E_f and $(z^{\min}, z^{\max})$ a strong autarky for E_g. Let $x^{\min} = x(z^{\min})$, $x^{\max} = x(z^{\max})$ by the mapping (42b). Then

$$\big(\forall x \neq (x \vee x^{\min}) \wedge x^{\max}\big) \;\; E_f((x \vee x^{\min}) \wedge x^{\max}) < E_f(x)\,. \tag{82}$$

Proof. Let $x \in \mathcal{L}$ and $z = z(x)$ as defined by mapping (42a). We will show that $(x \vee x^{\min}) \wedge x^{\max} = x((z \vee z^{\min}) \wedge z^{\max})$. The theorem will follow by the equivalence of E_f and E_g. Let us show that for any $z, z' \in \mathbb{B}^{\mathcal{V}\times\tilde{\mathcal{L}}}$ such that $z_s(i) \geq z_{s,i+1}$ and $z'_s(i) \geq z'_{s,i+1}$ it is $x(z \vee z') = x(z) \vee x(z')$. Indeed

$$x(z \vee z')_s = \sum_{i\in\tilde{\mathcal{L}}} \max(z_s(i), z'_s(i)) = \sum_{i\in\tilde{\mathcal{L}}} \Big([\![i{<}x(z)_s]\!] \vee [\![i{<}x(z')_s]\!] \Big) = \max(x(z)_s, x(z')_s)\,. \tag{83}$$

Similarly, we can verify that $x(z \wedge z') = x(z) \wedge x(z')$. □

The MQPBO method finds a strong autarky $(x^{\min}, x^{\max})$ using Theorem 14 and the strong autarky $(z^{\min}, z^{\max})$ found by QPBO for the binarized problem.

3.4 Autarkies for Submodular Problems

Let E_f be submodular and x^* be its minimizer. Then we have the following properties:

$$E_f(x \vee x^*) \leq E_f(x)\,, \tag{84a}$$
$$E_f(x \wedge x^*) \leq E_f(x)\,. \tag{84b}$$

These properties easily follow from submodularity, noting that $E_f(x \vee x^*) \geq E_f(x^*)$ and $E_f(x \vee x^*) \geq E_f(x^*)$. It follows that any pair of optimal solutions (x^{1*}, x^{2*}) is a weak autarky for this problem since $E_f((x \vee x^{1*}) \wedge x^{2*}) \leq E_f(x \vee x^{1*}) \leq E_f(x)$. Moreover, if we choose the *lowest* and the *highest* minimizers,

$$x^{\min} = \bigwedge \operatorname*{argmin}_x E_f(x)\,, \tag{85a}$$
$$x^{\max} = \bigvee \operatorname*{argmin}_x E_f(x)\,, \tag{85b}$$

then the autarky $(x^{\min}, x^{\max})$ is strong. This strong autarky can be determined from the solution of the corresponding MAXFLOW problem.

3.5 Auxiliary Submodular Problems

In the techniques (Kovtun 2003, 2004, 2011), a labeling x^* is found such that the mapping $x \mapsto x \vee x^*$ is improving for E_f. Since $x \vee x^* \geq x^*$, this guarantees the existence of a minimizer x^{**} satisfying $x^{**} \geq x^*$. This means that all partial assignments (s, x_s) such that $x_s < x^*_s$ can be eliminated as non-optimal. It can be seen that the improving mapping $x \mapsto x \vee x^*$ is a special case of an autarky, namely the autarky $(x^*, x^{\max})$ with $x^{\max}$ set to the maximum labeling.

According to the above §3.4, finding an improving mapping of this form for a submodular function E_g is easy. It is sufficient to calculate a minimizer of E_g or the "lowest" minimizer if we want to obtain strong domain constraints. To find such improving mapping for a non-submodular E_f, Kovtun (2003) proposed to construct an auxiliary energy E_g which would be submodular and have the property that its improving mappings are guaranteed to be improving also for E_f.

We will review the general method, constructing an auxiliary problem incrementally and two special methods, constructing binary auxiliary submodular problems, which are more useful in practice. For the second special case, we will later derive an optimal method to compare with.

Definition 7 (Kovtun 2011)**.** Let p denote the mapping $x \mapsto x \vee x^*$ for some x^*. The function E_g is called *auxiliary* for E_f if and p if

$$(\forall x \in \mathcal{L}) \quad E_f(p(x)) - E_f(x) \leq E_g(p(x)) - E_g(x)\,, \tag{86}$$

that is, the improvement by p in E_f is at least as big as the improvement in E_g.

If p is improving for E_g then the RHS is non-positive and so is the LHS. It follows that p is improving for E_f. However, such x^* that p is improving for E_g is not known in advance, at the point of constructing E_g. The idea is to require that (86) is satisfied for a family of mappings $\{x \mapsto x \vee x^* \,|\, x^* \in K\}$, where $K = \prod_s K_s$ and $K_s \subset \mathcal{L}_s$ are some subsets of labels. The condition (86) for all mappings in this family is replaced by the following simpler component-wise inequalities:

$$\begin{array}{cc} (\forall s \in \mathcal{V})\ (\forall i \in \mathcal{L}_s)\ (\forall i' \in K_s) & (f-g)_s(i \vee i') \leq (f-g)_s(i)\,, \\ (\forall st \in \mathcal{E})\ (\forall ij \in \mathcal{L}_{st})\ (\forall i'j' \in K_{st}) & (f-g)_{st}(i \vee i', j \vee j') \leq (f-g)_{st}(i,j)\,. \end{array} \tag{87}$$

It can be seen that the inequalities (87) imply (86) for arbitrary $x^* \in K$. They form a looser sufficient condition. Having constructed auxiliary problem E_g, it remains to find $x^* \in K$ such that the mapping $x \mapsto x \vee x^*$ is improving for E_g. As discussed in §3.4, if x^* is a minimizer (resp. the highest minimizer) of E_g, then the mapping $x \mapsto x \vee x^*$ is improving (resp. strictly improving) for E_g. It remains to make sure that the minimizer x^* is in K.

3.5.1 Iterative Algorithm

The iterative algorithm to construct E_g and $(K_s \,|\, s \in \mathcal{V})$ proposed by Kovtun (2004, 2011) is shown in Algorithm 3.

In step 3, for each edge $st \in \mathcal{E}$ a system of linear inequalities in g_{st} has to be solved. While (Kovtun 2004) provides an explicit solution, it will not be necessary for our consideration. If the condition $x^* \in K$ is not satisfied, the set K is enlarged. When the algorithm terminates, $x \mapsto x \vee x^*$ is strictly improving for E_f. It may stop, however, with $x^*_s = 0$ for all s, so that efficiently no constraints are derived. This algorithm runs

Algorithm 3: Iterative Construction of Auxiliary Problem (Kovtun 2004, 2011)

Set $K_s := \emptyset$, $s \in \mathcal{V}$;
repeat
  Construct a submodular g satisfying inequalities (87);
  Find $x^* = \bigwedge \operatorname*{argmin}_x E_g(x)$;
  if $x^*_s \in K_s$ *for all* $s \in \mathcal{V}$ **then**
    return x^*
  $K_s := K_s \cup \{x^*_s\} \quad \forall s \in \mathcal{V}$;
until;

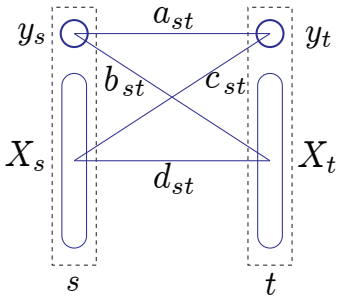


Figure 6 Illustration of the auxiliary one-against-all function g. The chosen label y competes with all other labels in every pixel. If y wins then it is guaranteed to be a part of any optimal assignment.

in polynomial time since minimization of E_g is polynomial and the algorithm performs no more than $|\mathcal{V}|$ iterations.

3.5.2 One-against-all Auxiliary Subproblem

A simpler non-iterative method proposed by Kovtun (2003) is as follows. Let y be a fixed proposal labeling. The algorithm attempts to identify pixels s where the label (s, y_s) is a strongly optimal partial assignment. The construction is motivated by the expansion move algorithm. Let us reorder labels in each pixel such that y_s becomes the highest label.

The problem E_g is constrained to have a form such that all unary and pairwise costs associated with labels from the set $X_s := \mathcal{L}_s \backslash \{y_s\}$ are equal (see Figure 6). More precisely, we assume that for all s the weights $g_s(X_s) = (g_s(i) \,|\, i \in X_s)$ are equal to the same value, denoted with some abuse of notation as $g_s(X_s)$. Similarly, for the pairwise terms we assume that

$$\begin{aligned} g_{st}(y_s, y_t) &= a_{st}\,, \\ g_{st}(y_s, X_t) &= b_{st}\,, \\ g_{st}(X_s, y_t) &= c_{st}\,, \\ g_{st}(X_s, X_t) &= d_{st}\,, \end{aligned} \tag{88}$$

where $a_{st}, b_{st}, c_{st}, d_{st} \in \mathbb{R}$. The problem E_g is thus defined by the weights $g_{s'}(y_{s'})$, $g_{s'}(X_{s'})$, a_{st}, b_{st}, c_{st}, d_{st} for s' ranging in $\mathcal{V}$ and st in $\mathcal{E}$.

The set K_s is chosen to be $\{0, y_s\}$. This corresponds to a family $\mathcal{P}$ of mappings p which either switch to label y_s or retain the current label in every pixel. At the same

time, for E_g in the restricted form, it is clear that its lowest minimizer takes either label 0 or label y_s. To be an auxiliary function for E_f w.r.t. any mapping from $\mathcal{P}$, g has to satisfy

$$\begin{aligned}
(\forall x_s \neq y_s) \qquad & g_s(X_s) - g_s(y_s) \leq f_s(x_s) - f_s(y_s)\,, \\
(\forall x_t \neq y_t) \qquad & b_{st} - a_{st} \leq f_{st}(y_s, x_t) - f_{st}(y_s, y_t)\,, \\
(\forall x_s \neq y_s) \qquad & c_{st} - a_{st} \leq f_{st}(x_s, y_t) - f_{st}(y_s, y_t)\,, \\
(\forall x_s \neq y_s)\ (\forall x_t \neq y_t) \qquad & d_{st} - a_{st} \leq f_{st}(x_s, x_t) - f_{st}(y_s, y_t)\,, \\
(\forall x_s \neq y_s)\ (\forall x_t \neq y_t) \qquad & d_{st} - b_{st} \leq f_{st}(x_s, x_t) - f_{st}(y_s, x_t)\,, \\
(\forall x_s \neq y_s)\ (\forall x_t \neq y_t) \qquad & d_{st} - c_{st} \leq f_{st}(x_s, x_t) - f_{st}(x_s, y_t)\,.
\end{aligned} \tag{89}$$

One of the solutions, enforcing also submodularity of E_g, is the following:

$$\begin{aligned}
& g_s(y_s) = f_s(y_s) - \min_{i \neq y_s} f_s(i)\,, \\
& a_{st} = f_{st}(y_s, y_t)\,, \quad b_{st} = \min_{j \neq y_t} f_{st}(y_s, j)\,, \quad c_{st} = \min_{i \neq y_s} f_{st}(i, y_t)\,, \\
& d_{st} = \min \Big\{ b_{st} + c_{st} - a_{st}, \\
& \qquad \min_{i \neq y_s, j \neq y_t} \Big[f_{st}(i,j) + \min\left\{ b_{st} - f_{st}(y_s, j), c_{st} - f_{st}(i, y_t) \right\} \Big] \Big\}\,.
\end{aligned} \tag{90}$$

The constructed auxiliary problem E_g has the property that its lowest minimizer $x^{\min} = \bigwedge \operatorname{argmin} E_g(x)$ is guaranteed to satisfy $(\forall s \in \mathcal{V})\ x^{\min} \in K_s$. Therefore $x' \mapsto z' \vee x^{\min}$ is improving for E_g.

3.5.3 Many-against-many Auxiliary Subproblem

The general construction proposed by Kovtun (2004, 2011) considers an arbitrary partition of the set of labels $\mathcal{L}_s$ into two subsets X_s and Y_s in each pixel. The labels are re-ordered such that $Y_s > X_s$ and $y_s = \min Y_s$. The labeling $x^{\min}$ is constrained to $K_s = \{0, y_s\}$ in every pixel. If $x' \mapsto x' \vee x^{\min}$ is improving, all labels (s, i) such that $i \in X_s$ and $x^{\min}_s = y_s$ can be eliminated. A binary auxiliary submodular problem is then constructed satisfying (87). The one-against-all construction is obtained as a special case by letting $Y_s = \{y_s\}$. That is, the set of potentially eliminated labels X_s contains all labels but y_s ($X_s = \mathcal{L}_s \backslash \{y_s\}$). We discuss the opposite special case below, when the set X_s contains a single candidate elimination label in every pixel s. This second case is interesting because of the family of mappings it considers. We will be able to derive the optimal method for this family and hence to show a strict improvement.

3.5.4 All-against-one Auxiliary Subproblem

In this section, we study a special case of the many-against-many construction, not explicitly considered by Kovtun (2011). The all-against-one auxiliary subproblem attempts to eliminate a single label in every pixel (as opposed to eliminating all but one labels). The number of nodes which can be eliminated is lower should the method succeed, however the sufficient conditions used in this case are tighter (there are fewer inequalities that g must fulfill).

Let y, z be fixed labellings and $\mathcal{A} \subset \mathcal{V}$. Let the mapping $p_{\mathcal{A}} \colon \mathcal{L} \to \mathcal{L}$ be defined pixel-wise as

$$\big(p_{\mathcal{A}}(x)\big)_s = \begin{cases} y_s & \text{if } x_s = z_s \text{ and } s \in \mathcal{A}\,, \\ x_s & \text{otherwise}\,. \end{cases} \tag{91}$$

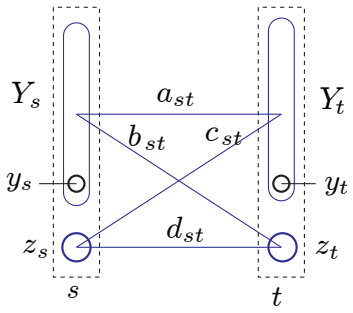


Figure 7 Illustration of the auxiliary all-against-one function E_g. The goal is to find pixels where the label z can be eliminated by replacing it with label y.

That is, mapping $p_{\mathcal{A}}$ replaces labels z with labels y for all pixels in $\mathcal{A}$. If $p_{\mathcal{A}}$ is improving, we can eliminate all nodes $((s, z_s) \,|\, s \in \mathcal{A},\ y_s \neq z_s)$ as non-optimal. For fixed labellings y, z, we are interested in finding the largest subset $\mathcal{A}$ such that the mapping $p_{\mathcal{A}}$ is improving. This mapping can be represented in the form $x \mapsto x \vee y'$ by ordering the labels such that $(\forall s)\ z_s < y_s = \min Y_s$ and letting $y' \in K_s = \{z_s, y_s\}$.

Following Kovtun (2011), we will construct an auxiliary problem E_g for the family of mappings $\{x \mapsto x \vee y' \,|\, y'_s \in K_s,\ \forall s \in \mathcal{V}\}$. As before, we restrict E_g in such a way that all labels $Y_s := \mathcal{L}_s \backslash \{z_s\}$ have the same associated unary and pairwise costs:

$$\begin{aligned} g_s(Y_s) &= g_s(y_s)\,, \\ g_{st}(Y_s, Y_t) &= a_{st}\,, \\ g_{st}(Y_s, z_t) &= b_{st}\,, \\ g_{st}(z_s, Y_t) &= c_{st}\,, \\ g_{st}(z_s, z_t) &= d_{st}\,. \end{aligned} \tag{92}$$

In this case E_g will be equivalent to a problem with two labels in each pixel s, deciding between z_s and y_s. The auxiliary property of E_g is ensured by the component-wise inequalities (87). We first consider pairs st, such that $y_s \neq z_s$. The required inequalities for such pairs are

$$\begin{aligned} g_s(z_s) - g_s(y_s) &\leq f_s(z_s) - f_s(y_s)\,, \\ c_{st} - a_{st} &\leq \min_{j \neq z_t} \left(f_{st}(z_s, j) - f_{st}(y_s, j)\right) =: \Delta_{st}\,, \\ b_{st} - a_{st} &\leq \min_{i \neq z_s} \left(f_{st}(i, z_t) - f_{st}(i, y_t)\right) = \Delta_{ts}\,, \\ d_{st} - a_{st} &\leq f_{st}(z_s, z_t) - f_{st}(y_s, y_t)\,, \\ d_{st} - b_{st} &\leq f_{st}(z_s, z_t) - f_{st}(y_s, z_t)\,, \\ d_{st} - c_{st} &\leq f_{st}(z_s, z_t) - f_{st}(z_s, y_t)\,. \end{aligned} \tag{93}$$

The last three inequalities account for possible mappings $p_{\mathcal{A}}$ depending on whether $\mathcal{A}$ includes s, t, or both. A solution to these inequalities and submodularity constraints can be obtained as

$$\begin{aligned} &g_s(z_s) = f_s(z_s) - f_s(y_s)\,, \quad g_s(y_s) = 0\,, \\ &a_{st} = 0\,, \quad b_{st} = \min_{i \neq z_s} \left(f_{st}(i, z_t) - f_{st}(i, y_t)\right), \quad c_{st} = \min_{j \neq z_t} \left(f_{st}(z_s, j) - f_{st}(y_s, j)\right), \\ &d_{st} = \min\Big(b_{st} + c_{st},\ f_{st}(z_s, z_t) + \min\left\{b_{st} - f_{st}(y_s, z_t), c_{st} - f_{st}(z_s, y_t)\right\}\Big)\,. \end{aligned} \tag{94}$$

In the case $y_s = z_s$, the label of pixel s should be fixed to 0 and hence s can be excluded from the problem. We have fewer inequalities in this case:

$$\begin{aligned} &b_{st} - a_{st} \leq \Delta_{ts}\,, \\ &d_{st} - c_{st} \leq f_{st}(z_s, z_t) - f_{st}(z_s, y_t)\,, \quad d_{st} \leq b_{st} + c_{st} - a_{st}\,. \end{aligned} \tag{95}$$

And we can choose a solution

$$\begin{aligned} &g_s(0) = g_s(1) = 0\,, \quad a_{st} = 0\,, \quad c_{st} = 0\,, \quad b_{st} = \Delta_{ts}\,, \\ &d_{st} = \min(\Delta_{ts}, f_{st}(z_s, z_t) - f_{st}(z_s, y_t)) = \min_i \left(f_{st}(i, z_t) - f_{st}(i, y_t)\right). \end{aligned} \tag{96}$$

Without loss of maximality (as will be defined and proven in §4.4 devoted to maximum projections), we can exclude pixel s and add the cost d_{st} to $g_t(z_t)$. It can be seen that this construction is equivalent to taking the worst case estimate for pixel s independently for all its neighbors $t \in \mathcal{N}(s) \cap \mathcal{A}$.

The methods reviewed above were proposed as sufficient conditions by Kovtun (2011). They do not pretend to be the best in any sense. In the sequel, we will show that replacing the labeling-wise inequalities (86) with component-wise inequalities (87) can be done without loss of generality and the submodular truncation is an optimal design step in the case of all-against-one auxiliary subproblem. However, the other design steps: 1) requiring that E_g is equivalent to a two-label problem and 2) requiring that E_g is auxiliary for E_f for all possible members of the considered family of mappings, are not optimal, leading to the final sufficient condition being over-constrained. In §4.4.3, we derive an optimal method for all-against-one problem, overcoming this limitation.

4 Unified Framework for Partial Optimality

In this chapter, we show that the different sufficient conditions used in the methods of DEE, (M)QPBO, and auxiliary submodular problems can be unified into a general class. We study the common properties of the unified class, and show the following. All the above methods can be derived as local sufficient conditions in a specially constructed reparametrization of the problem. Furthermore, they can be related to the standard LP relaxation. It is guaranteed that the found strong partial optimalities retain all optimal solutions of the LP relaxation, in other words, solutions of the LP relaxation automatically satisfy the derived partial optimalities. Therefore, LP relaxation cannot be tightened by these methods. A similar result holds for fixed points of the fusion move algorithm: they satisfy strong optimalities of a specific subclass.

In §4.1 we introduce our unified tool for reduction of the discrete search space, called a *projection*. It is a linear map of the set of relaxed labellings to itself that does not increase the energy and shrinks the possible selection of the labels. We give examples of projections corresponding to the individual methods and prove that the considered methods (DEE, QPBO, M-QPBO and auxiliary submodular problems) are all indeed special cases of our sufficient condition. We study general properties of projections and their relation to the LP relaxation. In §4.3, we prove the characterization of the sufficient condition as a simple local condition up to equivalent transformations. This reveals the principle of the method and offers a powerful tool for further analysis. In §4.4, we address the question of what the maximal projections (that provide maximal partial assignments) for a given problem are. While this is a difficult question in general, we give polynomial-time algorithms to find maximal projections in restricted subclasses. Our new algorithm, applicable to general energy functions, similarly to previous methods provides sufficient conditions for a partial optimal assignment. However, it is optimal for projections from a certain non-trivial class.

4.1 Projections

We observed in Chapter 3 that the optimality guarantees of the reviewed methods are obtained through the following mechanism. An improving mapping p of labelings is constructed. Given an arbitrary labeling x, p provides a labeling that has a better (or at least not worse) energy. In the case of Desmet's DEE, such a mapping changes the label in a single pixel s, replacing label α to label β in s. In the case of QPBO and MQPBO the mapping is of the form $x \mapsto (x \vee x^{\min}) \wedge x^{\max}$. In the case of auxiliary submodular problems, after a certain reordering of the labels, the mapping is of the form $x \mapsto x \vee x^{\min}$. As soon as mapping p is improving in the sense that $E_f(p(x)) \leq E_f(x)$, we know for sure that there exists an optimal labeling in $p(\mathcal{L})$. The considered mappings are such that $p(\mathcal{L})$ is expressed as pixel-wise domain constraints, eliminating part of the labels in each pixel. The following considerations lead us to a linearization of this construction and considering improving mappings of *relaxed* labelings instead.

The improving mappings of (M)QPBO and auxiliary problems may change labels of many pixels at a time and hence they are more general than the pixel-wise simple DEE mapping. At the same time, we note that general Goldstein's DEE condition is

more general in a different sense. It is based on a mapping of a label α to a convex combination of other labels rather than to a single label. This suggests the idea of considering improving mappings that map labelings onto the marginal polytope.

Even if a discrete mapping p is given, verification of the improving property is generally NP-hard. Thus we need a simpler sufficient condition to prove the improving property. For (M)QPBO and auxiliary problems we have seen that the sufficient conditions employed are in the form of component-wise inequalities on the energy vector f or its arc-consistent equivalent f^{φ}. These sufficient conditions ensure that p is improving for every unary and every pairwise term separately, which is easy to verify. We propose to extend the mapping p to relaxed labelings and demand the improving property on the local polytope Λ. By doing so, on the one hand we introduce a polynomially verifiable sufficient condition, and on the other hand we are able to show that this condition includes sufficient conditions of (M)QPBO, auxiliary problem and general Goldstein's DEE as special cases. Furthermore, we prove the following characterization: the mapping is Λ-improving iff there exists an equivalent transformation of the problem such that the simple component-wise sufficient conditions hold.

With such a linearized mapping we can establish a relation to the LP relaxation and the fusion move algorithm. Knowing that verification of the Λ-improving property is polynomially solvable (via LP), we pose the question how we can find such mappings that are maximal in the sense that they would allow to eliminate the maximal number of labels. There are two challenges. Firstly, even solving the verification LP is computationally expensive, so we would like to restrict ourselves to a family of mappings where this verification is even easier. Secondly, the Λ-improving property is considered now as a constraint and we wish to optimize over such mappings. In this challenging direction we analyze several restricted families of mappings.

At the beginning we are going to be more general and consider arbitrary mappings of Λ to itself and then we restrict ourselves to pixel-wise mappings which can be constructively defined. As we discussed, given a discrete improving mapping p, the search for the minimizer can be performed over $x \in p(\mathcal{L})$. The mappings of (M)QPBO, auxiliary problems and DEE are idempotent (satisfy $p(p(x)) = p(x)$). This is a natural property to keep in the generalization. It means that whenever a labeling was improved, the resulting labeling $p(x)$ cannot be improved further by p. On vector spaces, idempotent mappings are known as projections, so we borrow this term. There is a technical problem in that $\operatorname{aff}(\Lambda)$ is an affine space and not a vector space. We therefore define a linear map of $R^{\mathcal{I}}$ to itself and then consider its restriction to Λ.

Definition 8. A linear map $P\colon \mathbb{R}^{\mathcal{I}} \to \mathbb{R}^{\mathcal{I}}$ is called a *projection* if

$$P^2 = P\,, \tag{97a}$$

$$P(\mathcal{M}) \subset \mathcal{M}\,, \tag{97b}$$

$$P(\Lambda) \subset \Lambda\,. \tag{97c}$$

The first condition is the usual property of geometrical projections, idempotency of the map $\mu \mapsto P\mu$. The second condition requires that the marginal polytope $\mathcal{M}$ is closed under P. In particular, integer labellings have to be mapped to a convex combination of integer labellings at most. The third condition requires that Λ is also closed under P, *i.e.*, relaxed labelings are mapped to relaxed labelings.

Definition 9. A projection P is $\mathcal{M}$-*improving* for E_f if

$$(\forall \mu \in \mathcal{M}) \quad \langle f, P\mu \rangle \leq \langle f, \mu \rangle\,, \tag{98}$$

and *strictly $\mathcal{M}$-improving* if the inequality is strict for any $\mu \in \mathcal{M}$ such that $P\mu \neq \mu$.

Note, the condition (98) is equivalent to $\langle f, P\delta(x)\rangle \leq \langle f, \delta(x)\rangle$ for all $x \in \mathcal{L}$ or to $E_{P^\top f}(x) \leq E_f(x)$ for all $x \in \mathcal{L}$. Improving projections allow us to simplify the energy minimization problem as follows.

Theorem 15. Let

$$O = \operatorname*{argmin}_{\mu\in\mathcal{M}} \langle f, \mu\rangle\,, \qquad O' = \operatorname*{argmin}_{\mu\in P(\mathcal{M})} \langle f, \mu\rangle\,. \tag{99}$$

If P is $\mathcal{M}$-improving, then
1. $O' \subset O$;
2. $O' = P(O)$.

If P is strictly $\mathcal{M}$-improving then $O' = O = P(O)$.

Proof. From the $\mathcal{M}$-improving property we have $\langle f, P\mu\rangle \leq \langle f, P\mu\rangle$ and $P(\mathcal{M}) \subset \mathcal{M}$. Therefore, claim 1 follows and there holds equality of the minima of the two problems

$$\min_{\mu\in\mathcal{M}} \langle f, \mu\rangle = \min_{\mu\in P(\mathcal{M})} \langle f, \mu\rangle\,. \tag{100}$$

Let us show 2. It is easy to verify that $P(O) \subset O'$. Indeed, let $\mu \in P(O)$, then μ is optimal and $\mu \in P(\mathcal{M})$, therefore $\mu \in O'$. In the other direction, let $\mu \in O'$, *i.e.*, $\mu \in P(\mathcal{M})$ and is optimal. Assume for contradiction that $\mu \notin P(O)$. Then $\mu \in P(\mathcal{M})\backslash P(O) = P(\mathcal{M}\backslash O)$, where the set equality is implied by linearity. This contradicts the optimality of μ. Therefore, it must be $O' \subset P(O)$.

Now, consider a strictly $\mathcal{M}$-improving projection and assume $\mu \in O$ and $\mu \notin P(\mathcal{M})$. Since $P\mu \in P(\mathcal{M})$, we must have $P\mu \neq \mu$. Then the strict inequality holds $\langle f, P\mu\rangle < \langle f, \mu\rangle$ and states that μ is not optimal, which is a contradiction. Therefore, for strictly $\mathcal{M}$-improving projection, the set $P(\mathcal{M})$ retains all optimal marginal labelings, *i.e.*, $O' = O$. □

Note, since integer labelings are represented as a subset of $\mathcal{M}$ and the projection is a linear map, a strictly $\mathcal{M}$-improving projection preserves all optimal integer labelings, *i.e.*, we can write

$$\operatorname*{argmin}_{x\in\mathcal{L}} \langle f, \delta(x)\rangle = \operatorname*{argmin}_{x\in\mathcal{L}} \langle P^\top f, \delta(x)\rangle\,. \tag{101}$$

Example 1. Consider the energy minimization problem with 2 labels shown in Figure 8(a). Let us define $\bar{\mu} = (\mu_1(1),\, \mu_2(1),\, \mu_{12}(1,1),\, 1)$ as a minimal representation of the marginal vector (all other components are linearly dependent due to the constraints of $\mathcal{M}$). Let P be defined via the following linear map of this minimal representation:

$$P = \begin{pmatrix} 0.5 & -0.5 & 0.5 & 0.5 \\ -0.5 & 0.5 & 0.5 & 0.5 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{102}$$

This map projects (non-orthogonally) to a facet of the marginal polytope as shown in Figure 8(b). Assume optimal integer solutions are the two labelings $(0, 1)$ and $(1, 0)$. In that case, the projection P is strictly $\mathcal{M}$-improving. Another projection

$$P' = \begin{pmatrix} 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{103}$$

maps $\mathcal{M}$ to a single line (bold green in Figure 8(b)). It is $\mathcal{M}$-improving, as it retains one optimal point from $\mathcal{M}$, the point $(0.5, 0.5, 0, 1)$, which is a convex combination of the two optimal integer solutions. However, P' it is not strictly $\mathcal{M}$-improving.

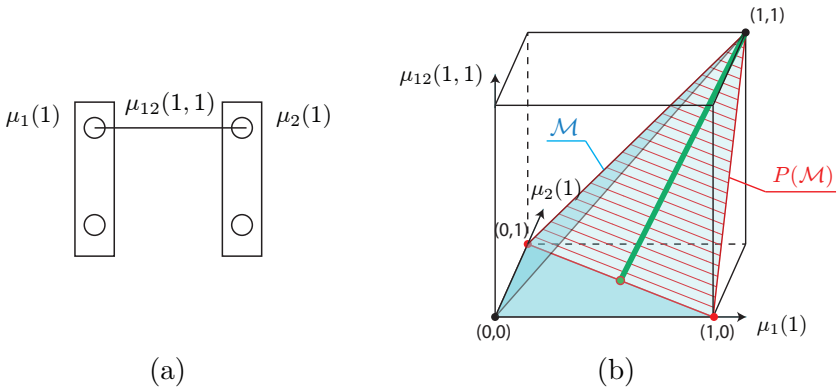


Figure 8 Illustration of the marginal polytope and a projection. (a) An energy minimization problem with two labels. The marginal vector μ is an element of $\mathbb{R}^9$, however, due to marginalization and normalization constraints there are only 3 linearly independent coordinates, *e.g.*, the ones shown in the figure. (b) Marginal polytope $\mathcal{M}$ (cyan) in the space of $\mu_1(1)$, $\mu_2(1)$ and $\mu_{12}(1,1)$. Possible labelings are marked as bold dots in this space. Suppose labelings $(1,0)$ and $(0,1)$ are optimal. Using strictly $\mathcal{M}$-improving projection P (example 1), the search space shrinks to $P(\mathcal{M})$ (the red hashed facet). Using $\mathcal{M}$-improving projection P', the search space shrinks to the bold green line, not containing any optimal integer solutions but containing their convex combination.

Now, we have seen that $\mathcal{M}$-improving projections simplify the problem while preserving optimal solutions. How do we find a projection that is $\mathcal{M}$-improving or at least verify whether a given projection is $\mathcal{M}$-improving? Verifying the $\mathcal{M}$-improving property (98) is equivalent to solving

$$\min_{\mu\in\mathcal{M}} \langle f - P^{\mathsf{T}} f, \mu\rangle \geq 0\,, \tag{104}$$

which can be as hard as the original energy minimization problem. However, we can get a simpler sufficient condition by ensuring the inequality over a larger polytope Λ and verify

$$\min_{\mu\in\Lambda} \langle f - P^{\mathsf{T}} f, \mu\rangle \geq 0\,. \tag{105}$$

This condition, in contrast to (104), is polynomially verifiable.

Definition 10. A projection P is called Λ-*improving* for E_f if it satisfies

$$(\forall \mu \in \Lambda) \quad \langle f, P\mu\rangle \leq \langle f, \mu\rangle\,, \tag{106}$$

and *strictly* Λ-*improving* if the inequality is strict for all $\mu \in \Lambda$ such that $P\mu \neq \mu$.

It will be shown that sufficient conditions in the reviewed partial optimality methods (QPBO, multi-label QPBO, auxiliary submodular problems and DEE singleton criteria) are special cases of Λ-improving projections. Hence, the properties that hold for Λ-improving projections, will be common to all of the mentioned methods. Not less importantly, we can design algorithms that are inbetween of the existing methods in certain sense or more general ones.

4.1.1 Pixel-wise Projections

While the above theoretical results are more general, in practice we restrict ourselves to projections of a simpler form that can be defined constructively. We consider projections

P such that $(P\mu)_s$ depends only on μ_s and $(P\mu)_{st}$ depends only on μ_{st}. To make sure that marginalization constraints are satisfied, the mapping for pairwise terms is induced by the mappings of unary terms. Such projections will be called *pixel-wise*.

Let us define for each $s \in \mathcal{V}$ a matrix $P_s \in \mathbb{R}^{\mathcal{L}_s \times \mathcal{L}_s}$ satisfying

$$\begin{aligned} (\forall ii' \in \mathcal{L}_s \times \mathcal{L}_s) \quad & P_{s,ii'} \in [0,\,1]\,, \\ & 1^\mathsf{T} P_s = 1\,, \\ & P_s P_s = P_s\,. \end{aligned} \tag{107}$$

The pixel-wise projection $P\colon \Lambda \to \Lambda$ is defined as follows,

$$\begin{aligned} (P\mu)_s(i) &= (P_s\mu_s)(i) = \sum_{i' \in \mathcal{L}_s} P_{s,ii'}\mu_s(i')\,, \\ (P\mu)_{st}(i,j) &= (P_s\mu_{st}P_t^\mathsf{T})(i,j) = \sum_{i' \in \mathcal{L}_s} \sum_{j' \in \mathcal{L}_t} P_{s,ii'}P_{t,jj'}\mu_{st}(i',j')\,. \end{aligned} \tag{108}$$

Theorem 16. The mapping P defined by (108) is a projection.

Proof. Idempotency follows from idempotency of individual matrices P_s. We have to show that $P(\Lambda) \subset \Lambda$ and $P(\mathcal{M}) \subset \mathcal{M}$.

