DISS. ETH N° 20310

COMBINATORIAL PROBLEMS WITH SUBMODULAR COUPLING
IN MACHINE LEARNING AND COMPUTER VISION

A dissertation submitted to

ETH ZURICH

for the degree of

Doctor of Sciences

presented by

STEFANIE SABRINA JEGELKA

Dipl.-Inf. (Eberhard Karls Universität Tübingen)

born 27th November 1981

citizen of Germany

accepted on the recommendation of

Prof. Dr. R. Andreas Krause, examiner
Prof. Dr. Jeffrey A. Bilmes, co-examiner
Prof. Dr. Bernhard Schölkopf, co-examiner

2012

To my parents.

Abstract

Numerous problems in machine learning and computer vision are discrete. As a complicating factor, they often involve large data sets and higher-order interactions between elements in the data. For example, segmenting an image into foreground and background requires assigning a label to each pixel in the image. As object and background commonly have significant wide-range coherency, the most probable label of a given pixel is not independent of the labels of other pixels. In general, if such interactions are important to a problem, it may be inappropriate to reduce it to efficiently solvable combinatorial problems like cuts in a neighborhood graph, because tractability frequently results from ignoring global interactions.

This thesis addresses a class of combinatorial problems that admit high-order interactions between elements. The interactions take the form of a submodular set function over edges in a graph. In particular, the thesis introduces *cooperative cuts* whose cost function is not a sum of edge weights, but a submodular function on edges. We show that cooperative cuts generalize and enhance graph-based models and applications in machine learning and computer vision.

The first part of the thesis studies theoretical and algorithmic questions, including upper and lower bounds on the approximation factor of the *minimum cooperative cut* problem. In addition to theoretical bounds, we empirically test the approximation algorithms on average and worst-case instances.

The second part investigates the impact of coupling edges in graph cuts. Graph cuts are frequently used for representing functions and thereby offer a tool for minimizing those functions efficiently. Cooperative cuts widen the range of representable functions, and we employ them to define energy functions. An energy function characterizes a probabilistic model and determines the complexity of inference in this model. Although cooperative cut energies possess none of the commonly used properties that imply tractability, the algorithms from Part I solve the inference problem within a bounded approximation factor.

Next, we explore an application. Cooperative cut energies encompass several recent models in the computer vision literature, and they establish the foundation for new models. In particular, the thesis introduces a new criterion for image segmentation that considers the homogeneity or *congruity* of the object boundary. This criterion remedies shortcomings of the popular graph cut method, notably, it preserves fine structures of the object even when the contrast is low.

The next result is motivated by a corpus subset selection problem. Even though this problem corresponds to minimizing a submodular function and is solvable in polynomial time, the complexity of state-of-the-art exact algorithms is too high for large data sets. The observation that cooperative cuts can represent any submodular function is the key to a faster algorithm for minimizing submodular functions approximately.

The third part of the thesis widens the scope and studies combinatorial problems with submodular cost functions in an online framework. Sequential decision problems ask to solve a problem repeatedly while an unknown cost function changes over time. The thesis proposes two generic Hannan-consistent algorithms building on the approximation methods discussed in Part I, and an algorithm for the sub-class of “label cost” functions. The results generalize commonly studied linear loss functions and apply to a variety of problems.

Zusammenfassung

Zahlreiche Optimierungsprobleme im Bereich des Maschinellen Lernens und Sehens sind diskret. Die Probleme werden dadurch erschwert, dass sie häufig Interaktionen von Datenelementen beinhalten und trotzdem auf großen Datensätzen gelöst werden müssen. Als Beispiel kann die Segmentierung eines Bildes in ein Vordergrundobjekt und Hintergrund herangezogen werden, wobei jedem Pixel ein entsprechendes Label zugewiesen wird. Da sowohl Objekt als auch Hintergrund meist großflächige Kohärenzen aufweisen, ist das Label eines gegebenen Pixels nicht von den Labeln anderer Pixel unabhängig. Mehrere Pixel müssen gemeinsam betrachtet werden. Wenn solche Interaktionen wichtig sind, dann kann es unangemessen sein, das Problem auf einfach lösbare kombinatorische Probleme zu reduzieren, deren Lösbarkeit auf dem Fehlen globaler Kopplungen von Variablen beruht.

Die vorliegende Arbeit beschäftigt sich mit kombinatorischen Problemen die weitrangige Kopplungen zulassen. Diese Kopplungen werden durch eine submodulare Mengenfunktion über Kanten in einem Graphen ausgedrückt. Insbesondere definiert die Arbeit *kooperative Schnitte*, deren Zielfunktion nicht die Summe der Kantengewichte, sondern eine submodulare Funktion über Kanten ist. In der Arbeit wird gezeigt, wie kooperative Schnitte auf Graphen basierende Modelle und Anwendungen im Maschinellen Lernen und Sehen erweitern.

Der erste Teil der Arbeit befasst sich mit theoretischen und algorithmischen Fragen wie unteren und oberen Schranken für Approximationsfaktoren. Neben der Herleitung theoretischer Schranken wird das Verhalten der beschriebenen Algorithmen empirisch untersucht.

Das Thema des zweiten Teils der Arbeit sind mathematische Modelle und Anwendungen, die von kooperativen Schnitte profitieren. Schnitte in Graphen werden unter anderem benutzt um Funktionen zu repräsentieren und auf diese Weise effizient zu minimieren. Kooperative Schnitte erweitern die Klasse der darstellbaren Funktionen. In der Arbeit werden sie angewandt um Energiefunktionen zu definieren. Eine solche Energiefunktion charakterisiert ein probabilistisches Modell und bestimmt, wie schwer das Inferenzproblem in diesem Modell ist. Die Klasse der durch kooperative Schnitte definierten Funktionen erfüllt keine der gängigen Kriterien, die Inferenz in Polynomzeit ermöglichen. Dennoch lösen die Algorithmen aus dem ersten Teil das Inferenzproblem approximativ.

Darüber hinaus werden Anwendungen beschrieben. Energiefunktionen aus kooperativen Schnitten beinhalten einige im Bereich des Maschinellen Sehens bekannte Modelle und bilden gleichzeitig die Grundlage für neue Modelle. Als konkretes Beispiel wird in der Arbeit ein neues Kriterium zur Segmentierung von Bildern eingeführt, welches die Einheitlichkeit des Objektrandes einbezieht. Dieses Kriterium verringert die Probleme der häufig verwendeten *Graph Cut*-Methode. Insbesondere feine Strukturen lassen sich nun viel besser segmentieren.

Ein weiteres Ergebnis ist durch ein Problem motiviert, eine Untermenge von Elementen eines Korpus auszuwählen. Dieses Problem ist ein submodulares Minimierungsproblem und in Polynomzeit lösbar, allerdings sind bekannte Algorithmen in der Praxis nicht auf großen Datensätzen anwendbar. Ein neuer, praktikablerer Approximationsalgorithmus entsteht aus der Beobachtung heraus, dass kooperative Schnitte jede submodulare Funktion darstellen können.

Der dritte Teil der Arbeit erweitert den Rahmen der behandelten Probleme und untersucht allgemeinere kombinatorische Probleme mit submodularen Kostenfunktionen, die als sequentielle Entscheidungsprobleme betrachtet werden. Bei sequentiellen Aufgaben muss ein gegebenes Problem wiederholt gelöst werden, wobei sich die unbekannte Kostenfunktion ändert. Die Arbeit beschreibt zwei generische, *Hannan-konsistente* Algorithmen, die auf den in Teil I eingeführten Techniken aufbauen. Desweiteren wird ein Algorithmus für die Unterklasse der *Label cost*-Funktionen entwickelt. Die Ergebnisse dieses Teils erweitern den Bereich bekannter Ergebnisse von linearen auf nicht-lineare (submodulare) Kostenfunktionen und umfassen eine Spanne von Problemstellungen.

Acknowledgments

This dissertation would have been very different without several people to whom I am deeply indebted. First, it was Jeff Bilmes who introduced me to submodularity. Jeff, thank you very much for your amazingly infinite patience, for your extraordinary support and for an awesome cooperation on cooperative cuts. Thanks for nights of discussions on Skype, for listening to all my ideas and counter-examples, for having me visit Seattle and for submodularity in every semi-free minute. And thanks, Katrin, Alexander and Sebastian for bearing this.
Then, all this work would not have been possible without the generosity of Bernhard Schölkopf, who let me work on wherever my interests led me. Thank you, Bernhard, for your support and the time at a great place for research. My gratitute also belongs to Andreas Krause, for great discussions on submodular problems and ideas, for his support to make this dissertation happen, and for very valuable advice.
Furthermore, I am very grateful to my first mentors at MPI, Arthur Gretton and Ulrike von Luxburg, for their support and for everything I could learn from them. Moreover, I would like to thank Hui Lin for his great efforts on our joint project. Even though early projects during my Ph.D. time are not included in this thesis, I want to thank my collaborators on these projects, Ule, Arthur, Bernhard, Bharath Sriperumbudur, Sebastian Nowozin, Karsten Borgwardt, Arindam Banerjee and Suvrit Sra. In particular, thanks to Sebastian for answering many questions about computer vision and more, and to Lumin Zhang and Zhikun Wang for help with OpenCV. Many thanks to Jeff, Andreas and Pradeep Ravikumar for the workshop co-organizations, and to Yoshinobu Kawahara for our reading group and discussions. The critical reading of Christoph Lampert, Lawrence Cayton, Andrew Guillory, Ajit Singh and Manuel Gomez Rodriguez improved papers and parts of this thesis, and, unknowingly, Yevgeny Seldin inspired a proof. Thanks also to all fellow students and researchers at MPI who made this time sociable. My thanks also go to Raymond Hemmecke and Ramin Zabih for their invitations, and to Erik Horvitz and Ashish Kapoor for hosting me as an intern. Finally, I would like to thank Richard Karp for the name "cooperative cut".
"Danke" to Joachim Buhmann, Rita Klute and Denise Spicher for guiding a novice through administrative jungles, and "danke" also to Sabrina Rehbaum.
Finally, I am very grateful to my family for supporting this long journey wherever they could. The deepest thanks go to my husband Suvrit, who answered so many questions and who was always there to support and encourage me. बहुत धन्यवाद, सब कुछ के लिये!

Contents

List of Figures

List of Tables

Chapter 1.

Introduction

Discrete and combinatorial problems pervade not only classical fields, they are also at the heart of many questions in computer vision and machine learning. To complicate matters, problems in machine learning often arise from applications that involve uncertainty and complex interactions between elements. Even though interactions can make these problems very complex, large data sets demand algorithms whose running time scales moderately in the size of the input.

This thesis addresses combinatorial problems that incorporate interactions between elements. The interactions are expressed by non-additive, submodular cost functions. Motivated by the demands of applications, it studies four questions: *(1) How can interactions be expressed in discrete problems so that they still admit to compute "good" solutions?*; *(2) What are appropriate algorithms, and how can they exploit properties of the problem?*; *(3) How does the proposed framework realize models in machine learning and computer vision?*; *(4) How does the framework affect results in applications?*

These questions are very wide, and we will make them concrete by specifying a class of problems that we name *cooperative cut*. Before formally defining cooperative cuts, we review some context and key questions.

Discrete problems in Machine Learning

In a broad sense, the goal of machine learning is to infer rules and make predictions based on observed data. The problem is formalized as a mathematical model, and fitting the model or making predictions (inference) are phrased as optimization problems. If the data are discrete, then so are model and associated optimization problems. Moreover, the mathematical model determines the complexity of the optimization problem.

Discrete problems materialize, for instance, in the form of clustering problems, graph partitioning, structured prediction, feature selection, active learning, or as learning and inference problems in discrete probabilistic models. Such problems are relevant to applications in biology, natural language processing, speech recognition or computer vision. Besides learning and inference, combinatorial problems can arise as sub-problems of designing optimization procedures. For example, distributed processing requires the partitioning of large models into appropriate chunks.

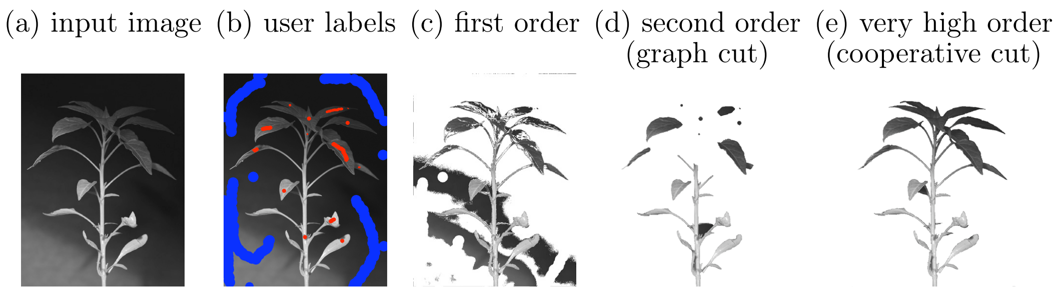

(a) input image
(b) user labels
(c) first order
(d) second order (graph cut)
(e) very high order (cooperative cut)

Figure 1.1. Often, an approximate solution to a complex high-order model (e) is preferable to the exact solution to a simple, low-order model (c),(d). This example from interactive image segmentation asks to infer "object" and "background" pixel labels based on the user input, the blue and red brush strokes. Solution (e) is an approximation, solutions (c),(d), are exact.