Let us first show $P(\Lambda) \subset \Lambda$. To verify that $P\mu$ satisfies the normalization constraints (15), check that

$$(\forall s \in \mathcal{V}) \quad \sum_{i \in \mathcal{L}_s} (P\mu)_s(i) = \sum_{i \in \mathcal{L}_s} \sum_{i' \in \mathcal{L}_s} P_{s,ii'}\mu_s(i') = \sum_{i' \in \mathcal{L}_s} \mu_s(i') = 1\,. \tag{109}$$

The marginalization constraints (14) are verified as follows:

$$(\forall st \in \mathcal{E}) \quad 1^\mathsf{T}(P\mu)_{st} = 1^\mathsf{T} P_s \mu_{st} P_t^\mathsf{T} = 1^\mathsf{T}\mu_{st}P_t^\mathsf{T} = \mu_t^\mathsf{T} P_t^\mathsf{T} = (P\mu)_t^\mathsf{T}\,. \tag{110}$$

Let us now prove $P(\mathcal{M}) \subset \mathcal{M}$. Because P is linear and $\mathcal{M}$ is the convex hull of integer labellings, it is sufficient to show that for any integer relaxed labeling $\mu \in \mathcal{M}$ there holds $\nu := P\mu \in \mathcal{M}$. By construction, we have

$$\begin{aligned} \nu &\geq 0\,, \\ 1^\mathsf{T}\nu_s &= 1\,, \\ 1^\mathsf{T}\nu_{st} &= \nu_t^\mathsf{T}\,, \\ \nu_{st} &= \nu_s\nu_t^\mathsf{T}\,. \end{aligned} \tag{111}$$

Let us choose a vertex s such that ν_s is not integer. We can expand ν as

$$\nu = \sum_{i \in \mathcal{L}_s} \nu_s(i)\nu^{(i)}\,, \tag{112}$$

where $\nu^{(i)} \in \Lambda$ is defined as

$$\begin{aligned} (\forall s' \in \mathcal{V}) \qquad \nu_{s'}^{(i)}(i') &= \begin{cases} [\![i{=}i']\!] & \text{if } s' = s\,, \\ \nu_{s'}(i') & \text{otherwise}\,, \end{cases} \\ (\forall st \in \mathcal{E}) \qquad \nu_{st}^{(i)} &= \nu_s^{(i)}(\nu_t^{(i)})^\mathsf{T}\,. \end{aligned} \tag{113}$$

This construction is illustrated in Figure 9. It can be verified that $\nu^{(i)}$ satisfies the same constraints as ν. Expansion (112) is straightforward to verify for singletons and pairs

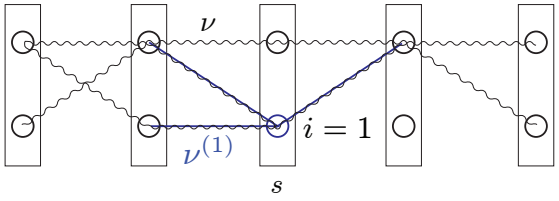


Figure 9 Illustration of the recursive construction to disassemble a relaxed labeling $\nu = P\mu$ (wave line) into a convex combination of integer labelings in order to prove $P(\mathcal{M}) \subset \mathcal{M}$. We expand ν as a combination of $|\mathcal{L}_s|$ relaxed labelings $(\nu^{(i)} \,|\, i \in \mathcal{L}_s)$ that are integer at s. Here, $v^{(1)}$ is shown by solid lines around s.

$s't'$ such that $s' \neq s$ and $t' \neq s$. To verify it for pairs $st \in \mathcal{E}$, check that

$$\sum_{i \in \mathcal{L}_s} \nu_s(i) \nu_{st}^{(i)} = \nu_s I (\nu_t^{(i)})^{\mathsf{T}} = \nu_s \nu_t^{\mathsf{T}} = \nu_{st} \,. \tag{114}$$

By construction, ν is a convex combination of $\nu^{(i)}$ and all $\nu^{(i)}$ are integral at vertex s. Applying this construction recursively to all vertices, we can represent ν as a convex combination of integral labelings, which proves that $\nu \in \mathcal{M}$. □

4.1.2 Examples

The following examples illustrate how different methods for partial optimality are interpreted as projections. The exact proofs that the projections constructed by these methods are Λ-improving will be given later.

Example 2 (α-domination)**.** Let us have 3 labels in $\mathcal{L}_s$ and let P_s project everything on a single label, *e.g.*, $\alpha = 2$. Then

$$P_s = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix} . \tag{115}$$

For any vector μ_s representing a relaxed labeling, *e.g.*, $(\,0.5\;0\;0.5\,)^{\mathsf{T}}$, the resulting projection $P_s\mu_s = (\,0\;1\;0\,)^{\mathsf{T}}$. Let P_t be identity for all $t \neq s$. If P is strictly Λ-improving, then optimal labeling in s must be α. In this case, the projection alters only pixel s and its improving property can be verified locally as in DEE. However, when the projection $P_{s'}$ is not identity for multiple pixels s' but has the form (115), the verification of the Λ-improving property is no longer local. Methods of Kovtun (2004) build projections of this type.

Example 3 (α-β domination)**.** Let us have 3 labels in $\mathcal{L}_s$ and let P_s project label $\alpha = 1$ onto label $\beta = 3$.

$$P_s = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} . \tag{116}$$

This is the type of projection constructed by simple DEE. Clearly, we have $(P_s\mu_s)(1) = 0$ for any μ_s. If P is Λ-improving then label 1 in s can be eliminated.

Example 4 (α-elimination)**.** Let

$$P_s = \begin{pmatrix} 0 & 0 & 0 \\ 0.5 & 1 & 0 \\ 0.5 & 0 & 1 \end{pmatrix} . \tag{117}$$

This projection corresponds to a general Goldstein's DEE with a fixed convex combination. If P is Λ-improving, label $\alpha = 1$ can be eliminated. See Figure 10(a).

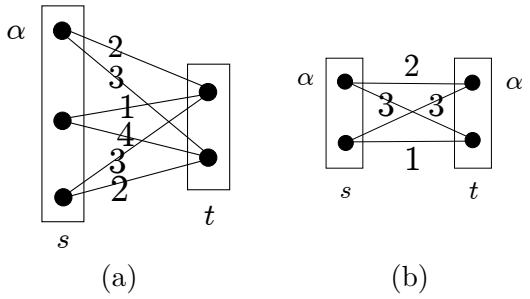


Figure 10 (a) Example of an energy in which label $\alpha = 3$ is not dominated by any other single label but can be eliminated by a convex combination of labels 1 and 2 by projection (117). The labeling $(\alpha, 2)$ has cost 2 which is smaller than the cost of labeling $(1, 2)$, so α cannot be improved by switch to label 1. On the other hand, labeling $(\alpha, 1)$ has a smaller cost than labeling $(2, 1)$, therefore α cannot be improved by switching to label 2 either. (b) Example of an energy in which label α cannot be eliminated by singles criterion in either s or t, but can be eliminated by simultaneous projection in both s and t.

Example 5 (multi-label QPBO)**.** This method provides interval constraints on the possible optimal labels in each pixel. It will be shown that it constructs an Λ-improving projection of the following form: for a pixel with 4 labels interval constraint $2 \leq x_s^* \leq 3$ will correspond to a projection

$$P_s = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}. \tag{118}$$

The constraint $x^{\min} \leq x^* \leq x^{\max}$ will correspond to all the interval projections simultaneously in all pixels.

4.1.3 Properties of Improving Projections

By design we have that Λ-improving property implies $\mathcal{M}$-improving and likewise in the strict case. However, the reverse implications do not hold. Example 6 illustrates an $\mathcal{M}$-improving projection that is not Λ-improving.

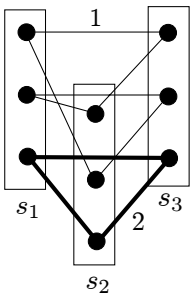


Figure 11 Example of an energy in which the projection onto labeling $(1, 1, 1)$ (bold) is $\mathcal{M}$-improving but not Λ-improving. The thin edges have the cost 1 while bold edges have the cost 2 and all the remaining edges have a large (∞) cost.

Example 6. Consider the energy shown in Figure 11. Consider a projection that maps any labeling to the labeling $(1, 1, 1)$, having cost 6. Because it is the optimal labeling, the projection is an $\mathcal{M}$-improving projection. On the other hand, there is a relaxed labeling with cost 3, which, however, does not belong to $\mathcal{M}$. When the projection is applied to this relaxed labeling, the energy increases.

Theorem 17. A Λ-improving projection preserves optimal solutions of the LP relaxations. Let

$$O = \operatorname*{argmin}_{\mu\in\Lambda} \langle f, \mu\rangle\,, \qquad O' = \operatorname*{argmin}_{\mu\in P(\Lambda)} \langle f, \mu\rangle\,. \tag{119}$$

Then $O' = P(O) \subset O$ if P is Λ-improving and $O' = P(O) = O$ in the strict case.

Proof. Similar to Theorem 15, substituting polytope Λ. □

This theorem implies that any of the partial optimality methods (DEE, MQPBO, auxiliary submodular problems) cannot be used to tighten the LP relaxation. They may only simplify it by reducing the number of variables, but preserving some (or all, depending on the specific method) initial optimal relaxed solutions.

Theorem 18. The projection preserves equivalence: $f \equiv g$ implies $P^{\mathsf{T}} f \equiv P^{\mathsf{T}} g$.

Proof.

$$(\forall \mu \in \Lambda) \quad \langle P^{\mathsf{T}} f, \mu\rangle = \langle f, P\mu\rangle = \langle g, P\mu\rangle = \langle P^{\mathsf{T}} g, \mu\rangle\,. \tag{120}$$

The equality $\langle f, P\mu\rangle = \langle g, P\mu\rangle$ holds since $f \equiv g$ and $P\mu \in \Lambda$. □

Theorem 19 (Linearity)**.** Let P be a (strictly) Λ-improving projection for E_f and Λ-improving for E_g. Then P is (strictly) Λ-improving for $E_f + E_g$.

Proof. We have

$$(\forall \mu \in \Lambda) \quad \langle f+g, P\mu\rangle = \langle f, P\mu\rangle + \langle g, P\mu\rangle \leq \langle f, \mu\rangle + \langle g, \mu\rangle = \langle f+g, \mu\rangle \tag{121}$$

and the inequality is strict if $\mu \in \Lambda \backslash P(\Lambda)$. □

Theorem 20. Let a projection P satisfy component-wise inequalities for f:

$$(\forall s \in \mathcal{V})\ (\forall i \in \mathcal{L}_s) \quad (P^{\mathsf{T}} f)_s(i) \leq f_s(i)\,, \tag{122a}$$

$$(\forall st \in \mathcal{E})\ (\forall ij \in \mathcal{L}_{st}) \quad (P^{\mathsf{T}} f)_{st}(i,j) \leq f_{st}(i,j)\,. \tag{122b}$$

Then P is Λ-improving for E_f.

Proof. Let $\mu \in \Lambda$. By multiplying (122a) by $\mu_s(i)$ and summing over s and i and multiplying (122b) by $\mu_{st}(i,j)$ and summing over st and ij we get

$$\langle P^{\mathsf{T}} f, \mu\rangle \leq \langle f, \mu\rangle\,, \tag{123}$$

which is equivalent to (98). □

Theorem 20 is very simple. It corresponds to the sufficient condition used in the method of auxiliary submodular problems (Kovtun 2004) reviewed in §3.5. A major result that we prove in §4.3 is that *any* Λ-improving projection for E_f does satisfy these component-wise inequalities for a specific equivalent transformation of E_f.

4.1.4 Relation to Fusion-Move Algorithm

We already observed that Λ-improving projections preserve the set of optimal solutions of LP relaxation. This property naturally follows from the definition. We will show next that a similar property holds for a special class of improving projections and the set of fixed points of the fusion move algorithm. It generalizes a weaker result in (Shekhovtsov and Hlaváč 2011) obtained for specific algorithms.

Theorem 21. Let P be a strictly Λ-improving projection for E_f of the pixel-wise form such that

$$P_s = I \quad \text{or} \quad P_{s,ii'} = [\![i{=}\alpha]\!] \,. \tag{124}$$

Let x be a fixed point of the fusion move Algorithm 2. Then

$$P\delta(x) = \delta(x) \,. \tag{125}$$

Proof. Consider the fusion move for labelings x and $y = \alpha$. The move energy is the restriction of f to the set of labels $K_s = \{x_s, \alpha\}$ in every pixel. The linear relaxation for the move energy (solved by QPBO) can be viewed as the constrained linear relaxation for f,

$$\min_{\mu \in \Lambda'} \langle f, \mu \rangle \,, \tag{126}$$

where $\Lambda' = \Lambda \cap \{\mu \,|\, \mu_s(i) = 0 \;\; \forall i \in \mathcal{L}_s \backslash \{x_s, \alpha\} \; \forall s \in \mathcal{V}\}$. Projection P maps Λ' to itself and $\{\delta(x') \,|\, x'_s \in K_s \; \forall s \in \mathcal{V}\}$ to itself. By definition we also have that $\langle f, P\mu \rangle \leq \langle f, \mu \rangle$ on $\Lambda \supset \Lambda'$ and the inequality is strict in the case $P\mu \neq \mu$. Therefore P is improving on Λ'.

Assume for contradiction that $P\delta(x) \neq \delta(x)$. There exists $s \in \mathcal{V}$ such that $x_s \neq \alpha$ and $(P\delta(x))_s(i) = [\![i{=}\alpha]\!]$. In that case optimal solutions of the relaxation (126) cannot have $\mu_{s,x_s} > 0$ and hence the fusion move from x to α will necessarily switch x_s to α, which contradicts the assumption of x being a fixed point. □

Suppose that the energy E_f satisfies metric condition (Boykov et al. 2001), so that pure expansion-move Algorithm 1 applies. In this case, QPBO provides the exact solution of the move subproblem and our proof above holds for the expansion move Algorithm 1. Therefore, if x is a fixed point of the expansion move, it satisfies $P\delta(x) = \delta(x)$.

The one-against-all algorithm of Kovtun (2003) reviewed in §3.5 can be related to the truncated expansion move algorithm. In (Shekhovtsov and Hlaváč 2011) we showed that fixed points of the expansion move with *any* truncation rule are preserved by the one-against-all method of Kovtun (2003). This result holds since the one-against-all method uses the weakest truncation in the sense discussed in §2.7.1.

4.2 Unification

In this section, we show how the optimality properties of different individual methods reviewed in Chapter 3 can be derived via improving projections introduced in §4.1. Despite of that, the methods are still different and deliver different partial optimality guarantees. The unification via improving projections reveals common properties of these methods, allows us to relate them to the LP relaxation and to the expansion move algorithm, and suggests a systematic study.

4.2.1 DEE as Improving Projection

We have already noticed in Examples 3 and 4 that the DEE conditions can be expressed via projection. We now give a formal proof of the improving property of this projection.

Statement 22. The projection constructed by Goldstein's general DEE is improving.

Proof. Let Goldstein's general DEE condition hold for a pair (s, α) for some coefficients $\bar{\beta} \geq 0$, $\sum_i \bar{\beta}(i) = 1$. The condition reads

$$\min_{x \in \mathcal{L}_{\mathcal{N}(s)}} \Big(f_s(\alpha) - \sum_i f_s(i)\bar{\beta}(i) + \sum_{t \in \mathcal{N}(s)} [f_{st}(\alpha, x_t) - \sum_i f_{st}(i, x_t)\bar{\beta}(i)] \Big) \geq 0\,. \tag{127}$$

We introduce the mapping P_s such that

$$P_{s,ii'} = \begin{cases} 1 & \text{if } \; i = i' \neq \alpha\,, \\ \bar{\beta}(i) & \text{if } \; i \neq \alpha,\; i' = \alpha\,, \\ 0 & \text{otherwise}\,. \end{cases} \tag{128}$$

Let k be an arbitrary label such that $k \neq \alpha$ and let μ_s be the indicator of this label, $\mu_s(i) = [\![i{=}k]\!]$. It holds that $P_s\mu_s = \mu_s$. For the indicator of label α with components $\mu_s(i) = [\![i{=}\alpha]\!]$, we have $P_s\mu_s = \bar{\beta}$. With such a projection, we can rewrite the constraint as

$$\min_{x \in \mathcal{L}} \langle f - P^{\mathsf{T}} f, \delta(x) \rangle \geq 0\,. \tag{129}$$

Indeed, for $x_s \neq \alpha$ the objective becomes $\langle f - f, \delta(x) \rangle \geq 0$, which holds trivially. For $x_s = \alpha$, the objective becomes that of (127). Additionally, we extended the minimization to include all variables $x \in \mathcal{L}$, not just the neighbors of s. This can be done because in the energy $f - P^{\mathsf{T}} f$ only the pairwise terms adjacent to s are non-zero. By the same argument, the resulting minimization problem has a star-structure, so we can relax the minimization to the local polytope without loosing tightness, writing (129) as

$$\min_{\mu \in \Lambda} \langle f - P^{\mathsf{T}} f, \mu \rangle \geq 0 \quad \Leftrightarrow \quad (\forall \mu \in \Lambda) \quad \langle f - P^{\mathsf{T}} f, \mu \rangle \geq 0\,, \tag{130}$$

which is the improving property of P. □

4.2.2 Autarky as Improving Projection

Roof dual, MQPBO and auxiliary submodular problems methods derive domain constraints in the form of autarkies (see Definition 5). General autarkies cannot be shown to be a special case of our sufficient condition as they are not polynomially verifiable. However, autarkies can be represented by an $\mathcal{M}$-improving projection and in the case of QPBO, MQPBO and submodular functions all autarkies are also Λ-improving. The case of submodular functions can then be used to show that the general sufficient condition formulated by Kovtun (2004) is a special case of Λ-improving projection as well.

Definition 11. The *linear extension* of mapping $p\colon \mathcal{L} \to \mathcal{L}$ is the linear mapping $P\colon \Lambda \to \Lambda$ satisfying

$$(\forall x \in \mathcal{L}) \quad P\delta(x) = \delta(p(x))\,. \tag{131}$$

Relation (131) means that the two mappings coincide on all integer vectors from Λ. Since $\operatorname{aff} \Lambda = \operatorname{aff} \mathcal{M} = \operatorname{aff} \delta(\mathcal{L})$, it can be verified that the linear extension exists and its restriction to Λ is unique.

Statement 23. The linear extension is unique on Λ.

Proof. Let P and P' both be linear extensions of p. They coincide on $\delta(\mathcal{L})$ and consequently on $\operatorname{aff} \delta(\mathcal{L})$. Since $\Lambda \subset \operatorname{aff} \Lambda = \operatorname{aff} \delta(\mathcal{L})$, they coincide on Λ. □

Statement 24. Let P and Q be linear extension of p and q, respectively. Then, PQ is the linear extension of the composition of p and q.

Proof.

$$(\forall x \in \mathcal{L}) \quad PQ\delta(x) = P(Q\delta(x)) = P\delta(q(x)) = \delta(p(q(x)))\,. \tag{132}$$

□

It follows that the linear extension of an idempotent mapping is idempotent.

Theorem 25. Let p be idempotent and pixel-wise: $p(x)_s = p_s(x_s)$. The linear extension P in this case is given by the pixel-wise projection (108) with the matrices

$$P_{s,ii'} = [\![p_s(i')=i]\!]\,. \tag{133}$$

Proof. We first verify that P is the linear extension of p.

$$\begin{aligned}(\forall x \in \mathcal{L}) \quad \Big(P\delta(x)\Big)_s(i) &= \sum_{i'} P_{s,ii'}[\![x_s=i']\!] \\ &= \sum_{i'} [\![p(i')=i]\!]\,[\![x_s=i']\!] = [\![p(x_s)=i]\!] \;= \Big(\delta(p(x))\Big)_s(i)\,.\end{aligned} \tag{134}$$

The equality for the pairwise components is verified similarly.

Matrices (133) satisfy the conditions $P_{s,ii'} \in [0,\ 1]$ and $\sum_i P_{s,ii'} = 1$. The property $P(\mathcal{M}) \subset \mathcal{M}$ follows by Theorem 16. Idempotency follows by Statement 24. □

Definition 12. Let (x, y) be a pair of labelings such that $x \leq y$. Let us define the following mappings:

$$\begin{aligned} &P_x^y, \quad \text{the extension of } \ x' \mapsto (x' \vee x) \wedge y\,, \\ &P_x, \quad \text{the extension of } \ x' \mapsto x' \vee x\,, \\ &P^y, \quad \text{the extension of } \ x' \mapsto x' \wedge y\,. \end{aligned} \tag{135}$$

By Theorem 25, these mappings are projections. It is easy to see that the mapping P_x^y is given by the following matrices (see Example 5)

$$(P_x^y)_{s,ii'} = \begin{cases} [\![i=x_s]\!] & \text{if } i' < x_s\,, \\ [\![i=i']\!] & \text{if } x_s \leq i' \leq y_s\,, \\ [\![i=y_s]\!] & \text{if } i' > y_s\,. \end{cases} \tag{136}$$

Mappings P_x (resp. P^y) are special cases of P_x^y when $y_s = |\mathcal{L}_s| - 1$ (resp. $x_s = 0$) for all s. It also can be verified that $P_x^y = P^y P_x = P_x P^y$ for all $x \leq y$.

Statement 26. Let $(x^{\min}, x^{\max})$ be an autarky for E_f. Then pixel-wise projection $P = P_{x^{\min}}^{x^{\max}}$ is $\mathcal{M}$-improving for E_f.

Proof. We have

$$\begin{aligned}(\forall x \in \mathcal{L}) \quad \langle f, P\delta(x)\rangle &= \langle f, \delta((x \vee x^{\min}) \wedge x^{\max})\rangle \\ &= E_f((x \vee x^{\min}) \wedge x^{\max}) \leq E_f(x)\,.\end{aligned} \tag{137}$$

Therefore, P is $\mathcal{M}$-improving. □

The Λ-improving property does not follow in general. However, it can be shown for all the specific methods of our consideration. We first show that the submodularity property

$$(\forall x, y \in \mathcal{L}) \quad E_f(x) + E_f(y) \geq E_f(x \vee y) + E_f(x \wedge y) \tag{138}$$

can be extended to Λ. Instead of a labeling x we consider a relaxed labeling μ and replace $E_f(x)$ with $\langle f, \mu \rangle$, $E_f(x \vee y)$ with $\langle f, P_y \mu \rangle$ and $E_f(x \wedge y)$ with $E_f(P^y \mu)$ and will prove the inequality

$$(\forall \mu \in \Lambda)\ (\forall y \in \mathcal{L}) \quad \langle f, \mu \rangle + E_f(y) \geq \langle f, P_y \mu \rangle + \langle f, P^y \mu \rangle \,. \tag{139}$$

Inequality (139) will include (138) in the special case $\mu = \delta(x)$.

To simplify further derivations, we give the following expression for $P_y \mu$:

$$\begin{aligned} (P_y \mu)_s(i) &= \sum_{i' < y_s} \mu_s(i') [\![i{=}y_s]\!] + \sum_{i \geq y_s} [\![i{=}i']\!] \mu_s(i') \\ &= [\![i{=}y_s]\!] \sum_{i' < y_s} \mu_s(i') + [\![i{\geq}y_s]\!] \mu_s(i) \,. \end{aligned} \tag{140}$$

Similarly,

$$(P^y \mu)_s(i) = [\![i{=}y_s]\!] \sum_{i' > y_s} \mu_s(i') + [\![i{\leq}y_s]\!] \mu_s(i) \,. \tag{141}$$

By extending the properties of autarkies of submodular problems to relaxed labellings, as presented next, we can show that P_x and P^y are Λ-improving projections and so is their composition.

Statement 27. If E_f is submodular then

$$(\forall \mu \in \Lambda)\ (\forall y \in \mathcal{L}) \quad \langle f, \mu \rangle + \langle f, \delta(y) \rangle \geq \langle f, P_y \mu \rangle + \langle f, P^y \mu \rangle \,. \tag{142}$$

Proof. We can rewrite (142) equivalently as

$$(\forall \mu \in \Lambda)\ (\forall y \in \mathcal{L}) \quad \langle f, \mu + \delta(y) - P_y \mu - P^y \mu \rangle \geq 0 \,. \tag{143}$$

This scalar product is a sum of unary terms and pairwise terms. We first show that the unary terms vanish:

$$\begin{aligned} \mu_s(i) + [\![i{=}y_s]\!] - [\![i{=}y_s]\!] \sum_{i' < y_s} \mu_s(i') - [\![i{\geq}y_s]\!] \mu_s(i) - [\![i{=}y_s]\!] \sum_{i' > y_s} \mu_s(i') - [\![i{\leq}y_s]\!] \mu_s(i) \\ = \mu_s(i)(1 - ([\![i{\geq}y_s]\!] + [\![i{<}y_s]\!])) + [\![i{=}y_s]\!](1 - \sum_{i'} \mu_{s,i'}) = 0 \,. \end{aligned} \tag{144}$$

Now consider the submodularity constraints:

$$\begin{aligned} &(\forall st \in \mathcal{E})\ (\forall ij \in \mathcal{L}_{st})\ (\forall y_{st} \in \mathcal{L}_{st}) \\ &\qquad f_{st}(i,j) + f_{st}(y_{st}) \geq f_{st}((i,j) \wedge y_{st}) + f_{st}((i,j) \vee y_{st}) \,. \end{aligned} \tag{145}$$

Multiplying this inequality by $\mu_{st}(i,j)$ and summing over ij, we obtain on the LHS

$$\sum_{ij} \mu_{st}(i,j) f_{st}(i,j) + f_{st}(y_{st}) = \sum_{ij} \mu_{st}(i,j) f_{st}(i,j) + \sum_{ij} [\![(i,j){=}y_{st}]\!] f_{st}(y_{st}) \tag{146}$$

and on the RHS

$$\sum_{ij} \mu_{st}(i,j)\big[f_{st}((i,j) \wedge y_{st}) + f_{st}((i,j) \vee y_{st})\big] = \sum_{ij} \big[(P^y\mu)_{st}(i,j) + (P_y\mu)_{st}(i,j)\big] f_{st}(i,j)\,, \tag{147}$$

where the equality is verified as follows:

$$\begin{aligned}&\sum_{ij} \mu_{st}(i,j) f_{st}((i,j) \wedge y_{st}) = \\ &\sum_{\substack{i<y_s\\ j<y_t}} \mu_{st}(i,j) f_{st}(i,j) + \sum_{\substack{i\geq y_s\\ j<y_t}} \mu_{st}(i,j) f_{st}(y_s,j) + \sum_{\substack{i<y_s\\ j\geq y_t}} \mu_{st}(i,j) f_{st}(i,y_t) + \sum_{\substack{i\geq y_s\\ j\geq y_t}} \mu_{st}(i,j) f_{st}(y_{st}) \\ &= \sum_{ij} (P^y\mu)_{st}(i,j) f_{st}(i,j)\,.\end{aligned} \tag{148}$$

The term with $\vee$ is rewritten similarly. By summing inequalities (146) $\geq$ (147) over $st \in \mathcal{E}$ we get the result. □

Theorem 28. Let E_f be submodular and $y \in \operatorname{argmin}_x E_f(x)$. Then P_y and P^y are improving for E_f:

$$(\forall \mu \in \Lambda) \quad \langle f, P_y\mu\rangle \leq \langle f, \mu\rangle\,, \tag{149a}$$

$$(\forall \mu \in \Lambda) \quad \langle f, P^y\mu\rangle \leq \langle f, \mu\rangle\,. \tag{149b}$$

Proof. Let us show (149a). The LP relaxation is tight for submodular problems. Thus for any $\mu' \in \Lambda$, there holds $\langle f, \mu'\rangle \geq E_f(y) = \langle f, \delta(y)\rangle$. In particular, for $\mu' = P^y\mu$ we have $\langle f, P^y\mu\rangle \geq \langle f, \delta(y)\rangle$, which combined with (142) implies the statement. □

We thus have shown that an autarky (x, y) for a submodular problem is always a Λ-improving projection P^y_x.

4.2.3 Roof dual as Improving Projection

The optimality guarantees of QPBO are in the form of an autarky (x, y). We are going to show that this autarky is specific such that the corresponding projection P^y_x is Λ-improving.

Statement 29. Let $O = \texttt{QPBO}(g)$, $x_s = \min O_s$, $y_s = \max O_s$. Then the projection P^y_x is strictly Λ-improving for E_g.

Proof. Recall the proof of partial optimality for QPBO (see page 30). It was shown that for an arc-consistent equivalent $\tilde{g}$ of g component-wise inequalities

$$\begin{aligned}(\forall ij)\ (\forall st) \quad & \tilde{g}_{st}(((i,j) \vee x_{st}) \wedge y_{st}) \leq \tilde{g}_{st}(i,j)\,,\\ (\forall s)\ (\forall i) \quad & \tilde{g}_s((i \vee x_s) \wedge y_s) \leq \tilde{g}_s(i)\end{aligned} \tag{150}$$

are satisfied and inequalities for unary terms are strict for $i \notin O_s$, or, equivalently, for $(i \vee x_s) \wedge y_s \neq i$. Thus sufficient conditions (122) are satisfied for P^y_x, hence P^y_x is Λ-improving for $E_{\tilde{g}}$ and so is for E_g. □

4.2.4 MQPBO as Improving Projection

In (Shekhovtsov et al. 2008, Theorem 2) it was shown that partial optimalities of MQPBO method extend to relaxed labelings and preserve solutions of the LP relaxation. We now recollect this result from the perspective of our unified framework and give a shorter proof.

In the construction of the binarized problem we introduced the mapping $x \mapsto z$ (42a) of multi-valued to binary labelings. Let us extend it to mapping Π of multi-label relaxed labelings μ to binary relaxed labelings ν. For index $i \in \mathcal{L}_s = \{0, 1, \dots, L-1\}$ define the following sets of labels:

$$L_s(i,0) = \{0, \dots, i\}\,, \quad L_s(i,1) = \{i+1, \dots, L-1\}\,.$$

Then we can define vector $\nu = \Pi\mu$ as

$$\nu_{(s,i)}(\alpha) = \sum_{i' \in L_s(i,\alpha)} \mu_s(i')\,, \qquad \nu_{(s,i)(t,j)}(\alpha,\beta) = \sum_{\substack{i' \in L_s(i,\alpha) \\ j' \in L_t(j,\beta)}} \mu_{st}(i',j')\,,$$

where i and j range in $\tilde{\mathcal{L}}_s = \tilde{\mathcal{L}}_t = \{0, 1, \dots, L-2\}$. It can be verified that mapping Π is consistent with the mapping $z_{s,i}(x) = [\![x_s > i]\!]$ in the sense that $\Pi\delta(x) = \delta(z(x))$.

Theorem 30. Let f be a multi-label problem and g the equivalent binary energy minimization problem via reduction in §2.4. Let $O = \texttt{QPBO}(g)$, $z_s^{\min} = \min O_s$, $z_s^{\max} = \max O_s$, $x^{\min} = x(z^{\min})$, $x^{\max} = x(z^{\max})$ according to the mapping $z \mapsto x$ (42b). Then the projection $P_{x^{\min}}^{x^{\max}}$ is strictly Λ-improving for E_f.

Proof. Using mapping Π, the equivalence of multi-label and binary problems $\forall x \in \mathcal{L}$ $E_f(x) = E_g(z(x))$ extends as

$$(\forall \mu \in \Lambda) \quad \langle f, \mu\rangle = \langle g, \Pi\mu\rangle\,. \tag{151}$$

Let $\tilde{g}$ be the arc-consistent equivalent of g, for which it is known that component-wise inequalities (150) hold. By equivalence (151), we have

$$(\forall \mu \in \Lambda) \quad \langle f, \mu\rangle = \langle g, \Pi\mu\rangle = \langle \tilde{g}, \Pi\mu\rangle = \langle \Pi^{\mathsf{T}}\tilde{g}, \mu\rangle \stackrel{\text{def}}{=} \langle \tilde{f}, \mu\rangle\,, \tag{152}$$

from which we conclude that $\tilde{f} \equiv f$. Because the mapping Π^{T} is component-wise, we have from $\tilde{f} = \Pi^{\mathsf{T}}\tilde{g}$

$$\begin{aligned} \tilde{f}_s(x_s) &= \sum_{i'} \tilde{g}_{(s,i')}(z_{s,i'}) \\ &\geq \sum_{i'} \tilde{g}_{(s,i')}((z_{s,i'} \vee z_{s,i'}^{\min}) \wedge z_{s,i'}^{\max}) \\ &= \tilde{f}_s((x_s \vee x^{\min})_s \wedge x_s^{\max})\,. \end{aligned} \tag{153}$$

Similarly, for pairwise terms:

$$\begin{aligned} \tilde{f}_{st}(x_{st}) &= \sum_{i'j'} \tilde{g}_{(s,i')(t,j')}(z_{(s,i')(t,j')}) \\ &\geq \sum_{i',j'} \tilde{g}_{(s,i')(t,j')}\Big((z_{(s,i')(t,j')} \vee z^{\min}_{(s,i')(t,j')}) \wedge z^{\max}_{(s,i')(t,j')}\Big) \\ &= \tilde{f}_{st}((x_{st} \vee x^{\min})_{st} \wedge x_{st}^{\max})\,. \end{aligned} \tag{154}$$

Therefore, component-wise inequalities hold for $\tilde{f}$. By the sufficient condition (122), we conclude that

$$(\forall \mu \in \Lambda) \quad \langle \tilde{f}, P_{x^{\min}}^{x^{\max}} \mu \rangle \leq \langle \tilde{f}, \mu \rangle \,, \tag{155}$$

therefore $P_{x^{\min}}^{x^{\max}}$ is Λ-improving for $E_{\tilde{f}}$ and so also for E_f. □

4.2.5 Auxiliary Problem as Improving Projection

In this section, we show that the sufficient conditions used by (Kovtun 2003) are also a special case of Λ-improving projection. First, we generalize the definition of an auxiliary problem given by equation (86) to arbitrary projections.

Definition 13 (Kovtun 2011, extended)**.** Function E_g is called *auxiliary* for E_f and P if

$$(\forall x) \quad E_{P^\mathsf{T} f}(x) - E_f(x) \leq E_{P^\mathsf{T} g}(x) - E_g(x) \,, \tag{156}$$

i.e., the improvement under projection P in E_f is at least as big as improvement in E_g. Clearly, if P is $\mathcal{M}$-improving for E_g then RHS is not positive and so is the LHS. In that case P is $\mathcal{M}$-improving for E_f. It is also clear that condition (156) is identical to the statement "P is $\mathcal{M}$-improving for $f - g$". Clearly, (156) is equivalent to (86) in the case of projection $P_{x^{\min}}$.

Now consider a stricter sufficient condition by requiring the inequality on the local polytope:

$$(\forall \mu \in \Lambda) \quad \langle P^\mathsf{T} f - f, \mu \rangle \leq \langle P^\mathsf{T} g - g, \mu \rangle \,, \tag{157}$$

which reads "P is Λ-improving for $f - g$".

Let us be even stricter and require component-wise inequalities

$$(P^\mathsf{T} - I)(f - g) \leq 0 \,. \tag{158}$$

On the one hand, inequalities (158) imply (157) via Theorem 20. On the other hand, inequalities (158) coincide with component-wise sufficient conditions (87) by (Kovtun 2004). We thus have the following theorem.

Theorem 31. Let E_g be submodular and $x^{\min}$ be its minimizers. Let component-wise inequalities (158) hold with $P = P_{x^{\min}}$. Then $P_{x^{\min}}$ is Λ-improving for E_f.

Proof. By Theorem 20 we have that $P_{x^{\min}}$ is Λ-improving for $f - g$. Since g is submodular, by Theorem 28 $P_{x^{\min}}$ is Λ-improving for g. By linearity Theorem 19, $P_{x^{\min}}$ is Λ-improving for E_f. □

By Theorem 4.2.5, the methods reviewed in §3.5, which are all based on component-wise sufficient conditions and submodular auxiliary functions, find a Λ-improving projection.

The approach of (Kovtun 2003) can be generalized to arbitrary projections as follows:

- Find a submodular E_g which is auxiliary for E_f and a class of projections $\mathcal{P}$.
- Find the "best" projection $P \in \mathcal{P}$ that is improving for E_g.
- Then P is improving for E_f.

In the next section, we will show that the component-wise inequalities (158) necessarily hold for a certain equivalent of f for an arbitrary improving projection. Therefore, the simplification of conditions from (157) to (158) is without loss of generality as soon as the final method is invariant to equivalent transformations of g.

4.3 Characterization of Improving Projections

We introduced the notion of an improving projection and showed that many domain constraints proposed in the literature are in fact improving projections. Notice that while proving this statement for different methods, we always considered a special reparametrization of the problem such that the local sufficient conditions of Theorem 20 were satisfied. There was such a reparametrization of the problem in each case that the constructed projection improved independently all unary and all pairwise terms. We are going to show now that it is not a coincidence: all Λ-improving projections of the pixel-wise form (107) can be obtained in this way.

Theorem 32. Let P be a projection of the pixel-wise form (107). Then P is Λ-improving for E_f iff there exists $g \equiv f$ such that local inequalities hold:

$$\begin{aligned} (P^\mathsf{T} g)_s &\leq g_s \,, \\ (P^\mathsf{T} g)_{st} &\leq g_{st} \,. \end{aligned} \tag{159}$$

The conditions (159) are, up to an equivalent transformation, necessary and sufficient for P to be Λ-improving. The "if" part follows trivially by Theorem 20. Prior to the proof of the "only if" part, let us give some clarification of this result. Note that conditions (159) can be shortly written as $g - P^\mathsf{T} g \geq 0$, where the inequalities are for all components, including the trivial inequality $g_0 - g_0 \geq 0$.

Let P be Λ-improving for f, then by definition

$$\min_{\mu \in \Lambda} \langle f - P^\mathsf{T} f, \mu \rangle \geq 0 \,, \tag{160}$$

which is equivalent (for a connected graph) to

$$(\exists \varphi) \quad (f - P^\mathsf{T} f) - A^\mathsf{T} \varphi \geq 0 \,, \tag{161}$$

i.e., some similar component-wise inequalities hold for a problem equivalent to $f - P^\mathsf{T} f$. However, this is not the desired result. Let the equivalent problem g be parametrized as $f - A^\mathsf{T} \varphi'$. We need to show that there exists φ' such that $(f - A^\mathsf{T} \varphi') - P^\mathsf{T}(f - A^\mathsf{T} \varphi') \geq 0$. The theorem claims that φ satisfying (161) exists such that it additionally satisfies $A^\mathsf{T} \varphi = (I - P^\mathsf{T}) A^\mathsf{T} \varphi'$ for some φ', which means that φ must be found from a constrained linear subspace.

Proof (of the "only if" part). First, we note that the following inequality holds for any projection P:

$$\min_{\mu \in \Lambda} \langle f, (I - P)\mu \rangle \leq 0 \,. \tag{162}$$

It can be verified as follows. Let us show that there exists a point μ' feasible to (162) such that the objective is zero. Pick arbitrary $\mu \in \Lambda$ and choose $\mu' = P\mu \in \Lambda$. We have $(I - P)\mu' = (P - PP)\mu = 0$. Therefore, the value of the minimization problem (162) is not positive. From this property, it follows that P is Λ-improving iff

$$\min_{\mu \in \Lambda} \langle (I - P)\mu, f \rangle = 0 \,. \tag{163}$$

We will prove that φ can be restricted in the dual maximization problem (as discussed above) by relaxing the primal problem. The key steps of the proof are given by the

following chain:

$$\begin{aligned} 0 &\overset{\text{(a)}}{=} \min_{\substack{A\mu=0\\ \mu\geq 0}} \langle f - P^{\mathsf{T}} f, \mu\rangle \overset{\text{(b)}}{=} \min_{\substack{AP\mu=0\\ A(I-P)\mu=0\\ \mu\geq 0}} \langle f - P^{\mathsf{T}} f, \mu\rangle \\ &\overset{\text{(c)}}{=} \min_{\substack{A(I-P)\mu=0\\ \mu\geq 0}} \langle f - P^{\mathsf{T}} f, \mu\rangle \overset{\text{(d)}}{=} \max_{\substack{\varphi\\ (I-P^{\mathsf{T}})(f-A^{\mathsf{T}}\varphi)\geq 0}} 0\,. \end{aligned} \tag{164}$$

Equality (a) is implied by the Λ-improving property (163) as follows. We know that $\geq$ holds in (a) because we dropped one constraint compared to (163). Assume for contradiction that there exists μ such that $A\mu = 0$, $\mu \geq 0$ and $\langle f - P^{\mathsf{T}} f, \mu\rangle < 0$. Let us first consider the case when the graph $(\mathcal{V}, \mathcal{E})$ is connected. Selecting $\mu' = \mu / \sum_{i\in\mathcal{L}_s} \mu_s(i)$ for some s ensures that $B\mu' = 1$ (due to marginalization constraints). Therefore $\mu' \in \Lambda$ and $\langle f - P^{\mathsf{T}} f, \mu'\rangle < 0$, which contradicts (163). In the case that graph $(\mathcal{V}, \mathcal{E})$ consists of several connected components, it can be shown that (163) holds for each component and so does (164a).

Equality (b) is verified as follows. Inequality $\leq$ holds because $A\mu - AP\mu = 0$ and $AP\mu = 0$ implies $A\mu = 0$. On the other hand, P preserves marginalization constraints: $A\mu = 0 \Rightarrow AP\mu = 0$.

Equality (c) is the key step. We removed one constraint, therefore $\geq$ trivially holds. Let us prove $\leq$. Let μ be feasible to RHS of equality (c). Let $\mu = \mu_1 + \mu_2$ such that $\mu_1 \in \operatorname{null}(I - P)$ and $\mu_2 \in \operatorname{null}(P)$. Let us construct μ_1' as follows:

$$\begin{aligned} \gamma &= \max_{st,ij} |\mathcal{L}_{st}|(\mu_1)_{st}(i,j)\,, \\ (\mu_1')_{st} &= \gamma / |\mathcal{L}_{st}|\,, \\ (\mu_1')_{s} &= \gamma / |\mathcal{L}_{s}|\,. \end{aligned} \tag{165}$$

By construction,

$$(\mu_1') \geq \mu_1 \qquad \text{and} \qquad A\mu_1' = 0\,. \tag{166}$$

Let $\mu_1'' = P\mu_1'$. Because $P \geq 0$, we have

$$\mu_1'' = P\mu_1' \geq P\mu_1 = \mu_1\,. \tag{167}$$

It also follows that $AP\mu_1'' = APP\mu_1' = AP\mu_1' = 0$ and $(I - P)\mu_1'' = (I - P)P\mu_1' = 0$. Let $\mu^* = \mu_1'' + \mu_2$. It preserves the objective,

$$\langle f - P^{\mathsf{T}} f, \mu^*\rangle = \langle f, (I - P)(\mu_1'' + \mu_2)\rangle = \langle f, (I - P)\mu_2\rangle = \langle f, (I - P)\mu\rangle\,. \tag{168}$$

We also have that

$$\begin{aligned} \mu^* = \mu_1'' + \mu_2 \geq \mu_1 + \mu_2 = \mu \geq 0\,, \\ A(I - P)\mu^* = A(I - P)\mu_2 = A(I - P)\mu = 0\,, \\ AP\mu^* = AP\mu_1'' = 0\,. \end{aligned} \tag{169}$$

Therefore, μ^* satisfies all constraints of the LHS of equality (c).

Equality (d) is the duality relation that asserts that the maximization problem on the RHS is feasible, which is the case iff

$$(\exists g \equiv f) \quad (I - P^{\mathsf{T}})g \geq 0\,. \tag{170}$$

□

Let us also note that the following simplified characterization of improving projection was obtained in the course of the proof.

Corollary 33. Pixel-wise projection P is Λ-improving iff

$$\min_{\substack{A(I-P)\mu=0 \\ \mu\geq 0}} \langle f - P^{\mathsf{T}} f, \mu \rangle = 0\,. \tag{171}$$

Compared to the Definition 10 of Λ-improving projection, this problem has relaxed constraints. The constraints $A\mu = 0$ and $B\mu = 1$ of Λ are relaxed to the constraint $A(I - P)\mu = 0$ in (171). This problem can be unbounded. In that case the equality does not hold and P is not Λ-improving. This form will be used later to simplify the problem of verification of the improving property. Relaxed constraints will allow us in some cases to decouple and eliminate part of the variables μ. This provides an algebraic approach to constructing partial optimality methods: select a subclass of projections and simplify the characterization problem starting from the form (171).

4.4 Maximum Projections

We say that projection P *dominates* Q if

$$P\mu = \mu \;\Rightarrow\; Q\mu = \mu\,. \tag{172}$$

For a given problem f, we would like to find the improving projection which dominates all other projections. This projection, if it exists, is referred to as *maximum*. Our goal is to find a projection that eliminates as many labels as possible. The maximum projection is optimal in this respect. In this section, we study classes of projections in which the maximum projection exists and can be found efficiently.

We say that two projections P and Q *commute* if $PQ = QP$.

Definition 14. A subset $\mathcal{P}$ of projections is a *commutative class* if

$$(\forall P, Q \in \mathcal{P}) \quad PQ = QP \;\text{ and }\; PQ \in \mathcal{P}\,. \tag{173}$$

Let $\mathcal{P}$ be a commutative class of projections. The product operation is idempotent, because $PP = P$, commutative and associative. Therefore $\mathcal{P}$ with product is a join-semilattice. If $QP \neq P$ then QP dominates both P and Q (has a larger null-space). We can therefore speak of the *maximum* projection in the class $\mathcal{P}$, which is the product of all projections in $\mathcal{P}$ and has the largest null-space.

We will consider several special classes of projections and study the problem of finding the maximum one in a class. As can be noticed, there are two requirements on the class: 1) it must be closed under the product and 2) the product must be commutative within the class. In general, even strictly improving pixel-wise projections of a single pixel may not commute. Regarding the closedness requirement, we have the following helpfull result.

Statement 34. Let $\mathcal{P}$ be a class of (strictly or not) Λ-improving projections for E_f. Then $(\forall P, Q \in \mathcal{P})\ PQ \in \mathcal{P}$.

Proof. For Λ-improving projections, the statement follows trivially by

$$(\forall \mu \in \Lambda) \quad E_f(PQ\mu) \leq E_f(Q\mu) \leq E_f(\mu)\,. \tag{174}$$

In the case of strictly Λ-improving projections, let $PQ\mu \neq \mu$. Then either $PQ\mu \neq Q\mu$ or $Q\mu \neq \mu$. It follows that one of the inequalities in (174) is strict and hence $E_f(PQ\mu) < E_f(\mu)$. □

We will now consider special cases.

4.4.1 Mappings $x \mapsto (x \vee y) \wedge z$

Consider the class $\mathcal{P}$ of strictly Λ-improving projections P_y^z, where $y, z \in \mathcal{L},\ y \leq z$. That is, the class induced by all strong autarkies for E_f.

Statement 35. $\mathcal{P}$ is a commutative class.

Proof. Let (x^1, x^2) and (y^1, y^2) be strong autarkies for f. The composition of mappings $x \mapsto (x \vee y^1) \wedge y^2$ and $x \mapsto (x \vee x^1) \wedge x^2$ is the mapping

$$x \mapsto \big(\big[(x \vee x^1) \wedge x^2\big] \vee y^1\big) \wedge y^2\,. \tag{175}$$

The product of projections $P_{y^1}^{y^2} P_{x^1}^{x^2}$ is given by the extension of this mapping to Λ. We need to show closedness and commutativity of $\mathcal{P}$.

Let x be an optimal labeling for E_f. By properties of strong autarkies, there holds

$$\begin{aligned} x^1 &\leq x \leq x^2\,, \\ y^1 &\leq x \leq y^2\,. \end{aligned} \tag{176}$$

It follows that

$$x^1 \leq y^2\,. \tag{177}$$

Furthermore, since $x^1 \leq x^2$, for any x we have $(x \vee x^1) \wedge x^2 = (x \wedge x^2) \vee x^1$. Using these relations we can rewrite the composition of autarkies in (175) as follows

$$\begin{aligned} \big(\big[(x \vee x^1) \wedge x^2\big] \vee y^1\big) \wedge y^2 &= \big(\big[x \wedge x^2\big] \vee \big[x^1 \vee y^1\big]\big) \wedge y^2 \\ &= \big[x \wedge (x^2 \wedge y^2)\big] \vee (x^1 \vee y^1) \\ &= \big[x \vee (x^1 \vee y^1)\big] \wedge (x^2 \wedge y^2). \end{aligned} \tag{178}$$

Therefore,

$$P_{y^1}^{y^2} P_{x^1}^{x^2} = P_{x^1 \vee y^1}^{x^2 \wedge y^2} = P_{x^1}^{x^2} P_{y^1}^{y^2}\,. \tag{179}$$

By Statement 34, it is a strictly Λ-improving projection. □

It follows that there exists the maximum strong autarky. In the case of binary energies, the maximum strictly Λ-improving projection (linear extension of the maximum strong autarky) is found by QPBO. Strictly Λ-improving projections preserve all optimal solutions of the LP relaxation. On the other hand, there exists an optimal relaxed labeling assigning non-zero weights to *all* nodes which are not eliminated by QPBO. Therefore, no more nodes can be eliminated by a strictly Λ-improving projection. Formally, this is detailed as follows.

Statement 36. Let f be binary and (x^1, x^2) the autarky computed by QPBO. Then $P = P_{x^1}^{x^2}$ is the maximum in $\mathcal{P}$.

Proof. Clearly, $P \in \mathcal{P}$. By the properties of the LP relaxation for binary problems, there exists an optimal relaxed solution μ such that

$$(\forall s \in \mathcal{V}) \quad (x^1_s \leq i \leq x^2) \;\Rightarrow\; \mu_s(i) > 0\,. \tag{180}$$

Assume for contradiction that P is not the maximum. Then there exists $Q = P^{y^2}_{y^1} \in \mathcal{P}$ such that $QP \neq P$. It implies that there exists $s \in \mathcal{V}$ such that $y^1_s > x^1_s$ or $y^2_s < x^2_s$. Without loss of generality, assume it is the first case. We have $\mu_s(x^1_s) > 0$ but $(Q\mu)_s(i) = 0$, therefore $Q\mu \neq \mu$ and it must be $E_f(Q\mu) < E_f(\mu)$, which contradicts optimality of μ. □

This was an expected result. The question of finding the maximum strong autarky for a multi-label problem is not resolved. The strong autarky found by MQPBO is not maximum since optimal solutions to the relaxation of the binarized problem may not be optimal to the relaxation of the multi-label problem (as we mentioned earlier, there is a gap between these two relaxations). However, in the case when the two relaxations coincide MQPBO does find the maximum strong autarky.

Statement 37. Let f be such that for each $st \in \mathcal{E}$ term f_{st} is either submodular or supermodular. Let (x^1, x^2) be the strong autarky computed by MQPBO. Then $P^{x^2}_{x^1}$ is the maximum in $\mathcal{P}$.

Proof. In (Shekhovtsov et al. 2008) we showed that under conditions of this statement there exists an optimal relaxed labeling to the multi-label problem that is half-integral and assigns $\mu_s(x^1_s) = \mu_s(x^2_s) = 1/2$ if $x^1_s < x^2_s$ and 1 if $x^1_s = x^2_s$. Similarly to Theorem 36, it follows that $P^{x^2}_{x^1}$ is maximum. □

Since the class of weak autarkies is not commutative, the maximum weak autarky does not exist (is not defined). In the following sections we restrict ourselves to "one-sided" autarkies $(x^{\min}, y)$ with $y_s = \max \mathcal{L}_s$. From the review in Chapter 3 we have seen that such autarkies are utilized by several methods. They form a commutative class and both maximum strong and maximum weak autarkies of this type exist. We will derive polynomial methods to determine these maximum elements under various further restrictions.

4.4.2 Mappings $x \mapsto (x \vee y)$

Let $\mathcal{P}$ be the class of projections of the form P_y (extending the map $x \mapsto x \vee y$). Clearly, such projections commute and the result belongs to $\mathcal{P}$, since $P_y P_z = P_{y \vee z} = P_z P_y$.

Submodular Energies

Statement 38. Let f be submodular. Then the maximum $\mathcal{M}$-improving projection P_{z^*} is given by the highest minimizer

$$z^* = \bigvee \operatorname*{argmin}_x E_f(x)\,; \tag{181}$$

and the maximum *strictly* $\mathcal{M}$-improving projection P_{z_*} is given by the lowest minimizer

$$z_* = \bigwedge \operatorname*{argmin}_x E_f(x)\,. \tag{182}$$

Proof. Let P_y be an $\mathcal{M}$-improving projection for E_f for some y. On the one hand, $E_f(P_y z^*) = E_f(y \vee z^*) \leq E_f(z^*)$, therefore $y \vee z^*$ is an optimal labeling for E_f. By construction, $z^* \geq y \vee z^*$ and hence $z^* \geq y$. On the other hand, $P_y P_{z^*} = P_{y \vee z^*} = P_{z^*}$. Hence P_{z^*} is maximum.

Let us consider the strictly improving case. Let P_y be strictly $\mathcal{M}$-improving for f and some y. Assume for contradiction that $z_* \vee y > z_*$. In that case $E_f(z_* \vee y) < E_f(z_*)$, which contradicts optimality of z_*. □

Statement 39. Let P_{z^*} be the maximum $\mathcal{M}$-improving projection for submodular f. Then P_{z^*} is the maximum Λ-improving.

Proof. Because f is submodular, for its arc-consistent equivalent f^φ there exists a labeling taking only locally minimal nodes and arcs. Since z^* is optimal to E_f, it also must take only locally minimal nodes and arcs. By the local sufficient conditions of Theorem 20, P_{z^*} is Λ-improving. Maximality follows from the fact that any Λ-improving projection is $\mathcal{M}$-improving. □

Let us also consider the following constrained case. Let $W \subset \mathcal{V}$ and let $\mathcal{P}$ be the class of projections P_y such that $y_s = 0$ for all $s \notin W$. This case arises when we want to restrict our consideration to a local window W of the full problem and also in the context of elimination move algorithm developed below. It is easy to see that $P_y P_z = P_z P_y = P_{y \vee z}$ belongs to the class if both P_y and P_z do. Therefore, the maximum projection is well-defined.

The maximum $\mathcal{M}$-improving projection P_{z^*} is given by the highest constrained minimizer of E_f, *i.e.*,

$$z^* = \bigvee \operatorname*{argmin}_{\substack{x \\ x_{\mathcal{V} \setminus W} = 0}} E_f(x) \,. \tag{183}$$

In addition, P_{z^*} is Λ-improving. The proof is similar to the unconstrained case.

General Binary Energies We will reduce the maximum Λ-improving projection problem for general binary energies to submodular ones. We will show that the construction of submodular auxiliary problem by truncation in (Kovtun 2011) is optimal for binary auxiliary problems considered there.

Let g be a *truncation* of f such that

$$g_{st}(0,0) = f_{st}(0,0) - \Delta_{st} \,, \tag{184}$$

where $\Delta_{st} = \max\{0, f_{st}(0,0) + f_{st}(1,1) - f_{st}(1,0) - f_{st}(0,1)\} \geq 0$ and all other components of g coincide with those of f. By construction, $\Delta_{st} = 0$ if f_{st} is submodular, and otherwise Δ_{st} is positive. In the later case, g_{st} becomes modular. Therefore, g is submodular.

Theorem 40. P_y is Λ-improving for E_f iff it is Λ-improving for its submodular truncation E_g (184).

Proof. First, we show the "if" part. By construction, the unary components of f and g are equal. For any x and any st we have

$$\begin{aligned} f_{st}(x_{st} \vee y_{st}) - g_{st}(x_{st} \vee y_{st}) &= \begin{cases} \Delta_{st} & \text{if } \; x_{st} = y_{st} = (0,0) \,, \\ 0 & \text{otherwise} \,, \end{cases} \\ &\leq f_{st}(x_{st}) - g_{st}(x_{st}) \,. \end{aligned} \tag{185}$$

By Theorem 20, projection P_y is Λ-improving for $f - g$. By linearity (Theorem 19), P_y is Λ-improving for f. The "if" part is proven. Note that inequalities (185) imply $E_f(x \vee y) - E_g(x \vee y) \leq E_f(x) - E_g(x)$, which means that g is an *auxiliary* function for f and projection P_y.

Let us now show the "only if" part. Let P_y be Λ-improving for E_f. By Theorem 32, an equivalent $f' = f - A^\mathsf{T}\varphi$ exists such that component-wise inequalities hold,

$$f'_s(x_s \vee y_s) \leq f'_s(x_s)\,, \quad f'_{st}(x_{st} \vee y_{st}) \leq f'_{st}(x_{st})\,. \tag{186}$$

Let $g' = g - A^\mathsf{T}\varphi = f - \Delta - A^\mathsf{T}\varphi = f' - \Delta$, the same reparametrization applied to g.

We are going to prove component-wise inequalities for g',

$$g'_s(x_s \vee y_s) \leq g'_s(x_s)\,, \quad g'_{st}(x_{st} \vee y_{st}) \leq g'_{st}(x_{st})\,. \tag{187}$$

Let us consider an edge st. If f_{st} is submodular, $g'_{st} = f'_{st}$ and (187) holds. Let us consider the case when f_{st} is not submodular. Denote $a = {f'}_{s,t}(1,1)$, $b = f'_{st}(1,0)$, $c = f'_{st}(0,1)$, $d = f'_{st}(0,0)$, $d' = g'_{st}(0,0) = d - \Delta_{st} = d - (a + d - b - c) = b + c - a$. Consider the following cases for y_{st}:

- Case $y_{st} = (0,0)$. Trivial, since $x_{st} \vee y_s t = x_{st}$.
- Case $y_{st} = (1,1)$. We have $a \leq d, b, c$. It follows that $d' = b + c - a \geq 2a - a = a$. Therefore, $a \leq d', b, c$, which are the required component inequalities for g'.
- Case $y_{st} = (1,0)$. We have $b \leq d$ and $a \leq c$. It follows that $d' = b + c - a \geq b + a - a = b$. Therefore, $b \leq d'$ and $a \leq c$.
- Case $y_{st} = (0,1)$ is similar to the above.

Since g' satisfies component-wise inequalities (187), we have that P_y is Λ-improving for E_g. □

We therefore have that the maximum Λ-improving projection for E_g coincides with the maximum Λ-improving for E_f. Note also that for submodular functions the classes of $\mathcal{M}$-improving and Λ-improving coincide.

In contrast to verification of Λ-improving property, verifying whether P_y is $\mathcal{M}$-improving for a binary problem is equivalent to checking global optimality of y, which is NP-hard.

The "if" part of the theorem extends to multi-label problems but the "only if" part does not. Of course, one can still use it in one direction and construct a sufficient condition using a submodular problem, similarly to the construction by Kovtun (2011), reviewed in §3.5. Alternatively, the theorem can be applied to the transformation of the multi-label energy to a binary one. The question of the maximum Λ-improving projection in the class $\{P_y \,|\, y \in \mathcal{L}\}$ remains open.

4.4.3 Mappings $x \mapsto (x \vee y')$, $y' \in \{y, z\}$

Let y, z be some fixed labelings and let $p\colon \mathcal{L} \to \mathcal{L}$ be defined pixel-wise as

$$p_s(x_s) = \begin{cases} y_s & \text{if } x_s = z_s\,, \\ x_s & \text{otherwise}\,. \end{cases} \tag{188}$$

The mapping p switches labels z_s to labels y_s in the pixels where $z_s \neq y_s$. If p is improving we can eliminate labels z_s in the set $\mathcal{W} = \{y_s \neq z_s\}$. It can be seen that this mapping is of the same form as in the all-against-one construction (§3.5.4), illustrated in Figure 7. Let us assume $y_s \neq z_s$ for all s and let the labels be reordered such that $z_s < y_s = \min\{\mathcal{L}_s \backslash y_s\}$. Using this ordering, mapping (188) takes the form

Algorithm 4: Eliminate(f,y,z)

```
1 v := −1; x := 0; y' := y; 𝒳 := 𝒱;
2 while v < 0 do
3 |   Construct binary energy ĝ from f and y', z by (189);
4 |   x := ⋀ argmin E_ĝ(x);                          /* lowest minimum cut */
5 |   v := E_ĝ(x);                                  /* cost of the minimum cut */
6 |   for s ∈ 𝒳 such that x_s = 1 do
7 |   |   y'_s := z_s;
8 |   𝒳 := 𝒳 ∩ {s ∈ 𝒱 | x_s = 1};
```

$x \mapsto x \vee y$, corresponding to the projection P_y. The ordering is needed purely for convenience of notation and does not influence the result. Let us consider the class $\mathcal{P} = \{P_{y'} \,|\, y'_s \in \{y_s, z_s\}\}$. Depending on the choice of y', the projection can select pixels in which to eliminate z_s by replacing it with y_s.

We have the following results in this case:

- For a given $P \in \mathcal{P}$, verification of the Λ-improving property can be reduced to MAXFLOW.
- Finding the maximum Λ-improving projection in $\mathcal{P}$ can be reduced to a series of MAXFLOW problems.

The algorithm for determining the maximum Λ-improving projection $P \in \mathcal{P}$ is given in Algorithm 4. The algorithm iteratively shrinks the set of vertices by those which cannot be a part of any Λ-improving projection of the considered type. An auxiliary binary energy $\hat{g}$ in the algorithm is constructed as follows.

$$\begin{aligned}
\Delta_{st}(y_s) &= \min_{j \neq z_t} (f_{st}(z_s, j) - f_{st}(y_s, j)) \,, \\
\Delta_{ts}(y_t) &= \min_{i \neq z_s} (f_{st}(i, z_t) - f_{st}(i, y_t)) \,, \\
\hat{g}_s(1) &= f_s(z_s) - f_s(y_s) + \sum_{t \,|\, st \in \tilde{\mathcal{E}}} \Delta_{st}(y_s) \,, \\
\hat{g}_{st}(1,1) &= \min \Big\{ 0, f_{st}(z_s, z_t) - f_{st}(y_s, y_t) - \Delta_{st}(y_s) - \Delta_{ts}(y_t) \Big\}
\end{aligned} \tag{189}$$

and the other components of $\hat{g}$ are set to zero. Correctness and optimality of this algorithm is proven in §4.4.4. The output of the algorithm is a subset of vertices $\mathcal{X}$ (possibly empty), in which the label z can be eliminated. The set of vertices $\mathcal{X}$ is the maximal one for a given pair (y, z). In every iteration of the algorithm the set of potentially eliminatable vertices $\mathcal{X}$ shrinks at least by one so that it terminates in at most $|\mathcal{V}|$ iterations. Interestingly, this algorithm has a similar structure as the ad-hock iterative algorithm of Kovtun (2004), described in§3.5.1. In the actual implementations, the minimization problem in step 4 can be solved via MAXFLOW and warm-started from the previous iteration.

This algorithm was verified experimentally on random and stereo vision problems. Preliminary results indicate that partial optimalities it can determine are complementary to the ones found by one-against-all method of (Kovtun 2004) and DEE. This suggests applying different methods sequentially, gradually reducing the problem. Each method, in turn, can potentially eliminate more labels, if the problem is already reduced. However, for our algorithm, there is an open choice of the test labelings y and z. We experimented with several heuristics but did not came to a definite conclusion.

It is possible to show that when y and z are equal respectively to label α and β everywhere and we run the algorithm repeatedly for all pairs α and β, it will then subsume the simple Goldstein's DEE method. We feel these ideas need further investigation and thorough experimental validation.

4.4.4 Optimality of the Elimination Algorithm

We first show how the verification of Λ-improving property of P can be reduced to the MAXFLOW. It is obtained by substituting the form of projection into the simplified characterization (171) given by Theorem 32 and a subsequent reduction. The characterization (171) constitutes verifying whether the minimization problem

$$\min_{\substack{A(I-P)\mu=0\\ \mu\geq 0}} \langle f, (I-P)\mu\rangle \tag{190}$$

is bounded and attains value 0.

For the unary components of $(I-P)\mu$, we have:

$$(\mu - P\mu)_s(i) = \mu_s(z_s)([\![i{=}z]\!] - [\![i{=}y]\!])\,. \tag{191}$$

Minimization problem (171) will therefore only involve variables μ_{s,z_s} and not $\mu_s(i)$ for any $i \neq z_s$ and any y_s. The pairwise components of $(I-P)\mu$ can be written as follows:

$$\begin{aligned}(\mu - P\mu)_{st}(i,j) &= \mu_{st}(z_s,j)([\![i{=}z_s]\!] - [\![i{=}y_s]\!]) + \mu_{st}(i,z_t)([\![j{=}z_t]\!] - [\![j{=}y_t]\!])\\ &+ \mu_{st}(z_s,z_t)([\![i{=}y_s,\ j{=}z_t]\!] + [\![i{=}z_s,\ j{=}y_t]\!] - [\![i{=}z_s,\ j{=}z_t]\!] - [\![i{=}y_s,\ j{=}y_t]\!])\,.\end{aligned} \tag{192}$$

Reduced marginalization constraints $A(I-P)\mu = 0$ simplifies to

$$\begin{aligned}&[\![y_s{\neq}z_s]\!]\Big(\mu_s(z_s) - \sum_j \mu_{st}(z_s,j)\Big) = 0 \quad \forall st \in \mathcal{E}\,,\\ &[\![y_t{\neq}z_t]\!]\Big(\mu_t(z_t) - \sum_i \mu_{st}(i,z_t)\Big) = 0 \quad \forall st \in \mathcal{E}\,.\end{aligned} \tag{193}$$

There are at most two constraint equations per edge st depending whether $y_s \neq z_s$ and $y_t \neq z_t$. The contribution to the objective of (171) from the singleton s expresses simply as

$$\sum_i ((I-P)\mu)_s(i) f_s(i) = \mu_s(z_s)(f_s(z_s) - f_s(y_s))\,. \tag{194}$$

The contribution from a pair st expresses as follows,

$$\begin{aligned}&\sum_{ij}((I-P)\mu)_{st}(i,j) f_{st}(i,j) =\\ &\quad= \sum_j \mu_{st}(z_s,j)(f_{st}(z_s,j) - f_{st}(y_s,j)) + \sum_i \mu_{st}(i,z_t)(f_{st}(i,z_t) - f_{st}(i,y_t))\\ &\quad+ \mu_{st}(z_s,z_t)(-f_{st}(z_s,z_t) - f_{st}(y_s,y_t) + f_{st}(y_s,z_t) + f_{st}(z_s,y_t))\,.\end{aligned} \tag{195}$$

We can then eliminate variables $(\mu_{st}(z_s,j) \mid j \neq z_t)$ from the characterization (190) by solving the subminimization problem where they are involved, explicitly as follows:

$$\begin{aligned}&\min_{(\mu_{st}(z_s,j)\,|\,j\neq z_t)} \quad \sum_{j\neq z_t} \mu_{st}(z_s,j)(f_{st}(z_s,j) - f_{st}(y_s,j))\\ &\text{subj.}\ \begin{cases}[\![y_s{\neq}z_s]\!](\mu_s(z_s) - \sum_j \mu_{st}(z_s,j)) = 0\,,\\ \mu \geq 0\end{cases}\end{aligned} \tag{196}$$

$$
\begin{aligned}
&= \begin{cases} 0 & \text{if } y_s{=}z_s\,, \\ (\mu_s(z_s) - \mu_{st}(z_s, z_t))\Delta_{st}(y_s) & \text{if } \mu_s(z_s) \geq \mu_{st}(z_s, z_t) \wedge y_s \neq z_s\,, \\ \infty & \text{otherwise} \end{cases} \\
&= \begin{cases} (\mu_s(z_s) - \mu_{st}(z_s, z_t))\Delta_{st}(y_s) & \text{if } [\![y_s{\neq}z_s]\!](\mu_s(z_s) - \mu_{st}(z_s, z_t)) \geq 0\,, \\ \infty & \text{otherwise}\,. \end{cases}
\end{aligned} \tag{197}
$$

where

$$
\Delta_{st}(y_s) = \min_{j \neq z_t}(f_{st}(z_s, j) - f_{st}(y_s, j))\,. \tag{198}
$$

We can now write down the reduced characterization (190) as the linear program

$$
\begin{aligned}
&\min_{\bar{\mu}} \sum_{s \in \mathcal{V}} \bar{\mu}_s \bar{g}_s + \sum_{st \in \mathcal{E}} \bar{\mu}_{st} \bar{g}_{st}\,, \\
&\text{subj.} \begin{cases} \bar{\mu} \geq 0 & \\ \bar{\mu}_s \geq \bar{\mu}_{st} & \forall st \in \mathcal{E} \,|\, y_s \neq z_s\,, \\ \bar{\mu}_t \geq \bar{\mu}_{st} & \forall st \in \mathcal{E} \,|\, y_s \neq z_s\,, \end{cases}
\end{aligned} \tag{199}
$$

where we denoted $\bar{\mu}_s = \mu_s(z_s)$, $\bar{\mu}_{st} = \mu_{st}(z_s, z_t)$ and

$$
\begin{aligned}
\bar{g}_s(y_s) &= f_s(z_s) - f_s(y_s) + \sum_{t \,|\, st \in \tilde{\mathcal{E}}} \Delta_{st}(y_s)\,, \\
\bar{g}_{st}(y_{st}) &= f_{st}(z_s, z_t) - f_{st}(y_s, y_t) - \Delta_{st}(y_s) - \Delta_{ts}(y_t)\,, \\
\Delta_{ts}(y_t) &= \min_{i \neq z_s}(f_{st}(i, z_t) - f_{st}(i, y_t))\,.
\end{aligned} \tag{200}
$$

The dependence of g on y is made explicit since we are going to optimize in y. Because we have $\bar{g}_s(y_s) = 0$ for $y_s{=}z_s$, we can safely extend the constraints of the problem to all edges, independently of the set $\mathcal{W}$. Moreover, if $\bar{g}_{st} \geq 0$ setting $\bar{\mu}_{st}$ to 0 does not lead to an increase of the objective. We therefore can exclude all such pairs from the problem by replacing $\bar{g}_{st}$ with

$$
\hat{g}_{st} = \min(0, \bar{g}_{st})\,. \tag{201}
$$

After this simplification we claim the following.