As a concrete example of a discrete problem, imagine we would like to design an algorithm for segmenting the plant in Figure 1.1(a) from its background. We phrase this problem mathematically and assign a variable to each pixel, which will take the label "object" or "background". To infer the assignments from given labels, we define a model that determines the probability $p(\boldsymbol{x}|\bar{\boldsymbol{z}})$ of an assignment $\boldsymbol{x}$ given the statistics of pixels labeled by a user and the observed image $\bar{\boldsymbol{z}}$. By this model, the assignment of choice is the one maximizing $p(\boldsymbol{x}|\bar{\boldsymbol{z}})$. Finding this maximizer is the *inference* problem.

Efficiency, complexity and structure

When solving the inference problem, there is an exponential number of assignments to choose from. In general, such discrete problems can be notoriously hard, and solving them requires time exponential in the size of the input. Even worse, it may even take exponential time to provide a "provably good" solution. Often however, problems in machine learning involve large data sets and thereby rule out computationally expensive methods.

To avoid having to search through exponentially many assignments, the function p must have certain properties. For example, if all pixels are judged independently and p is a product of single-pixel terms $p(x_i|\bar{\boldsymbol{z}})$, then we can optimize each term separately. Commonly, one additionally considers pairs of pixels, so that it is less likely that the labels of neighboring pixels disagree. It turns out that then the inference problem is equivalent to finding a minimum cut in a grid graph. In the graph, each pixel is a node, and the cut is the object boundary. The Minimum Cut problem is a combinatorial problem.

Formally, a *combinatorial problem* consists of a ground set $\mathcal{E}$ of elements, a family $\mathcal{S}$ of feasible solutions $S \subseteq \mathcal{E}$ which are subsets of $\mathcal{E}$, and a cost function f defined on sets $S \subseteq \mathcal{E}$. We aim to find a feasible solution $S \in \mathcal{S}$ with minimum cost $f(S)$. The elements could be edges in a graph and the family $\mathcal{S}$ might consist, for example, of all cuts in the graph. Both $\mathcal{S}$ and the cost f can induce relations between elements. Combinatorial problems can also be phrased in terms of variables, and instead of picking subsets one assigns discrete labels.

Relating the inference problem to graphs reveals *structure* that helps solve the problem combinatorially. Several combinatorial problems admit efficient algorithms, and Minimum Cut is one of them. In general, it may be an option to phrase the problem at hand in terms of a model that fits a "simple" combinatorial problem.

The image segmentation model corresponding to cuts profits computationally because it has much *separability*: the logarithm of the posterior probability $p(\boldsymbol{x}|\bar{\boldsymbol{z}})$, which is usually considered, is a polynomial of degree two and only involves pairs of neighboring pixels [Picard and Ratliff, 1975, Greig et al., 1989, Boykov and Jolly, 2001]. While this model has been successfully applied, Figure 1.1 illustrates its limits: models of order one or two fail to identify the true boundary of the plant. They cannot capture important *higher-order* coherency or, in other words, take into account dependencies between many pixels jointly.

Separability can be limiting not only to image segmentation. Section 2.5 lists a selection of combinatorial problems that consider elements not separately but jointly. The cost functions that express such coupling are not separable like a sum of element-wise weights, they are nonlinear.

While complex interactions between variables may suit the data better, they usually lead to nonlinear cost functions and almost always to NP-hard optimization problems. For such hard problems, we must revert to approximations. Figure 1.1, drawn from Chapter 6, illustrates that an approximate solution to a more complex model may still be more appropriate than an exact solution to an oversimplified model that disregards essential structural properties of the application. In fact, the problem need not even be NP-hard. For large data, an algorithm that runs in high-order polynomial time can be practically infeasible. Chapter 7 explores approximation algorithms for such a problem.

An *approximation algorithm* returns a solution S whose cost is within a bounded factor α of the cost of the optimal solution S^*: for a minimization problem with cost function f, this means that $f(S^*) \leq f(S) \leq \alpha f(S^*)$ with $\alpha \geq 1$. The factor α may be a function of the size of the input data. A key question is then:

> I. What are efficient algorithms for which the approximation factor is as small as possible?

If one allows arbitrary constraints and cost functions, nothing can be said about bounds on the approximation factor. Approximations, like efficient algorithms, rely on the problem having *structure* that can be exploited. Combinatorial objects may therefore be relevant to approximations too. Certainly, the graph cuts used for image segmentation provide good structure for optimization, but they are too restrictive. Important questions are then:

II. Which models have suitable structure that optimally trades offefficiency and sufficient expressive richness?
III. How can one make use of such structure?

Implicitly, Question II also asks whether there are frameworks beyond frequently used combinatorial problems that express more complex yet approximately manageable mathematical models with richer interactions. In particular, this thesis focuses on a model that combines two known structures in a new way: submodularity and graphs.

After investigating the above questions theoretically and in applications, the thesis explores how the discussed approximations can be used in online problems where there is uncertainty about the cost function.

Cooperative Cuts

We address the above questions by introducing interactions between elements in combinatorial problems, namely, by allowing interactions between edges in a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ or, equivalently, between pairs of variables. To express these interactions, we define a submodular set function over sets of edges. A set function $f : 2^{\mathcal{E}} \to \mathbb{R}$ defined on subsets $A \subseteq \mathcal{E}$ of a ground set $\mathcal{E}$ is submodular if it satisfies *diminishing marginal costs*[1]: for any sets $A \subseteq B \subset \mathcal{E}$ and $e \in \mathcal{E} \setminus B$, it holds that

$$f(A \cup \{e\}) - f(A) \geq f(B \cup \{e\}) - f(B). \tag{1.1}$$

A set function f is *nondecreasing* if $A \subseteq B \subseteq \mathcal{E}$ implies that $f(A) \leq f(B)$. Submodular functions have long been an important concept in combinatorial optimization [Frank, 1989, 1993, Vondrák, 2007, Fujishige, 2003], operations research and engineering [Iri and Fujishige, 1981, Narayanan, 1997], and game theory [Shapley, 1971], and they have recently gained attention in machine learning as well [Kempe et al., 2003, Narasimhan et al., 2006, Narasimhan and Bilmes, 2005, Krause et al., 2008, Bach, 2010, Das and Kempe, 2011, Golovin and Krause, 2011].

The largest part of this thesis focuses on a minimum cut problem with submodular instead of additive cost functions. We refer to the resulting problem as *minimum cooperative cut*

Definition 1.1 (Minimum cooperative cut (MinCoopCut))**.** Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with terminal nodes $s, t \in \mathcal{V}$ and a nondecreasing submodular function $f : 2^{\mathcal{E}} \to \mathbb{R}_+$, find an (s,t)-cut $C \subseteq \mathcal{E}$ with minimum cost $f(C)$. An (s,t)-cut $C \subseteq \mathcal{E}$ is a set of edges whose removal disconnects s and t.

The "cooperation" in cooperative cuts emerges from the diminishing cost property of the cost function. Certain edges occurring together in a cut can have a much lower cost than the sum of their individual costs.

[1]The common term is *diminishing marginal returns*, but "costs" suits the context here better.

With respect to this cooperation property, submodular functions can be viewed as being at the boundary of tractability[2]. Diminishing marginal costs imply that for any sets $A, B \subseteq \mathcal{E}$ it holds that $f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$. Whereas a submodular function can be minimized over the power set of $\mathcal{E}$ in polynomial time, a function that only satisfies $f(A) + f(B) \geq f(A \cup B)$ may in polynomial time not even admit a solution with a bounded approximation factor.

The combination of submodular functions and graphs extends the realm of either concept. As opposed to standard additive cost functions, a submodular function allows the coupling of edges in a graph and thereby the modeling of complex interactions. Graph cuts have been used for modeling and optimization in a wide range of areas. As cooperative cuts employ the same graph structures as standard graph cuts, they easily integrate into and extend many of the ubiquitous applications of graph cuts. Chapters 6 and 7 explore two such examples. Graph cuts also provide the link to probabilistic models, because, as indicated above, they represent a class of probabilistic models. Cooperative cuts extend this class to include models with variable interactions of arbitrarily high order.

Even though two chapters of this thesis are devoted to modeling with energy functions, cooperative cuts have more applications. For example, the algorithms in this thesis also approximately solve the problem of finding minimum-entropy separators, a problem arising when performing inference in dynamic graphical models. This problem is outlined in Section 2.5.

More generally, combinatorial problems where the sum-of-(edge-)weights has been replaced by a submodular cost function have recently gained attention in theoretical computer science [Iwata and Nagano, 2009, Goel et al., 2009]. We will refer to these problems as *submodular-cost combinatorial problems*.

Some results in the sequel affect submodular-cost combinatorial problems beyond cooperative cuts. The algorithm in Section 4.2.3 applies to any combinatorial problem whose standard, sum-of-weights version is polynomial-time solvable. Chapter 8 extends the scope to several submodular-cost combinatorial problems in an online setting. Indeed, submodular-cost problems arise in a variety of applications, as illsutrated by the selection listed in Section 2.5.

Organization of the thesis

This thesis is structured into three parts and a chapter on background information. Chapter 2 summarizes basic results and concepts and states examples of submodular-cost combinatorial problems. More specific related work is discussed in the respective chapters.

The first two technical parts address cooperative cuts, and Part III (Chapter 8) is dedicated to online algorithms for general submodular-cost combinatorial problems. Part I (Chapter 3–4) discusses theoretical questions such as complexity and

[2]Strictly speaking, the class of XOS or fractionally subadditive functions lies in between submodular and subadditive functions [Lehmann et al., 2006], but they are not our focus here.

approximating f		simplifying the constraints	
additive approx.	$\frac{\lvert C^*\rvert}{1+(\lvert C^*\rvert-1)\nu(C^*)} = O(m)$	convex relaxation	$n-1$
ellipsoid	$O(\sqrt{m}\log m)$		
partition & flow	$n/2$	randomized greedy	$n-1$
iterative	$\frac{\lvert C^*\rvert}{1+(\lvert C^*\rvert-1)\nu(C^*)} = O(m)$		

Table 1.1. Approximation algorithms for MinCoopCut from Chapter 4 and upper bounds on their worst-case approximation factors for $n = |\mathcal{V}|$ and $m = |\mathcal{E}|$.

approximation methods. Part II explores the usefulness of edge interactions, i.e., applications that use the algorithms from Part I. This entails the modeling perspective, i.e., a new family of high-order energy functions (Chapter 5), and practical applications, i.e., a specific model for image segmentation (Chapter 6) as well as an algorithm for efficient approximate submodular minimization (Chapter 7).

1.1. Summary of the thesis

For better orientation, we outline the problems discussed in each chapter and the respective results.

1.1.1. Part I: Algorithms and complexity of MinCoopCut

In short, Chapters 3 and 4 address the following topics:

- approximation algorithms for MinCoopCut,
- hardness results for MinCoopCut.

Chapter 3 shows that MinCoopCut is NP-hard. This hardness can be attributed to two modifications compared to polynomial-time solvable problems: the cost function or the constraints. First, the standard Minimum (s,t)-cut problem whose cost function is a *sum of weights* is not NP-hard. Second, submodular function minimization without any *constraints* is not NP-hard either. The approximation methods in Chapter 4 take one of these two viewpoints, and we categorize them by which of the two modifications they relax. Table 1.1 summarizes the algorithms and the bounds on their worst-case approximation factors. We also empirically evaluate the algorithms on average-case and worst-case data.

The first group of algorithms replaces the "difficult" submodular cost function f by an approximation $\hat{f}$ and solves a minimum cut with respect to the cost $\hat{f}$. We examine four variants. The simplest is a straightforward additive approximation. The second builds on a generic approximation $\hat{f}$ by Goemans et al. [2009]. The third exploits the graph structure for a locally exact approximation. It is minimized via its dual problem, a generalized maximum flow. The fourth algorithm iteratively defines and minimizes upper bounds on f. Even though its worst-case approximation factor is the same as that of the first variant, on most other data its approximations are much better and competitive to those of the other algorithms. This algorithm scales to large data sets and is thus efficient enough to be the basis for the applications in Chapters 6 and 7.