Statement 41. Problem (199) attains 0 if and only if

$$
\min_{x \in \{0,1\}^{\mathcal{V}}} \Big(\sum_{s \in \mathcal{V}} \hat{g}_s x_s + \sum_{st \in \mathcal{E}} \hat{g}_{st} x_s x_t \Big) = 0\,, \tag{202}
$$

where we introduced $\hat{g}_s = \bar{g}_s$ for consistency of notation.

Proof. It can be verified that any solution of (202) yields a feasible solution to (199) by assigning $\bar{\mu}_s = x_s$ and $\bar{\mu}_{st} = x_s x_t$. We have to show that a feasible solution to (199) of a negative cost allows to construct a $0{-}1$ solution of negative cost. Let $\bar{\mu}$ be feasible to (199) and has negative cost. Clearly $\gamma\bar{\mu}$ for any $\gamma > 0$ is also feasible and has negative cost. We therefore can assume $\bar{\mu} \leq 1$. With this constraint, and non-positivity of $\hat{g}_{st}$ we recognize that the problem is the linear relaxation of submodular quadratic pseudo-Boolean function minimization, which is tight. □

It will be convenient to consider the objective of (202) as a set function. Let $\hat{E}_g(\mathcal{A}, y)$ denote the objective of (202) for $\mathcal{A} = \{s \in \mathcal{V} \,|\, x_s = 1\}$. It can be expanded as

$$
\hat{E}_g(\mathcal{A}, y) = \sum_{s \in \mathcal{A}} \hat{g}_s(y_s) + \sum_{st \in (\mathcal{A}, \bar{\mathcal{A}})_{\mathcal{E}}} \hat{g}_{st}(y_{st})\,. \tag{203}
$$

As noted above, this function is submodular in the first argument. Consider the intersection of all minimizers (the minimal element) $\mathcal{A} = \bigcap \operatorname{argmin}_{\mathcal{A}} \hat{E}_g(\mathcal{A}, y)$. Suppose $\hat{E}_g(\mathcal{A}, y) < 0$ which means that the projection is not improving. It is easy to show that it must be $y_s \neq z_s$ for s from $\mathcal{A}$. Our goal now is to show that for labelings y' which are modifications of y keeping the label or switching to z in any pixel, P cannot be improving unless $y'_{\mathcal{A}} = z_{\mathcal{A}}$. That is, pixels in $\mathcal{A}$ may not be eliminated by any projection of this form. By following this way, we can discard sequentially all the pixels which cannot be eliminated by any projection of the form until there left only those which can.

Let $\mathcal{U}$ denote the set of pixels, where the labelings y' coincide with y. If only $\mathcal{U} \cap \mathcal{A} \neq 0$ the projection corresponding to y' will not be improving, as proven next.

Theorem 42. Let $\mathcal{A} = \bigcap \operatorname{argmin}_{\mathcal{A}} \hat{E}_g(\mathcal{A}, y)$. Let $\mathcal{U} \subset \mathcal{V}$ such that $\mathcal{U} \cap \mathcal{A} \neq \emptyset$. Let $y'_s = y_s$ if $s \in \mathcal{U}$ and $y'_s = z_s$ if $s \notin \mathcal{U}$.

Then $\hat{E}_g(\mathcal{A}, y') < 0$.

Proof. Assume for contradiction that $\hat{E}_g(\mathcal{A}, y) \geq 0$. We know that $\hat{E}_g(\emptyset, y) = 0$. From the minimality of $\mathcal{A}$ (it is the intersection of all minimizers), it must be that $\hat{E}_g(\mathcal{A}, y) = 0$ and $\mathcal{A} \subset \emptyset$. This contradicts to $\mathcal{U} \cap \mathcal{A} \neq \emptyset$. We therefore have $\hat{E}_g(\mathcal{A}, y) < 0$.

If $A \subset \mathcal{U}$ we have $\hat{E}_g(x, y') = \hat{E}_g(x, y) < 0$ and the theorem is proven. Otherwise we have $A \backslash \mathcal{U} \subsetneq \mathcal{A}$.

Since $\mathcal{A}$ is minimizer it must be $\hat{E}_g(\mathcal{A} \backslash \mathcal{U}, y) \geq \hat{E}_g(\mathcal{A}, y)$. Because $\mathcal{A} \backslash \mathcal{U} \subsetneq \mathcal{A}$, an equality would contradict to minimality of $\mathcal{A}$. We have theretofore $\hat{E}_g(\mathcal{A} \backslash \mathcal{U}, y) > \hat{E}_g(\mathcal{A}, y)$.

To prove the theorem we will show that the following inequality holds:

$$\hat{E}_g(\mathcal{A}, y') \leq \hat{E}_g(\mathcal{A}, y) - \hat{E}_g(A \backslash \mathcal{U}, y)\,. \tag{204}$$

Because the RHS is < 0 it would imply the desired result. Let us expand all summands of the inequality as follows

$$\begin{aligned}
&\hat{E}_g(\mathcal{A}, y) - \hat{E}_g(\mathcal{A} \backslash \mathcal{U}, y) - \hat{E}_g(\mathcal{A}, y') \\
&= \sum_{s \in \mathcal{A}} \hat{g}_s(y) + \sum_{st \in (\mathcal{A}, \mathcal{A})_{\mathcal{E}}} \hat{g}_{st}(y) - \sum_{s \in \mathcal{A} \backslash \mathcal{U}} \hat{g}_s(y) - \sum_{st \in (\mathcal{A} \backslash \mathcal{U}, \mathcal{A} \backslash \mathcal{U})_{\mathcal{E}}} \hat{g}_{st}(y) - \sum_{s \in \mathcal{A}} \hat{g}_s(y') - \sum_{st \in (\mathcal{A}, \mathcal{A})_{\mathcal{E}}} \hat{g}_{st}(y') \\
&= -\sum_{s \in \mathcal{A} \backslash \mathcal{U}} \hat{g}_s(y') + \sum_{st \bar{\in} (\mathcal{A}, \mathcal{A} \backslash \mathcal{U})_{\mathcal{E}}} [\hat{g}_{st}(y) - \hat{g}_{st}(y')] \\
&\overset{(a)}{=} \sum_{st \bar{\in} (\mathcal{A}, \mathcal{A} \backslash \mathcal{U})_{\mathcal{E}}} [\hat{g}_{st}(y) - \hat{g}_{st}(y')]\,,
\end{aligned} \tag{205}$$

Where equality (a) follows from that for $s \in A \backslash \mathcal{U}$, we have $y'_s = z_s$ and from

$$\begin{aligned}
\Delta_{st}(y'_s) &= \min_{j \neq z_t} (f_{st}(z_s, j) - f_{st}(z_s, j)) = 0\,, \\
\bar{g}_s(y'_s) &= f_s(z_s) - f_s(z_s) + \sum_{st \bar{\in} \mathcal{E}} \Delta_{st}(y'_s) = 0\,.
\end{aligned} \tag{206}$$

It remains to verify that for every $st \bar{\in} (\mathcal{A}, \mathcal{A} \backslash \mathcal{U})_{\mathcal{E}}$ there holds

$$\min(0, \bar{g}_{st}(y'_{st})) \leq \min(0, \bar{g}_{st}(y_{st}))\,. \tag{207}$$

We will consider the following cases:

- Case $y'_{st} = (z_s, z_t)$. We have $\bar{g}_{st}(y'_{st}) = 0$. The inequality (207) follows from that $\min(0, g_{st}(y_{st})) \leq 0$.
- Case $y'_{st} = (y_s, z_t)$. Let us verify that the following inequality holds:

$$\bar{g}_{st}(y') \leq \bar{g}_{st}(y)\,. \tag{208}$$

Because function $\min(0, \cdot)$ is non-decreasing, it would imply (207). Expanding $\bar{g}_{st}$, we obtain

$$\begin{aligned}
&\bar{g}_{st}(y') - \bar{g}_{st}(y) \\
&= \Big(f_{st}(z_s, z_t) - f_{st}(y_s, z_t) - \Delta_{st}(y_s) - 0\Big) \\
&- \Big(f_{st}(z_s, z_t) - f_{st}(y_s, y_t) - \Delta_{st}(y_s) - \Delta_{ts}(y_t)\Big) \\
&= \Delta_{ts}(y_t) - (f_{st}(y_s, z_t) - f_{st}(y_s, y_t)) \\
&= \min_{i \neq z_s} (f_{st}(i, z_t) - f_{st}(i, y_t)) - (f_{st}(y_s, z_t) - f_{st}(y_s, y_t)) \leq 0\,,
\end{aligned} \tag{209}$$

which holds true.

- Case $y'_{st} = (z_s, y_t)$ is expanded similarly to the above.

By backtracking the chain of implications, we obtain that $\hat{E}_g(\mathcal{A}, y') < 0$. □

By this theorem, any y', which does not coincide with z on the subset $\mathcal{A}$, corresponds to a projection that is not Λ-improving. We therefore arrive at the Algorithm 4.

5 Distributed Mincut/Maxflow Algorithms

Minimum *s-t* cut (MINCUT) is a classical combinatorial problem with applications in many areas of science and engineering. In this chapter, we develop a novel distributed algorithm for the MINCUT problem. Motivated by applications like volumetric segmentation in computer vision, we aim at solving large sparse problems. When the problem does not fully fit in the memory, we need to either process it by parts, looking at one part at a time, or distribute across several computers. Many MINCUT/MAXFLOW algorithms are designed for the shared memory architecture and do not scale to the setting, when the memory has to be divided among the computation units. We start by a more detailed overview of models and optimization techniques in computer vision, where the MINCUT problem is employed, and give examples of our test problems.

MINCUT in Computer Vision In some cases, an applied problem is formulated directly as a MINCUT. More often, however, MINCUT problems in computer vision originate from the Energy minimization framework (maximum a posteriori solution in a Markov random field model). Submodular Energy minimization problems completely reduce to MINCUT (Ishikawa 2003; Schlesinger and Flach 2006). When the energy minimization is intractable, MINCUT is employed in relaxation and local search methods. The linear relaxation of pairwise Energy minimization with 0-1 variables reduces to MINCUT (Boros et al. 1991; Kolmogorov and Rother 2007) as well as the relaxation of problems reformulated in 0-1 variables (Kohli et al. 2008). Expansion-move, swap-move (Boykov et al. 1999) and fusion-move (Lempitsky et al. 2010) algorithms formulate a local improvement step as a MINCUT problem. Many applications of MINCUT in computer vision use graphs of a regular structure, with vertices arranged into an N-D grid and edges uniformly repeated, *e.g.*, 3D segmentation models illustrated in Figure 12(c), 3D reconstruction models, Figure 12(b). Because of such regular structure, the graph itself need not be stored in the memory. Only the edge capacities need to be stored, allowing relatively large instances to be solved by a specialized implementation. However, in many cases, it is advantageous to have a non-regular structure, *e.g.*, in stereo with occlusions in Figure 12(a), in 3D reconstruction with adaptive tetrahedral volume (Labatut et al. 2009; Jancosek and Pajdla 2011). Such applications would benefit from a large-scale generic MINCUT solver.

Distributed Computation The previous research mostly focused on speeding up MINCUT by parallel computation in the shared memory model. We consider a *distributed* memory model, which assumes that the computation units have their own separate memory and exchanging the information between them is expensive. A distributed algorithm has therefore to divide the computation and the problem data between the units and keep the communication rate low.

We will consider distributed algorithms, operating in the following two practical usage modes:

- Sequential (or *streaming*) mode, which uses a single computer with a limited memory and a disk storage, reading, processing and writing back a portion of data at a time.

- Parallel mode, in which the units are thought as computers in a network.

We propose new algorithms for both cases, prove their correctness and termination guarantees. In the assessment of complexity, we focus on the costly operations such as load-unload of the data in the streaming mode or message exchanges in the parallel mode. More specifically, we call a *sweep* the event when all units of a distributed algorithm recalculate their data once. Sweeps in our algorithms correspond to outer iterations and their number is roughly proportional to the amount of communication in the parallel mode or disk operations in the streaming mode. While there are algorithms with better bounds in terms of elementary operations, our algorithms achieve lower communication rates.

Previous Work A variant of path augmentation algorithm was shown by Boykov and Kolmogorov (2004) to have the best performance on computer vision problems among sequential solvers. There were several proposals how to parallelize it. Partially distributed implementation (Liu and Sun 2010) augments paths within disjoint regions first and then merges regions hierarchically. In the end, it still requires finding augmenting paths in the whole problem. Therefore, it cannot be used to solve a large problem by distributing it over several computers or by using a limited memory and a disk storage. For the shared memory model Liu and Sun (2010) reported a near-linear speed-up with up to 4 CPUs for 2D and 3D segmentation problems.

Strandmark and Kahl (2010) obtained a distributed algorithm using a dual decomposition approach. The subproblems are MINCUT instances on the parts of the graph (regions) and the master problem is solved using the subgradient method. This approach requires solving MINCUT subproblems with real valued capacities and does not have a polynomial bound on the number of iterations. The integer algorithm proposed by Strandmark and Kahl (2010) is not guaranteed to terminate. Our experiments (§5.6.3) showed that it did not terminate on some of the instances in 1000 sweeps. In §5.9, we relate dual variables in this method to flows.

The push-relabel algorithm (Goldberg and Tarjan 1988) performs many local atomic operations, which makes it a good choice for a parallel or distributed implementation. A distributed version (Goldberg 1991) runs in $O(n^2)$ time using $O(n)$ processors and $O(n^2\sqrt{m})$ messages, where n is the number of vertices and m is the number of edges in the problem. However, for a good practical performance it is crucial to implement the gap relabel and the global relabel heuristics (Cherkassky and Goldberg 1994). The global relabel heuristic can be parallelized (Anderson and Setubal 1995), but it is difficult to distribute. We should note, however, that the global relabel heuristic was not essential in the experiments with computer vision problems we made (§5.6.2). Delong and Boykov (2008) proposed a coarser granulation of push-relabel operations, associating a subset of vertices (a region) to each processor. Push and relabel operations inside a region are decoupled from the rest of the graph. This allows to process several non-interacting regions in parallel or run in a limited memory, processing few regions at a time. The gap and relabel heuristics, restricted to the regions (Delong and Boykov 2008) are powerful and distributed at the same time. Our work was largely motivated by Delong and Boykov (2008) and the remark that their approach might be extendible to augmenting path algorithms. However, our first attempt to prioritize augmentation to the boundary vertices by the shortest distance to the sink did not lead to a correct algorithm.

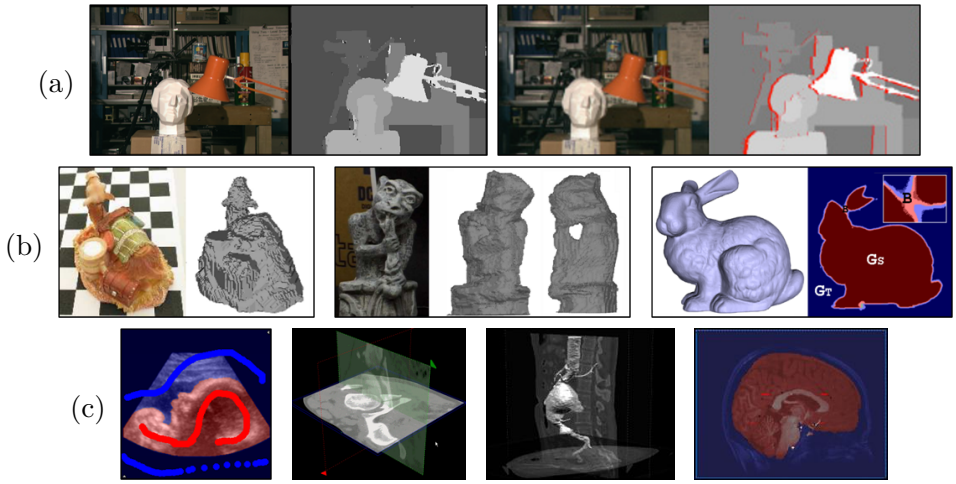


Figure 12 Examples of labeling problems in computer vision solved via MAXFLOW. (a) Stereo and stereo with occlusions (Boykov et al. 1998), (Kolmogorov and Zabih 2001). (b) 3D reconstruction (Boykov and Lempitsky 2006; Lempitsky et al. 2006) and surface fitting (Lempitsky and Boykov 2007). (c) 3D segmentation (Boykov and Jolly 2001; Boykov 2003; Boykov and Funka-Lea 2006). The instances are published at the University of Western Ontario web pages (2008) for benchmarking MAXFLOW implementations.

Other Related Work The following works do not consider a distributed implementation but are relevant to our design. The Partial Augment-Relabel algorithm (PAR) by Goldberg (2008) augments a path of length k in each step. It may be viewed as a lazy variant of push-relabel, where actual pushes are delayed until it is known that a sequence of k pushes can be executed. The algorithm by Goldberg and Rao (1998) incorporates the notion of a length function and a valid labeling w.r.t. this length. It can be seen that the labeling maintained by our algorithm corresponds to the length function assigning 1 to boundary edges and 0 to intra-region edges. Goldberg and Rao (1998) used such generalized labeling in the context of the blocking flow algorithm but not within the push-relabel.

Outline We first revisit the algorithm of Delong and Boykov (2008) for the case of a fixed partition into regions (§5.2). We study a sequential variant and a novel parallel variant of their algorithm, which allows running computations concurrently on neighboring interacting regions using a conflict resolution similar to the asynchronous parallel push-relabel (Goldberg 1991). We prove that both variants have a tight $O(n^2)$ bound on the number of sweeps. The new algorithm, we construct (§5.3), works with the same partition of the graph into regions but is guided by a different distance function than the push-relabel one.

Given a fixed partition into regions, we introduce a distance function, which counts the number of region boundaries crossed by a path to the sink. Intuitively, it corresponds to the amount of costly operations – network communications or loads-unloads of the regions in the streaming mode. The algorithm maintains a labeling, which is a lower bound on the distance function. Within a region, we first augment paths to the sink and then paths to the boundary vertices prioritized by the lowest label. Thus the flow is pushed out of the region in the direction given by the distance estimate. We present a sequential and parallel versions of the algorithm, which terminate in $O(|\mathcal{B}|^2)$ sweeps, where $\mathcal{B}$ is the set of all boundary vertices (incident to inter-region edges).

We describe additional heuristics and an efficient implementation of both push-relabel

and augmented-path based distributed algorithms in §5.4. The proposed algorithms are evaluated on instances of MINCUT problems collected and published by the Computer Vision Research Group at the University of Western Ontario (illustrated in Figure 12). The results are compared against the state-of-the-art sequential and parallel solvers (§5.6). We also studied the behavior of the algorithms w.r.t. problem size, granularity of the partition, *etc.* In §5.7, we study the question to which extend the exchange of information between parts is necessary. It turns out, that for simple problems (*e.g.*, stereo) a large part of the optimal solution can be recovered from the individual parts, while for more complicated problems (3D segmentation) this is no longer the case. The interactions between regions are really necessary. In §5.8, we construct an example of MINCUT, on which the push-relabel discharge achieves its worst case bound on the number of sweeps, while the new algorithm terminates in a constant number of sweeps. In §5.9, we show how the dual decomposition approach for min-cut by (Strandmark and Kahl 2010) can be formulated in terms of flows and reveal the relation to our partitioning scheme.

5.1 MINCUT and Push-Relabel

> *"Needless to say, their version not only has its own real beauty, but is somewhat "sexy" running depth first search on the layered network constructed by (extended) breadth first search".*
>
> Y. Dinitz,
> about Even and Itai's version of the maximum flow algorithm.

We solve MINCUT problem by finding a maximum preflow[1]. In this section, we give basic definitions and introduce the push-relabel framework of Goldberg and Tarjan (1988). While we assume the reader is familiar with mincut/maxflow, we explain some known results using the notation adjusted for the needs of this chapter.

In the classical framework of minimum cut and maximum flow, the flow augmentation transforms a minimum cut problem into an equivalent one on the residual network (preserving costs of all cuts up to a constant). However, there is no equivalent minimum cut problem corresponding to the augmentation of a *preflow*. In the push-relabel approach of Goldberg and Tarjan (1988), this is not essential, as only single residual arcs need to be considered and algorithms can be formulated as working with a pair of a network and a preflow. In this work, we need to deal with residual paths and the reachability in the residual network. We therefore use the extended definition of the minimum cut problem, which includes a flow excess (or supply) in every vertex. After this extension, the family of equivalent min-cut problems becomes closed under preflow augmentations. This allows us to formulate algorithms more conveniently as working with the current residual network and constructing a preflow increment. This point is illustrated in Figure 13.

By a *network* we call the tuple $G = (V, E, s, t, c, e)$, where V is a set of vertices; $E \subset V \times V$ is the set of edges, thus (V, E) is a directed graph; $s, t \in V$, $s \neq t$, are *source* and *sink*, respectively; $c\colon E \to \mathbb{N}_0$ is a capacity function; and $e\colon V \to \{0, 1, \ldots, \infty\}$, $e(t) = 0$, $e(s) = \infty$ is an *excess* function. We also denote $n = |V|$ and $m = |E|$.

[1] A maximum preflow can be completed to a maximum flow using the flow decomposition, in $O(m \log m)$ time. Because we are primarily interested in the minimum cut, we do not consider this step or whether it can be distributed.

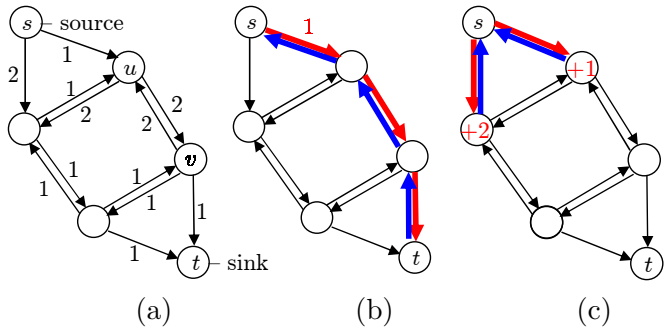


Figure 13 (a) Example of a network with indicated edge capacity function. (b) Augmenting path approach: send flow from the source to the sink along a path. The residual network defines an equivalent min-cut problem. (c) Push-relabel approach: the preflow is pushed over arcs in all directions, prioritized by the shortest distance to the sink. The equivalent min-cut problem is defined by a network with excess.

For any sets $X, Y \subset V$ we denote $(X, Y)_E = (X \times Y) \cap E$. For $C \subset V$ such that $s \in C$, $t \notin C$, the set of edges $(C, \bar{C})_E$, with $\bar{C} = V \backslash C$ is called an *s-t cut*. The MINCUT problem is

$$\min \Big\{ \sum_{(u,v)\in(C,\bar{C})_E} c(u,v) + \sum_{v\in\bar{C}} e(v) \;\Big|\; C \subset V,\, s \in C,\, t \in \bar{C} \Big\}. \tag{210}$$

The objective is called the *cost* of the cut. Note, that excess in this problem can be equivalently represented as additional edges from the source, but we prefer the explicit form. Without the loss of generality, we assume that E is symmetric. If not, the missing edges are added and assigned a zero capacity.

A *preflow* in G is the function $f\colon E \to \mathbb{Z}$ satisfying the following constraints:

$$f(u,v) \le c(u,v) \quad \forall (u,v) \in E\,, \tag{211a}$$

(capacity constraint)

$$f(u,v) = -f(v,u) \quad \forall (u,v) \in E\,, \tag{211b}$$

(antisymmetry)

$$e(v) + \sum_{u\,|\,(u,v)\in E} f(u,v) \ge 0 \quad \forall v \in V\,. \tag{211c}$$

(preflow constraint)

The constraint (211b) removes the redundancy in the otherwise independent flow values on (u, v) and (v, u) (positive flows should naturally cancel each other) and shortens the equations at the same time.

A *residual network* w.r.t. preflow f is a network $G_f = (V, E, s, t, c_f, e_f)$ with the capacity and excess functions given by

$$c_f = c - f\,, \tag{212a}$$

$$e_f(v) = e(v) + \sum_{u\,|\,(u,v)\in E} f(u,v), \quad \forall v \in V\backslash\{s,t\}\,. \tag{212b}$$

By constraints (211), it is $c_f \ge 0$ and $e_f \ge 0$. The costs of all s-t cuts differ in G and G_f by a constant called the *flow value*, $|f| = \sum_{u\,|\,(u,t)\in E} f(u,t)$. This can be easily verified by substituting c_f and e_f into (210) and expanding. Network G_f is thus *equivalent* to network G up to the constant $|f|$. Since all cuts in G_f are non-negative, $|f|$ is a lower

bound on the cost of a cut in G. The problem of maximizing this lower bound, *i.e.* finding a maximum preflow:

$$\max_f |f| \quad \text{subj. constraints (211)} \tag{213}$$

is dual to MINCUT. The value of a (non-unique) maximum preflow equals to the cost of a (non-unique) minimum cut. The solutions are related as explained below.

We say that $w \in V$ is *reachable* from $v \in V$ in the network G_f if there is a path (possibly of length 0) from v to w composed of edges with strictly positive residual capacities c_f (a *residual path*). This relation is denoted by $v \to w$.

Let us consider a residual path from v to w such $e_f(v) > 0$. *Augmentation* increases the flow by $\Delta > 0$ on all forward edges of the path, and decreases it on all reverse edges, where Δ does not exceed the residual capacities of the forward arcs or $e_f(v)$. In the result, the excess $e_f(v)$ is decreased and excess $e_f(w)$ is increased. Augmenting paths to the sink increases the flow value. In the augmenting path approach, the problem (213) is solved by repeatedly augmenting residual paths from vertices having excess (*e.g.*, source) to the sink.

If w is not reachable from v in G_f we write $v \not\to w$. For any $X, Y \subset V$, we write $X \to Y$ if there exist $x \in X$, $y \in Y$ such that $x \to y$. Otherwise we write $X \not\to Y$.

A preflow f is maximum iff there is no residual path to the sink which can be augmented. This can be written as $\{v \,|\, e_f(v) > 0\} \not\to t$ in G_f. In this case, the cut $(\bar{T}, T)_E$ with $T = \{v \in V \,|\, v \to t \text{ in } G_f\}$ has the value 0 in G_f. Because all cuts are non-negative, it is a minimum cut.

General Push-relabel A *Distance* function $d^*\colon V \to \{0, 1, \ldots, n\}$ in G_f assigns to $v \in V$ the length of the shortest residual path from v to t, or n if no such path exists. The (non-unique) shortest path cannot have loops, thus its length is not greater than $n - 1$. Let us denote $d^\infty = n$.

A *labeling* $d\colon V \to \{0, 1, \ldots, d^\infty\}$ is *valid* in G_f if $d(t) = 0$ and $d(u) \le d(v) + 1$ for all $(u, v) \in E$ such that $c_f(u, v) > 0$. Any valid labeling is a lower bound on the distance d^* in G_f, however not every lower bound is a valid labeling.

A vertex v is called *active* w.r.t. (f, d) if $e_f(v) > 0$ and $d(v) < d^\infty$.

The definitions of reachability and validity are given w.r.t. the residual network G_f, however expressions like "$v \to w$ in G" or "d is valid in G" are also correct, and will be needed later on. In particular, we will consider algorithms making some large steps, where a preflow increment f is computed and then applied to the initial network by assigning $G := G_f$. After that, the algorithm continues with G and resets f.

To ensure that residual paths do not go through the source (and for reasons of efficiency) we make all edges from the source saturated during the following `Init` procedure, common to all algorithms in this chapter.

Procedure Init

```
   /* saturate source edges                                   */
1  f(s,v) := c(s,v) ∀(s,v) ∈ E;
2  G := G_f; f := 0;                                  /* apply preflow */
3  d := 0, d(s) := d^∞;                             /* initialize labels */
```

The generic push-relabel algorithm by Goldberg and Tarjan (1988) maintains a preflow f and a valid labeling d. It starts with `Init` and applies the following `Push` and `Relabel` operations while possible:

- `Push`(u, v): applicable if u is active and $c_f(u, v) > 0$ and $d(u) = d(v) + 1$. The operation increases $f(u, v)$ by Δ and decreases $f(v, u)$ by Δ, where $\Delta = \min(e_f(u), c_f(u, v))$.
- `Relabel`(u): applicable if u is active and $(\forall v | (u, v) \in E, c_f(u, v) > 0)\ d(u) \leq d(v)$. It sets $d(u) := \min\left(d^\infty, \min\{d(v) + 1 \mid (u, v) \in E,\ c_f(u, v) > 0\}\right)$.

If u is active then either `Push` or `Relabel` operation is applicable to u. The algorithm preserves validity of the labeling and stops when there are no active vertices. For any u such that $e_f(u) > 0$, we have $d(u) = d^\infty$. Therefore $d^*(u) = d^\infty$ and $u \nrightarrow t$ in G_f, so f is a (non-unique) maximum preflow.

5.2 Region Discharge Revisited

We now review the approach of Delong and Boykov (2008) and reformulate it for the case of a fixed graph partition. We introduce generic sequential and parallel algorithms, which will be applied with both push-relabel and augmenting path approaches.

Delong and Boykov (2008) introduced the following operation. The *discharge* of a *region* $R \subset V \backslash \{s, t\}$ applies `Push` and `Relabel` to $v \in R$ until there are no active vertices left in R. This localizes computations to R and its *boundary*, defined as

$$B^R = \{w \mid \exists u \in R\ (u, w) \in E, w \notin R,\ w \neq s, t\}\,. \tag{214}$$

When a `Push` is applied to an edge $(v, w) \in (R, B^R)$, the flow is sent out of the region. We say that two regions $R_1, R_2 \subset V \backslash \{s, t\}$ *interact* if $R_1 \cap R_2 \neq \emptyset$ or $R_1 \cap B^{R_2} \neq \emptyset$. That is, when they share vertices or they are connected by an edge. Because `Push` and `Relabel` operations work on the individual edges, discharges of non-interacting regions can be performed in parallel. The algorithm proposed by Delong and Boykov (2008) repeats the following steps until there are no active vertices in V:

1. Select several non-interacting regions, containing active vertices.
2. Discharge the selected regions in parallel, applying region-gap and region-relabel heuristics.
3. Apply the global gap heuristic.

All heuristics (global-gap, region-gap, region-relabel) serve to improve the distance estimate. They are very important in practice, but do not affect the theoretical properties and will be discussed in §5.4 devoted to the implementation.

5.2.1 Region Network

We now take a different perspective on the algorithm. We consider each region discharge as a proper subproblem to be solved. Given the states of the boundary edges on the input (labels and excess), the region discharge of region R returns a flow and a labeling. To define it formally, we first single out the subnetwork, on which region discharge will work.

Region network $G^R = (V^R, E^R, s, t, c^R, e^R)$ has set of vertices $V^R = R \cup B^R \cup \{s, t\}$; set of edges $E^R = (R \cup \{s, t\}, R \cup \{s, t\})_E \cup (R, B^R)_E \cup (B^R, R^R)_E$; capacities $c^R(u, v) = c(u, v)$ if $(u, v) \in E^R \backslash (B^R, R)_E$ and 0 otherwise; and excess $e^R = e|_{R \cup \{s,t\}}$ (the restriction of function e to its subdomain $R \cup \{s, t\}$). This subnetwork is illustrated in Figure 14(b). Note that the capacities of edges coming from the boundary, $(B^R, R)_E$,

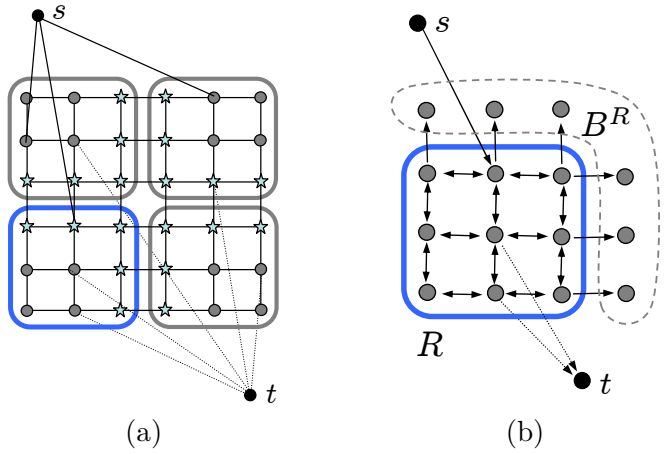


Figure 14 (a) Partition of a network into 4 regions and the boundary set $\mathcal{B}$ depicted by stars. (b) The region network corresponding to the highlighted region in (a).

are set to zero. Indeed, these edges belong to a neighboring region network. The region discharge operation of Delong and Boykov (2008), which we refer to as Push-relabel Region Discharge (PRD), can now be defined as shown in Procedure `PRD`.

Procedure $(f, d) = \text{PRD}(G^R, d)$

`/* assume` $d\colon V^R \to \{0, \dots, d^\infty\}$ `valid in` G^R `*/`
1 **while** $\exists v \in R$ *active* **do**
2 apply `Push` or `Relabel` to v; `/* changes` f `and` d `*/`
3 apply region gap heuristic (§5.4); `/* optional */`

5.2.2 Generic Region Discharging Algorithms

While Delong and Boykov (2008) selected the regions dynamically, trying to divide the work evenly between CPUs in each iteration and cover the most of the active vertices, we restrict ourselves to a fixed collection of regions $(R^k)_{k=1}^K$ forming a partition (disjoint union) of $V \backslash \{s, t\}$. With respect to this partition, we will use shortened notations B^k, G^k, V^k, E^k, c^k, e^k to denote the corresponding region boundary B^{R_k}, region network G^{R_k} and the respective compounds of region network G^{R_k}. We also define the *boundary* $\mathcal{B} = \bigcup_k B^k$, which is the set of all vertices incident to inter-region edges (Figure 14(a)).

We now define generic sequential and parallel algorithms which use a black-box `Discharge` function as a subroutine. The sequential algorithm (Algorithm 5) takes regions from the partition one-by-one and applies the `Discharge` operation to them until there are no active vertices in either region left. The parallel algorithm (Algorithm 6) calls `Discharge` for all regions concurrently and then resolves conflicts in the flow similarly to the asynchronous parallel push-relabel of Goldberg and Tarjan (1988). A conflict occurs if two interacting regions increase their labels on the vertices of a boundary edge (u, v) simultaneously and try pushing flow over it (in their respective region networks). In such a case, we accept the labels, but do not allow the flow to cross the boundary in one of the directions by the following construction. In step 5 of Algorithm 6, boundary edges, where the validity condition is potentially violated, are assigned $\alpha = 0$. The flow fusion in step 6 disables the flow on such edges (the flow

Algorithm 5: Sequential Discharging

```
1 Init;
2 while there are active vertices do                                  /* a sweep */
3     for k = 1, ... K do
4         Construct G^k from G;
5         (f', d') := Discharge(G^k, d|_{V^k});
6         G := G_{f'};                                          /* apply f' to G */
7         d|_{R^k} := d'|_{R^k};                                /* update labels */
8         apply global gap heuristic (§5.4);                        /* optional */
9 Compute the reachability v → t in G, ∀v (§5.4.2);
```

Algorithm 6: Parallel Discharging

```
1  Init;
2  while there are active vertices do                                 /* a sweep */
       /* discharge all regions in parallel                                    */
3      (f'_k, d'_k) := Discharge(G^k, d|_{V^k}) ∀k;
4      d'|_{R^k} := d'_k|_{R^k} ∀k;                                /* fuse labels */
       /* determine valid pairs                                                */
5      α(u, v) := [[d'(u) ≤ d'(v) + 1]]  ∀(u, v) ∈ (B, B);
       /* fuse flows                                                           */
6      f'(u, v) := { α(v, u) f'_k(u, v) + α(u, v) f'_j(u, v)   if (u, v) ∈ (R^k, R^j),
                   { f'_k(u, v)                                 if (u, v) ∈ (R^k, R^k);
7      G := G_{f'};                                          /* apply f' to G */
8      d := d';                                              /* update labels */
9      global gap heuristic (§ 5.4);                             /* optional */
10 Compute the reachability v → t in G, ∀v (§5.4.2);
```

going "upwards"). As will be proven later, this correction restores the validity. The actual implementation does not maintain the full network G, only the separate region networks. This is in contrast to Delong and Boykov (2008), who perform all operations in the global network G.

In the case when the abstract `Discharge` procedure is implemented by `PRD`, the sequential and parallel algorithms correspond to the push-relabel approach and will be referred to as S-PRD and P-PRD respectively. S-PRD is a sequential variant of Delong and Boykov (2008) and P-PRD is a novel parallel variant. As was mentioned above, the original algorithm by Delong and Boykov (2008) allows to discharge only non-interacting regions in parallel (in this case there are no conflicts). To discharge all regions, this approach would require sequentially selecting subsets of non-interacting regions for processing.[2] Our parallel algorithm applies ideas of Goldberg and Tarjan (1988) to remove this limitation and process all regions in parallel.

We prove below that both S-PRD and P-PRD terminate with a valid labeling in at most $2n^2$ sweeps. Parallel variants of push-relabel (Goldberg 1987) have the same bound on the number of sweeps. However, they perform much simpler sweeps, processing every

[2] The number of sequential phases required is equal to the minimal coloring of the region interaction graph in a general case, *i.e.* 2 for bipartite graph, and so on.

vertex only once, compared to S/P-PRD. A natural question is whether $O(n^2)$ bound cannot be tightened for S/P-PRD. In §5.8, we give the example of a graph, its partition into two regions and a valid sequence of `Push` and `Relabel` operations, implementing S/P-PRD which takes $\Omega(n^2)$ sweeps to terminate.[3] The number of inter-region edges in this example is constant, which shows that a better bound in terms of this characteristic is not possible.

5.2.3 Complexity of Sequential Push-relabel Region Discharging

Our proof follows the main idea of the similar result for parallel push-relabel by Goldberg (1987). The main difference is that we try to keep `Discharge` operation as abstract as possible. Indeed, it will be seen that proofs of termination of other variants follow the same pattern, using several important properties of the `Discharge` operation, abstracted from the respective algorithm. Unfortunately, to this end we do not have a unified proof, so we will analyze all cases separately.

Statement 43 (Properties of PRD)**.**
Let $(f', d') = \texttt{PRD}(G^R, d)$, then
1. there are no active vertices in R w.r.t. (f', d'), (*optimality*)
2. $d' \geq d$, $d'|_{B^R} = d|_{B^R}$, (*labeling monotony*)
3. d' is valid in $G^R_{f'}$, (*labeling validity*)
4. $f'(u, v) > 0 \Rightarrow d'(u) > d(v)$, $\forall (u, v) \in E^R$. (*flow direction*)

Proof. 1. Optimality. This is the stopping condition of `PRD`.
2,3. Labeling validity and monotony: labels are never decreased and the `Push` operation preserves labeling validity (Goldberg and Tarjan 1988). Labels not in R^k are not modified.
4. Flow direction: let $f'(u, v) > 0$, then there was a push operation from u to v in some step. Let $\tilde{d}$ be the labeling in this step. We have $\tilde{d}(u) = \tilde{d}(v) + 1$. Because labels never decrease, $d'(u) \geq \tilde{d}(u) > \tilde{d}(v) \geq d(v)$. □

These properties are sufficient to prove correctness and the complexity bound of S-PRD. They are abstract from the actual sequence of `Push` and `Relabel` operation performed by `PRD` and for a given pair (f', d') they are easy to verify. For correctness of S-PRD, we need to verify that it maintains a labeling that is valid globally.

Statement 44. Let d be a valid labeling in G. Let f' be a preflow in G^R and d' be a labeling satisfying properties 2 and 3 of Statement 43. Extend f' to E by letting $f'|_{E \setminus E^R} = 0$ and extend d' to V by letting $d'|_{V \setminus V^R} = d|_{V \setminus V^R}$. Then d' is valid in $G_{f'}$.

Proof. We have that d' is valid in $G^R_{f'}$. For edges outside the region network, $(u, v) \in (V \setminus R, V \setminus R)_E$, it is $f'(u, v) = 0$ and d' coincides with d on $V \setminus R$. It remains to verify the validity on the boundary edges $(v, u) \in (B^R, R)_E$ in the case $c^R_f(v, u) = 0$ and $c_f(v, u) > 0$. These are the incoming boundary edges, which are zeroed in the network G^R. Because $0 = c^R_f(v, u) = c^R(v, u) - f(v, u) = -f(v, u)$, we have $c_f(v, u) = c(v, u)$. Since d was valid in G, $d(v) \leq d(u) + 1$. The new labeling d' satisfies $d'(u) \geq d(u)$ and $d'(v) = d(v)$. It follows that $d'(v) = d(v) \leq d(u) + 1 \leq d'(u) + 1$. Hence d' is valid in $G_{f'}$. □

[3] An algorithm is said to be $\Omega(f(n))$ if for some numbers c' and n_0 and all $n >= n_0$, the algorithm takes at least $c' f(n)$ time on *some* problem instance. Here we measure complexity in sweeps.

Similar to Goldberg (1987), we introduce the *potential function*

$$\Phi = \max\{d(v) \,|\, v \in V,\ v \text{ is active in } G\}. \tag{215}$$

This value may increase and decrease during the algorithm run, but the total number of times it can change is bounded. We first show that its increase is bounded for a region discharge on R by the total increase of the labeling.

Statement 45. Let (f', d') satisfy properties 2-4 of Statement 43. Let f' be extended to E by setting $f'|_{E\setminus E^R} = 0$ and d' be extended to V by setting $d'|_{V\setminus V^R} = d|_{V\setminus V^R}$. Let $G' = G_{f'}$ and Φ' be the new potential computed for the network G' and labeling d'. Then

$$\Phi' - \Phi \le \sum_{v\in R} [d'(v) - d(v)]\,. \tag{216}$$

Proof. Let the maximum in the definition of Φ' be attained at a vertex v, so $\Phi' = d'(v)$. Then either $v \notin V^R$, in which case $\Phi' \le \Phi$ (because the label and the excess of v in G and G' are the same), or $v \in V^R$ and there exists a path $(v_0, v_1, \dots v_l)$, $v_l = v$, $v_0, \dots v_{l-1} \in R$, such that $f'(v_{i-1}, v_i) > 0$, $i = 1 \dots l$ and v_0 is active in G. We have $\Phi \ge d(v_0)$, therefore

$$\begin{aligned} \Phi' - \Phi \le d'(v_l) - d(v_0) &= \sum_{i=1}^{l} [d'(v_i) - d'(v_{i-1})] + [d'(v_0) - d(v_0)] \\ &\overset{(a)}{\le} \sum_{i=0}^{l} [d'(v_i) - d(v_i)] \overset{(b)}{\le} \sum_{v\in R\cup B^R} [d'(v) - d(v)] \overset{(c)}{=} \sum_{v\in R} [d'(v) - d(v)]\,, \end{aligned} \tag{217}$$

where inequality (a) is due to the flow direction property (Statement 43.4) which implies $d'(v_{i-1}) > d(v_i)$. The inequality (b) is due to monotony property (Statement 43.2) and due to $v_i \subset R \cup B^R$. The equality (c) is due to $d'|_{B^R} = d|_{B^R}$. □

We can now state the termination.

Theorem 46. S-PRD terminates in at most $2n^2$ sweeps.

Proof. Labeling d does not exceed n for every vertex. Because there are n vertices, d can be increased n^2 times at most.

From Statement 45 it follows that the increase of Φ after the discharge of region R^k is bounded by the total increase of $d|_{R^k}$. Since regions are disjoint, the total increase of Φ after a sweep of S-PRD is bounded by the total increase of d.

If d has not increased during a sweep ($d' = d$) then Φ decreases at least by 1. Indeed, let us consider the set of vertices having the label greater or equal to the label of the highest active vertex, $H = \{v \,|\, d(v) \ge \Phi\}$. These vertices do not receive flow during all discharge operations due to the flow direction property. After discharging R^k, there are no active vertices in $R^k \cap H$ (Statement 43.1). Therefore, there are no active vertices in H after the full sweep.

In the worst case, Φ can increase by one n^2 times and decrease by one n^2 times. Therefore, there are no active vertices in G in at most $2n^2$ sweeps and the algorithm terminates. □

When the algorithm terminates, it outputs a network G, equivalent to the initial one, a labeling d valid in G and guarantees that there are no active vertices w.r.t. d. This implies that in the current network G there are no paths from the vertices having

positive excess to the sink and the cut $(\bar{T}, T)_E$, with $T = \{v \mid v \to t \text{ in } G\}$ is one of the minimum cuts. The issue, how to compute the reachability $v \to t$ in G in a distributed fashion, utilizing d, rather than by breadth-first search in G is discussed in §5.4.2. This is the purpose of the last step in both of the algorithms.

5.2.4 Complexity of Parallel Push-relabel Region Discharging

We will now prove the validity and termination of the parallel algorithm using the results of previous section. Properties similar to Statement 43 will be proven for the fused flow and labeling (constructed at step 6 of Algorithm 6). The bound on the increase of the potential will follow for the whole network as if it was a single region.

Statement 47. Let d be a valid labeling in the beginning of a sweep of P-PRD. Then the pair of the fused flow and the labeling, (f', d'), satisfies:

1. $d' \geq d$; (*labeling monotony*)
2. d' is valid in $G_{f'}$; (*labeling validity*)
3. $f'(u, v) > 0 \Rightarrow d'(u) > d(v)\ \forall (u, v) \in E$. (*flow direction*)

Proof.
1. We have $d'|_{R^k} \geq d|_{R^k}$ for all k.
2. We have to prove the validity for the boundary edges, where the flow and the labeling are fused from different regions. It is sufficient to study the two regions case. Denote the regions R^1 and R^2. The situation is completely symmetric w.r.t. orientation of a boundary edge (u, v). Let $u \in R^1$ and $v \in R^2$. Let only $d'(v) \leq d'(u) + 1$ be satisfied and not $d'(u) \leq d'(v) + 1$. By the construction in step 6 of Algorithm 6, flow f_2 is canceled and $f'(u, v) = f'_1(u, v) \geq 0$. Suppose $c_{f'_1}(u, v) > 0$, then we have that $d'_1(u) \leq d'_1(v)+1$, because d'_1 is valid in $G^1_{f'_1}$. It follows that $d'(u) = d'_1(u) \leq d'_1(v)+1 = d(v)+1 \leq d'_2(v)+1 = d'(v)+1$, where we also used labeling monotonicity property. The inequality $d'(u) \leq d'(v) + 1$ is a contradiction, therefore it must be that $c_{f'}(u, v) = 0$. The labeling d' is valid on (u, v) in this case. Note that inequalities $d'(v) \leq d'(u) + 1$ and $d'(u) \leq d'(v) + 1$ cannot be violated simultaneously. In the remaining case, when both inequalities are satisfied, the labeling is valid for arbitrary flow on (u, v), so no flow is canceled in the flow fusion step.
3. If $f'(u, v) > 0$ then $f'_k(u, v) > 0$ and there was a push operation from u to v in the discharge of region $R^k \ni u$. Let $\tilde{d}_k$ be the labeling in G^k on this step. We have $d'(u) \geq \tilde{d}_k(u) = \tilde{d}_k(v) + 1 \geq d(v) + 1 > d(v)$. □

Theorem 48. P-PRD terminates in $2n^2$ sweeps at most.

Proof. As before, the total increase of d is at most n^2. As shown above, the labeling monotony, labeling validity and flow direction are satisfied for the fused flow and labeling (f', d') on the region $R = V \backslash \{s, t\}$. Applying Statement 45, we get that the total increase of the potential is bounded from above by the total increase of d during a sweep.

If d has not increased during a sweep ($d' = d$) then $\alpha(u, v) = 1$ for all $(u, v) \in (\mathcal{B}, \mathcal{B})_E$ (all boundary pairs are valid). The flow direction property implies that the flow goes only "downwards" the labeling. So no flow is canceled in the fusion step. Let $H = \{v \mid d(v) \geq \Phi\}$. These vertices are above any active vertices, so they cannot receive flow. After the sweep, all active vertices that were in H, are discharged and must

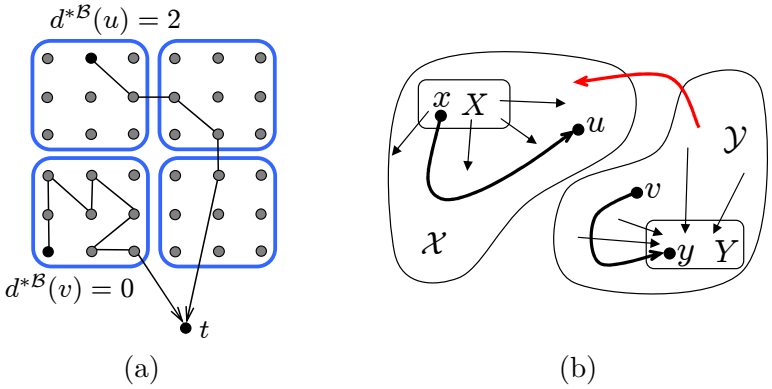


Figure 15 (a) Illustration of the region distance. (b) Illustration of Lemma 51: augmentation on paths from x to u or from v to y preserves $X \nrightarrow Y$, but not the augmentation on the red path.

become inactive. Because there is no active vertices with label Φ or above left, it is $\Phi' < \Phi$. It follows that the algorithm will terminate in $2n^2$ sweeps at most. □

5.3 Augmented path Region Discharge

In this section, we introduce the core of our new algorithm, which combines path augmentation and push-relabel approaches. We will give a new definition to the distance function, the validity of a labeling and introduce the new `Discharge` operation to be used within the generic sequential and parallel algorithms (Algorithms 5 and 6). With these modifications the algorithms will be proven correct and will fulfill a tighter bound on the number of sweeps.

5.3.1 A New Distance Function

Consider the fixed partition $(R^k)_{k=1}^K$. Let us introduce the distance function that counts only inter-region edges and not inner edges. The *region distance* $d^{*\mathcal{B}}(u)$ in G is the minimal number of inter-region edges contained in a path from u to t, or $|\mathcal{B}|$ if no such path exists:

$$d^{*\mathcal{B}}(u) = \begin{cases} \min\limits_{P=((u,u_1),\dots,(u_r,t))} |P \cap (\mathcal{B},\mathcal{B})_E| & \text{if } u \to t\,, \\ |\mathcal{B}| & \text{if } u \nrightarrow t\,. \end{cases} \tag{218}$$

This distance corresponds well to the number of region discharge operations required to transfer the excess to the sink (see Figure 15(a)).

Statement 49. If $u \to t$ then $d^{*\mathcal{B}}(u) < |\mathcal{B}|$.

Proof. Let P be a path from u to t given as a sequence of edges. If P contains a loop then it can be removed from P and $|P \cap (\mathcal{B},\mathcal{B})_E|$ will not increase. A path without loops goes through each vertex once at most. For $\mathcal{B} \subset V$, there are $|\mathcal{B}| - 1$ edges in the path at most that have both endpoints in $\mathcal{B}$. □

We now let $d^\infty = |\mathcal{B}|$ and redefine a valid labeling w.r.t. the new distance. A labeling

Procedure ARD(G^R,d)

/* assume $d\colon V^R \to \{0,\dots,d^\infty\}$ valid in G^R */
for $i = 0, 1, \dots, d^\infty$ **do** /* stage i */
  $T_i = \{t\} \cup \{v \in B^R \,|\, d(v) < i\}$;
  Augment(R, T_i);
/* Region-relabel */
$d(u) := \begin{cases} \min\{i \,|\, u \to T_i\} & \text{if } u \in R,\ u \to T_{d^\infty}, \\ d^\infty & \text{if } u \in R,\ u \nrightarrow T_{d^\infty}, \\ d(u) & \text{if } u \in B^R. \end{cases}$

Procedure Augment(X,Y)

while *there exist a path* $(v_0, v_1, \dots, v_l)$ *such that*
$c_f(v_{i-1}, v_i) > 0$, $e_f(v_0) > 0$, $v_0 \in X$, $v_l \in Y$ **do**
  augment $\Delta = \min(e_f(v_0), \min_i c_f(v_{i-1}, v_i))$ units along the path.

$d\colon V \to \{0,\dots,d^\infty\}$ is *valid* in G if $d(t) = 0$ and for all $(u,v) \in E$ such that $c_f(u,v) > 0$:

$$\begin{aligned} d(u) &\le d(v) + 1 \quad \text{if } (u,v) \in (\mathcal{B},\mathcal{B})_E\,, \\ d(u) &\le d(v) \quad\ \ \text{if } (u,v) \notin (\mathcal{B},\mathcal{B})_E\,. \end{aligned} \tag{219}$$

Statement 50. A valid labeling d is a lower bound on the region distance $d^{*\mathcal{B}}$.

Proof. If $u \nrightarrow t$ then $d(u) \le d^{*\mathcal{B}}$. Otherwise, let $P = ((u, v_1), \dots, (v_l, t))$ be one of the shortest paths w.r.t. $d^{*\mathcal{B}}$, *i.e.* $d^{*\mathcal{B}}(u) = |P \cap (\mathcal{B},\mathcal{B})_E|$. Applying the validity property to each edge in this path, we have $d(u) \le d(t) + |P \cap (\mathcal{B},\mathcal{B})_E| = d^{*\mathcal{B}}(u)$. □

5.3.2 A New Region Discharge

In this subsection, reachability relations "$\to$", "$\nrightarrow$", residual paths, and labeling validity will be understood in the region network G^R or its residual G^R_f.

The new Discharge operation, called Augmented path Region Discharge (ARD), works as follows. It first pushes the excess to the sink along augmenting paths inside the network G^R. When it is no longer possible, it continues to augment paths to vertices in the region boundary B^R in the order of their increasing labels. This is represented by the sequence of nested sets $T_0 = \{t\}$, $T_1 = \{t\} \cup \{v \in B^R \,|\, d(v) < 1\}$, $\dots$, $T_{d^\infty} = \{t\} \cup \{v \in B^R \,|\, d(v) < d^\infty\}$. Set T_i is the destination of augmentations in stage i. As we prove below, in stage $i > 0$ residual paths may exist only to the set $T_i \backslash T_{i-1} = \{v \,|\, d(v) = i - 1\}$.

The labels on the boundary $d|_{B^R}$ remain fixed during ARD and the labels $d|_R$ inside the region do not participate in augmentations and therefore are updated only at the end.

We claim that ARD terminates with no active vertices inside the region, preserves validity and monotonicity of the labeling, and pushes flow from higher labels to lower labels w.r.t. the new labeling. These properties will be required to prove finite termi-

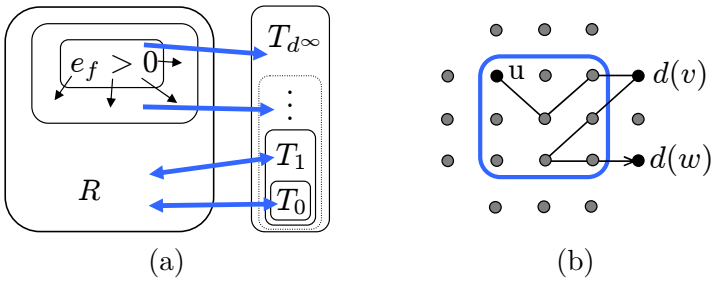


Figure 16 (a) Reachability relations in the network G_f^R at the end of stage 1 of ARD: $\{v \,|\, e_f(v) > 0\} \nrightarrow T_1$; $T_{d^\infty} \setminus T_1 \nrightarrow R$. (b) Example of a path in the network G_f^R for which by Corollary 54 it must be $d(v) \leq d(w)$. Note, such a path is not possible at the beginning of ARD, but in the middle it may exist since residual capacities of edges $(B^R, R)_E$ may become non-zero.

nation and correctness of S-ARD. Before we prove them (Statement 57) we need the following intermediate results:

- Properties of the network G_f^R maintained by ARD (Statement 52, Corollaries 53 and 54).
- Properties of valid labellings in the network G_f^R (Statement 55).
- Properties of the labeling constructed by region-relabel (line 4 of `ARD`) in the network G_f^R (Statement 56).

Lemma 51. Let $X, Y \subset V^R$, $X \cap Y = \emptyset$, $X \nrightarrow Y$. Then $X \nrightarrow Y$ is preserved after i) augmenting a path $(x, \dots, v)$ with $x \in X$ and $v \in V^R$; ii) augmenting a path $(v, \dots, y)$ with $y \in Y$ and $v \in V^R$.

Proof. (See Figure 15(b)). Let $\mathcal{X}$ be the set of vertices reachable from X. Let $\mathcal{Y}$ be the set of vertices from which Y is reachable. Clearly $\mathcal{X} \cap \mathcal{Y} = \emptyset$, otherwise $X \rightarrow Y$. Therefore, the cut $(\mathcal{X}, \bar{\mathcal{X}})_E$ separates X and Y and has all edge capacities equal zero. Any residual path starting in X or ending in Y cannot cross the cut and its augmentation cannot change the edges of the cut. Hence, X and Y will stay separated. □

Statement 52. Let $v \in V^R$ and $v \nrightarrow T_a$ in G_f in the beginning of stage i_0 of ARD, where $a, i_0 \in \{0, 1, \dots d^\infty\}$. Then $v \nrightarrow T_a$ holds until the end of ARD, *i.e.*, during all stages $i \geq i_0$.

Proof. We need to show that $v \nrightarrow T_a$ is not affected by augmentations performed by ARD. If $i_0 \leq a$, we first prove $v \nrightarrow T_a$ holds during stages $i_0 \leq i \leq a$. Consider augmentation of a path $(u_0, u_1, \dots, u_l)$, $u_0 \in R$, $u_l \in T_i \subset T_a$, $e_f(u_0) > 0$. Assume $v \nrightarrow T_a$ before augmentation. By Lemma 1 with $X = \{v\}$, $Y = T_a$ (noting that $X \nrightarrow Y$ and the augmenting path ends in Y), after the augmentation $v \nrightarrow T_a$. By induction, it holds till the end of stage a and hence in the beginning of stage $a + 1$.

We can assume now that $i_0 > a$. Let $A = \{u \in R \,|\, e_f(u) > 0\}$. At the end of stage $i_0 - 1$, we have $A \nrightarrow T_{i_0 - 1} \supset T_a$ by construction. Consider augmentation in stage i_0 on a path $(u_0, u_1 \dots, u_l)$, $u_0 \in R$, $u_l \in T_{i_0}$, $e_f(u_0) > 0$. By construction, $u_0 \in A$. Assume $\{v\} \cup A \nrightarrow T_a$ before augmentation. Apply Lemma 51 with $X = \{v\} \cup A$, $Y = T_a$ (we have $X \nrightarrow Y$ and $u_0 \in A \subset X$). After augmentation it is $X \nrightarrow T_a$. By induction, $X \nrightarrow T_a$ till the end of stage i_0. By induction on stages, $v \nrightarrow T_a$ until the end of the

ARD procedure. □

Corollary 53. If $w \in B^R$ then $w \nrightarrow T_{d(w)}$ throughout the ARD procedure.

Proof. At initialization, it is fulfilled by construction of G^R due to $c^R((B^R, R)_E) = 0$. It holds then during ARD by Statement 52. □

In particular, we have $B^R \nrightarrow t$ during ARD.

Corollary 54. Let $(u, v_1 \dots v_l, w)$ be a residual path in G^R_f from $u \in R$ to $w \in B^R$ and let $v_r \in B^R$ for some r. Then $d(v_r) \le d(w)$.

Proof. We have $v_r \nrightarrow T_{d(v_r)}$. Suppose $d(w) < d(v_r)$, then $w \in T_{d(v_r)}$ and because $v_r \to w$ it is $v_r \to T_{d(v_r)}$ which is a contradiction. □

The properties of the network G^R_f established by Statement 52 and Corollary 54 are illustrated in Figure 16.

Statement 55. Let d be a valid labeling, $d(u) \ge 1$, $u \in R$. Then $u \nrightarrow T_{d(u)-1}$.

Proof. Suppose $u \to T_0$. There exist a residual path $(u, v_1 \dots v_l, t)$, $v_i \in R$ (by Corollary 53 it cannot happen that $v_i \in B^R$). By validity of d we have $d(u) \le d(v_1) \le \dots \le d(v_l) \le d(t) = 0$, which is a contradiction.

Suppose $d(u) > 1$ and $u \to T_{d(u)-1}$. Because $u \nrightarrow T_0$, it must be that $u \to w$, $w \in B^R$ and $d(w) < d(u) - 1$. Let $(v_0 \dots v_l)$ be a residual path with $v_0 = u$ and $v_l = w$. Let r be the minimal number such that $v_r \in B^R$. By validity of d we have $d(u) \le d(v_1) \le \dots \le d(v_{r-1}) \le d(v_r) + 1$. By Corollary 54 we have $d(v_r) \le d(w)$, hence $d(u) \le d(w) + 1$, which is a contradiction. □

Statement 56. For d computed on line 4 of Procedure ARD and any $u \in R$ it holds:

1. d is valid;
2. $u \nrightarrow T_a \Leftrightarrow d(u) \ge a + 1$.

Proof.

1. Let $(u, v) \in E^R$ and $c(u, v) > 0$. Clearly $u \to v$. Consider four cases:
   - case $u \in R$, $v \in B^R$: Then $u \to T_{d(v)+1}$, hence $d(u) \le d(v) + 1$.
   - case $u \in R$, $v \in R$: If $v \nrightarrow T_{d^\infty}$ then $d(v) = d^\infty$ and $d(u) \le d(v)$. If $v \to T_{d^\infty}$, then $d(v) = \min\{i \mid v \to T_i\}$. Let $i = d(v)$, then $v \to T_i$ and $u \to T_i$, therefore $d(u) \le i = d(v)$.
   - case $u \in B^R$, $v \in R$: By Corollary 53, $u \nrightarrow T_{d(u)}$. Because $u \to v$, it is $v \nrightarrow T_{d(u)}$, therefore $d(v) \ge d(u) + 1$ and $d(u) \le d(v) - 1 \le d(v) + 1$.
   - case when $u = t$ or $v = t$ is trivial.

2. The "⇐" direction follows by Statement 55 applied to d, which is a valid labeling. The "⇒" direction: we have $u \nrightarrow T_a$ and $d(u) \ge \min\{i \mid u \to T_i\} = \min\{i > a \mid u \to T_i\} \ge a + 1$. □

Statement 57 (Properties of ARD)**.** Let d be a valid labeling in G^R. The output (f', d') of ARD satisfies:

1. There are no active vertices in R w.r.t. (f', d'); (*optimality*)
2. $d' \ge d$, $d'|_{B^R} = d|_{B^R}$; (*labeling monotonicity*)
3. d' is valid in $G^R_{f'}$; (*labeling validity*)

4. f' is a sum of path flows, where each path is from a vertex $u \in R$ to a vertex $v \in \{t\} \cup B^R$ and it is $d'(u) > d(v)$ if $v \in B^R$. (*flow direction*)

Proof.
1. In the last stage, the ARD procedure augments all paths to T_{d^∞}. After this augmentation a vertex $u \in R$ either has excess 0 or there is no residual path to T_{d^∞} and hence $d'(u) = d^\infty$ by construction.
2. For $d(u) = 0$, we trivially have $d'(u) \geq d(u)$. Let $d(u) = a + 1 > 0$. By Statement 55, $u \nrightarrow T_a$ in G^R and it holds also in $G^R_{f'}$ by Statement 52. From Statement 56.2, we conclude that $d'(u) \geq a + 1 = d(u)$. The equality $d'|_{B^R} = d|_{B^R}$ is by construction.
3. Proven by Statement 56.1.
4. Consider a path from u to $v \in B^R$, augmented in stage $i > 0$. It follows that $i = d(v) + 1$. At the beginning of stage i, it is $u \nrightarrow T_{i-1}$. By Statement 52, this is preserved till the end of ARD. By Statement 56.2, $d'(u) \geq i = d(v) + 1 > d(v)$. □

Algorithms 5 and 6 for `Discharge` being ARD will be referred to as S-ARD and P-ARD, respectively.

5.3.3 Complexity of Sequential Augmented path Region Discharging

Statement 44 holds for S-ARD as well, so S-ARD maintains a valid labeling.

Theorem 58. S-ARD terminates in $2|\mathcal{B}|^2 + 1$ sweeps at most.

Proof. The value of $d(v)$ does not exceed $|\mathcal{B}|$ and d is non-decreasing. The total increase of $d|_{\mathcal{B}}$ during the algorithm is at most $|\mathcal{B}|^2$.

After the first sweep, active vertices are only in $\mathcal{B}$. Indeed, discharging region R^k makes all vertices $v \in R^k$ inactive and only vertices in $\mathcal{B}$ may become active. So by the end of the sweep, all vertices $V \backslash \mathcal{B}$ are inactive.

Therefore, after the first sweep, the potential can be equivalently written as

$$\Phi = \max\{d(v) \,|\, v \in \mathcal{B},\ v \text{ is active in } G\}\,. \tag{220}$$

We will prove the following two cases for each sweep but the first one:

1. If $d|_{\mathcal{B}}$ is increased then the increase in Φ is no more than total increase in $d|_{\mathcal{B}}$. Consider discharge of R^k. Let Φ be the value before ARD on R^k and Φ' the value after. Let $\Phi' = d'(v)$. It must be that v is active in G'. If $v \notin V^k$, then $d(v) = d'(v)$ and $e(v) = e_{f'}(v)$ so $\Phi \geq d(v) = \Phi'$.
   Let $v \in V^k$. After the discharge, vertices in R^k are inactive, so $v \in B^k$ and it is $d'(v) = d(v)$. If v was active in G then $\Phi \geq d(v)$ and we have $\Phi' - \Phi \leq d'(v) - d(v) = 0$. If v was not active in G, there must exist an augmenting path from a vertex v_0 to v such that $v_0 \in R^k \cap \mathcal{B}$ was active in G. For this path, the flow direction property implies $d'(v_0) \geq d(v)$. We now have $\Phi' - \Phi \leq d'(v) - d(v_0) = d(v) - d(v_0) \leq d'(v_0) - d(v_0) \leq \sum_{v \in R^k \cap \mathcal{B}} [d'(v) - d(v)]$. Summing over all regions, we get the result.
2. If $d|_{\mathcal{B}}$ is not increased then Φ is decreased at least by 1. We have $d' = d$. Let us consider the set of vertices having the highest active label or above, $H = \{v |\, d(v) \geq \Phi\}$. These vertices do not receive the flow during all discharge operations due to the flow direction property. After the discharge of R^k, there are no active vertices left in $R^k \cap H$ (Statement 57.1). After the full sweep, there are no active vertices in H.

In the worst case, starting from sweep 2, Φ can increase by one $|\mathcal{B}|^2$ times and decrease by one $|\mathcal{B}^2|$ times. There are no active vertices left in at most $2|\mathcal{B}|^2+1$ sweeps. □

On termination, we have that the labeling is valid and there are no active vertices in G. Therefore, the accumulated preflow is maximal and a minimum cut can be found by analyzing the reachability in G (see discussion for S-PRD §5.2.3 and implementation details §5.4.2).

5.3.4 Complexity of Parallel Augmented Path Region Discharging

Statement 59 (Properties of Parallel ARD)**.** Let d be a valid labeling at the beginning of a sweep of P-ARD. The pair of fused flow and labeling (f', d') satisfies:

1. Vertices in $V\backslash\mathcal{B}$ are not active in $G_{f'}$; (*optimality*)
2. $d' \geq d$; (*labeling monotony*)
3. d' is valid; (*labeling validity*)
4. f' is the sum of path flows, where each path is from a vertex $u \in V$ to a vertex $v \in \mathcal{B}$, satisfying $d'(u) \geq d(v)$. (*weak flow direction*)

Proof.
1. For each k there are no active vertices in R^k w.r.t. (f'_k, d'_k). The fused flow f' may differ from f'_k only on the boundary edges $(u,v) \in (\mathcal{B},\mathcal{B})_E$. So there are no active vertices in $V\backslash\mathcal{B}$ w.r.t. (f', d').
2. By construction.
3. Same as in P-PRD.
4. Consider the augmentation of a path from $u \in R^k$ to $v \in B^k$ during `ARD` on G^k and canceling of the flow on the last edge of the path during the flow fusion step. Let the last edge of the path be (w, v). We need to prove that $d'(u) \geq d(w)$. Let $\tilde{d}$ be the labeling in G^k right before augmentation, as if it was computed by region-relabel. Because $\tilde{d}$ is valid it must be that $\tilde{d}(w) \leq \tilde{d}(v) + 1$. We have $d'_k(u) > d(v) = \tilde{d}(v) \geq \tilde{d}(w) - 1 \geq d(w) - 1$. So $d'(u) \geq d(w)$. □

Theorem 60. P-ARD terminates in $2|\mathcal{B}|^2 + 1$ sweeps.

Proof. As before, total increase of $d|_\mathcal{B}$ is at most $|\mathcal{B}|^2$.
After the first sweep, active vertices are only in $\mathcal{B}$ by Statement 59.1.
For each sweep after the first one:

- If $d|_\mathcal{B}$ is increased then increase in Φ is no more than the total increase of $d|_\mathcal{B}$. Let Φ' be the value of the potential in the network $G' = G_{f'}$. Let $\Phi' = d'(v)$. It must be that v is active in G' and $v \in \mathcal{B}$.