The second group of algorithms simplifies the constraints, and both derived algorithms in this group build on a relation between cuts and covers. This relation leads to a rounding technique for solutions to the convex relaxation of MinCoopCut, and to an efficient randomized greedy algorithm.

The hardness results in Chapter 3 help place the approximation bounds in context. We prove an information-theoretic lower bound of $\Omega(\sqrt{|\mathcal{V}|}/\log|\mathcal{V}|)$ ($\Omega(\sqrt{|\mathcal{E}|}/\log|\mathcal{E}|)$ for sparse graphs). Before showing this bound, we prove that MinCoopCut is NP-hard. The reduction in the proof is entirely new, and, as it uses submodularity in a non-standard way, might be of interest on its own.

1.1.2. Part II: Applications of CoopCut

Chapters 5, 6 and 7 address applications of MinCoopCut. The applications are motivated by typical applications of graph cuts for representing set functions, regularization and for image segmentation:

- a new class of high-order energy functions defined by cooperative cuts;
- a new model for image segmentation that significantly improves the results;
- a non-uniform smoothing criterion for continuous problems;
- an efficient algorithm for approximate submodular function minimization by representing submodular functions as cooperative cuts.

A new class of non-submodular global energy functions

Chapter 5 draws connections between cooperative cuts and energy functions of probabilistic models. An *energy function* $E(\boldsymbol{x})$ defines a distribution $p(\boldsymbol{x}) \propto \exp(-E(\boldsymbol{x}))$ for variables $\boldsymbol{x}$ in a discrete domain $\mathcal{D}^n$. The energy determines the complexity of the *most probable explanation* problem (MPE) of finding the assignment $\operatorname{argmax}_{\boldsymbol{x}} p(\boldsymbol{x})$ that maximizes the probability $p(\boldsymbol{x})$.

In certain cases, the energy can be equated with cuts in an appropriate graph. We extend this equivalence to cooperative cuts. In the simplest case, the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ corresponding to energy $E : \{0,1\}^n \to \mathbb{R}^+$ has one node v_i for each random variable x_i, and two additional terminal nodes s, t. Let $X(\boldsymbol{x}) = \{v_i \mid x_i = 1\} \cup \{s\}$ be the set of nodes whose variables are assigned label one, and let $\delta(X) = \{(v_i, v_j) \in \mathcal{E} \mid v_i \in X \cup \{s\}, v_j \notin X\}$ denote the cut around the set X. Then the equivalence between energy and cuts can be phrased as

$$E(\boldsymbol{x}) = w(\delta(X(\boldsymbol{x}))) = \sum_{e \in \delta(X(\boldsymbol{x}))} w(e). \tag{1.2}$$

model	original reference	$f(C)$
Graph Cut	Boykov and Jolly [2001]	$w(C)$
(binary) P^n fct.	Kohli et al. [2007]	$g(\|C\|)$
P^n Potts	Kohli et al. [2007]	$\max_{e \in C} w(e)$
robust P^n	Kohli et al. [2009b]	$\sum_j \min\{\|C \cap S_j\|,\ q\}\gamma/q$
class labels	Delong et al. [2011]	$g_L(\bigcup_{e \in C} \ell(e))$
random walker (discretized)	Grady [2006]	$\sqrt{w^2(C)}$
ℓ_∞	Sinop and Grady [2007]	$\max_{e \in C \cap \mathcal{E}_n} w(e)$
watershed cuts	Cousty et al. [2009], Allène et al. [2009]	$\max_{e \in C} w(e)$
total variations (discretized)	Rudin et al. [1992], Chambolle and Darbon [2009], Couprie et al. [2011]	$\sum_j g_j(C \cap S_j)$
congruent boundaries	Chapter 6	$\sum_j g_\theta(w(C \cap S_j))$
general submodular	Chapter 7	polymatroid part of function

Table 1.2. Examples of cooperative cuts explained in Sections 5.3 and 6.2.

Based on this equivalence, we define the family of *cooperative cut energies* as those energy functions E_f for which there exists a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a nondecreasing submodular function $f : 2^{\mathcal{E}} \to \mathbb{R}_+$ such that

$$E_f(\boldsymbol{x}) = f(\delta(X(\boldsymbol{x}))) = f(\{e \mid e \in \delta(X(\boldsymbol{x}))\}). \tag{1.3}$$

Cooperative cut energies satisfy none of the simplifying properties that are commonly used for tractable inference. Nevertheless, the MPE problem becomes a minimum cooperative cut problem, and therefore the algorithms from Chapter 4 apply for inference with bounded approximation factors.

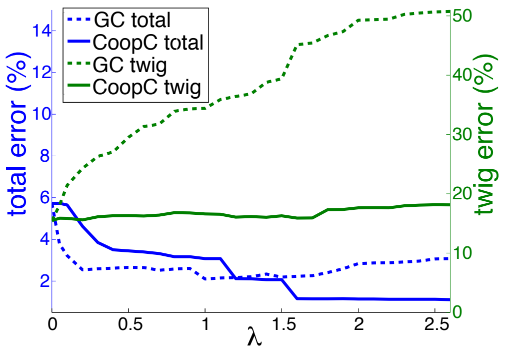
The graph cut analogy holds for binary random variables. We additionally show algorithms that minimize cooperative cut energies for multi-label models within an approximation bound.

To put this new family of functions in context, we demonstrate that a number of recently defined higher-order energies from computer vision are special cases of cooperative cut energies (Table 1.2).

Applications in image segmentation and smoothing

Chapter 6 develops the idea of smoothness criteria via cooperative cuts. It shows a practical application of cooperative cut energies and a new criterion for image

Figure 1.2 Regularization curve for standard Graph Cut (GC) and cooperative cut. As the regularization coefficient λ increases, the average total error shrinks for both methods. For GC, the average twig error increases, indicating that fine object parts are cut off. Cooperative cut leaves the segmentation of fine structures largely unaffected and minimizes both total and twig errors simultaneously.



segmentation. Figure 1.1 illustrates that second-order models with energies of the form (1.2) can result in segmentations that fail to identify delicate parts of the object, in particular if the contrast is low. The segmentation in Figure 1.1(e) uses the congruity of the object boundary to transfer information from high-contrast areas to low-contrast areas of the image. This congruity criterion is a cooperative cut energy.

The iterative algorithm from Chapter 4 efficiently provides approximate minimizing solutions. The efficiency relies on two properties: the cooperative energies use the same grid graph structure as standard graph cut models, and, despite interactions of arbitrarily high order, no auxiliary variables are needed.

A detailed empirical evaluation demonstrates that the global uniformity criterion qualitatively and quantitatively improves segmentation results. On difficult grayscale images with low contrast regions, the segmentation error is reduced by up to 70%. The regularization curve in Figure 1.2 shows that the congruity criterion reduces the global segmentation error while preserving fine structures, and suggests that here, the cooperative cut is a more appropriate smoothness criterion than Graph Cut. These benefits rely on diminishing costs and in particular on the group structure of edge interactions.

The pairwise terms in standard graph cuts can be viewed as a regularizing bias. We extend this regularization viewpoint and, building on cooperative cut energies, define a generic *edge-norm* criterion for structured regularization of differences of discrete or continuous variables. Convex losses with an edge-norm can be minimized via a proximal splitting method. In addition, we show that edge-norms unify a variety of formulations in the literature.

Approximate submodular minimization

Minimizing a submodular function is not an NP-hard problem [Grötschel et al., 1981]. However, if “efficiency” is more pragmatically defined as having low-order polynomial running time, then to date general submodular minimization is not an efficiently solvable problem either.

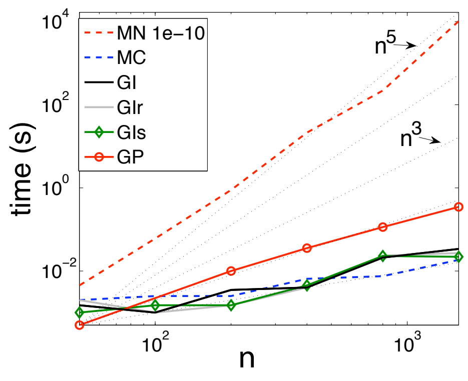


Figure 1.3 Average running times for the minimum norm point algorithm (MN) and the algorithms described in Chapter 7 on a speech corpus subset selection problem (log-log plot). The slopes of the black dotted lines indicate complexities n, n^2, n^3, n^4 and n^5.

Certain submodular functions can be represented as graph cuts, and as such be minimized much more efficiently. But graph cut representations using formulation (1.2) do not comprise all submodular functions [Živný et al., 2009]. Nevertheless, *any* submodular function can be represented by a cooperative cut. Based on this representation, we treat submodular function minimization as an NP-hard problem and solve it *approximately* with an extension of the iterative algorithm from Chapter 4. The resulting method approximates a submodular function by a sequence of graph-representable submodular functions. Its empirical running time is by up to two orders of magnitude faster than that of the commonly used minimum norm point algorithm (see Figure 1.3), while it retains a small empirical approximation factor. On the theoretical side, we prove that any element selected by the algorithm is a member of the maximal optimal solution.

1.1.3. Part III: Sequential decision problems beyond linear costs

Chapter 8 derives algorithms for general submodular-cost combinatorial problems when encountered as sequential decision problems. It shows

- two generic algorithms for online submodular-cost combinatorial problems;
- a special algorithm for the class of *label cost* functions.

Sequential decision problems ask to solve the same problem repeatedly in each of T time steps (e.g., playing moves in a game or daily selecting the route to work). In each step t, we choose a solution $S_t \in \mathcal{S}$ from a decision space $\mathcal{S}$. Only afterwards do we observe the loss function $f_t : \mathcal{S} \to \mathbb{R}_+$ and incur the loss $f_t(S)$. The overall goal is to be competitive with the best fixed solution in hindsight, that is, to minimize the (external) *regret*

$$R_T = \frac{1}{T}\Big(\sum_{t=1}^{T} f_t(S_t) - \min_{S\in\mathcal{S}} \sum_{t=1}^{T} f_t(S)\Big). \tag{1.4}$$

	I.a subgradient descent (Sec. 8.2)	I.b Follow-the-pert.-leader (Sec. 8.3)	II. label costs (Sec. 8.4)
Set Cover	$O(\underline{k}\sqrt{m/T})$	–	$O(\underline{\ln \lvert\mathcal{U}\rvert}\sqrt{\lvert\mathcal{L}\rvert/T})$
Vertex Cover	$O(\underline{2}\sqrt{m/T})$	–	$O(\underline{\ln \lvert\mathcal{E}\rvert}\sqrt{\lvert\mathcal{L}\rvert/T})$
(s,t)-cut	$O(\underline{n}\sqrt{m/T})$	$O(\underline{n}m/\sqrt{T})$	$O(\sqrt{\underline{m}\lvert\mathcal{L}\rvert/T})$
Spanning Tree	–	$O(\underline{n}m/\sqrt{T})$	$O(\underline{\ln n}\sqrt{\lvert\mathcal{L}\rvert/T})$
Perfect Matching	–	$O(\underline{n}m/\sqrt{T})$	$O(\underline{\lvert\mathcal{L}\rvert}\sqrt{\lvert\mathcal{L}\rvert/T})^*$
monotone MSCA	$O(\underline{\log m}\sqrt{m/T})$	–	–
submodular MP	$O(\underline{2}\sqrt{m/T})$	–	–

Table 1.3. Regret bounds of two generic algorithms (I) and an algorithm for label costs (II) derived in Chapter 8, when applied to a range of problems. The approximation factor α is underlined. A graph has n nodes and m edges; for set cover, m is the number of sets. MSCA is the minimum submodular-cost allocation problem, which subsumes e.g. submodular-cost facility location. MP stands for "multiway partition". *Result for complete bipartite graphs.

For approximations, a variant called α-regret compares to the best solution that is achievable in polynomial time (with approximation factor α). The regret of a *Hannan-consistent* algorithm vanishes as T increases.

Chapter 8 addresses problems where the decision space has exponential size and consists of combinatorial structures. Almost all former work on combinatorial sequential problems restricts f_t to be additive (linear). Previous work on non-linear loss functions f_t only applies to very simple constraints.

Building on approximation techniques for submodular-cost combinatorial problems, Chapter 8 shows the first Hannan-consistent algorithms for a variety of combinatorial sequential problems with *submodular* costs. Table 1.3 shows example regret bounds implied by those algorithms. We describe three generic online algorithms for submodular-cost combinatorial problems. The first two build on two main approximation strategies and lead to regret bounds of $O(\alpha m/\sqrt{T})$ and $O(\alpha\sqrt{m/T})$, respectively. As a side effect, Corollary 8.1 of Theorem 8.1 tightens a bound for unconstrained online minimization by Hazan and Kale [2009] to $O(\sqrt{m/T})$. Finally, the sub-class of "label cost" functions often admits better approximations. Therefore, the final algorithm is dedicated to this special setting. The algorithm can use any corresponding batch algorithm as a black box and achieves a regret bound of $O(\alpha\sqrt{L/T})$ for L labels.