  If v was active in G then $\Phi \geq d(v)$ and we have $\Phi' - \Phi \leq d'(v) - d(v)$.

  If v was not active in G then there must exist a path flow in f' from a vertex v_0 to v such that $v_0 \in \mathcal{B}$ was active in G. For this path, the weak flow direction property implies $d'(v_0) \geq d(v)$. We have $\Phi' - \Phi \leq d'(v) - d(v_0) = d'(v) - d(v) + d(v) - d(v_0) \leq d'(v) - d(v) + d'(v_0) - d(v_0) \leq \sum_{v\in\mathcal{B}}[d'(v) - d(v)]$.
- If $d|_\mathcal{B}$ is not increased then Φ is decreased at least by 1. In this case, f' satisfies the strong flow direction property and the proof of Theorem 58 applies.

After total of $2|\mathcal{B}|^2 + 1$ sweeps, there are no active vertices left. □

5.4 Implementation

In this section, we first discuss heuristics for improving the distance labeling (making it closer to the true distance at a cheap cost) commonly used in the push-relabel framework. They are essential for the practical performance of the algorithms. We then describe our base implementations of S-ARD/S-PRD and the solvers they rely on. In the next section, we describe an efficient implementation of ARD, which is more sophisticated but has a much better practical performance. All of the labeling heuristics can only increase the labels and preserve validity of the labeling. Therefore, they do not break theoretical properties of the respective algorithms.

5.4.1 Heuristics

Region-relabel heuristic. This heuristic computes labels $d|_R$ of the region vertices, given the distance estimate on the boundary, $d|_{B^R}$. There is a slight difference between PRD and ARD variants (using distance d^* and $d^{*\mathcal{B}}$, resp.), displayed by the corresponding "if" conditions.

Procedure $d|_R = \text{Region-relabel}(G^R, d|_{B^R})$

```
   /* init                                                              */
 1 d(t) := 0; O := {t}; d|_R := d^∞; d^c := 0;
 2 if ARD then d|_{B^R} := d|_{B^R} + 1;                  /* (for ARD) */
   /* O is a list of open vertices, having the current label d^c       */
 3 d^max := max{d(w) | w ∈ B^R, d(w) < d^∞};
 4 while O ≠ ∅ or d^c < d^max do
       /* if O is empty raise d^c to the next seed                     */
 5     if O = ∅ then d^c := min{d(w) | w ∈ B^R, d(w) > d^c, d(w) < d^∞};
       /* add seeds to the open set                                    */
 6     O := O ∪ {w ∈ B^R | d(w) = d^c};
       /* find all unlabeled vertices from which O can be reached      */
 7     O := {u ∈ R | (u, v) ∈ E^R, v ∈ O, c(u, v) > 0, d(u) = d^∞};
 8     if PRD then d^c ← d^c + 1;                         /* (for PRD) */
 9     d|_O := d^c;                        /* label the new open vertices */
10 if ARD then d|_{B^R} := d|_{B^R} − 1;                  /* (for ARD) */
```

In the implementation, the set of boundary vertices is sorted in advance, so that Region-relabel runs in $O(|E^R| + |V^R| + |B^R| \log |B^R|)$ time and uses $O(|V^R|)$ space. The resulting labeling d' is valid and satisfies $d' \geq d$ for arbitrary valid d.

Global gap heuristic. Let us briefly explain the global gap heuristic (Cherkassky and Goldberg 1994). It is a sufficient condition to identify that the sink is unreachable from a set of vertices. Let there be no vertices with label $g > 0$: $\forall v \in V\ d(v) \neq g$, and let $d(u) > g$. For a valid labeling d, it follows that there is no vertex v for which $c(u, v) > 0$ and $d(v) < g$. Assuming there is such a vertex, we will have $d(u) \leq d(v) + 1 \leq g$, which is a contradiction. Therefore the sink is unreachable from all vertices $\{u \mid d(u) > g\}$ and their labels may be set to d^∞.

Region gap heuristic of Delong and Boykov (2008) detects if there are no vertices inside region R having label $g > 0$. Such vertices can be connected to the sink in the whole network only through one of the boundary vertices, so they may be relabeled up

to the closest boundary label. Here is the algorithm[4]:

Procedure $d|_R = \text{Region-gap}(G^R, d|_{R \cup B^R}, g)$

```
  /* Input:  region network G^R, labeling d,                    */
  /* gap g such that ∀v ∈ R  d(v) ≠ g.                          */
1 d^next := min{d(w) | w ∈ B^R, d(w) > g.};
2 for v ∈ R such that g < d(v) < d^next do
3 |  d(v) := d^next+1;
```

If no boundary vertex is above the gap, then $d^{\text{next}} = d^\infty$ in step 1 and all vertices above the gap are disconnected from the sink in the network G. Interestingly, this sufficient condition does not imply a global gap. In our implementation of PRD, we detect the region-gap efficiently after every vertex relabel operation by discovering an empty bucket (see the implementation of S/P-PRD in §5.4.5).

5.4.2 Reachability/Exact distance Computation

At the termination of our algorithms (S/P-PRD, S/P-ARD), we have found a maximum preflow and we know that $(\bar{T}, T)_E$, defined by $T = \{v \mid v \to t \text{ in } G\}$, is a minimum cut. However, we only know the lower bound d on the true distance d^* (resp. $d^{*\mathcal{B}}$). Therefore the reachability relation $v \to t$ is not fully known at this point. When $d(v) = d^\infty$, we are sure that $v \not\to t$ in G and hence v must be in the source set of a minimum cut, but if $d(v) < d^\infty$, it is still possible that $v \not\to t$ in G. Therefore, we need to do some extra work to make d the exact distance and in this way to find the minimum cut. For this purpose we execute several extra sweeps, performing only region-relabel and gap heuristics until labels stop changing. We claim that at most d^∞ such extra sweeps are needed. We give a proof for the case of push-relabel distance.

Proof. Let us call labels $d(v)$ *loose* if $d(v) < d^*(v)$ and exact if $d(v) = d^*(v)$. Consider the lowest loose label, $L = \min\{d(v) \mid d(v) < d^*(v)\}$ and the set of loose vertices having this label, $\mathcal{L} = \{v \mid L = d(v) < d^*(v)\}$. Let us show that after a sweep of region-relabel, the value of L increases at least by 1. Let $v \in \mathcal{L}$, $(v, w) \in E$ and $c(v, w) > 0$. If $d(w)$ is loose, we have $d(w) \geq L$ by construction. Assume that $d(w)$ is exact. Since $d(v) < d^*(v)$ and $d^*(v) \leq d^*(w) + 1$, we have $d(w) \geq d(v) = L$. Therefore, all neighbors of v have label L or above. After the elementary Relabel of v or Region-relabel of the region including v, its label will increase at least by 1 (recall that Relabel of v performs $d(v) := \min_w\{d(w) \mid (v, w) \in E,\ c(v, w) > 0\} + 1$). Because this holds for all vertices from $\mathcal{L}$, the value L will increase at least by 1 after elementary Relabel of all vertices or a sweep of Region-relabel. Because L is bounded above by d^∞, after at most d^∞ sweeps, d will be the exact distance. □

This proof can be suitably modified for the case of region distance (used in ARD) by replacing the pair (v, w) with a path from v to a boundary vertex w. In this case, we have the bound $d^\infty = |\mathcal{B}|$ sweeps. In the experiments, we observed that in order to compute the exact distance, only a few extra sweeps were necessary (from 0 to 2) for S/P-ARD and somewhat more for S/P-PRD. Note, to compute the final reachability relation in S/P-PRD, the region distance and ARD Region-relabel could be employed. However,

[4] Region-gap-relabel (Delong and Boykov 2008, fig. 10) seems to contain an error: only vertices above the gap should be processed in step 3.

we did not implement this improvement. In §5.5, we describe how ARD Region-relabel is replaced by a dynamic data structure (search trees), allowing for quick recomputation during the sweeps.

5.4.3 Referenced Implementations

Boykov-Kolmogorov (BK) The reference augmenting path implementation by Boykov and Kolmogorov (2004) (v3.0, `http://www.cs.adastral.ucl.ac.uk/~vnk/software.html`). We will also use the possibility of dynamic updates in this code due to Kohli and Torr (2005). There is only a trivial $O(mn^2|C|)$ complexity bound known for this algorithm[5], where C is the cost of a minimum cut.

Highest level Push-Relabel (HIPR) The reference push-relabel implementation by Cherkassky and Goldberg (1994) (v3.6, `http://www.avglab.com/andrew/soft.html`). This implementation has two stages: finding the maximum preflow / minimum cut and upgrading the maximum preflow to a maximum flow. Only the first stage was executed and benchmarked. We tested two variants with frequency of the global relabel heuristic (the frequency parameter roughly corresponds to the proportion of time spent on global updates versus push/relabel) equal to 0.5 (the default value in HIPR v3.6) and equal to 0. These variants will be denoted HIPR0.5 and HIPR0, respectively. HIPR0 executes only one global update at the beginning. Global updates are essential for difficult problems. However, HIPR0 was always faster than HIPR0.5 in our experiments with real test instances[6]. The worst case complexity is $O(n^2\sqrt{m})$.

5.4.4 S/P-ARD Implementation

The basic implementation of `ARD` simply invokes BK solver as follows. On stage 0, we compute the maximum flow in the network G^R by BK, augmenting paths from source to the sink. On the stage i, infinite capacities are added from the boundary vertices having label $i-1$ to the sink, using the possibility of dynamic changes in BK. The flow augmentation to the sink is then continued, reusing the search trees. The Region-relabel procedure is implemented as described earlier in this section. In the beginning of next discharge, we clear the infinite link from the boundary to the sink and repeat the above. Some parts of the sink search tree, linked through the boundary vertices, get destroyed, but the larger part of it and the source search tree are reused. A more efficient implementation is described in §5.5. It includes additional heuristics and maintenance of separate boundary search trees.

S-ARD. In the streaming mode, we keep only one region in the memory at a time. After a region is processed by `ARD`, all the internal data structures have to be saved to the disk and cleared from the memory until the region is discharged next time. We manage this by allocating all the region's data into a fixed page in the memory, which can be saved and loaded preserving the pointers. By doing the load/unload manually (rather than relying on the system swapping mechanism), we can accurately measure the pure time needed for computation (CPU) and the amount of disk I/O. We also can use 32bit pointers with larger problems.

[5]The worst-case complexity of the breadth-first search shortest path augmentation algorithm is just $O(m|C|)$. The tree adaptation step, introduced by Boykov and Kolmogorov (2004) to speed-up the search, does not have a good bound and introduces an additional n^2 factor.

[6]There is a discrepancy with Delong and Boykov (2008, Figure 4) regarding the results for the basic push-relabel. The main implementation difference is in the order of processing (HIPR versus FILO). It is also possible that their plot is illustrative, showing results without the gap heuristic.

A region with no active vertices is skipped. The global gap heuristic is executed after each region discharge. Because this heuristic is based on labels of boundary vertices only, it is sufficient to maintain a label histogram with $|\mathcal{B}|$ bins to implement it. S-ARD uses $O(|\mathcal{B}|+|(\mathcal{B},\mathcal{B})_E|)$ "shared" memory and $O(|V^R+E^R|)$ "region" memory, to which regions are loaded one at a time.

To solve large problems that do not fit in the memory, we have to create region graphs without ever loading the full problem. We implemented a tool called *splitter*, which reads the problem from a file and writes edges corresponding to the same region to the region's separate "part" file. Only the boundary edges (linking different regions) are withheld to the memory.

P-ARD. We implemented this algorithm for a shared-memory system using OpenMP language extension. All regions are kept in the memory, the discharges are executed concurrently in separate threads, while the gap heuristic and messages exchange are executed synchronously by the master thread.

5.4.5 S/P-PRD Implementation

To solve region discharge subproblems in PRD in the highest label first fashion, we designed a special reimplementation of HIPR, which will be denoted **HPR**. We intended to use the original HIPR implementation to make sure that PRD relies on the state-of-the art core solver. It was not possible directly. A subproblem in PRD is given by a region network with fixed distance labels on the boundary (let us call them *seeds*). Distance labels in PRD may go up to n in the worst case. The same applies to region subproblems as well. Therefore, keeping an array of buckets corresponding to possible labels (like in HIPR), would not be efficient. It would require $O(|V|)$ memory and an increased complexity. However, because a region has only $|V^R|$ vertices, there are no more than $|V^R|$ distinct labels at any time. This allows to keep buckets as a doubly-linked list with at most $|V^R|$ entries. The highest label selection rule and the region-gap heuristic can then be implemented efficiently with just a small overhead. We tried to keep other details similar to HIPR (current arc data structure, etc.). HPR with arbitrary seeds has the worst case complexity $O(|V^R|^2\sqrt{|E^R|})$ and uses $O(|V^R|+|V^E|)$ space. When the whole problem is taken as a single region, HPR should be equivalent to HIPR0. Though the running time on the real instances can be somewhat different.

S-PRD This is our reimplementation of the algorithm by Delong and Boykov (2008) for an arbitrary graph and a fixed partition, using HPR as a core solver. It uses the same memory model, paging mechanism and the splitter tool as S-ARD. The region discharge is always warm-started. We found it inefficient to run the region-relabel after every discharge. In the current experiments, motivated by performance of HIPR0, we run it once at the beginning and then only when a global gap is discovered. To detect a global gap, we keep a histogram of all labels, $O(n)$ memory, and update it after each region discharge (in $O(|V^R|)$ time). In practice, this $O(n)$ memory is not a serious limitation – labels are usually well below n. If they are not then we should consider a weaker gap heuristic with a smaller number of bins. Applying the gap (raising the corresponding vertices to d^∞) for all regions is delayed until they are loaded. So we keep the track of the best global gap detected for every region. Similar to how the sequential Algorithm 5 represents both S-ARD and S-PRD, it constitutes a piece of generic code in our implementation, where the respective discharge procedure and gap heuristics are plugged.

P-PRD This is our implementation of parallel `PRD` for shared-memory system with OpenMP.

5.5 Efficient ARD Implementation

The basic implementation of S-ARD, as described in the previous section, worked reasonably fast (comparable to BK) on simple problems like 2D stereo and 2D random segmentation (§5.6.1). However, the performance was unexpectedly bad on some 3D problems. For example, to solve `LB07-bunny-lrg` instance (§5.6.2) the basic implementation required 32 minutes of CPU time. In this section, we describe an efficient implementation which is more robust and is comparable in speed with BK on all tested instances. In particular, to solve `LB07-bunny-lrg` it takes only 15 seconds of CPU time. The problem, why the basic implementation is so slow, is in the nature of the algorithm: sometimes it has to augment the flow to the boundary, without knowing of whether it is a useful work or not. If the particular boundary was selected wrongly, the work is wasted. This happens in `LB07-bunny-lrg` instance, where the data seeds are sparse. A huge work is performed to push the flow around in the first few iterations, before a reasonable labeling is established. We introduce two heuristics how to overcome this problem: the boundary-relabel heuristic and partial discharges. An additional speed-up is obtained by dynamically maintaining boundary search trees and the current labeling.

5.5.1 Boundary Relabel Heuristic

We would like to have a better distance estimate, but we cannot run a global relabel because implementing it in a distributed fashion would take several full sweeps, which would be too wasteful. Instead, we go for the following cheaper lower bound. Our implementation keeps all the boundary edges (including their flow and distance labels of the adjacent vertices) in the shared memory. Figure 17(a) illustrates this boundary information. We want to improve the labels by analyzing only this boundary part of the graph, not looking inside the regions. Since we do not know how the vertices are connected inside the regions, we have to assume that every boundary vertex might be connected to any other one within the region, except of the following case. If u and v are in the same region R and $d(u) > d(v)$ then we know for sure that $u \not\rightarrow v$ in G^R. It follows from the validity of labeling d (as defined for ARD in §5.3). We can calculate now a lower bound on the distance $d^{*\mathcal{B}}$ in G assuming that all the rest of the vertices are potentially connected within the regions.

We will now construct an auxiliary directed graph $\bar{G}$ with arcs having length 0 or 1 and show that the distance in this graph (according to the arc lengths) lower bounds $d^{*\mathcal{B}}$. If $d(v) = d(u)$ we have to assume that $v \rightarrow u$ and $u \rightarrow v$ in G^R, therefore the new lower bound for u and v will coincide. Hence we group vertices having the same label within a region together as shown in Figure 17(b). In the case $d(v) < d(u)$, we know that $u \not\rightarrow v$ but have to assume $v \rightarrow u$ in R. We thus add a directed arc of length zero from the group of v to the group of u (Figure 17(b)). Let $d_1 < d_2 < d_3$ be labels of groups within one region. There is no need to create an arc from d_1 to d_3, because two arcs from d_1 to d_2 and from d_2 to d_3 of length zero are an equivalent representation. Therefore it is sufficient to connect only groups having consecutive labels. We then add all residual edges (u, v) between the regions to $\bar{G}$ with length 1. We can calculate the distance to vertices with label 0 in $\bar{G}$ by running Dijkstra's algorithm. Let this distance

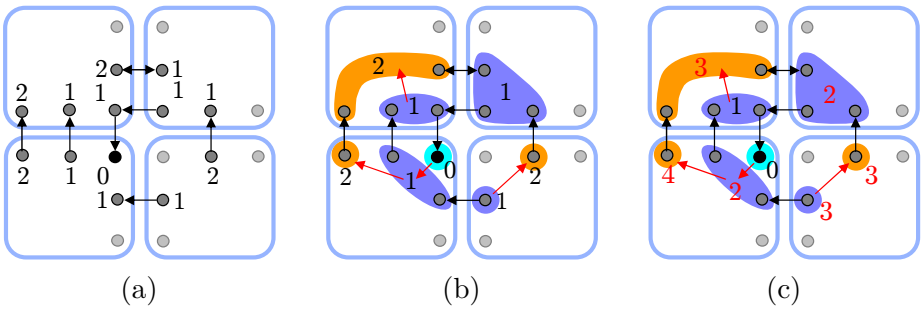


Figure 17 Boundary relabel heuristic: (a) Boundary vertices of the network and a valid labeling. Directed arcs correspond to non-zero residual capacities. Vertices without numbers have label d^∞ and do not participate in the construction. (b) Vertices having the same label are grouped together within each region and arcs of zero length (of red color) are added from a group to the next label's group. It is guaranteed that *e.g.*, vertices with label 1 are not reachable from vertices with label 2 within the region, hence there is no arc 2→1. Black arcs have the unit length. (c) The distance in the auxiliary graph is a valid labeling and a lower bound on the distance in the original network.

be denoted d'. We then update the labels as

$$d(u) := \max\{d(u), d'(u)\}\,. \tag{221}$$

We have to prove the following two points:

1. d' is a valid labeling;
2. If d and d' are valid labellings, then $\max(d, d')$ is valid.

Proof. 1. Let $c(u,v) > 0$. Let u and v be in the same region. It must be that $d(u) \le d(v)$. Therefore either u and v are in the same group or there is an arc of length zero from group of u to group of v. It must be $d'(u) \le d'(v)$ in any case. If u and v are in different regions, there is an arc of length 1 from group of u to group of v and therefore $d'(u) \le d'(v) + 1$.

2. Let $l(u,v) = 1$ if $(u,v) \in (\mathcal{B}, \mathcal{B})$ and $l(u,v) = 0$ otherwise. We have to prove that if $c(u,v) > 0$ then

$$\max\{d(u), d'(u)\} \le \max\{d(v), d'(v)\} + l(u,v)\,. \tag{222}$$

Let $\max\{d(u), d'(u)\} = d(u)$. From validity of d we have $d(u) \le d(v) + l(u,v)$. If $d(v) \ge d'(v)$, then $\max\{d(v), d'(v)\} = d(v)$ and (222) holds. If $d(v) < d'(v)$ then $d(u) \le d(v) + l(u,v) < d'(v) + l(u,v)$ and (222) holds again.

□

The complexity of the boundary relabel heuristic is $O(|(\mathcal{B}, \mathcal{B})|)$. It is relatively inexpensive and can be run after each sweep. It does not affect the correctness and the worst case bound on the number of sweeps of S/P-ARD.

5.5.2 Partial Discharge

Another heuristic, which proved very efficient, was simply to postpone path augmentations to higher boundary vertices to further sweeps. This allows to save a lot of unnecessary work, especially when used in combination with boundary-relabel. More precisely, the `ARD` procedure is allowed to execute only stages up to s on sweep s. This way, in sweep 0, only paths to the sink are augmented and not any path to the boundary. Vertices which cannot reach the sink (but can potentially reach the boundary) get

label 1. These initial labels may already be improved by boundary-relabel. In sweep 1, paths to the boundary with label 0 are allowed to be augmented and so on.

Note that this heuristic does not affect the worst case complexity of S/P-ARD. Because labels can grow only up to $|\mathcal{B}|$, after at most $|\mathcal{B}|$ sweeps the heuristic turns into full discharge. Therefore, the worst case bound of $O(|\mathcal{B}^2|)$ sweeps remains valid. In practice, we found that it increases the number of sweeps slightly, while significantly reduces the total computation time. Similarly, in the case of push-relabel, it would make sense to perform several sweeps of Region-relabel before doing any pushes to get a better estimate of the distance.

5.5.3 Boundary Search Trees

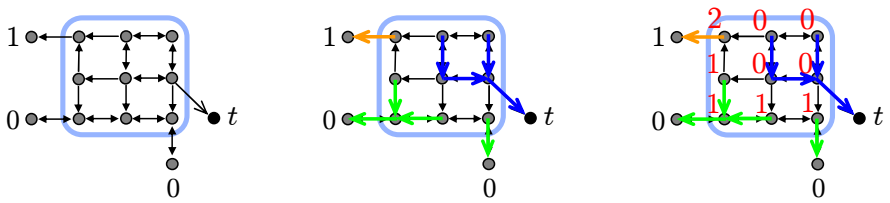


Figure 18 Search trees. (a) A region with some residual arcs. The region has only 3 boundary vertices, for simplicity, the numbers correspond to the labels. (b) Search trees of the sink and boundary vertices: when a vertex can be reached by several trees, it choses the one with the lowest label of the root. The sink is assigned a special label -1. The source search tree is empty in this example. (c) Labels of the inner vertices are determined as their tree's root label+1.

We now redesign the implementation of `ARD` such that not only the sink and source search trees are maintained but also the search trees of boundary vertices. This allows to save computation when the labeling of many boundary vertices remains constant during the consequent sweeps, with only a small fraction changing. Additionally, knowing the search tree for each inner vertex of the region determines its actual label, so the region-relabel procedure becomes obsolete. The design of the search tree data structures, their updates and other detail are the same as proposed by Kolmogorov (2004b), only few changes to the implementation are necessary. For each vertex $v \in R$, we introduce a mark $\tilde{d}(v)$ which corresponds to the root label of its tree or is set to a special *free* mark if v is not in any tree. For each tree, we keep a list of open vertices (called active by Kolmogorov (2004b)). A vertex is *open* if it is not blocked by the vertices of the trees with the same or lower root label (more precisely, v is open if it is not free and there is a residual edge (u, v) such that u is free or its root label is higher than that of v). The trees may grow only at the open vertices.

Figure 18 shows the correspondence between search trees and the labels. The sink search tree is assigned label -1. In the stage 0 of `ARD`, we grow the sink tree and augment all found paths if the sink tree touches the source search tree. Vertices that are added to the sink tree are marked with label $\tilde{d} = -1$. In stage $i+1$ of `ARD`, we grow trees with root at a boundary vertices w with label $d(w) = i$, all vertices added to the tree are marked with $\tilde{d} = i$. When the tree touches the source search tree, the found path is augmented. If the tree touches a vertex u with label $\tilde{d}(u) < i$, it means that u is already in the search tree with a lower root and no action is taken. It cannot happen that a vertex is reached with label $\tilde{d} > i$ during growth of a search tree with root label

i, this would contradict to the properties of `ARD`. The actual label of a vertex v at any time is determined as $\tilde{d}(v) + 1$ if $v \in R$ and $d(v)$ if $v \in B^R$.

Let us now consider the situation in which region R has built some search trees and the label of a boundary vertex w is risen from $d(w)$ to $d'(w)$ (as a result of update from the neighboring region or one of the heuristics). All the vertices in the search tree starting from w were previously marked with $d(w)$ and have to be declared as free vertices or adopted to any other valid tree with root label $d(w)$. The adaptation is performed by the same mechanism as in BK. The situation when a preflow is injected from the neighboring region and (a part of) a search tree becomes disconnected is also handled by the orphan adaptation mechanism.

The combination of the above improvements allows S-ARD to run in about the same time as BK on all tested vision instance (§5.6.2), sometimes being even significantly faster (154sec. vs. 245sec. on `BL06-gargoyle-lrg`).

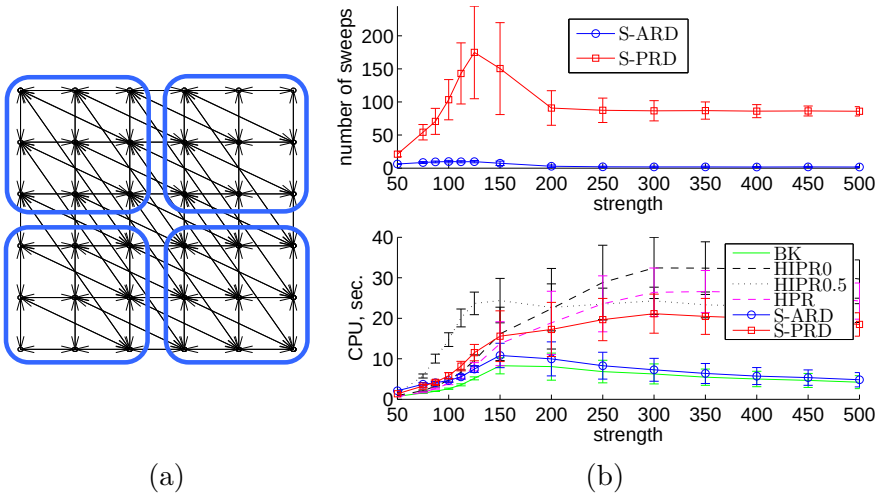
5.6 Experiments

All experiments were conducted on a system with Intel Core 2 Quad CPU@2.66Hz, 4GB memory, Windows XP 32bit and Microsoft Visual C++ compiler. Our implementation and instructions needed to reproduce the experiments can be found at `http://cmp.felk.cvut.cz/~shekhovt/d_maxflow`. We conducted 3 series of experiments:

- Synthetic experiments, in which we observe general dependencies of the algorithms, with some statistical significance, *i.e.* not being biased to a particular problem instance. It also serves as an empirical validation, as thousands of instances are solved. Here, the basic implementation of S-ARD was used.
- Sequential competition. We study sequential versions of the algorithms, running them on real vision instances (University of Western Ontario web pages 2008). Only a single core of the CPU is utilized. We fix the region partition and study how much disk I/O it would take to solve each problem when only one region can be loaded in the memory at a time. In this and the next experiment, we used the efficient implementation of ARD. Note, in the preceeding publication (Shekhovtsov and Hlavac 2011) we reported worse results with the earlier implementation.
- Parallel competition. Parallel algorithms are tested on the instances which can fully fit in 2GB of memory. All 4 cores of the CPU are allowed to be used. We compare our algorithms with two other state-of-the-art distributed implementations.

5.6.1 General Dependences: Synthetic Problems

We generated simple synthetic problems to validate the algorithms. The network is constructed as a 2D grid with a regular connectivity structure. Figure 19(a) shows an example of such a network. The edges are added to the vertices at the following relative displacements $(0,1)$, $(1,0)$, $(1,2)$, $(2,1)$, $(1,3)$, $(3,1)$, $(2,3)$, $(3,2)$, $(0,2)$, $(2,0)$, $(2,2)$, $(3,3)$, $(3,4)$, $(4,2)$. By *connectivity* we mean the number of edges incident to a vertex far enough from the boundary. Adding pairs $(0,1)$, $(1,0)$ results in connectivity 4 and so on. Each vertex is given an integer excess/deficit distributed uniformly in the interval $[-500\ 500]$. A positive number means a source link and a negative number a sink link. All edges in the graph are assigned a constant capacity, called *strength*. The network is partitioned into regions by slicing it in s equal parts in both dimensions. Thus we have 4 parameters: the number of vertices, the connectivity, the strength and the number of regions. We generate 100 instances for each value of the parameters.



(a) (b)

Figure 19 (a) Example of a synthetic problem: a network of the size 6×6, connectivity 8, partitioned into 4 regions. The source and sink are not shown. (b) Dependence on the interaction strength, for size 1000×1000, connectivity 8 and 4 regions. Plots show mean values and intervals containing 70% of the samples.

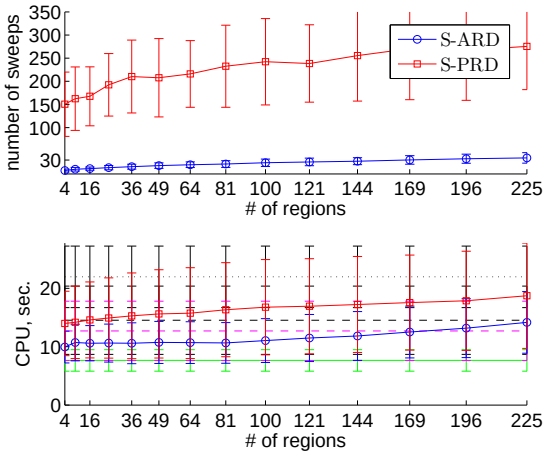


Figure 20 Dependence on the number of regions, for size 1000×1000, connectivity 8, strength 150.

Let us first look at the dependence on the strength shown in Figure 19(b). Problems with small strength are easy, because they are very local – long augmentation paths do not occur. On the other hand, long paths needs to be augmented for problems with large strength. However, finding them is easy because bottlenecks are unlikely. Therefore BK and S-ARD have a maximum in the computation time somewhere in the middle. It is more difficult to transfer the flow over long distances for push-relabel algorithms. This happens when the global relabel heuristic becomes efficient and HIPR0.5 outperforms HIPR0. The region-relabel heuristic of S-PRD allows it to outperform other push-relabel variants.

We consider all such random 2D networks are too easy in general. Nevertheless, they are useful and instructive to show basic dependences. We now select the "difficult" point for BK with the strength 150 and study other dependencies:

- The number of regions (Figure 20). For this problem family, both the number of

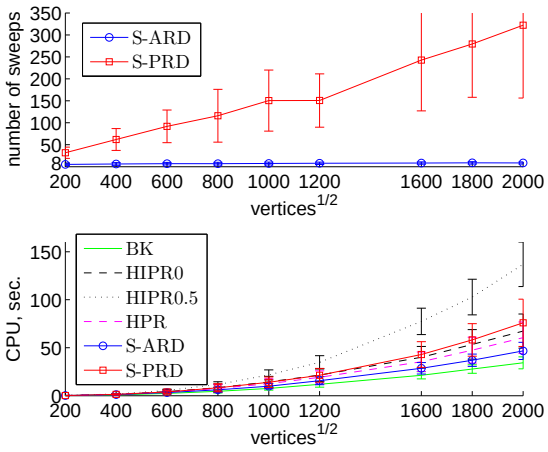


Figure 21 Dependence on the problem size, for connectivity 8, strength 150, 4 regions.

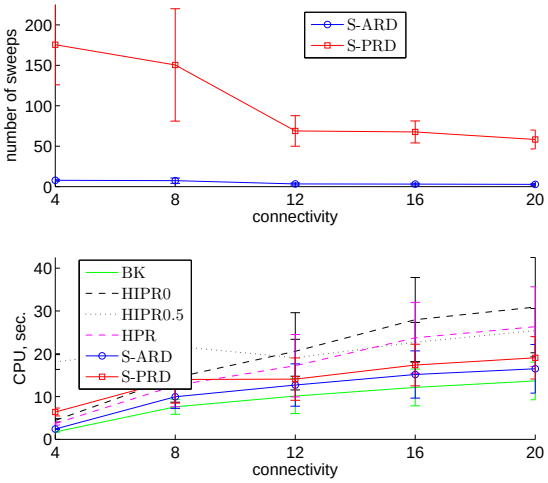


Figure 22 Dependence on the connectivity, for size 1000×1000, strength = (150 · 8)/connectivity, 4 regions.

sweeps and the computation time grows slowly with the number of regions.

- The problem size (Figure 21). Computation efforts of all algorithms grow proportionally. However, the number of sweeps shows different asymptotes. It is almost constant for S-ARD but grows significantly for S-PRD.
- Connectivity (Figure 22). Connectivity is not independent of the strength. Roughly, 4 edges with capacity 100 can transmit as much flow as 8 edges with capacity 50. Therefore while increasing the connectivity we also decrease the strength as $150 \cdot 8$ divided by connectivity in this plot.
- Workload (Figure 5.6.1). This plot shows how much time each of the algorithms spends performing different parts of computation. Note that the problems are solved on a single computer with all regions kept in memory, therefore the time on sending messages should be understood as updates of dynamic data structure of the region w.r.t. the new labeling and flow on the boundary. For S-PRD more sweeps are needed, so the total time spent in messages and gap heuristic is increased. Additionally, the gap heuristic has to take into account all vertices, unlike only

the boundary vertices in S-ARD.

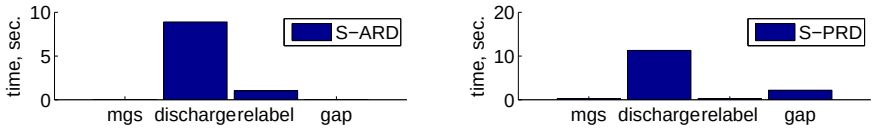


Figure 23 Workload distribution, for size 1000×1000, connectivity 8, 4 regions, strength 150. *msg* – passing the messages (updating flow and labels on the boundary), *discharge* – work done by the core solver (BK for S-ARD and HPR for S-PRD), *relabel* – the region-relabel operation, *gap* – the global gap heuristic.

5.6.2 Sequential Competition

We tested our algorithms on the MAXFLOW problem instances in computer vision obtained from University of Western Ontario web pages (2008). The data consist of typical max-flow problems in computer vision, graphics, and biomedical image analysis. *Stereo* instances are sequences of subproblems (arising in the expansion move algorithm) for which the total time should be reported. There are two models: BVZ (Boykov et al. 1998), in which the graph is a 4-connected 2D grid, and KZ2 (Kolmogorov and Zabih 2001), in which there are additional long-range links. *Multiview* 3D reconstruction models LB06 (Lempitsky et al. 2006) and BL06 (Boykov and Lempitsky 2006). Graphs of these problems are cellular complexes subdividing the space into 3D cubes and each cube into 24 smaller cells. *Surface* fitting instances LB07 (Lempitsky and Boykov 2007) are 6-connected 3D grid graphs. And finally, there is a collection of volumetric *segmentation* instances BJ01 (Boykov and Jolly 2001), BF06 (Boykov and Funka-Lea 2006), BK03 (Boykov 2003) with 6-connected and 26-connected 3D grid graphs.

To test our streaming algorithms, we used the `regulargrid` hint available in the definition of the problems to select the regions by slicing the problem into 4 parts in each dimension – into 16 regions for 2D BVZ grids and into 64 regions for 3D segmentation instances. Problems KZ2 do not have such a hint (they are not regular grids), so we sliced them into 16 pieces just by the vertex number. We did the same for the multiview LB06 instances. Though they have a `size` hint, we failed to interpret the vertex layout correctly (the separator set $\mathcal{B}$ was unexpectedly large when trying to slice along the dimensions). So we sliced them purely by the vertex number.

One of the problems we faced is the pairing of arcs that are reverse of each other. While in stereo, surface and multiview problems, the reverse arcs are consequent in the files, and can be easily paired, in 3D segmentation they are not. For a generic algorithm, not being aware of the problem's regularity structure, it is actually a non-trivial problem requiring at least the memory to read all of the arcs first. Because our goal is a relative comparison, we did not pair the arcs in 3D segmentation. This means we kept twice as many arcs than necessary for those problems. This is seen in Table 1, *e.g.*, for `babyface.n26c100`, which is 26-connected, but we construct a multigraph (has parallel arcs) with average vertex degree of 49. For some other instances, however, this is not visible, because there could be many zero arcs, *e.g.*, `liver.n26c10` which is a 26-connected grid too, but has the average vertex degree of 10.4 with unpaired arcs. The comparison among different methods is correct, since all of them are given exactly the same multigraph.

The results are presented in Table 1. We measured the real time of disk I/O. However,

it depends on the hard drive performance, other concurrently running processes as well as on the system file caching (which has effect for small problems). We therefore report total bytes written/loaded and give an estimate of the full running time for the disk speed of 100MB/s (see Table 2). Note that disk I/O is not proportional to the number of sweeps, because some regions may be inactive during a sweep and thus skipped. We did not monitor the memory usage for HIPR. It is slightly higher than that of HPR, because of keeping initial arc capacities.

For verification of solvers, we compared the flow values to the ground truth solution provided in the dataset. Additionally, we saved the cut output from each solver and checked its cost independently. Verifying the cost of the cut is relatively easy: the cut can be kept in memory and the edges can be processed form the DIMACS problem definition file on-line. An independent check of (pre-)flow feasibility would be necessary for full verification of a solver. However, it would require storing the full graph in the memory and was not implemented.

Table 1. Sequential Competition. CPU – the time spent purely on computation, excluding the time for parsing, construction and disk I/O. The total time to solve the problem is not shown. K – number of regions. RAM – memory taken by the solver; for BK in the case it exceeds 2GB limit, the expected required memory; for streaming solvers the sum of shared and region memory. I/O – total bytes read or written to the disk.

problem			BK	HIPR0	HIPR 0.5	HPR	S-ARD			S-PRD		
name	n(10^6)	$\frac{m}{n}$	CPU	CPU	CPU	CPU	CPU	sweeps	K	CPU	sweeps	K
size			RAM	RAM	RAM	RAM	RAM		I/O	RAM		I/O
stereo												
BVZ-sawtooth(20)	0.2	4.0	0.68s	3.0s	7.7s	3.8s	0.63s	6	16	3.7s	32	16
434×380			14MB			17MB	0.3+0.9MB		114MB	0.8+1.1MB		0.7GB
BVZ-tsukuba(16)	0.1	4.0	0.36s	1.9s	4.9s	2.6s	0.35s	5	16	2.1s	29	16
384×288			9.7MB			11MB	0.2+0.6MB		71MB	0.5+0.8MB		373MB
BVZ-venus(22)	0.2	4.0	1.2s	5.7s	15s	6.2s	1.1s	6	16	6.6s	36	16
434×383			15MB			17MB	0.3+0.9MB		119MB	0.8+1.1MB		0.9GB
KZ2-sawtooth(20)	0.3	5.8	1.8s	7.1s	22s	6.1s	2.2s	6	16	7.4s	23	16
38MB			33MB			36MB	1.2+2.0MB		280MB	1.8+2.5MB		1.2GB
KZ2-tsukuba(16)	0.2	5.9	1.1s	5.3s	20s	4.4s	1.4s	6	16	5.9s	18	16
26MB			23MB			25MB	1.1+1.4MB		186MB	1.4+1.7MB		0.7GB
KZ2-venus(22)	0.3	5.8	2.8s	13s	39s	10s	3.4s	8	16	14s	36	16
38MB			34MB			37MB	1.2+2.1MB		330MB	1.9+2.5MB		1.8GB
multiview												
BL06-camel-lrg	18.9	4.0	81s				63s	11	16	308s	418	16
100×75×105×24			1.6GB				19+103MB		28GB	86+122MB		0.6TB
BL06-camel-med	9.7	4.0	25s	29s	77s	59s	20s	12	16	118s	227	16
80×60×84×24			0.8GB			1.0GB	31+53MB		16GB	46+63MB		225GB
BL06-camel-sml	1.2	4.0	0.98s	1.5s	6.3s	1.8s	0.96s	9	16	4.2s	47	16
40×30×42×24			106MB			124MB	8.0+7.0MB		1.4GB	6.9+8.2MB		9.1GB
BL06-gargoyle-lrg	17.2	4.0	245s			91s	154s	21	16	318s	354	16
80×112×80×24			1.5GB			1.7GB	23+95MB		35GB	82+112MB		0.8TB
BL06-gargoyle-med	8.8	4.0	115s	17s	58s	37s	73s	16	16	143s	340	16
64×90×64×24			0.8GB			0.9GB	37+50MB		14GB	44+58MB		235GB
BL06-gargoyle-sml	1.1	4.0	6.1s	1.2s	3.0s	1.7s	3.9s	10	16	4.4s	55	16
32×45×32×24			97MB			114MB	9.3+6.6MB		1.3GB	6.9+7.7MB		9.4GB
surface												

Continued on next page

Table 1 – continued from previous page

LB07-bunny-lrg 401×396×312	49.5	6.0	 5.7GB				15s 130+87MB	6	64 49GB	416s 226+99MB	43	64 276GB
LB07-bunny-med 202×199×157	6.3	6.0	1.6s 0.7GB	20s	41s	26s 0.8GB	2.1s 33+12MB	10	64 6.5GB	16s 43+863MB	27	64 0.0MB
LB07-bunny-sml 102×100×79	0.8	5.9	0.17s 95MB	0.80s	1.8s	1.1s 101MB	0.32s 8.2+1.6MB	9	64 0.8GB	0.86s 7.9+1.9MB	19	64 2.0GB
segm												
liver.n26c10 170×170×144	4.2	10.4	6.4s 0.8GB	18s	18s	34s 0.7GB	14s 36+12MB	13	64 13GB	39s 30+13MB	157	64 82GB
liver.n26c100 170×170×144	4.2	11.1	12s 0.8GB	26s	28s	39s 0.7GB	24s 38+13MB	15	64 16GB	35s 30+14MB	98	64 66GB
liver.n6c10 170×170×144	4.2	9.8	7.2s 0.7GB	17s	25s	40s 0.7GB	14s 33+12MB	16	64 15GB	36s 28+13MB	151	64 79GB
liver.n6c100 170×170×144	4.2	10.5	15s 0.8GB	30s	34s	44s 0.7GB	19s 35+12MB	17	64 14GB	32s 29+13MB	94	64 70GB
babyface .n26c10 250×250×81	5.1	47.3	 3.7GB				179s 156+56MB	38	64 102GB	222s 173+58MB	169	64 0.6TB
babyface .n26c100 250×250×81	5.1	49.0	 3.8GB				231s 156+56MB	44	64 115GB	262s 180+57MB	116	64 0.6TB
babyface .n6c10 250×250×81	5.1	11.1	6.8s 1.0GB	38s	51s	68s 0.9GB	20s 22+16MB	17	64 19GB	100s 37+17MB	275	64 261GB
babyface .n6c100 250×250×81	5.1	11.5	13s 1.0GB	71s	65s	87s 0.9GB	24s 22+16MB	19	64 18GB	74s 37+17MB	191	64 189GB
adhead.n26c10 256×256×192	12.6	31.5	 6.2GB				128s 153+83MB	17	64 84GB	224s 195+86MB	109	64 0.8TB
adhead.n26c100 256×256×192	12.6	31.6	 2.5GB				174s 34+36MB	21	64 36GB	269s 77+39MB	129	64 354GB
bone.n26c10 256×256×119	7.8	32.3	 4.0GB				25s 122+61MB	12	64 35GB	96s 147+63MB	148	64 470GB
bone.n26c100 256×256×119	7.8	32.4	 4.0GB				29s 122+61MB	14	64 39GB	68s 147+63MB	124	64 321GB
bone.n6c10 256×256×119	7.8	11.5	7.7s 1.5GB	5.7s	17s	12s 1.4GB	7.2s 62+23MB	9	64 13GB	37s 52+25MB	195	64 188GB
bone.n6c100 256×256×119	7.8	11.6	9.1s 1.6GB	9.1s	22s	14s 1.5GB	8.7s 62+23MB	10	64 13GB	23s 52+25MB	65	64 104GB
bone_subx .n6c100 128×256×119	3.9	11.8	7.1s 0.8GB	6.3s	12s	6.4s 0.7GB	5.5s 39+12MB	12	64 7.1GB	9.4s 29+13MB	42	64 42GB
bone_subxy .n26c100 128×128×119	1.9	32.2	5.9s 1.0GB	3.9s	6.1s	4.6s 0.8GB	7.3s 92+13MB	13	64 10GB	8.7s 50+16MB	33	64 39GB
abdomen _long.n6c10 512×512×551	144.4	11.8	 29GB				179s 410+403MB	11	 196GB		> 35	 >1TB
abdomen _short.n6c10 512×512×551	144.4	11.8	 29GB				82s 410+403MB	11	 138GB			

Our new algorithms computed flow values for all problems matching those provided in the dataset, except for the following cases:

- `LB07-bunny-lrg`: no ground truth solution available (we found flow/cut of cost 15537565).
- `babyfacen26c10` and `babyfacen26c100`: we found higher flow values than those which were provided in the dataset (we found flow/cut of cost 180946 and 1990729 resp.).

The latter problems appear to be the most difficult for S-ARD in terms of both time and number of sweeps. Despite this, S-ARD requires much fewer sweeps, and conse-

problem	S-ARD		S-PRD	
	mem, MB	time	mem, MB	time
BVZ-sawtooth	1.1	1.7s	1.9	11s
BL06-camel-lrg	122	6min	208	1.8h
BL06-gargoyle-lrg	118	8.5min	194	2.4h
LB07-bunny-lrg	217	8.6min	325	54min
babyface.n26c10	212	20min	231	1.9h
adhead.n26c10	236	16min	281	2.3h
adhead.n26c100	70	9min	116	1h
bone.n26c10	183	6.3min	210	1.4h
abdomen long.n6c10	813	36min	∼800	>3h

Table 2 Estimated running time for the algorithms in the streaming mode, including the time for Disk I/O. The estimate is computed for a disk speed of 100MB/s and does not include initial problem splitting. The table also gives the total amount of memory used by the solver.

quently much less disk I/O operations than the push-relabel variant. This means that in the streaming mode, where read and write operations take a lot of time, S-ARD is clearly superior. Additionally, we observe that the time it spends for computation is comparable to that of BK, sometimes even significantly smaller.

Next, we studied the dependency of computation time and number of sweeps on the number of regions in the partition. We selected 3 representative instances of different problems and varied the number of regions in the partition. The results are presented in the Figure 24. The instance `BL06-gargoyle-sml` was partitioned by the vertex index and the remaining two problems were partitioned according to their 3D vertex layout using variable number of slices in each dimension. The results demonstrate that the computation time required to solve these problems is stable over a large range of partitions and the number of sweeps required does not grow rapidly. Therefore, for the best practical performance the partition for S-ARD can be selected to meet other requirements: memory consumption, number of computation units, *etc.* We should note however, that with refining the partition the amount of shared memory grows proportionally to the number of boundary edges. In the limit of single-vertex regions, the algorithm will turn into a very inefficient implementation of pure push-relabel.

5.6.3 Parallel Competition

In this section, we test parallel versions of our algorithms and compare them with two state-of-the-art methods. The experiments are conducted on the same machine as above (Intel Core 2 Quad CPU@2.66Hz) but allowing the use of all 4 CPUs. The goal is to see how the distributed algorithms perform in the simplified setting when they are run not in the network but on a single machine. For P-ARD/PRD we expect that the total required work would increase compared to the sequential versions because the discharges are executed concurrently. The relative speed-up therefore would be sublinear even if we managed to distribute the work between CPUs evenly. The tests are conducted on small and medium size problems (taking under 2GB of memory). For P-ARD and P-PRD we use the same partition into regions as in Table 1. For other solvers, discussed next, we tried to meet better their requirements.

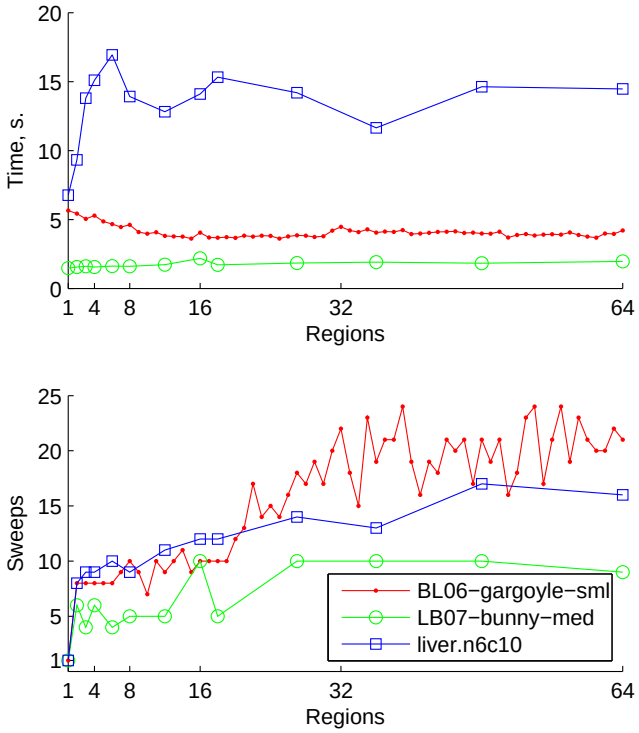


Figure 24 Dependence on the number of regions for the representative instances of multiview, stereo and segmentation. Top: CPU time used. Bottom: number of sweeps.

DD The integer dual decomposition algorithm by Strandmark and Kahl (2010)[7] uses adaptive vertex-wise step rule and randomization. With or without randomization, this algorithm is not guaranteed to terminate in polynomial time. Without the randomization, there exist a simple example with 4 vertices such that the algorithm never terminates. With the randomization, there is always a chance that it terminates, however, there is no reasonable bound on the number of iterations. Interestingly, in all of the stereo problems the algorithm terminated in a small number of iterations. However, on larger problems partitioned into 4 regions it exceeded the internal iterations bound (1000) in many cases and returned without the optimal flow/cut. In such a case, it provides only an approximate solution to the problem. Whether such a solution is of practical value is beyond us. We tested it with partitions into 2 and 4 regions (denoted **DDx2** and **DDx4** resp.). Naturally, with 2 regions the algorithm can utilize only 2 CPUs. When the number of regions is increased, the random event of termination is expected to happen less likely, which is confirmed by our experiments.

RPR The implementation of Region Push Relabel (Delong and Boykov 2008) by Sameh Khamis (v1.01, `http://vision.csd.uwo.ca/code/`) was published shortly before this work.

For RPR we constructed partition of the problem into smaller blocks. Because regions in RPR are composed dynamically out of blocks (default is 8 blocks per re-

[7] Multi-threaded maxflow library `http://www.maths.lth.se/matematiklth/personal/petter/cppmaxflow.php`

gion) we partitioned 2D problems into $64 = 8^2$ blocks and 3D problems into $512 = 8^3$ blocks. This partition was also empirically faster than a coarser one. The parameter `DischargesPerBlock` was set by recommendation of authors to 500 for small problems (stereo) and to 15000 for big problems. The implementation is specialized for regular grids, therefore multiview and KZ2 problems which do not have `regulargrid` hint cannot be solved by this method. Because of the fixed graph layout in RPR, arcs which are reverse of each other are automatically grouped together, so RPR computes on a reduced graph compared to other methods. Let us also note that because of the dynamic regions, RPR is not fully suitable to run in a distributed system.

The method of Liu and Sun (2010) (parallel, but not distributed) would probably be the fastest one in this competition (as could be estimated from the results reported by Liu and Sun (2010)), however the implementation is not publicly available.

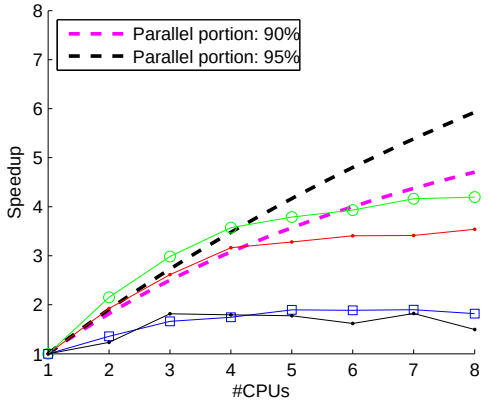
Table 3. Parallel Competition. The mark X denotes that the method returned only an approximate solution due to limited iterations.

problem	BK	DDx2		DDx4		P-ARD		P-PRD		RPR	
	time	time, sweeps									
stereo											
BVZ-sawtooth(20)	0.68s	0.52s	7	0.37s	11	0.30s	7	2.4s	31	4.8s	274
BVZ-tsukuba(16)	0.36s	0.28s	6	0.20s	8	0.17s	5	1.5s	33	2.1s	197
BVZ-venus(22)	1.2s	0.84s	7	0.59s	9	0.50s	7	4.9s	36	8.0s	466
KZ2-sawtooth(20)	1.8s	1.2s	11	0.91s	16	0.96s	6	4.9s	23		
KZ2-tsukuba(16)	1.1s	0.67s	7	0.52s	11	0.70s	8	4.9s	22		
KZ2-venus(22)	2.8s	1.9s	7	1.3s	12	1.5s	10	10s	39		
multiview											
BL06-camel-med	25s	18s	221	13s	260	8.7s	14	81s	322		
BL06-camel-sml	0.98s	0.63s	11	0.49s	27	0.49s	10	2.5s	70		
BL06-gargoyle-lrg	245s	120s	517	mem		58s	23	mem			
BL06-gargoyle-med	115s	59s	20	38s	50	27s	21	79s	219		
BL06-gargoyle-sml	6.1s	3.0s	19	1.9s	19	1.6s	10	2.4s	52		
surface											
LB07-bunny-med	1.6s	1.3s	11	1.1s	11	1.3s	13	12s	35	37s	349
LB07-bunny-sml	0.17s	0.12s	11	0.12s	11	0.21s	8	0.58s	21	3.5s	99
segm											
liver.n6c10	7.2s	X 7.6s	1000	X 22s	1000	8.9s	23	23s	164	5.1s	1298
liver.n6c100	15s	17s	31	X 21s	1000	12s	17	23s	102	7.3s	1722
babyface.n6c10	6.8s	8.8s	61	X 24s	1000	12s	22	61s	135	17s	4399
babyface.n6c100	13s	16s	338	X 20s	1000	17s	23	61s	179	22s	4833
bone.n6c10	7.7s	5.2s	22	X 8.2s	1000	4.9s	17	16s	182	6.3s	918
bone.n6c100	9.1s	5.3s	12	4.1s	17	6.2s	13	14s	70	7.9s	1070
bone_subx.n6c100	7.1s	6.3s	24	5.2s	34	3.9s	17	5.8s	48	1.5s	747
bone_subxy.n26c100	5.9s	3.4s	11	3.2s	12	5.8s	16	6.0s	37	hang	

Results The results are summarized in Table 3. The time reported is the wall clock time passed in the calculation phase, not including any time for graph construction. The number of sweeps for DD has the same meaning as for P-ARD/PRD, it is the number of times all regions are synchronously processed. RPR however is asynchronous and uses dynamic regions. For it, we define sweeps = `block_discharges/number_of_blocks`.

Comparing to Table 1, we see that P-ARD on 4 CPUs is about $1.5 - 2.5$ times faster than S-ARD. The speed-up over BK varies from 0.8 on `livern6c10` to more than 4 on `gargoyle`.

We see that DD gets lucky some times and solves the problem really quickly, but often it fails to terminate. We also observe that our variant of P-PRD (based on highest first selections rule) is a relatively slow, but robust distributed method. RPR, which is based on LIFO selection rule, is competitive on the 3D segmentation problems but is slow on other problems, despite its compile-time optimization for the particular graph structure. It also uses a relatively higher number of blocks. The version we



problem	regions	MEM	1CPU	8CPUs
BL06-gargoyle-lrg	16	1.6GB	137.4s	38.82s
BL06-camel-lrg	16	1.8GB	67.14s	16.00s
LB07-bunny-lrg	64	12GB	12.94s	7.11s
liver.n6c100	64	0.9GB	18.41s	12.21s

Figure 25 Speedup of P-ARD with the number of CPUs used. The extended legend shows the time to solve each problem with 1 and 8CPUs (does not include initialization). Dashed lines correspond to the speedup in the ideal case (Amdahl's law) when the parallel portion of the computation is 90% and 95%.

tested always returned the correct flow value but often a wrong (non-optimal) cut. Additionally, for 26 connected `bone_subxy.n26c100` it failed to terminated within 1 hour.

5.6.4 Scalability with Processors

We performed additional tests of P-ARD in the shared memory mode using 1-8 CPUs. This experiment was conducted on a system with Intel(R) Core(TM)i7 CPU 870@2.9GHz, 16GB memory, linux 64bit and gcc compiler. The plot in Figure 25 shows the speedup of solving the problem (excluding initialization) using multiple CPUs over the time needed by a single CPU. For this test, we selected medium and large size problems of different kind that can fully fit in 16GB of memory. The two problems which were taking longer in the serial implementation scaled relatively well. On the other side, the largest `LB07-bunny` problem did not scale well. We believe that the limiting factor here is the memory bandwidth. We inspected that the sequential part of the computation (boundary relabel heuristic, synchronous message exchange) occupy less than 10% of the total time for all four problems. The fully parallel part should exhibit a linear speed-up in the ideal case of even load. The load for `LB07-bunny` should be relatively even, since we have enough regions (64) to be processed with 8 CPUs. Still, there is no speed-up observed in the parallel part of the first sweep (where most of the work is done) when scaling from 4 to 8 CPUs.

It is most probable that reducing memory requirements (*e.g.*, by having dedicated graph implementation for regular grids) would also lead to a speed-up of the parallel solver. We also observed that the 32 bit compilation (pointers take 32 bits) runs faster than the 64 bit compilation. It is likely that our implementation can be optimized further for the best parallel performance. We should however consider that preparing the data for the problem and splitting it into regions is another time-consuming part,

which needs to be parallelized.

5.7 Region Reduction

In this section, we attempt to reduce the region network as much as possible by identifying and eliminating vertices which can be decided optimally regardless of the reminder of the full network outside of the region. If it was possible to decide about many vertices globally optimally inside a region network, the whole problem would simplify a lot. It would require less memory and could be potentially solved without distributing and or partitioned again into larger regions. We propose an improved algorithm for such a reduction and its experimental verification. This preprocessing is studied separately and was not applied in the tests of distributed algorithms above. Experiments with vision problems (Table 4) showed that while 2D problems can be significantly reduced, many of the higher-dimension problems do not allow a substantial reduction.

Some vertices become disconnected from the sink in the course of the studied algorithms (S/P-ARD, S/P-PRD). If they are still reachable from the source, they must belong to the source set of any optimal cut. Such vertices do not participate in further computations and the problem can be reduced by excluding them. Unfortunately, the opposite case, when a vertex must be strictly in the sink set is not discovered until the very end of the algorithms.

The following algorithm attempts to identify as many vertices as possible for a given region. It is based on the following simple consideration: if a vertex is disconnected from the sink in G^R as well as from the region boundary, B^R, then it is disconnected from the sink in G; if a vertex is not reachable from the source in G^R as well as from B^R then it is not reachable from the source in G.

Let us say that a vertex v is a *strong source vertex* (resp. a *strong sink vertex*) if for any optimal cut $(C, \bar{C})$, $v \in C$ (resp. $v \in \bar{C}$). Similarly, v will be called a *weak source vertex* (resp. *weak sink vertex*), if there exists an optimal cut $(C, \bar{C})$ such that $v \in C$ (resp. $v \in \bar{C}$).

Kovtun (2004) suggested to solve two auxiliary problems, modifying network G^R by adding infinite capacity links from the boundary vertices to the sink and in the second problem adding infinite capacity links from the source to the boundary vertices. In the first case, if v is a strong source vertex in the modified network G^R, it is also a strong source vertex in G. Similarly, the second auxiliary problem allows to identify strong sink vertices in G. It requires solving a MAXFLOW problem on G^R twice. We improve this construction by reformulating it as the following algorithm finding a single flow in G^R.

Statement 61. Sets B^S and B^T constructed in step 2 are disjoint.

Proof. We have $s \nrightarrow t$ after step 1, hence there cannot exist simultaneously a path from s to v and a path from v to t. □

After step 1, the network G^R is split into two disconnected networks: with vertices reachable from s and vertices from which t is reachable. Therefore, any augmentations occurring in steps 4 and 5 act on their respective subnetworks and can be carried independently of each other. On the output of Algorithm 7, we have: $s \nrightarrow B^R \cup \{t\}$ and $B^R \cup \{s\} \nrightarrow t$. The classification of vertices is shown in Figure 26.

Augmenting on (s, t) in step 1 and on (s, B^S) in the step 4 is the same work as done in `ARD` (where (s, B^S) paths are augmented in the order of labels of B^S). This is

Algorithm 7: Region Reduction (G^R, B^R)

```
   /* Input: network G^R, boundary B^R.                          */
1  Augment(s, t);
2  B^S := {v | v ∈ B^R, s → v};          /* source boundary set */
3  B^T := {v | v ∈ B^R, v → t};            /* sink boundary set */
4  Augment(s, B^S);
5  Augment(B^T, t);
6  foreach v ∈ R do
7      if s → v then v is strong source vertex;
8      if v → t then v is strong sink vertex;
9      otherwise
10         if v ↛ B^R then v is weak source vertex;
11         if B ↛ v^R then v is weak sink vertex;
```

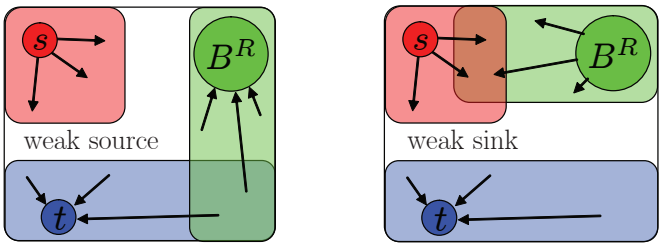


Figure 26 Classification of vertices in V^R build by Algorithm 7. Vertices reachable from s are strong source vertices. Vertices from which t is reachable are strong sink vertices. The remaining vertices can be classified as weak source vertices (a) if they cannot reach boundary, or as weak sink vertices (b) if they are not reachable from the boundary. Some vertices are both: weak source and weak sink, this means they can be on both sides of a (non-unique) optimal cut (but not independently).

not a coincidence, these algorithms are very much related. However, the augmentation on (B^T, t) in step 5 cannot be executed during ARD. It would destroy validity of the labeling. We therefore consider Algorithm 7 as a separate general preprocessing.

If v is a weak source vertex, it follows that it is not a strong sink vertex. In the preflow pushing algorithms, we find the cut $(\bar{T}, T)$, where T is the set of all strong sink vertices in G. We consider that v is *decided* if it is a strong sink or a weak source vertex.

Table 4 gives the percentage of how many vertices are decided (and hence can be excluded from the problem) by Algorithm 7 for computer vision problems. It is seen that in stereo problems, a large percent of vertices is decided. These problems are rather local and potentially can be fully solved by applying Algorithm 7 on several overlapping windows. In contrast, only a small fraction can be decided locally for many other problems.

5.8 Tightness of $O(n^2)$ bound for PRD

In this section, we give an example of a network, its partition into regions and a sequence of valid push and relabel operations, implementing PRD, such that S/P-PRD runs in

BVZ-sawtooth(20)	80.0%	LB07-bunny-sml	15.6%
BVZ-tsukuba(16)	72.8%	liver.n26c10	7.1%
BVZ-venus(22)	70.2%	liver.n26c100	5.3%
KZ2-sawtooth(20)	85.0%	liver.n6c10	7.2%
KZ2-tsukuba(16)	69.9%	liver.n6c100	5.3%
KZ2-venus(22)	75.8%	babyface.n26c10	29.3%
BL06-camel-lrg	2.0%	babyface.n26c100	30.9%
BL06-camel-med	2.3%	babyface.n6c10	35.4%
BL06-camel-sml	4.6%	babyface.n6c100	33.7%
BL06-gargoyle-lrg	6.0%	adhead.n26c10	0.3%
BL06-gargoyle-med	2.4%	adhead.n26c100	0.3%
BL06-gargoyle-sml	9.8%	adhead.n6c10	0.2%
LB07-bunny-lrg	11.4%	adhead.n6c100	0.1%
LB07-bunny-med	13.1%	bone.n26c10	8.7%
bone.n26c100	6.9%	bone_subxyz.n6c100	6.6%
bone.n6c10	8.8%	bone_subxyz_subx.n26c10	7.9%
bone.n6c100	7.0%	bone_subxyz_subx.n26c100	6.6%
bone_subx.n26c10	6.6%	bone_subxyz_subx.n6c10	8.2%
bone_subx.n26c100	6.6%	bone_subxyz_subx.n6c100	6.6%
bone_subx.n6c10	6.3%	bone_subxyz_subxy.n26c10	11.3%
bone_subx.n6c100	6.3%	bone_subxyz_subxy.n26c100	9.5%
bone_subxy.n26c10	6.6%	bone_subxyz_subxy.n6c10	12.7%
bone_subxy.n26c100	6.6%	bone_subxyz_subxy.n6c100	9.3%
bone_subxy.n6c10	6.4%	abdomen_long.n6c10	1.7%
bone_subxy.n6c100	6.3%	abdomen_short.n6c10	6.3%
bone_subxyz.n26c10	6.6%	bone_subxyz.n6c10	6.6%
bone_subxyz.n26c100	6.6%		

Table 4 Percentage of vertices which can be decided by preprocessing. The problems are partitioned into regions the same way as in Table 1. The average number over subproblems is shown for stereo problems.

$\Omega(n^2)$ sweeps.

We start by an auxiliary example, in which the preflow is transferred from a vertex to a boundary vertex with a higher label. In this example, some inner vertices of a region are relabeled, but not any of the boundary vertices. It will imply that the total number of sweeps cannot be bounded by the number of relabellings of boundary vertices alone.

Example 7. Consider a network of 6 regular vertices in Figure 27. Assume all edges have infinite capacity, so only non-saturating pushes occur. There are two regions $R_1 = \{1, 2, 3, 4, 5\}$ and $R_2 = \{6\}$. Figure 27 shows a sequence of valid push and relabel operations. We see that some vertices get risen due to relabel, but the net effect is that flow excess from vertex 1 is transferred to vertex 6, which had a higher label initially. Moreover, none of the boundary vertices (vertices 5,6) are relabeled.

Example 8. Consider the network in Figure 28. The first step corresponds to a sequence of push and relabel operations (same as in Figure 27) applied to the chain $(1, 2a, 3a, 4a, 5, 6)$. Each next step starts with the excess at vertex 1. Chains are selected in turn in the order a, b, c. It can be verified from Figure 28 that each step is a valid possible outcome of PRD applied first to R_1 and then to R_2. The last configuration repeats the first one with all labels raised by 6. The same loop exactly may be repeated many times.

It is seen that vertices $1, 5, 6$ are relabeled only during pushes in the chains a and b and never during pushes in chain c. If there were more chains like chain c, it would take many iterations (= number of region discharge operations) before boundary vertices are

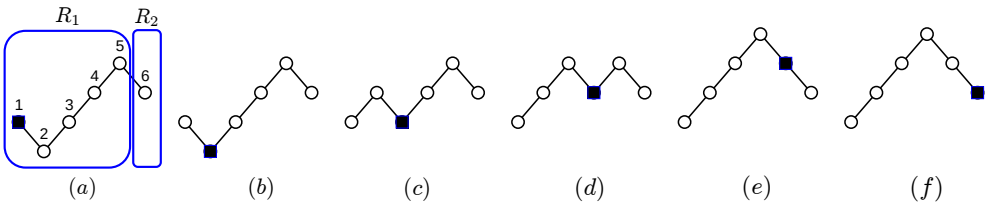


Figure 27 Steps of Example 1. The height of a vertex corresponds to its label. The black box shows the vertex with excess in each step. The source and the think vertices are not shown. (a)-(b) flow excess is pushed to vertex 2; (c) vertex 2 is relabeled, so that two pushes are available and excess is pushed to vertex 3; (d-f) similar.

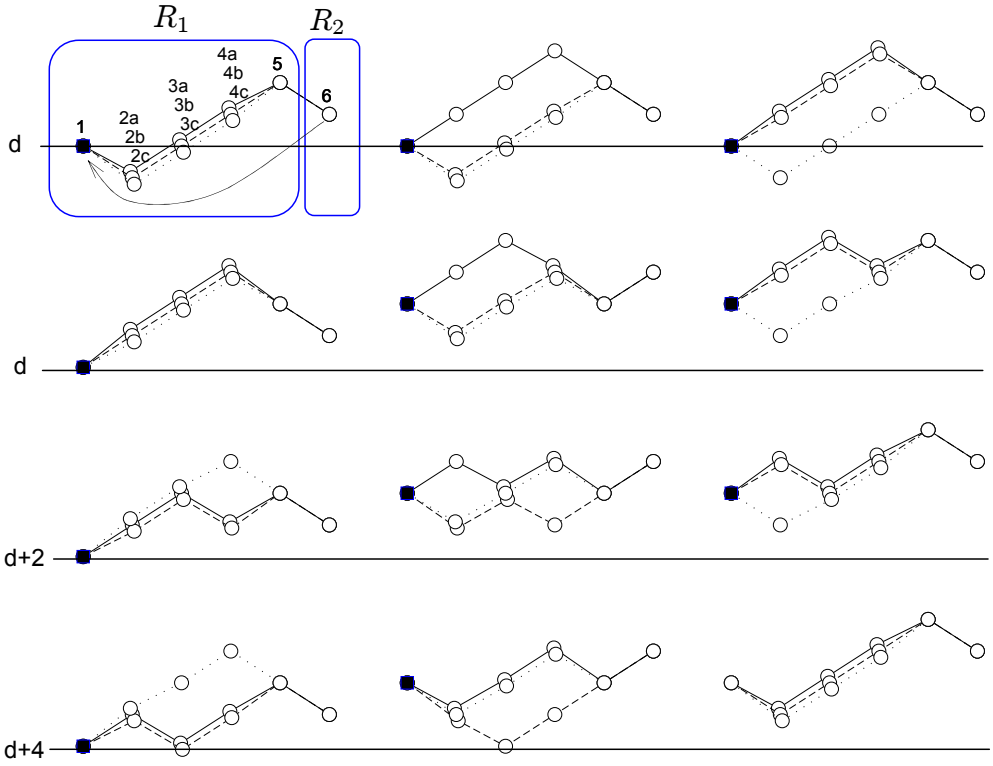


Figure 28 Steps of Example 2. Top left: a network with several chains of vertices like in Example 1. Vertices 1, 5, 6 are common for all chains but there are separate copies of vertices 2, 3, 4 denoted by letters. In addition, there is a reverse arc from vertex 6 to vertex 1. From left to right, top to bottom: one step of transferring a flow from vertex 1 to vertex 6 using one of the chains and then pushing it through the arc (6,1), relabeling 6 when necessary. The label of the first vertex is increased three times by 2.

risen. Let there be k additional chains in the graph (denoted d, e,...) handled exactly the same way as chain c. The total number of vertices in the graph is $n = 3k + \text{const}$. Therefore, it will take $\Omega(n)$ region discharges to complete each loop raising all vertices by a constant value. The number of discharges needed in order that vertex 1 reaches label D, is $\Omega(nD)$. To make the example complete, we add a chain of vertices initially having labels $1, 2, 3, \ldots, D$ to the graph such that there is a path from vertex 1 to the sink through a vertex with label D. Clearly, we can arrange that $D = \Omega(n)$. The algorithm needs $\Omega(n^2)$ discharges on this example.