1.2. Publications contained in this thesis

This thesis covers material from the following publications:

S. Jegelka and J. Bilmes. Approximation Bounds for Inference using Cooperative Cuts. *International Conference on Machine Learning (ICML)*, 2011.
(Chapters 3, 4)

S. Jegelka and J. Bilmes. Submodularity Beyond Submodular Energies: Coupling Edges in Graph Cuts. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.
(Chapters 3, 4, 5, 6)

S. Jegelka and J. Bilmes. Multi-Label Cooperative Cuts. *CVPR Workshop on Inference with Structured Potentials*, 2011.
(Chapter 5)

S. Jegelka, H. Lin and J. Bilmes. On Fast Approximate Submodular Minimization. *Advances in Neural Information Processing Systems (NIPS)*, 2011.
(Chapter 7)

S. Jegelka and J. Bilmes. Online Submodular Minimization for Combinatorial Structures. *International Conference on Machine Learning (ICML)*, 2011.
(Chapter 8)

Chapter 2.

Background

This chapter reviews basic definitions and properties of submodular functions and graph cuts. For reference, a table of notation is also contained in Appendix A.

2.1. Notation

We consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $n = |\mathcal{V}|$ nodes and $m = |\mathcal{E}|$ edges. The set of edges is commonly the ground set of elements we use, and the cost function $f : 2^{\mathcal{E}} \to \mathbb{R}$ is defined over the power set $2^{\mathcal{E}}$ of $\mathcal{E}$. By $\mathbb{R}^{\mathcal{E}}$ we denote the set of all vectors of length $|\mathcal{E}|$ whose entries are indexed by elements in $\mathcal{E}$, that is, the set of all modular (linear) functions over $2^{\mathcal{E}}$ that are zero for the empty set. A vector $w \in \mathbb{R}^{\mathcal{E}}$ corresponds to a function $w(S) = \sum_{e \in S} w(e)$ for all $S \subseteq \mathcal{E}$. The letter w always denotes a vector of weights and the equivalent additive function.

Capital letters A, B, C, S, T refer to sets, and lowercase letters a, b, u, v, x, y, z to elements or variables. For ease of notation, we will also drop the curly braces around single elements and write e for $\{e\}$. We will also write $A + e$ for $A \cup \{e\}$.

2.2. Polynomiality and approximations

We will address minimization problems of a set function $f : 2^{\mathcal{E}} \to \mathbb{R}_+$ over a family $\mathcal{S} \subseteq 2^{\mathcal{E}}$ of feasible sets. Let $S^* \in \operatorname{argmin}_{S \in \mathcal{S}} f(S)$ be an optimal solution. An *approximation algorithm* with *approximation factor* α is an algorithm that returns a solution $\hat{S} \in \mathcal{S}$ such that $f(S^*) \leq f(\hat{S}) \leq \alpha f(S^*)$.

An algorithm runs in *polynomial time* if its running time is upper bounded by a function that is a polynomial in the size of the input, here, in the size $|\mathcal{E}|$ of the ground set. Algorithms that minimize general submodular functions usually assume the function to be given in the form of an oracle, and include the time to evaluate the function as a parameter τ. The running time of a *pseudo-polynomial* algorithm can in addition depend polynomially on the length of the numerical input, that is, for example, the logarithm of the largest function value or edge weight.

Running times are usually given in the *big O* notation. We write $t(n) = O(g(n))$ if there exists an integer n_0 and a constant γ such that for all $n \geq n_0$, it holds that $t(n) \leq \gamma g(n)$. Lower bounds are written using big Omega: $t(n) = \Omega(g(n))$ means that there exists a positive constant γ and an integer n_0 such that for all $n \geq n_0$, it holds that $t(n) \geq \gamma g(n)$.

2.3. Submodular functions

Submodular functions have been important in several fields, such as combinatorics, computer vision and economics. Early work includes work on matroids [Whitney, 1935] (more information on matroids can be found in [Welsh, 1976, Oxley, 1992]), on generalizations of matroids [Edmonds, 1970], and the work by Choquet [1955], who calls submodularity "strong subadditivity". Submodular functions have also been referred to as "semi-modular" functions and "sub-valuations" [Choquet, 1955]. The subsequent paragraphs summarize some basic properties and results about submodular functions. For proofs and further details, the reader is referred to texts such as [Fujishige, 2005, Lovász, 1983, Narayanan, 1997, Schrijver, 2004].

A set function $f : 2^{\mathcal{E}} \to \mathbb{R}$ is submodular if for all A, $B \subseteq \mathcal{E}$, it holds that

$$f(A \cup B) + f(A \cap B) \leq f(A) + f(B). \tag{2.1}$$

A function that satisfies Condition (2.1) with equality is called *modular*. A modular function f_m is additive, meaning that $f_m(A) = f_m(\emptyset) + \sum_{e \in A}(f_m(e) - f_m(\emptyset))$. If inequality (2.1) holds in the other direction, the function is *supermodular*. An alternative definition of submodularity uses marginal costs. Define the *marginal* value of f for all $A \subseteq \mathcal{E}$ with respect to a set $B \subseteq \mathcal{E}$ as

$$\rho_f(A|B) \triangleq f(A \cup B) - f(B) \tag{2.2}$$

We occasionally drop the subscript if the involved function f is clear from context. Submodularity is equivalent to *diminishing marginal costs*: for all $A \subseteq B \subseteq \mathcal{E} \setminus \{e\}$ and all $e \in \mathcal{E}$, it holds that

$$\rho_f(e|B) \leq \rho_f(e|A). \tag{2.3}$$

In the literature, this inequality is often termed *diminishing marginal returns*. Since in this work, the submodular function is viewed as a cost function, we refer to it as diminishing marginal *costs*.

A set function is *monotone* or *nondecreasing* if for all $A \subseteq B \subseteq \mathcal{E}$, it holds that $f(A) \leq f(B)$. It is *normalized* if $f(\emptyset) = 0$. A nonnegative, normalized submodular function is also *subadditive*:

$$f(A \cup B) \leq f(A) + f(B). \tag{2.4}$$

Set functions and pseudo-boolean functions

There is a straightforward correspondence between set functions and pseudo-boolean functions. A *pseudo-boolean function* maps from $\{0,1\}^n$ to $\mathbb{R}$ (a boolean function would map to $\{0,1\}$). Those functions have mostly been considered by their representation as multilinear polynomials. The pseudo-boolean equivalent of a set function $f : 2^{\mathcal{E}} \to \mathbb{R}$ is the pseudo-boolean function $f_p : \{0,1\}^n \to \mathbb{R}$ that acts on indicator vectors, that is, $f_p(\chi_S) \triangleq f(S)$ for all $S \subseteq \mathcal{E}$. Conversely, the set function corresponding to a pseudo-boolean function is $f(S) \triangleq f_p(\chi_S)$ for all $S \subseteq \mathcal{E}$. The pseudo-boolean polynomial of a submodular function is computable via the Möbius inversion formula (described in Section 7.2.1). The *order* of a function is the degree of its corresponding polynomial.

Regularity

In the field of computer vision, submodularity is also known as *regularity* [Kolmogorov and Zabih, 2004]. Regularity there refers to energy functions $E : \{0,1\}^n \to \mathbb{R}$ of binary Markov random fields, and such functions are pseudo-boolean functions. The property is usually defined via restrictions or projections. The *projection* $E_{i,j,y}$ for $y \in \{0,1\}^{n-2}$ is the restriction of E to variables x_i and x_j, where all other variables x_k with $k \neq i,j$ are fixed as $x_k = y_k$. An energy E is regular if for all i,j and all $y \in \{0,1\}^{n-2}$ it holds that

$$E_{i,j,y}(0,0) + E_{i,j,y}(1,1) \leq E_{i,j,y}(1,0) + E_{i,j,y}(0,1), \tag{2.5}$$

$$\text{equivalently,} \quad E_{i,j,y}(1,1) - E_{i,j,y}(1,0) \leq E_{i,j,y}(0,1) - E_{i,j,y}(0,0). \tag{2.6}$$

Let f_E be the set function equivalent to E, and let A be such that $y = \chi_A$. Then Condition (2.6) is equivalent to

$$f(A \cup e_i \cup e_j) - f(A \cup e_i) \leq f(A \cup e_j) - f(A).$$

This can be shown to be equivalent to diminishing marginal costs, Inequality (2.3).

2.3.1. Polyhedra and extensions

An important concept for submodular functions is the *submodular polyhedron*. The polyhedron P_f of f is the set of all normalized modular functions (vectors) $x \in \mathbb{R}^{\mathcal{E}}$ that lower-bound f. Recall that the entries of such a vector are indexed by the elements $e \in \mathcal{E}$. Formally,

$$P_f = \{x \in \mathbb{R}^{\mathcal{E}} \mid x(A) \leq f(A) \; \forall A \subseteq \mathcal{E}\}. \tag{2.7}$$

The base polytope $B_f \subset P_f$ is the set of $x \in P_f$ whose entries sum up to $f(\mathcal{E})$:

$$B_f = \{x \in P_f \mid x(\mathcal{E}) = f(\mathcal{E})\}. \tag{2.8}$$

The base polytope has in general an exponential number of vertices, each of which corresponds to a certain permutation of the ground set. The "greedy algorithm" shows how a vertex relates to a permutation.

The "greedy algorithm" for linear optimization over P_f

In this section, we assume that the submodular function f is normalized. Edmonds [1970] showed that a linear function can be optimized over the polyhedron P_f in $O(m \log m)$ time[1]. Let $c \in \mathbb{R}_+^{\mathcal{E}}$ be a nonnegative[2] vector. To solve the problem

$$\max \quad c \cdot y \ \text{ s.t. } y \in P_f, \tag{2.9}$$

we sort the entries of c in nonincreasing order such that $c_{\pi(1)} \geq c_{\pi(2)} \geq \ldots \geq c_{\pi(m)}$. The order defines a chain of sets $A_1 \subset A_2 \subset \ldots A_m$, with $A_1 = \{e_{\pi(1)}\}$ and $A_i = A_{i-1} \cup \{e_{\pi(i)}\}$ for $i > 0$. The entries of a maximizing vector $y^* \in P_f$ are $y^*_{\pi(1)} = f(A_1)$ and $y^*_{\pi(i)} = f(A_i) - f(A_{i-1}) = \rho_f(e_{\pi(i)}|A_{i-1})$ for $i > 1$. Edmonds [1970] derives this result via the dual of the linear programming problem (2.9). If c is positive, then the vector y^* is always in B_f, because by construction it satisfies $y^*(\mathcal{E}) = f(\mathcal{E})$. By construction and diminishing marginal costs, it also satisfies $y^*(e) \leq f(e)$ for all $e \in \mathcal{E}$.

If the maximizer y^* for a cost vector c is unique, then it is a vertex of the base polytope, because linear functions over polytopes have an optimum that is a vertex. As the vertex y^* only depends on the permutation of the ground set, the above construction also shows a one-to-one correspondence between permutations of $\mathcal{E}$ and vertices of P_f [Fujishige, 2005, Edmonds, 1970, Shapley, 1971, Lovász, 1983]. If f is nondecreasing, then y^* is nonnegative. This implies that for nondecreasing submodular functions, all vectors in B_f are nonnegative.

Extensions

Two different constructions are commonly used that extend submodular functions from a discrete domain of characteristic vectors to a continuous domain: the Lovász extension and a multilinear extension. The *characteristic vector* $\chi_A \in \{0,1\}^{\mathcal{E}}$ of a set $A \subseteq \mathcal{E}$ is a vector whose eth entry is one if and only if $e \in A$. Given a vector $x \in \mathbb{R}^{\mathcal{E}}$, let $\{\beta_1, \ldots, \beta_k\}$ be its distinct entries. The level sets of x are the index sets $B_j = \{i \mid x(i) \leq \beta_j\}$. Any vector $x \in \mathbb{R}^{|\mathcal{E}|}$ can be written as a unique combination of the characteristic vectors of its level sets:

$$x = \sum\nolimits_j \lambda_j \chi_{B_j}. \tag{2.10}$$

[1]Strictly speaking, the 1970 paper assumes a polymatroid rank function, but the same procedure works for general normalized submodular functions; Lovász [1983] only assumes a normalized function for the greedy algorithm.

[2]If c has a negative entry, then the optimal solution is unbounded. Edmonds [1970] initially proved the results here for polymatroid rank functions and the polymatroid, which is restricted to nonnegative vectors. In that case, c can be negative, and the greedy algorithm still works as presented here, with the modification that for each $c_i < 0$, we set $y^*_i = 0$. If we aim for a solution in the base polytope, we proceed as above.