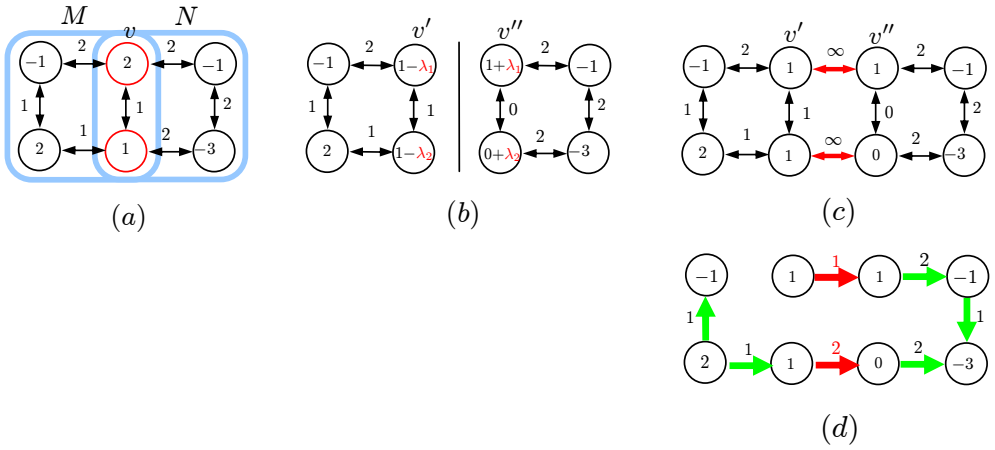


Figure 29 Interpretation of the dual decomposition. (a) Example of a network with denoted capacities. Terminal capacities are shown in circles, where "+" denotes s-link and "−" denotes t-link. $M \cap N$ is a separator set. (b) The network is decomposed into two networks holding copies of the separator set. The associated capacities are divided (not necessarily evenly) between two copies. The variable λ_1 is the Lagrangian multiplier of the constraint $x_v = y_v$. (c) Introducing edges of infinite capacity enforces the same constraint, that v' and v'' are necessarily in the same cut set of any optimal cut. (d) A maximum flow in the network (c), the flow value on the red edges corresponds to the optimal value of the dual variables λ.

Because there is only one active vertex at any time, the example is independent of the rule used to select the active vertex (highest label, FIFO, *etc.*). By the same reason, it also applies to parallel PRD. Because the number of regions is constant, the number of sweeps required is also $\Omega(n^2)$.

For a comparison, noting that the number of boundary vertices is 3, we see that S-ARD algorithm will terminate in a constant number of sweeps for arbitrary k in this example.

5.9 Relation to the Dual Decomposition

In our approach, we partition the set of vertices into regions and couple the regions by sending the flow through the inter-region edges. In the dual decomposition for MINCUT (Strandmark and Kahl 2010) detailed below, a separator set of the graph is selected and each subproblem gets a copy of the separator set. The coupling is achieved via the constraint that the cut of the separator set must be consistent across the copies. We show now how the dual variables of Strandmark and Kahl (2010) can be interpreted as a flow, thus relating their approach to ours. While they solve the same MINCUT problem as we do, the network they construct is slightly different, in that it has extra vertices connected by infinite capacity edges.

Decomposition of the MINCUT problem into two parts is formulated by Strandmark and Kahl (2010) as follows. Let $M, N \subset V$ are such that $M \cup N = V$, $\{s, t\} \subset M \cap N$ and there are no edges in E from $M \backslash N$ to $N \backslash M$ and vice-versa. Let $x \colon M \to \{0, 1\}$ and $y \colon N \to \{0, 1\}$ be the indicator variables of the cut set, where 0 corresponds to the

source set. Then the MINCUT problem without excess can be reformulated as:

$$\begin{aligned} &\min_{x,y} C^M(x) + C^N(y)\,, \\ &\text{subj. } \begin{cases} x_s = y_s = 0\,, \\ x_t = y_t = 1\,, \\ x_i = y_i \quad \forall i \in M \cap N\,, \end{cases} \end{aligned} \tag{223}$$

where

$$\begin{aligned} C^M(x) &= \sum_{(i,j)\in E^M} c^M(i,j)(1-x_i)x_j\,, \\ C^N(y) &= \sum_{(i,j)\in E^N} c^N(i,j)(1-y_i)y_j\,, \end{aligned} \tag{224}$$

$$\begin{aligned} &c^M(i,j) + c^N(i,j) = c(i,j)\,, \\ &c^M(i,j) = 0 \quad \forall i,j \in N\backslash M\,, \\ &c^N(i,j) = 0 \quad \forall i,j \in M\backslash N\,, \end{aligned} \tag{225}$$

$E^M = (M \times M) \cap E$ and $E^N = (N \times N) \cap E$. The minimization over x and y decouples once the constraint $x_i = y_i$ is absent. The dual decomposition approach is to solve the dual problem:

$$\max_{\lambda} \Big[\min_{\substack{x \\ x_s=0 \\ x_t=1}} \Big(C_M(x) + \sum_{i\in M\cap N} \lambda_i(1-x_s)x_i \Big) + \min_{\substack{y \\ y_s=0 \\ y_t=1}} \Big(C_N(y) - \sum_{i\in M\cap N} \lambda_i(1-y_s)y_i \Big) \Big]\,, \tag{226}$$

where the dual variable λ is multiplied by the extra terms $(1-x_s) = (1-y_s) = 1$ to show explicitly that the inner minimization problems are instances of the minimum cut problem.

We observe that dual variables λ correspond to the flow on the artificial edges of infinite capacity between the copies of the vertices of the separator set as illustrated by Figure 29. Indeed, consider a vertex v in the separator set. The dual variable λ_v contributes to the increase of the terminal link (v', t) in the subproblem M and to the decrease of the terminal link (v'', t) in the subproblem N. This can be equivalently represented as an augmentation of flow of λ_v on the cycle v', t', v'' in the network Figure 29(c). The optimal flow in the network Figure 29(c) on the constraint edges will therefore correspond to the optimal λ. This construction could be easily extended to the case when a vertex v from the separator set is shared by more than two subproblems.

There exist an optimal integer flow for a problem with integer capacities. This observation provides an alternative proof of the theorem (Strandmark and Kahl 2010, Theorem 2)[8], stating that there exist an integer optimal λ. Despite the existence of an integer solution, the integer subgradient algorithm (Strandmark and Kahl 2010) is not guaranteed to find it.

The algorithms we introduced can be applied to such a decomposition by running them on the extended graph Figure 29(c), where vertices of the separator set are duplicated and linked by additional edges of infinite capacity. Our algorithms are guaranteed to terminate and find the minimum cut and the optimal dual variables λ. It

[8] Strandmark and Kahl (2010) stated their theorem for even integer costs in the case of two-subproblem separator sets. They remarked that a multiple of 4, resp., 8 is needed in the cases of decompositions for 2D and 3D grids. However, this multiplication is unnecessary if we choose to split the cost unevenly but preserving the integrality (like we did in the example).

could be observed, however, that this construction does not allow to reduce the number of boundary vertices or the number of inter-region edges, while the size of the regions increases. Therefore, it is more beneficial to use the plain graph partition scheme with our algorithms, not using duplicated vertices.

6 Conclusion

Partial Optimality We presented a new sufficient condition for deriving partial optimal assignment in a multi-label energy minimization. Our framework allows for unified analysis of the methods previously proposed in the literature (Desmet et al. 1992; Kovtun 2003, 2004; Boros et al. 2006; Kohli et al. 2008). The proposed sufficient condition include the conditions used by the methods in the literature as special cases. At the same time, it is a polynomially verifiable one. The verification is expressed via a special LP relaxation. A main unifying property for partial optimality methods in this form is that they preserve all solutions of the LP relaxation, *i.e.*, LP relaxation cannot be tightened by these methods. Theorem 32 proves existence of an equivalent transformation of the problem such that the sufficient condition is expressed in local inequalities. This allows to derive a simplified LP for verification of the condition. Theorem 21 proves for a subclass of partial optimality methods (including methods of Kovtun (2003)) that the fixed points of the fusion move algorithm satisfy the derived partial optimality guarantees. We studied subclasses, in which the maximum improving projections can be found efficiently. Theorem 40 proves that submodular truncation for binary variables preserves all projections of the form $x \to x \vee y$, allowing to find maximum improving projection of this type by reduction to a submodular problem. Algorithm 4 finds in polynomial time the maximum improving projection of the type "eliminate z by switching to y" for multilabel problems.

Interestingly, DEE and auxiliary submodular problems are not connected with lower bounds. They are nevertheless unified now with (M)QPBO methods, which are derived via lower bounds. In methods of Kovtun (2003), to obtain partial optimalities several sufficient conditions have to be satisfied simultaneously. QPBO method and approaches with submodular lower bounds (Kahl and Strandmark 2011; Kolmogorov 2012a) maximize a lower bound and obtain optimality guarantees as a by-product. Our approach is more direct: we formalize the maximality of optimal partial solutions in a given class and derive methods achieving it.

We believe that a large part of the approach (including the characterization) is extendible to higher-order models and higher-order relaxations. However, there is still a number of open questions arising from our approach in the pairwise case. We did not fully characterize strictly improving projections and consequently some of the theorems in section 4.4 are lacking the strict counterpart. We proposed that the projection may have non-integer weights, but considered only integer projections (except for showing DEE). The set of improving projections for a given function is represented by inequalities which are linear in the projection itself (except for the idempotency, which can be omitted). Hence, in principle, we can optimize some criterion over this convex set. In practice, we are lacking a low-dimensional convex parametric family of projections that would be flexible enough. An ideal criterion would be to maximize the dimensionality of the null space of the projection (equivalent to rank minimization), however even a simpler approximate criterion can provide exact guarantees to the initial energy minimization problem.

Distributed Mincut We developed a new algorithm for MINCUT problem on sparse graphs, which combines augmenting paths and push-relabel approaches. We proved the worst case complexity guarantee of $O(|\mathcal{B}|^2)$ sweeps for the sequential and parallel variants of the algorithm (S/P-ARD). There are many algorithms in the literature with complexities in terms of elementary arithmetic operations better than we can prove. Nevertheless, we showed that our algorithms are fast and competitive in practice, even in the shared memory model.

We proposed an improved algorithm for the local problem reduction (§5.7) and determined that most of our test instances are difficult enough in the sense that very few vertices can be decided optimally by looking at individual regions. The result that S/P-ARD solves test problems in few tens of sweeps is thus non-trivial. We also gave a novel parallel version of the region push-relabel algorithm of Delong and Boykov (2008). We provided a number of auxiliary results to relate our approach to the state-of-the-art.

Both in theory and practice (randomized test), S-ARD has a better asymptote in the number of sweeps than the push-relabel variant. Experiments on real instances showed that when run on a single CPU and the whole problem fits into the memory S-ARD is comparable in speed with the non-distributed MAXFLOW implementation by Boykov and Kolmogorov (2004), and is even significantly faster in some cases. When only a single region is loaded into memory at a time, S-ARD uses much fewer disk I/O than S-PRD. We also demonstrated that the running time and the number of sweeps are very stable with respect to the partition of the problem into up to 64 regions. In the parallel mode, using 4 CPUs, P-ARD achieves a relative speedup of about $1.5 - 2.5$ times over S-ARD and uses just slightly larger number of sweeps. P-ARD compares favorably to other parallel algorithms, being a robust method suitable for a use in a distributed system.

Our algorithms are implemented for generic graphs. Clearly, it is possible to specialize the implementation for grid graphs. It would reduce the memory consumption and might reduce the computation time as well.

A practically useful mode could be actually a combination of a parallel and sequential processing, when several regions are loaded into the memory at once and processed in parallel. There are several particularly interesting combinations of algorithm parallelization and hardware, which may be exploited: 1) parallel on several CPUs, 2) parallel on several network computers, 3) sequential, using Solid State Drive, 4) sequential, using GPU for solving region discharge.

There is the following simple way how to allow region overlaps in our framework. Consider a sequential algorithm that is allowed to keep 2 regions in memory at a time. It can then load pairs of regions $(1, 2)$, $(2, 3)$, $(3, 4)\ldots$, and alternate between the regions in a pair until both are discharged. With PRD, this is efficiently equivalent to discharging twice larger regions with a $1/2$ overlap and may significantly decrease the number of sweeps required. In the case of a 3D grid, it would take 8 times more regions to allow overlaps in all dimensions. However, to meet the same memory limit, the regions have to be 8 times smaller. It has to be verified experimentally whether it is beneficial. In fact, the Region Push-Relabel implementation of Delong and Boykov (2008) uses exactly this strategy: a dynamic region is composed out of a number of smaller blocks and blocks are discharged until the whole region is not discharged. It is likely that with this approach we could further reduce the disk I/O in the case of the streaming solver.

Bibliography

Hammer, P. L. (1965). Some network flow problems solved with pseudo-Boolean programming. In: *Operation research* 13, pp. 388–399.

Hammersley, J. M. and Clifford, P. (1971). *Markov fields on finite graphs and lattices.*

Nemhauser, G. and Trotter Jr., L. (1975). Vertex packings: structural properties and algorithms. English. In: *Mathematical programming* 8 (1), pp. 232–248.

Koval, V. and Schlesinger, M. (1976). Two-dimensional programming in image analysis problems. In: *Automatics and telemechanics* 2, pp. 149–168. In Russian, See review by Werner (2007).

Shlezinger, M. (1976). Syntactic analysis of two-dimensional visual signals in the presence of noise. In: *Cybernetics and systems analysis* 4, pp. 113–130. See review by Werner (2007).

Fisher, M., Nemhauser, G., and Wolsey, L. (1978). An analysis of approximations for maximizing submodular setfunctions i. In: *Mathematical programming*, pp. 265–294.

Hammer, P., Hansen, P., and Simeone, B. (1984). Roof duality, complementation and persistency in quadratic 0-1 optimization. In: *Mathematical programming*, pp. 121–155.

Goldberg, A. (1987). Efficient graph algorithms for sequential and parallel computers. PhD thesis. Massachusetts Institute of Technology.

Lu, S. H. and Williams, A. C. (1987). Roof duality for polynomial 0-1 optimization. In: *Mathematical programming* 37(3), pp. 357–360.

Goldberg, A. V. and Tarjan, R. E. (1988). A new approach to the maximum flow problem. In: *Journal of the ACM* 35.

Greig, D., Porteous, B., and Seheult, A. (1989). Exact maximum a posteriori estimation for binary images. In: *Journal of the royal statistical society, series B* 51(2), pp. 271–279.

Boros, E., Hammer, P. L., and Sun, X. (1991). *Network flows and minimization of quadratic pseudo-Boolean functions.* Tech. rep. RRR 17-1991. RUTCOR.

Goldberg, A. V. (1991). Processor-efficient implementation of a maximum flow algorithm. In: *Information processing letters* 38(4), pp. 179–185.

Desmet, J. et al. (1992). The dead-end elimination theorem and its use in protein side-chain positioning. In: *Nature* 356, pp. 539–542.

Cherkassky, B. V. and Goldberg, A. V. (1994). *On implementing push-relabel method for the maximum flow problem.* Tech. rep.

Goldstein, R. F. (1994). Efficient rotamer elimination applied to protein side-chains and related spin glasses. In: *Biophysical journal* 66(5), pp. 1335–1340.

Anderson, R. and Setubal, J. C. (1995). A parallel implementation of the push-relabel algorithm for the maximum flow problem. In: *Journal of parallel and distributed computing* 29(1), pp. 17–26.

Bertsekas, D. (1995). *Nonlinear programming.* Athena Scientific.

Lasters, I., De Maeyer, M., and Desmet, J. (1995). Enhanced dead-end elimination in the search for the global minimum energy conformation of a collection of protein side chains. In: *Protein engineering* 8(8), pp. 815–22.

Adams, W. P., Lassiter, J. B., and Sherali, H. D. (1998). Persistency in 0-1 polynomial programming. In: *Mathematics of operations research* 23(2), pp. 359–389.

Boykov, Y., Veksler, O., and Zabih, R. (1998). Markov random fields with efficient approximations. In: *Computer vision and pattern recognition conference*, pp. 648–655.

Goldberg, A. V. and Rao, S. (1998). Beyond the flow decomposition barrier. In: *Journal of ACM* 45 (5), pp. 783–797.

Koster, A. M., Hoesel, S., and Kolen, A. (1998). The partial constraint satisfaction problem: facets and lifting theorems. In: *Operations research letters* 23, 89–97(9).

Lauritzen, S. L. (1998). *Graphical models*. 17. Oxford Science Publications.

Boykov, Y., Veksler, O., and Zabih, R. (1999). Fast approximate energy minimization via graph cuts. In: *International conference on computer vision*. Vol. 1, pp. 377–384.

Boykov, Y. and Jolly, M.-P. (2000). Interactive organ segmentation using graph cuts. In: *Medical image computing and computer-assisted intervention (MICCAI)*, pp. 276–286.

Felzenszwalb, P. F. and Huttenlocher, D. P. (2000). Efficient matching of pictorial structures. In: *Computer vision and pattern recognition conference*, pp. 2066–2076.

Pierce, N. A. et al. (2000). Conformational splitting: a more powerful criterion for dead-end elimination. In: *Journal of computational chemistry* 21(11), pp. 999–1009.

Roy, S. and Govindu, V. (2000). MRF solutions for probabilistic optical flow formulations. In: *International conference on pattern recognition*, p. 7053.

Schlesinger, M. I. and Flach, B. (2000). Some solvable subclasses of structural recognition problems. In: *Czech pattern recognition workshop 2000, invited paper*. Chezh Pattern Recognition Society.

Boykov, Y. and Jolly, M.-P. (2001). Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images. In: *International conference on computer vision*. Vol. 1, pp. 105–112.

Boykov, Y., Veksler, O., and Zabih, R. (2001). Fast approximate energy minimization via graph cuts. In: *IEEE transactions on pattern analysis and machine intelligence* 23(11), pp. 1222–1239.

Chekuri, C. et al. (2001). Approximation algorithms for the metric labeling problem via a new linear programming formulation. In: *In symposium on discrete algorithms*, pp. 109–118.

Kolmogorov, V. and Zabih, R. (2001). Computing visual correspondence with occlusions via graph cuts. In: *International conference on computer vision*, pp. 508–515.

Lafferty, J. D., McCallum, A., and Pereira, F. C. N. (2001). Conditional random fields: probabilistic models for segmenting and labeling sequence data. In: *International conference on machine learning*, pp. 282–289.

Looger, L. L. and Hellinga, H. W. (2001). Generalized dead-end elimination algorithms make large-scale protein side-chain structure prediction tractable: implications for protein design and structural genomics. In: *Journal of molecular biology* 307, pp. 429–445.

Boros, E. and Hammer, P. (2002). Pseudo-Boolean optimization. In: *Discrete applied mathematics* 1-3(123), pp. 155–225.

Schlesinger, M. I. and Flach, B. (2002). Analysis of optimal labelling problems and application to image segmentation and binocular stereovision. In: *International east-west-vision workshop*.

Boykov, Y. (2003). Computing geodesics and minimal surfaces via graph cuts. In: *International conference on computer vision*, pp. 26–33.

Ishikawa, H. (2003). Exact optimization for Markov random fields with convex priors. In: *IEEE transactions on pattern analysis and machine intelligence* 25(10), pp. 1333–1336.

Kim, J., Kolmogorov, V., and Zabih, R. (2003). Visual correspondence using energy minimization and mutual information. In: *International conference on computer vision*, pp. 1033–1040.

Kovtun, I. (2003). Partial optimal labeling search for a NP-hard subclass of (max, +) problems. In: *DAGM-symposium*, pp. 402–409.

Kwatra, V. et al. (2003). Graphcut textures: image and video synthesis using graph cuts. In: *ACM transactions on graphics, SIGGRAPH* 22(3), pp. 277–286.

Wainwright, M., Jaakkola, T., and Willsky, A. (2003). Exact MAP estimates by (hyper)tree agreement. In: *Advances in neural information processing systems 15*, pp. 809–816.

Boykov, Y. and Kolmogorov, V. (2004). An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. In: *IEEE transactions on pattern analysis and machine intelligence*. Vol. 26, pp. 1124–1137.

Kolmogorov, V. and Zabih, R. (2004). What energy functions can be minimized via graph cuts? In: *IEEE transactions on pattern analysis and machine intelligence* 26(2), pp. 147–159.

Kolmogorov, V. (2004a). *Convergent tree-reweighted message passing for energy minimization*. Tech. rep. MSR-TR-2004-90. Microsoft Research Cambridge.

Kolmogorov, V. (2004b). Graph based algorithms for scene reconstruction from two or more views. PhD thesis.

Kovtun, I. (2004). Image segmentation based on sufficient conditions of optimality in NP-complete classes of structural labelling problem. In Ukrainian. PhD thesis. IRTC ITS National Academy of Sciences, Ukraine.

Rother, C., Kolmogorov, V., and Blake, A. (2004). GrabCut: interactive foreground extraction using iterated graph cuts. In: vol. 23. 3, pp. 309–314.

Kohli, P. and Torr, P. (2005). Efficiently solving dynamic Markov random fields using graph cuts. In: *International conference on computer vision*. Vol. 2, pp. 922–929.

Kolmogorov, V. N. and Wainwright, M. J. (2005). On optimality of tree-reweighted max-product message-passing. In: *Uncertainty in artificial intelligence*.

Kolmogorov, V. (2005). *Primal-dual algorithm for convex Markov random fields*. Tech. rep. MSR-TR-2005-117. Microsoft Research Cambridge.

Komodakis, N. and Tziritas, G. (2005). A new framework for approximate labeling via graph cuts. In: *International conference on computer vision*, pp. 1018–1025.

Narasimhan, M. and Bilmes, J. (2005). A supermodular-submodular procedure with applications to discriminative structure learning. In: *Uncertainty in artificial intelligence*.

Rother, C. et al. (2005). Digital tapestry. In: *Computer vision and pattern recognition conference*, pp. 589–596.

Wainwright, M., Jaakkola, T., and Willsky, A. (2005). MAP estimation via agreement on (hyper)trees: message-passing and linear-programming approaches. In: *IEEE transactions on information theory* 51(11), pp. 3697–3717.

Werner, T. (2005). *A linear programming approach to max-sum problem: A review*. Tech. rep. CTU–CMP–2005–25. Center for Machine Perception, Czech Technical University, p. 40.

Boros, E., Hammer, P. L., and Tavares, G. (2006). *Preprocessing of unconstrained quadratic binary optimization*. Tech. rep. RRR 10-2006. RUTCOR.

Boykov, Y. and Lempitsky, V. (2006). From photohulls to photoflux optimization. In: *British machine vision conference*. Vol. 3, p. 1149.

Boykov, Y. and Funka-Lea, G. (2006). Graph cuts and efficient N-D image segmentation. In: *International journal of computer vision* 70 (2), pp. 109–131.

Georgiev, I., Lilien, R. H., and Donald, B. R. (2006). Improved pruning algorithms and divide-and-conquer strategies for dead-end elimination, with application to protein design. In: *Bioinformatics* 22 (14), pp. 174–183.

Juan, O. and Boykov, Y. (2006). Active graph cuts. In: *Computer vision and pattern recognition conference*. Vol. 1, pp. 1023–1029.

Kolmogorov, V. (2006). Convergent tree-reweighted message passing for energy minimization. In: *IEEE transactions on pattern analysis and machine intelligence* 28(10), pp. 1568–1583.

Lempitsky, V. et al. (2006). Oriented visibility for multiview reconstruction. In: *European conference on computer vision*, pp. 226–238.

Schlesinger, D. and Flach, B. (2006). *Transforming an arbitrary minsum problem into a binary one*. Tech. rep. TUD-FI06-01. Dresden University of Technology.

Shekhovtsov, A. (2006). Supermodular decomposition of structural labeling problem. In: *Control systems and computers* 1, pp. 39–48.

Avidan, S. and Shamir, A. (2007). Seam carving for content-aware image resizing. In: *ACM Transactions on graphics* 26(3), p. 10.

Johnson, J. K., Malioutov, D. M., and Willsky, A. S. (2007). Lagrangian relaxation for MAP estimation in graphical models. In: *Computing research repository* (abs/0710.0013).

Kolmogorov, V. (2007). *A note on the primal-dual method for the semi-metric labeling problem*. Research Report. Microsoft Research Cambridge.

Kolmogorov, V. and Rother, C. (2007). Minimizing non-submodular functions with graph cuts – a review. In: *IEEE transactions on pattern analysis and machine intelligence*.

Komodakis, N., Paragios, N., and Tziritas, G. (2007). MRF optimization via dual decomposition: message-passing revisited. In: *International conference on computer vision*, pp. 1–8.

Lempitsky, V. and Boykov, Y. (2007). Global optimization for shape fitting. In: *Computer vision and pattern recognition conference*. Vol. 6.

Rother, C. et al. (2007). Optimizing binary MRFs via extended roof duality. In: *Computer vision and pattern recognition conference*.

Schlesinger, D. (2007). Exact solution of permuted submodular minsum problems. In: *International conference on energy minimization methods in computer vision and pattern recognition*. Vol. 4679, pp. 28–38.

Schlesinger, M. I. and Giginyak, V. V. (2007). Solution to structural recognition (max,+)-problems by their equivalent transformations. In: *Control systems and computers* (1,2), pp. 3–15.

Werner, T. (2007). A linear programming approach to max-sum problem: A review. In: *IEEE transactions on pattern analysis and machine intelligence* 29(7), pp. 1165–1179.

Delong, A. and Boykov, Y. (2008). A scalable graph-cut algorithm for N-D grids. In: *Computer vision and pattern recognition conference*, pp. 1–8.

Goldberg, A. V. (2008). The partial augment–relabel algorithm for the maximum flow problem. In: *Proceedings of the 16th annual european symposium on algorithms*, pp. 466–477.

Kohli, P. et al. (2008). On partial optimality in multi-label MRFs. In: *International conference on machine learning*, pp. 480–487.

Komodakis, N. and Paragios, N. (2008). Beyond loose LP-relaxations: optimizing MRFs by repairing cycles. In: *Eccv (3)*, pp. 806–820.

Lempitsky, V. S., Roth, S., and Rother, C. (2008). Fusionflow: discrete-continuous optimization for optical flow estimation. In: *Computer vision and pattern recognition conference*.

Shekhovtsov, A. and Hlaváč, V. (2008). A lower bound by one-against-all decomposition for Potts model energy minimization. In: *Computer vision winter workshop*.

Shekhovtsov, A. et al. (2008). *LP-relaxation of binarized energy minimization*. Tech. rep. CTU–CMP–2007–27. Czech Technical University.

University of Western Ontario web pages (2008). *Computer vision research group. max-flow problem instances in vision*. `http://vision.csd.uwo.ca/maxflow-data/`.

Vineet, V. and Narayanan, P. (2008). CUDA cuts: fast graph cuts on the GPU. In: *Computer vision GPU workshop at ECCV*, pp. 1–8.

Wainwright, M. J. and Jordan, M. I. (2008). Graphical models, exponential families, and variational inference. In: *Foundations and trends in machine learning* 1(1-2), pp. 1–305.

Werner, T. (2008). High-arity interactions, polyhedral relaxations, and cutting plane algorithm for soft constraint optimisation (MAP-MRF). In: *Computer vision and pattern recognition conference*.

Delong, A. and Boykov, Y. (2009). Globally optimal segmentation of multi-region objects. In: *International conference on computer vision*, pp. 285–292.

Kumar, M. P., Kolmogorov, V., and Torr, P. H. S. (2009). An analysis of convex relaxations for MAP estimation of discrete MRFs. In: *Journal of machine learning research* 10, pp. 71–106.

Labatut, P., Pons, J.-P., and Keriven, R. (2009). Robust and efficient surface reconstruction from range data. In: *Computer graphics forum* 28(8), pp. 2275–2290.

Li, S. Z. (2009). *Markov random field modeling in image analysis*. 3rd. Springer Publishing Company, Incorporated.

Liu, J., Sun, J., and Shum, H.-Y. (2009). Paint selection. In: *ACM transactions on graphics* 28(69) (3), pp. 1–7.

Arora, C. et al. (2010). An efficient graph cut algorithm for computer vision problems. In: *European conference on computer vision*, pp. 552–565.

Kolmogorov, V. (2010). Generalized roof duality and bisubmodular functions. In: *Advances in neural information processing systems*, pp. 1144–1152.

Lempitsky, V. et al. (2010). Fusion moves for Markov random field optimization. In: *IEEE transactions on pattern analysis and machine intelligence* 32, pp. 1392–1405.

Liu, J. and Sun, J. (2010). Parallel graph-cuts by adaptive bottom-up merging. In: *Computer vision and pattern recognition conference*, pp. 2181–2188.

Sontag, D. (2010). Approximate inference in graphical models using LP relaxations. PhD thesis. Massachusetts Institute of Technology.

Strandmark, P. and Kahl, F. (2010). Parallel and distributed graph cuts by dual decomposition. In: *Computer vision and pattern recognition conference*, pp. 2085–2092.

Vineet, V. and Narayanan, P. J. (2010). Solving multilabel MRFs using incremental α-expansion on the GPUs. In: *Asian conference on computer vision*. Vol. 3, pp. 633–643.

Fix, A. et al. (2011). A graph cut algorithm for higher-order Markov random fields. In: *International conference on computer vision*, pp. 1020–1027.

Goldberg, A. V. et al. (2011). Maximum flows by incremental breadth-first search. In: *European conference on algorithms*, pp. 457–468.

Ishikawa, H. (2011). Transformation of general binary MRF minimization to the first-order case. In: *IEEE transactions on pattern analysis and machine intelligence* 33(6), pp. 1234–1249.

Jancosek, M. and Pajdla, T. (2011). Robust, accurate and weakly-supported-surfaces preserving multi-vew reconstruction. In: *Computer vision and pattern recognition conference*.

Kahl, F. and Strandmark, P. (2011). Generalized roof duality for pseudo-Boolean optimization. In: *International conference on computer vision*, pp. 255–262.

Kovtun, I. (2011). Sufficient condition for partial optimality for (max, +) labeling problems and its usage. In: *Control systems and computers* (2). special issue.

Kumar, M. P., Veksler, O., and Torr, P. H. (2011). Improved moves for truncated convex models. In: *Journal of machine learning research* 12, pp. 31–67.

Lempitsky, V., Vedaldi, A., and Zisserman, A. (2011). A pylon model for semantic segmentation. In: *Advances in neural information processing systems*.

Osokin, A., Vetrov, D., and Kolmogorov, V. (2011). Submodular decomposition framework for inference in associative Markov networks with global constraints. In: *Computing research repository* (abs/1103.1077).

Savchynskyy, B. et al. (2011). A study of Nesterov's scheme for Lagrangian decomposition and MAP labeling. In: *Computer vision and pattern recognition conference*, pp. 1817–1823.

Schlesinger, M., Vodolazskiy, E., and Lopatka, N. (2011). Stop condition for subgradient minimization in dual relaxed (max,+) problem. In: *International conference on energy minimization methods in computer vision and pattern recognition*, pp. 118–131.

Shekhovtsov, A. and Hlaváč, V. (2011). On partial opimality by auxiliary submodular problems. In: *Control systems and computers* (2). special issue.

Shekhovtsov, A. and Hlavac, V. (2011). A distributed mincut/maxflow algorithm combining path augmentation and push-relabel. In: *International conference on energy minimization methods in computer vision and pattern recognition*, p. 14.

Delong, A. et al. (2012). Fast approximate energy minimization with label costs. In: *International journal of computer vision* 96(1), pp. 1–27.

Jamriška, O., Sýkora, D., and Hornung, A. (2012). Cache-efficient graph cuts on structured grids. In: *Computer vision and pattern recognition conference*, pp. 3673–3680.

Kahl, F. and Strandmark, P. (2012). Generalized roof duality. In: *Discrete applied mathematics* 160(16-17), pp. 2419–2434.

Kappes, J. H., Savchynskyy, B., and Schnörr, C. (2012). A bundle approach to efficient MAP-inference by Lagrangian relaxation. In: *Computer vision and pattern recognition conference*, pp. 1688–1695.

Kolmogorov, V. (2012a). Generalized roof duality and bisubmodular functions. In: *Discrete applied mathematics* 160(4-5), pp. 416–426.

Kolmogorov, V. (2012b). The power of linear programming for valued csps: a constructive characterization. In: *Computing research repository* (abs/1207.7213).

Shekhovtsov, A. and Hlaváč, V. (2012). A distributed mincut/maxflow algorithm combining path augmentation and push-relabel. In: *International journal of computer vision*, p. 28. Authorship: 90-10.

Thapper, J. and Zivny, S. (2012). The power of linear programming for valued CSPs. In: *Computing research repository* (abs/1204.1079).

Veksler, O. (2012). Multi-label moves for MRFs with truncated convex priors. In: *International journal of computer vision* 98(1), pp. 1–14.

Verma, T. and Batra, D. (2012). Maxflow revisited: an empirical comparison of maxflow algorithms for dense vision problems. In: *British machine vision conference*, p. 12.

Vineet, V., Warrell, J., and Torr, P. H. S. (2012). A tiered move-making algorithm for general pairwise MRFs. In: *Computer vision and pattern recognition conference*, pp. 1632–1639.