Lovász extension. The *Lovász extension* $\tilde{f}$ of a set function f uses the decomposition (2.10) for an equivalent sum of function values [Lovász, 1983]:

$$\tilde{f}(x) = \sum\nolimits_j \lambda_j f(B_j). \tag{2.11}$$

The extension $\tilde{f}$ is convex if and only if f is submodular. It is positively homogeneous, and the extension of $f + g$ is $\tilde{f} + \tilde{g}$. Moreover, the definition (2.11) implies that $\tilde{f}(\chi_A) = f(A)$ for all sets $A \subseteq \mathcal{E}$. The Lovász extension is equivalent to the *Choquet integral*

$$\tilde{f}(x) = (C)\int x df \triangleq \int_{-\infty}^{0} (f(\{e_i \mid x_i \geq \theta\}) - f(\mathcal{E}))d\theta + \int_{0}^{\infty} f(\{e_i \mid x_i \geq \theta\})d\theta. \tag{2.12}$$

For submodular functions, the Lovász extension is closely tied to the optimization problem (2.9). If f is submodular, then the coefficients λ_j in the sum within (2.11) are an optimal solution of the dual of Problem (2.9) [Edmonds, 1970]. This means that for submodular functions, the Lovász extension can equivalently be written as

$$\tilde{f}(x) = \max_{y \in P_f} \quad y \cdot x. \tag{2.13}$$

The maximizing y is computed by the greedy algorithm, and the level sets of x are parts of the chain of A_i described above, with $\lambda_j = (x_{\pi(j)} - x_{\pi(j+1)})$ and the ordering π determined by sorting y.

Subgradient. Let $g_x = \operatorname{argmax}_{y \in P_f} y \cdot x$ be a maximizer in Equation (2.13). By the definition of the Lovász extension, it holds that $g_x \cdot x' \leq \tilde{f}(x')$ for all $x' \in \mathbb{R}_+^{\mathcal{E}}$. In consequence, g_x is a subgradient of $\tilde{f}$ at x: it satisfies the condition

$$\tilde{f}(x') - \tilde{f}(x) \geq g_x \cdot (x' - x). \tag{2.14}$$

Via the greedy algorithm, a subgradient of $\tilde{f}$ can thus be computed in $O(m \log m)$ time.

Lovász extension and sampling. The definition (2.11) via level sets and the Choquet integral point at another interpretation of the Lovász extension: It can be viewed as the expected function value when randomly drawing level sets with probabilities proportional to the coefficients λ_j. Let $x \in [0,1]^{\mathcal{E}}$. Assume we pick a threshold $\theta \in [0,1]$ uniformly at random, and let $A_\theta = \{e \mid x(e) \geq \theta\}$. Then $x = \mathbb{E}_\theta[\chi_{A_\theta}]$ and

$$\tilde{f}(x) = \mathbb{E}_\theta[f(A_\theta)]. \tag{2.15}$$

Multilinear extension. An alternative extension results from a different way of sampling sets. Consider entry $x(e)$ in a vector $x \in [0,1]^{\mathcal{E}}$ as the probability of element e being chosen. The probabilities of different elements are independent. The resulting extension is [Calinescu et al., 2011, Vondrák, 2008a]

$$\check{f}(x) = \mathbb{E}[f(x)] = \sum_{S \subseteq \mathcal{E}} f(S) \prod_{e \in S} x(e) \prod_{e \notin S} (1 - x(e)). \tag{2.16}$$

This extension is concave in any nonnegative direction and has been used for maximization problems. In the sequel, we use the Lovász extension.

2.3.2. Matroids, polymatroids and submodular functions

Submodular functions can be seen as a generalization of matroid rank functions. A matroid itself is a generalization of the linear dependence structure of columns of a matrix.

Matroids

Definition 2.1 (Matroid)**.** Given a ground set $\mathcal{E}$ of elements, the tuple $\mathcal{M} = (\mathcal{E}, \mathcal{I})$ is a *matroid* if the following holds for its family $\mathcal{I} \subseteq 2^{\mathcal{E}}$:

1. $\emptyset \in \mathcal{I}$.
2. If $I_1 \subseteq I_2 \in \mathcal{I}$, then $I_1 \in \mathcal{I}$.
3. If $I_1, I_2 \in \mathcal{I}$ and $|I_1| < |I_2|$, then there exists an element $e \in I_2 \setminus I_1$ such that $I_1 \cup \{e\} \in \mathcal{I}$.

Each $I \in \mathcal{I}$ is called an *independent set*, and the family $\mathcal{I}$ is the *family of independent sets* of the matroid. A set $J \notin \mathcal{I}$ is said to be *dependent*.

If the elements in the ground set $\mathcal{E}$ are the columns of a matrix, then each set of linearly independent columns is an independent set. Analogous to the rank of a matrix, we can define the rank function of a matroid.

Definition 2.2 (Matroid rank function)**.** The *rank function* $r : 2^{\mathcal{E}} \to \mathbb{R}_+$ of a matroid $\mathcal{M} = (\mathcal{E}, \mathcal{I})$ is defined as

$$r(S) = \max\{|I| \mid I \subseteq S, I \in \mathcal{I}\}.$$

A matroid rank function is a nondecreasing, normalized, nonnegative and integral submodular function. For all $e \in \mathcal{E}$, the element-wise marginal costs $\rho_f(e|S)$ of a matroid rank function are always zero or one. This property is called the *unit-increase property.*

Definition 2.2 shows that the independent sets $I \in \mathcal{I}$ are those for which it holds that $r(I) = |I|$. The *bases* of a matroid are the maximal independent sets, that means, if we add any element $e \notin B$ to a base $B \in \mathcal{I}$, then the resulting set $B \cup \{e\}$ is dependent. This also means that $r(\mathcal{E}) = r(B)$ for any base B. All bases have the same cardinality. The bases of a matroid satisfy an *exchange property*: For all bases B_1, B_2 and for any $e_1 \in B_1 \setminus B_2$, there exists an element $e_2 \in B_2 \setminus B_1$ such that replacing e_1 by e_2 results in another base $B_1 - e_1 + e_2$.

The *matroid polytope* $\overline{P}$ is the convex hull of the characteristic vectors of all independent sets. Equivalently, it consists of all nonnegative vectors in the submodular polyhedron P_r of the rank function.

Polymatroids

Polymatroids generalize matroids [Edmonds, 1970].

Definition 2.3 (Polymatroid rank function)**.** A function $f : 2^{\mathcal{E}} \to \mathbb{R}$ over subsets of a ground set $\mathcal{E}$ is a polymatroid (rank) function if

1. $f(\emptyset) = 0$.
2. f is nondecreasing: $S \subseteq T \subseteq \mathcal{E}$ implies that $f(S) \leq f(T)$.
3. f is submodular.

Edmonds [1970] calls these functions β-functions. Matroid rank functions are special cases of polymatroid functions for which the unit increase property holds. The pair $(\mathcal{E}, f)$ is called a *polymatroid*.

To see the relation between polymatroids and matroids, recall that in a matroid, (1) every subset of an independent set is independent, and (2) all bases have the same cardinality. From the viewpoint of characteristic vectors, (1) means that if S is independent and $\chi_S \in P_r$, then $\chi_T \in P_r$ for all $T \subseteq S$ or $\chi_T \leq \chi_S$ ("$x \leq y$" means that the inequality holds entry-wise). Now consider the following definition by Edmonds [1970]: A polymatroid (polytope) $\overline{P} \subset \mathbb{R}_+^{\mathcal{E}}$ is a compact non-empty set such that

1. $\overline{P}$ is *down-monotone*, i.e., if $0 \leq x \leq y$ and $y \in \overline{P}$, then also $x \in \overline{P}$.
2. For any vector $y \in \mathbb{R}_+^{\mathcal{E}}$, every maximal $x \leq y$ with $x \in \overline{P}$ has the same component sum $\sum_{e\in\mathcal{E}} x(e)$.

Edmonds calls the sum in (2.) the rank $r(y)$ of y. The connection to set functions follows from the polymatroid associated with a function f. The polymatroid $\overline{P}_f$ of a function f satisfying Definition 2.3 is

$$\overline{P}_f = \big\{x \in \mathbb{R}_+^{\mathcal{E}} \mid \sum_{e\in S} x(e) \leq f(S) \text{ for all } S \subseteq \mathcal{E}\big\}. \tag{2.17}$$

Let χ_S be the characteristic vector of a set $S \subseteq \mathcal{E}$, and define a vector z_S such that $z_e = f(e)$ if $e \in S$, and $z_e = 0$ otherwise. If $f(e) > 0$ for all elements e, then this vector has the same pattern of nonzeros as χ_S. If f is the rank function of a matroid without self-loops (elements with $f(e) = 0$), then $z_S = \chi_S$. Moreover, $x \leq z_S$ for all $x \in \overline{P}_f$ with $x(e) = 0$ for all $e \notin S$. The rank of z_S is

$$r(z_S) \;=\; \max_{y\in\overline{P}_f, y\leq z_S} y(\mathcal{E}) \;=\; \max_{y\in\overline{P}_f} y(S) \;=\; \max_{y\in P_f} y\cdot\chi_S \;=\; f(S).$$

The greedy algorithm [Edmonds, 1970] implies the last equality. The vector y maximizing the dot product over the submodular polyhedron (also called extended polymatroid) of f must be nonnegative; this again follows from the greedy algorithm and the fact that f is nondecreasing. Thus, maximizing over P_f or $\overline{P}_f$ is equivalent here. The equations show how the vector rank r yields the value of the polymatroid function.

General submodular functions

A *submodular* set function generalizes polymatroid rank functions further as it neither needs to be nondecreasing nor normalized. It only still satisfies Inequality (2.1) and of course also diminishing marginal costs.

Submodularity is sometimes defined as a property of functions over a distributive lattice. For sets, the "meet" $\wedge$ and "join" $\vee$ operations are set intersection and inclusion, respectively. As this thesis only considers submodular set functions, general lattices are only mentioned here. For a lattice $\mathcal{P} = (\mathcal{E}, \preceq)$ with partial order $\preceq$, the equivalent of Inequality (2.1) is

$$f(a \vee b) + f(a \wedge b) \leq f(a) + f(b) \tag{2.18}$$

for all $a, b \in \mathcal{P}$.

2.3.3. Examples of submodular functions

For illustration, this section lists some examples of submodular functions that are well-known or that will be relevant in subsequent chapters.

Matroid rank functions

Column matroid/linear matroid. Consider a matrix M whose columns are indexed by elements in a ground set $\mathcal{E}$. A set $S \subseteq \mathcal{E}$ is independent if the corresponding columns are linearly independent, otherwise it is dependent.
Uniform matroid. The perhaps simplest matroid is the uniform matroid. In a uniform matroid of rank k, all sets with cardinality less than or equal to k, $|S| \leq k$, are independent. Its rank function is $r(S) = \min\{|S|, k\}$. A free matroid is a uniform matroid with $k = |\mathcal{E}|$ in which all sets are independent.
Partition matroid. Consider a partition of the ground set $\mathcal{E}$ into m disjoint sets E_i. Each E_i is assigned a rank k_i. A partition matroid defines its independent sets as

$$\mathcal{I} = \{I \subseteq \mathcal{E} \mid (I \cap E_i) \leq k_i \text{ for all } 1 \leq i \leq m\}. \tag{2.19}$$

We can view this matroid as a direct sum of uniform matroids. The direct sum of these uniform rank functions is the rank function of the partition matroid:

$$r(S) = \sum_{i=1}^{m} \min\{|S \cap E_i|, k_i\}. \tag{2.20}$$

A simple example of a partition matroid is the following: assume each element in the ground set has a color, and the rank of a set is the number of colors occurring in that set. In this matroid, we have one E_i for each color, and $k_i = 1$ for all i.
Bipartite matching or bipartite connected components. The following matroid will become important in Chapter 3. The elements in the ground set are

vertices in a graph, and each vertex has degree one. This graph implements a bipartite matching. A set is independent if it does not contain any two connected nodes. Indeed, these are the independent sets of the graph. For this particular graph, the rank function can be seen to define a matroid by relating the graph to a partition matroid. The edges are the sets E_i of the partition, each has two elements. With $k_i = 1$ for each edge, the resulting partition matroid is equivalent to the bipartite matching matroid. (Remark: this matroid is different from what is commonly referred to as a "matching matroid".)

Graphic matroid. A frequently occurring matroid is the graphic matroid. Its ground set is the set of edges $\mathcal{E}$ in a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. A set of edges is independent if it defines a cycle-free subgraph of $\mathcal{G}$. In other words, all independent sets are forests, and, if $\mathcal{G}$ is connected, then the bases are the spanning trees of $\mathcal{G}$. We will encounter this matroid in Chapter 8.

Polymatroid rank functions

Covers. Assume each element e in the ground set $\mathcal{E}$ has an associated area $\mathrm{ar}(e)$ that it covers, for example, sensors operating in a particular area. The function $\mathrm{ar} : 2^{\mathcal{E}} \to \mathbb{R}_+$ measures the area that is jointly covered by the elements in a set. This function obviously satisfies the condition of diminishing marginal function values.

Neighborhoods in bipartite graphs. The discrete analogue of cover functions are neighborhoods in a bipartite graph $\mathcal{G} = (\mathcal{V}_1, \mathcal{V}_2, \mathcal{F})$. The ground set is $\mathcal{V}_1$. We define the *neighborhood function* $\mathcal{N} : 2^{\mathcal{V}_1} \to 2^{\mathcal{V}_2}$, illustrated in Figure 2.1, as

$$\mathcal{N}(S) = \{v \in \mathcal{V}_2 \mid v \text{ is connected to a node } u \in S \text{ by an edge } (u, v) \in \mathcal{F}\}. \quad (2.21)$$

The function $f : 2^{\mathcal{V}_1} \to \mathbb{R}_+$ defined as $f(S) = |\mathcal{N}(S)|$ is normalized and nondecreasing submodular.

Maximum. A simple example of a polymatroid rank function is the maximum. Assume each element $e \in \mathcal{E}$ has a nonnegative weight $w(e)$. Then the maximum function charges the weight of the heaviest element in the set: $f(S) = \max_{e \in S} w(e)$. For consistency, we set $f(\emptyset) = 0$.

Discounted price functions. Let $w : 2^{\mathcal{E}} \to \mathbb{R}_+$ be a normalized modular function. Goel et al. [2010] define a discounted price function as a nondecreasing concave scalar function g composed with w, i.e., $f(S) = g(w(S)) = g(\sum_{e \in S} w(e))$. It must also hold that $g(0) = 0$. It has long been known that $h(w(S))$ for a scalar function h is submodular if h is concave [Shapley, 1971]; for a polymatroid rank function, the scalar function must also be nondecreasing.

Entropy. A well-known example of a polymatroid rank function is the joint Shannon entropy. Here, each element in the ground set $\mathcal{E}$ is a random variable, and the function $f(S)$ is the joint entropy of the variables in S. Since $f(\emptyset) = 0$, this function is a polymatroid rank function.

Submodular functions

Cut function. Given an undirected weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$, the ground set is the set $\mathcal{V}$ of nodes. The cut $\delta(S)$ around a set of nodes $S \subseteq \mathcal{V}$ is the weight of the edges that have only one end-point in S, that is,

$$\delta(S) = \{e \in \mathcal{E} \mid |e \cap S| = 1\}. \tag{2.22}$$

The cut function cut : $2^{\mathcal{V}} \to \mathbb{R}_+$ measures the weight of a cut around a set, i.e., $\mathrm{cut}(S) = w(\delta(S)) = \sum_{e \in \delta(S)} w(e)$. This function is submodular. A similar definition holds for a directed cut in a directed graph.
Concave function of a sum. A scalar, not necessarily monotone concave function $g : \mathbb{R}_+ \to \mathbb{R}$ composed with a sum, i.e., nonnegative modular function $w : 2^{\mathcal{E}} \to \mathbb{R}_+$, is submodular [Shapley, 1971]. Such a function is defined as $f(S) = g(w(S))$. Shapley [1971] calls the supermodular equivalent of such a function, derived as a convex function of a sum, a *convex measure game.*

A short discussion

The above examples already offer a glimpse of the myriad of different types of submodular functions. For these functions, there are additional distinctions other than being monotone or not. First of all, *symmetric* submodular functions that satisfy $f(S) = f(\mathcal{E} \setminus S)$ seem to be easier to handle: they can be minimized in cubic time [Queyranne, 1998], whereas the best known worst-case running time for general submodular minimization is at least $O(n^6)$ [Orlin, 2009]. Even for submodular maximization, symmetric submodular functions admit better or simpler approximation results than general submodular functions [Feige et al., 2011].

Furthermore, the identifiability of "interactions" varies. As an example, for covers or for discounted price functions with a strictly concave g, the "interaction structure" of the elements is rather simple. By considering a pair of elements alone one can see whether they interact or not, i.e., whether their joint presence in a set S will make the cost of S less than the sum of the costs of single elements in S. Formally, the cost is reduced as $f(S) < \sum_{e \in S} f(e)$ if and only if there exist $e_i, e_j \in S$ with $f(\{e_i, e_j\}) < f(e_i) + f(e_j)$. In contrast, for the rank of a graphic matroid on a graph without multi-edges or self loops, it holds that $f(\{e_i, e_j\}) = f(e_i) + f(e_j) = 2$ for any pair of edges in a cycle C, even though $f(C) = |C| - 1 < |C|$. In some sense, the functions are harder to optimize if interactions cannot be identified from small groups, at least if the underlying structure is unknown. This "hiding" property can be exploited for proving lower bounds as demonstrated in Chapter 3.

Moreover, the uniform matroid appears to be simpler than, say, the graphic matroid. The rank of the uniform matroid only depends on the cardinality of the argument and therefore the function is invariant to permutations of the ground set. This is another type of symmetry. Other functions with more underlying structure do not treat all elements equally. The notion of symmetry with respect to permutations has been studied with respect to the hardness of submodular maximization, where the “symmetry gap” plays a role [Vondrák, 2011].

2.3.4. Operations and construction of additional submodular functions

Given the basic submodular functions above, several operations allow to define additional submodular functions. Some examples are outlined next.

Complement

If f is submodular, then the function $\bar{f}$ defined by taking complements, $\bar{f}(S) = f(\mathcal{E} \setminus S)$ is also submodular.

Sum

It is simple to check that any linear combination $\sum_{i=1}^{m} \alpha_i f_i$ of submodular functions f_i with positive coefficients α_i is submodular.

Contraction

The *contraction* of a function f by a set $A \subseteq \mathcal{E}$ is defined for any $S \subseteq \mathcal{E} \setminus A$ as the marginal cost

$$f_A(S) = \rho_f(S|A) = f(A \cup S) - f(A). \tag{2.23}$$

The contraction of a submodular function is submodular.

Direct sum

Given two submodular functions $f_1 : 2^{\mathcal{E}_1} \to \mathbb{R}_+$ and $f_2 : 2^{\mathcal{E}_2} \to \mathbb{R}_+$ on disjoint ground sets $\mathcal{E}_1 \cap \mathcal{E}_2 = \emptyset$, we can define their direct sum $f_1 \oplus f_2 : 2^{\mathcal{E}_1 \cup \mathcal{E}_2} \to \mathbb{R}_+$:

$$(f_1 \oplus f_2)(S) = f_1(S \cap \mathcal{E}_1) + f_2(S \cap \mathcal{E}_2). \tag{2.24}$$

The direct sum is again a submodular function. Conversely, a set function f is *decomposable* with respect to a partition $\{E_1, \dots, E_k\}$ of the ground set $\mathcal{E}$ if

$$f(S) = \sum_{i=1}^{k} f(S \cap E_i) \tag{2.25}$$

for all $S \subseteq \mathcal{E}$. The sets E_i are called *separators* of f. The direct sum construction makes $\mathcal{E}_1$ and $\mathcal{E}_2$ the separators of $f_1 \oplus f_2$. In the sequel, a *separable* function will be a function that is decomposable this way with respect to any partition, that means a modular function $f(S) = \sum_{e \in \mathcal{E}} f(S \cap e)$.

As mentioned above, the rank of a partition matroid can be viewed as a direct sum of uniform matroid rank functions. Similarly, the label costs defined in Section 2.5 can be constructed as a direct sum of polymatroid rank functions of the form

$$f_i(S) = \begin{cases} 0 & \text{if } S = \emptyset \\ \gamma_i \min(|S|, 1) > 0 & \text{otherwise.} \end{cases} \tag{2.26}$$

Direct sum constructions derived from a partition of a large ground set will be used in particular for the applications to image segmentation in Chapter 6.

Truncation

The truncation of a nondecreasing normalized submodular function f at threshold $\gamma > 0$ is defined as $g(S) = \min\{f(S), \gamma\}$ and again submodular. As an example, a uniform matroid is the truncation of the cardinality function at k.

Convolution

The (lower) convolution of two submodular functions f, g is

$$(f * g)(S) = \min_{T \subseteq S} \quad f(T) + g(S \setminus T). \tag{2.27}$$

The convolution is submodular if f or g is modular, but otherwise not necessarily. If f and g are matroid rank functions, then $f * g$ is the rank function of the intersection of the matroids. Matroid intersections are not always matroids.

Truncations will occur in Chapter 4. Moreover, a truncation is a convolution with a function of the form (2.26).

Functions induced by a graph

Given a bipartite graph $\mathcal{G} = (\mathcal{V}_1, \mathcal{V}_2, \mathcal{F})$, the neighborhood $\mathcal{N}(S)$ of a set of nodes $S \subseteq \mathcal{V}_1$ was defined in Equation (2.21) as the set of nodes in $\mathcal{V}_2$ reachable from nodes $X \subseteq \mathcal{V}_1$ via edges $\mathcal{F}$. Figure 2.1 illustrates such a neighborhood.

Proposition 2.1. *Let $f_2 : 2^{\mathcal{V}_2} \to \mathbb{R}_+$ be a nondecreasing submodular function on subsets of $\mathcal{V}_2$. Then the induced function $f_1 : 2^{\mathcal{V}_1} \to \mathbb{R}_+$,*

$$f_1(S) = f_2(\mathcal{N}(S)) \tag{2.28}$$

is nondecreasing submodular.

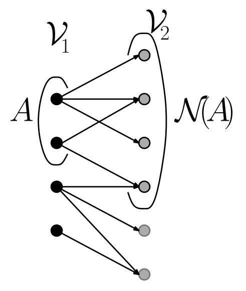


Figure 2.1 Bipartite graph and the neighborhood $\mathcal{N}(A) \subseteq \mathcal{V}_2$ of set $A \subseteq \mathcal{V}_1$.

Proof. The proof relies on diminishing marginal costs (see also [Schrijver, 2004, §44.6g]). Let $S \subseteq T \subseteq \mathcal{V}_1 \setminus \{v\}$. By the submodularity of f_2, it holds that

$$\begin{aligned}
&f_1(T \cup \{v\}) - f_1(T) && (2.29)\\
&= f_2\big(\mathcal{N}(S) \cup \mathcal{N}(T \setminus S) \cup (\mathcal{N}(v) \setminus \mathcal{N}(T))\big) - f_2(\mathcal{N}(S) \cup \mathcal{N}(T \setminus S)) && (2.30)\\
&\leq f_2\big(\mathcal{N}(S) \cup (\mathcal{N}(v) \setminus \mathcal{N}(T))\big) - f_2(\mathcal{N}(S)) && (2.31)\\
&\leq f_2\big(\mathcal{N}(S) \cup \mathcal{N}(v)\big) - f_2(\mathcal{N}(S)) && (2.32)\\
&= f_1(S \cup \{v\}) - f_1(S). && (2.33)
\end{aligned}$$

For the second inequality, we used that f_2 is nondecreasing. □

2.3.5. Minimizing submodular functions

Optimization of submodular functions is an ongoing topic of research. Whereas maximizing submodular functions is NP-hard, minimizing submodular functions is not. As this thesis focuses on minimization problems, we give an overview of results for minimization. Further details of unconstrained submodular minimization are discussed in Chapter 7, and submodular minimization with combinatorial constraints is addressed in Chapters 3 and 4.

Unconstrained submodular function minimization

The first polynomial-time algorithm for minimizing general submodular functions uses the ellipsoid method and was devised by Grötschel et al. [1981]. The initial pseudo-polynomial algorithm was later turned into a strongly polynomial one [Grötschel et al., 1988, p. 311]. The question about a polynomial-time combinatorial algorithm remained open for longer. Cunningham [1984] resolved it for the special case of a difference between a matroid rank function and a rational modular function[3]. In a different article [Cunningham, 1985a], he showed a flow-like framework that led to a pseudo-polynomial time algorithm and that inspired further developments. The first strongly polynomial-time combinatorial algorithms were

[3]Strictly speaking, Edmonds [1965] already solves the minimization of a matroid rank minus a scaled cardinality function.

then presented by Iwata et al. [2001] and Schrijver [2000]. Several papers followed with further algorithms and improvements, summarized in [Fleischer, 2000, McCormick, 2006]. The currently (as of February 2012) fastest strongly polynomial combinatorial algorithm has a worst-case running time of $O(n^5\tau + n^6)$ for a ground set of size n and time τ of querying the function oracle [Orlin, 2009].

An alternative with to date unknown complexity is the minimum norm point algorithm, also called the Fujishige-Wolfe algorithm [Fujishige et al., 2006, Fujishige and Isotani, 2011]. However, an example in Chapter 7 indicates that the worst-case running time of the minimum norm point algorithm may not always be practical either.

Lower bounds on the number of function oracle queries needed for submodular minimization are still an open problem, apart from the obvious bound of $\Omega(m)$ [McCormick, 2006, Harvey, 2008].

Not all submodular functions pose the same algorithmic complications as the general case. Symmetric submodular functions, for which $f(S) = f(\mathcal{E} \setminus S)$ for all $S \subseteq \mathcal{E}$ can be minimized in cubic time [Queyranne, 1998]. The same technique, a generalization of a minimum cut algorithm, applies to posimodular functions, which satisfy $f(S \setminus T) + f(T \setminus S) \leq f(S) + f(T)$ [Nagamochi and Ibaraki, 1998]. Minimum cut in a graph is another well-studied special case of symmetric submodular function minimization.

Constrained submodular function minimization

Whereas unconstrained submodular minimization is solvable in polynomial time, even simple constraints very quickly render the problem NP-hard. When considering that cut functions are a special case of submodular functions, this observation, however, is not so surprising. Minimum cuts can be computed in low-order polynomial time, but most additional size constraints lead to very hard problems [Wagner and Wagner, 1993, Khot, 2004].

Minimizing submodular functions over a ring[4] or crossing family can be reduced to submodular minimization over a power set [Grötschel et al., 1988, Chapter 10]. Similarly, the constraint that the solution should have odd cardinality does not make the problem NP-hard [Grötschel et al., 1981, 1984], [Grötschel et al., 1988, Section 10.4]. When looking at the more general notion of lattices (instead of power sets), submodular minimization is feasible over finite distributive lattices (if the smallest and largest element in the lattice and the preorder are computable in polynomial time) [Schrijver, 2004, Section 49.3]. Krokhin and Larose [2008] show a number of non-distributive and product lattices for which submodular minimization is polynomial-time solvable.

Size constraints, in contrast, result in difficult problems. Svitkina and Fleischer [2008] assume a modular weight function w on the ground set and define SUBMOD-

[4]A ring family is a collection of subsets that is closed under union and intersection.

ULAR BALANCED CUT as minimizing a submodular function $f(S)$ subject to the constraint that $w(S) \geq \gamma w(\mathcal{E})$ and $w(\mathcal{E} \setminus S) \geq \gamma w(\mathcal{E})$ for a fraction $\gamma \in [0, 1]$. The approximation factor for this problem has a lower bound of $\Omega(\sqrt{n/\log n})$ in the oracle model. A similar lower bound holds for bicriteria approximations and a constraint $w(S) \geq W$, proved in the same paper. The equality constraint $|S| = k$ has long been known to render minimum cut (a special case of submodular minimization) NP-hard [Garey et al., 1976], and this does not only hold for $k = n/2$ [Feige et al., 2003, Bui and Jones, 1992]. Nagano et al. [2011] show that the vector in B_f with minimum norm gives the optimal solution for a subset of possible k, however, this subset cannot be controlled.

An upper bound on the cardinality of the solution too makes the problem hard, but only an NP-hardness result is known [Svitkina and Fleischer, 2008]. In this setting, symmetric submodular functions are again "easier" and can be minimized with cardinality upper bounds in polynomial time. Goemans and Soto [2010] prove that this holds for all constraints that define downward-monotone families of sets. Being *downward-monotone* means that if the family contains a set S, then it also contains all subsets of S.

Combinatorial constraints, such as that the solution should be a spanning tree or a cut in a given graph, are even harder. Chapter 3 summarizes results for such constraints and shows new results for cut constraints.

2.4. Graph cuts

A wide range of problems in operations research, machine learning, computer vision and network analysis can be phrased as graph cuts. A *cut* in a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with nodes $\mathcal{V}$ and edges $\mathcal{E}$ is a set of edges $C \subseteq \mathcal{E}$ whose removal partitions the graph, so that the resulting graph has at least one more connected component than without the edge removal. This definition holds for directed and undirected graphs. An (s,t)-cut for distinguished terminal nodes $s, t \in \mathcal{V}$ is a set of edges whose removal disconnects all paths from s to t.

In general, we are seeking a *minimal* cut, that means a cut $C \subseteq \mathcal{E}$ such that no proper subset $B \subset C$ is a cut. A minimal cut in a connected graph is always the *boundary* $\delta(S)$ of a set of nodes $S \subseteq \mathcal{V}$, that is, $C = \delta(S) = \{(u, v) \in \mathcal{E} \mid u \in S, v \in \mathcal{V} \setminus S\}$. The standard minimum (s,t)-cut problem reads as follows:

Problem 2.1 (Minimum (s,t)-cut (MINCUT)). *Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with edge weights $w : \mathcal{E} \to \mathbb{R}_+$ and two distinguished terminal nodes $s, t \in \mathcal{V}$, find the (s,t)-cut $C \subseteq \mathcal{E}$ with minimum weight $w(C) = \sum_{e \in C} w(e)$.*

The MINCUT problem can be stated in a general form as

$$\min \; f(C) \quad \text{s.t. } C \text{ is an } (s,t)\text{-cut.} \tag{2.34}$$

The standard MinCut problem uses an additive, nondecreasing cost $f(C) = w(C) = \sum_{e \in C} w(e)$.

If the weights w are nonnegative, then MinCut with such an additive cost function can be solved in polynomial time in e.g. $O(n^3)$ or $O(n^2\sqrt{m})$ [Ahuja et al., 1993], usually via its dual problem, MaxFlow. Empirically, the running times for special graphs (for adapted algorithms) can be much better. Boykov and Kolmogorov [2004], for example, report near-linear running times for grid graphs used in computer vision. If edge weights are allowed to be negative, then the minimum cut problem is NP-hard, as it includes MaxCut, and as Correlation Clustering for 2 clusters can be reduced to it. For correlation clustering, the graph edges are relations “similar” or “dissimilar” between adjacent nodes. The task is to group the nodes to minimize the similarity relations across clusters and the dissimilarity relations within clusters. This problem was proven to be NP-hard by Shamir et al. [2002], Giotis and Guruswami [2006].

Furthermore, the minimum cut problem becomes hard when (polynomial) size constraints on the partition are added [Wagner and Wagner, 1993]. Chapter 3 shows that more general non-additive cost functions can also make the problem hard, in fact, they lead to lower bounds that are larger than for many of the size-constrained cut problems.

Minimum cut and Maximum flow

The dual problem to MinCut is the MaxFlow problem [Ford and Fulkerson, 1954, 1956, Dantzig and Fulkerson, 1955, 1956]. This duality result has had great impact, and a generalization of it will become important in Chapter 4. Therefore we review it duality here.

The MaxFlow problem asks to maximize the flow from node s (the source) to node t (the sink), while regarding the constraints that the flow on each edge e should not exceed the capacity $w(e)$.

We begin with the linear program (LP) for MinCut which has variables $\pi \in [0,1]^{\mathcal{V}}$ for nodes and $x \in [0,1]^{\mathcal{E}}$ for edges. An edge is in the cut if $x(e) = 1$.

$$\begin{aligned} \min_{x,\pi} \quad & \sum_{e \in \mathcal{E}} w(e)x(e) \\ \text{s.t.} \quad & \pi(v) - \pi(u) + x(e) \geq 0 \quad \forall e = (u,v) \in \mathcal{E} \\ & \pi(s) - \pi(t) \geq 1 \\ & 0 \leq x, \pi \leq 1 \end{aligned}$$

This LP always has an integral optimal solution (because the constraint matrix is totally unimodular), and so nothing changes if we relax the constraints $x \in \{0,1\}^{\mathcal{E}}$ and $\pi \in \{0,1\}^{\mathcal{V}}$ to those variables being between zero and one. In the optimal solution, π is the indicator vector of the nodes reachable from s, that is, the s-side of the cut, and $x(e) = 1$ if and only if edge e crosses the cut from the s-side ($\pi(v) = 1$) to the t-side ($\pi(v) = 0$).

The dual problem reads as follows, using constants $d(v) = 0$ if $v \in \mathcal{V} \setminus \{s,t\}$, $d(u) = 1$ if $u = s$, and $d(u) = -1$ if $u = t$:

$$\begin{aligned} &\max_{\nu,\varphi} \nu \\ \text{s.t.} \quad &\varphi(e) \leq w(e) \quad \forall e \in \mathcal{E} \\ \sum_{e \in \delta^+ v} \varphi(e) - \sum_{e' \in \delta^- v} &\varphi(e') = d(v)\nu \quad \text{for all } v \in \mathcal{V} \\ &\varphi \geq 0. \end{aligned}$$

The variable $\nu \in \mathbb{R}$ captures the total flow out of the source and into the sink. For $v \neq s,t$, the second inequality says that the flow into and out of a node should be the same. The flow on an edge e is denoted by $\varphi(e)$. In a directed graph, we denote the boundary of outgoing edges by $\delta^+ S = \{(u,v) \in \mathcal{E} \mid u \in S, v \notin S\}$ and that of incoming edges by $\delta^- S = \{(v,u) \in \mathcal{E} \mid u \in S, v \notin S\}$. We see that the weight $w(e)$ of an edge reappears in the flow program as the capacity of an edge.

Cooperative Cut

The common cost in the MinCut problem is the sum $f(C) = w(C)$ of weights of the cut edges. This is a separable, modular function, meaning that it is a sum of terms that involve only one edge each. The results in Chapter 3 suggest that, given f is nonnegative and nondecreasing, this separability is an essential ingredient for MinCut being tractable. While enabling efficient algorithms on the one hand, such an additive cost function can, on the other hand, be very limiting. An example are the simple models in Figure 1.1 that rely on additive unary or pairwise energies (the latter are closely related to minimum cuts).

In the following chapters we allow f to be non-separable. In particular, f can be any nondecreasing, nonnegative *submodular* function. We re-state a central problem in this thesis:

Problem 2.2 (Minimum Cooperative Cut (MinCoopCut)). *Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, two distinguished terminal nodes $s,t \in \mathcal{V}$ and a nonnegative, nondecreasing submodular function $f : 2^{\mathcal{E}} \to \mathbb{R}_+$, find an (s,t)-cut $C \subseteq \mathcal{E}$ minimizing the cost $f(C)$.*

We will say that edge e *cooperates* with the edges in $B \subseteq \mathcal{E}$ if its marginal cost with respect to B is smaller than its individual cost: $\rho_f(e|B) < f(e)$. Then it holds that the joint cost $f(B \cup e) < \sum_{e' \in B \cup e} f(e')$ is smaller than the sum of element-wise costs.

2.5. Examples of combinatorial problems with submodular cost functions

Apart from the new applications of CoopCut that will be encountered in subsequent chapters, there is a range of combinatorial problems whose cost function is not a sum of weights. The following list points to some examples. All of these examples are combinatorial problems with submodular cost functions.

Label Costs

With label costs, each element e in the ground set $\mathcal{E}$ has a set of labels (features) $\pi(e) \subset \mathcal{L}$, and the cost $f(S)$ of a set $S \subseteq \mathcal{E}$ is the cost of the labels of its elements: $f(S) = c(\bigcup_{e \in S} \pi(e))$. Labels are shared among several elements, and the cost of a set of labels L is additive: $c(L) = \sum_{\ell \in L} c(\ell)$. This function is an instance of the neighborhood functions defined above: in Figure 2.1, $\mathcal{V}_1$ are the elements and $\mathcal{V}_2$ the labels, and an element is connected to its labels. Label costs will enjoy special attention in Chapter 8.

In a network, the labels can correspond to transportation media or edges maintained by the same company, and choosing paths or trees that span few labels lowers the cost. That is, labels or classes implement fixed costs. Another application is reliable connectivity structures in networks: links do not break independently, but share physical resources or other common sources of failure. They belong to "Shared Risk Link Groups" [Yuan et al., 2005], modeled by common labels. Related ideas have surfaced in computer security. *Attack graphs* are state graphs modeling the steps of an intrusion, where transition edges are labeled by atomic actions. In such a graph, the minimum label cut indicates the lowest-cost prevention of an intrusion [Jha et al., 2002].

Finally, the Minimum Hitting Set of Bundles problem [Angel et al., 2009] is a label cost problem with multiple labels. For this problem, we are given a ground set $\mathcal{I}$ of items i, each with a cost $c(i)$, and bundles of items. In addition, there is a collection of sets S, and each S is a set of bundles b, as illustrated in Figure 2.2. The aim is to find the cheapest selection of items such that this selection covers at least one bundle in each S. In other words, we are trying to find a selection of bundles B, one bundle for each S, that has minimum cost $f(B) = \sum_{i \in b \text{ for a } b \in B} c(i)$. The latter is a label cost problem: assign to each bundle b a set of labels (items) $\pi(b) = \{i \mid i \in b\}$. Now we can write $f(B) = \sum_{i \in \bigcup_{b \in B} \pi(b)} c(i)$. Minimum Hitting Set of Bundles subsumes problems such as the multiple query optimization problem, where a set S is a query, its bundles are different plans of action that lead to answering the query, and the items in a bundle, i.e., plan, are the tasks necessary to execute that plan [Angel et al., 2009]. A low-cost set of bundles will answer all queries with the minimum number or cost of tasks,

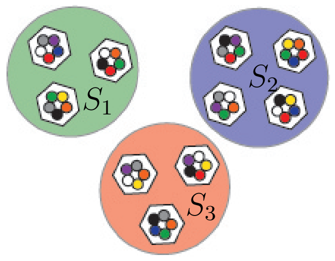


Figure 2.2 Minimum hitting set of bundles. The bundles are the hexagons, and one item is denoted by one small colored disk.

exploiting that some tasks are shared. A further special case is the Min-k-SAT problem.

Categorized bottleneck problems

Related to label cost problems are categorized bottleneck path problems [Averbakh and Berman, 1994], where the edges in a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ are partitioned into categories E_i. Each edge $e \in \mathcal{E}$ has a weight $w(e)$, and the cost of a path $P \subseteq \mathcal{E}$ is the sum of the maximal weights used from each category: $c(P) = \sum_i \max_{e \in P \cap E_i} w(e)$. This problem models "right of passage" or "usage" settings [Averbakh and Berman, 1994]. For example, the categories indicate different companies or means of transportation that operate the respective edges in that category. The companies charge customers via "day pass" tickets, and those tickets cost according to the longest edge used from the category. Similarly, the categorized bottleneck assignment problem [Aggarwal et al., 1986, Punnen, 2004, Seshan, 1981] derived from the sum assignment problem, the categorized bottleneck spanning tree problem [Richey and Punnen, 1992] and the categorized bottleneck traveling salesman problem [Punnen, 1992] are combinatorial problems with submodular costs.

Bottleneck problems for only one category have received attention in the literature, too, for example in the context of routing and communication network design [Hochbaum and Shmoys, 1986, Hochbaum and Pathria, 1996]. They also occur in computer vision, as we will see in Chapter 6.

Minimum power assignment or minimum-energy broadcasting

In wireless ad-hoc networks $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we seek a connectivity structure (e.g., a spanning tree) $S \subseteq \mathcal{E}$ that has minimum power requirement. The power consumption of a node depends on the most expensive edge it is using, $p(v|S) = \max_{e=(v,u) \in S} c(e)$, and the total cost is the sum of the node costs, $f(S) = \sum_{v \in V} p(v|S)$ [Calinescu et al., 2003, Wan et al., 2002, Wieselthier et al., 2000].

Stochastic optimization

In discrete mean-risk minimization, one aims to minimize a stochastic cost function over a collection of structures $\mathcal{S} \subseteq 2^{\mathcal{E}}$ while avoiding risks. The resulting optimiza-

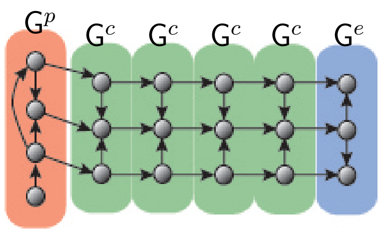


Figure 2.3. Illustration of a dynamic graphical model.

tion problems have cost functions of the form $f(S) = \sum_{e \in S} \mu_e + \Omega\sqrt{\sum_{e \in S} \sigma_e^2}$ — a submodular function. Instead of the square root, other concave functions can arise that yield submodular set functions [Atamtürk and Narayanan, 2008].

Separators in dynamic graphical models

One motivating application for the study of MinCoopCut was that of finding separators in dynamic graphical models. A graphical model $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ is a graph that defines a family of probability distributions. It has a node v_i for each random variable x_i, and any represented distribution $p(x)$ must factor with respect to the edges of the graph as $p(x) \propto \prod_{(\mathsf{v}_i,\mathsf{v}_j) \in \mathsf{E}} \psi_{ij}(x_i, x_j)$. A *dynamic graphical model* consists of three template parts: a prologue $\mathsf{G}^p = (\mathsf{V}^p, \mathsf{E}^p)$, a chunk $\mathsf{G}^c = (\mathsf{V}^c, \mathsf{E}^c)$ and an epilogue $\mathsf{G}^e = (\mathsf{V}^e, \mathsf{E}^e)$. Given a length τ, an *unrolling* of the template is a model that begins with G^p on the left, followed by $\tau + 1$ repetitions of the "chunk" part G^c and ending in the epilogue G^e. The prologue G^p may be connected to nodes from G^c, but not to G^e.

Obviously, the unrolled models are quite repetitive, as the same chunk is periodically repeated. When performing inference, such models are commonly cut into repeating segments, and then the segments are processed separately before merging the results, which means, for example, stitching together separately triangulated segments [Bilmes, 2010]. The separating interface are the nodes at the boundary that are connected to two segments and are included in one or both adjacent segments. The segments or "slices" form a junction tree, and they need not correspond to the chunks. In fact, the properties of the interfaces determine the complexity of inference (as those variables must become a clique).

The cost of inference grows with the size of the joint state space of the interface variables. A "small" separator corresponds to a minimum vertex cut in the graphical model, where the cost function measures the size of the joint state space. Vertex cuts can be rephrased as standard edge cuts. Often, a modular cost function suffices for good results. Sometimes, however, a more general cost function is needed, e.g., Bilmes and Bartels [2003] demonstrate that it can be beneficial to use a state space function that considers determinism between variables.

An example of a function that respects determinisms is the following. In a Bayesian network that has determinism, let D be the subset of fully deterministic nodes. That means any $x_i \in D$ is a deterministic function of the variables corresponding to its parent nodes $\mathrm{par}(\mathsf{v}_i)$: $p(x|x_{\mathrm{par}(\mathsf{v}_i)}) = \mathbf{1}[x_i = g(x_{\mathrm{par}(\mathsf{v}_i)})]$ for some function g. Let $\mathcal{D}_i$ be the state space of variable x_i. Furthermore, given a set A of variables, let $A_D = \{x_i \in A \cap D \mid \mathrm{par}(\mathsf{v}_i) \subseteq A\}$ be its subset of fully determined variables. If the state space of a deterministic variable is not restricted by fixing a subset of its parents, then the function measuring the state space of a set of variables A is $f(A) = \prod_{x_i \in A \setminus A_D} |\mathcal{D}_i|$. The logarithm of this function is a submodular function. This means we aim to find a vertex separator with minimum submodular cost, or, rephrased in terms of edges, a minimum cooperative cut.

More generally, as the same slicing can be used for parallel processing, we may seek a separator that requires little information to be transferred from one segment to another, i.e., from one processor to another. A good proxy for such "compressibility" might be entropy. The resulting optimization problem is again a cooperative cut. The algorithms in Chapter 4 apply to both entropy and the state space function for finding suitable efficient separators.

Chapter 3.

Hardness of MinCoopCut

Before exploring algorithms for solving the Minimum Cooperative Cut problem, we establish results on the hardness of the problem. We prove that MinCoopCut is NP-hard, and then show a lower bound on the approximation factor that puts the upper bounds in Chapter 4 in context. While the non-constant lower bound is the mathematically stronger result, the reduction for proving NP-hardness may be of interest for its construction.

3.1. Related hardness results

The introductory Chapter 2 already lists some hardness results for constrained submodular minimization. Here we review results for submodular minimization with combinatorial constraints.

Many constrained submodular problems are known to be NP-hard, and Theorem 3.1 adds MinCoopCut to the list of these problems. Moreover, several lower bounds have been shown recently: Iwata and Nagano [2009] consider covering constraints, and Goemans et al. [2009] study submodular-cost Minimum Spanning Tree, Perfect Matching, Shortest Path and Edge Cover. These lower bounds, summarized in Table 3.1, are mostly polynomial. In that respect, Theorem 3.2 fits well within the set of existing results. These results are worst-case results and hold for the entire class of nondecreasing submodular functions.

Special cases in the wide family of these non-modular combinatorial problems have been studied separately. The Minimum Hitting Set of Bundles problem can be formulated as a submodular-cost hitting set problem, as described in Section 2.5. This problem does not admit an approximation factor better than $N-1-\epsilon$ ($N \geq 3$), where N is the maximum number of bundles per set [Angel et al., 2009]. A further well-studied special case are label cost problems, many of which are also NP-hard. They are mostly neither constant-factor approximable, but, for certain constraint classes, admit logarithmic instead of polynomial approximation factors. Table 3.2 summarizes a selection of hardness results. In particular, Minimum Label Cut, a special case of MinCoopCut, has been shown to be NP-hard [Jha et al., 2002] via a reduction from Hitting Set. This result is an alternative proof to Theorem 3.1 below, however, the proof below illustrates the expressive richness of CoopCut via a different, new construction. Zhang et al. [2011] prove a lower

Problem	Lower Bound	Reference
Set Cover	$\Omega(\ln \lvert\mathcal{U}\rvert)$	Iwata and Nagano [2009]
Minimum Spanning Tree	$\Omega(n)$	Goel et al. [2009]
Shortest Path	$\Omega(n^{2/3})$	Goel et al. [2009]
Perfect Matching	$\Omega(n)$	Goel et al. [2009]
Minimum Cut	$\Omega(\sqrt{n})$	Theorem 3.2

Table 3.1. Hardness results for combinatorial problems with submodular costs, where n is the number of nodes, and $\mathcal{U}$ the universe to cover.

Problem	Lower bound	Reference
Min. Spanning Tree	$\Omega(\log n)$	Krumke and Wirth [1998]
Shortest Path	$2^{(\log m)^{1-(\log\log m)^{-c}}}$, $c < 0.5$	Zhang et al. [2011]
Perfect Matching ($K_{n,n}$)	$(0.5-\epsilon)\log n$, any $\epsilon > 0$	Monnot [2005]
Minimum cut	$2^{(\log m)^{1-(\log\log m)^{-c}}}$, $c < 0.5$	Zhang et al. [2011]

Table 3.2. Lower bounds for label cost problems in graphs with n nodes and m edges.

bound on the approximation factor for Minimum Label Cut. Their proof, a reduction from Label Cover, makes different assumptions and is in general different from the information-theoretic technique used to prove Theorem 3.2. We proved Theorems 3.1 and 3.2 before knowing of the results for label cuts.

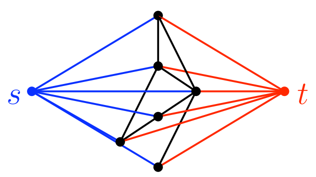
Finally, Goel et al. [2010] study hardness results in a multi-agent setting, where the cost of each agent is one discounted price function. Problems that are closer to covers, such as edge cover or spanning tree (a submodular cover), admit logarithmic approximation factors, whereas Perfect Matching and Shortest Path remain harder, with polylogarithmic lower bounds. This pattern resembles that of label costs, and the proof techniques too are similar to those used for label costs [Hassin et al., 2007]. Other restricted classes of (submodular) functions admit an FPTAS [Nikolova, 2010, Goyal and Ravi, 2008, Mittal and Schulz, 2012], even though the running times for these algorithms are in general not very practical either.

3.2. Minimum Cooperative (s,t)-Cut is NP-hard

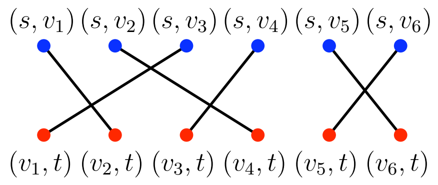
Theorem 3.1. *Minimum Cooperative (s,t)-Cut is NP-hard.*

The proof of this Theorem is a reduction from Graph Bisection, which is known to be NP-hard [Garey et al., 1976].

Definition 3.1 (Graph Bisection). Given a graph $\mathcal{G}_B = (\mathcal{V}_B, \mathcal{E}_B)$ with edge weights $w_B : \mathcal{E}_B \to \mathbb{R}_+$, find a partition $V_1 \dot\cup V_2 = \mathcal{V}_B$ with $|V_1| = |V_2| = |\mathcal{V}_B|/2$ with minimum cut weight $w(\delta(V_1))$.

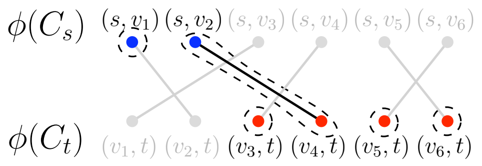


(a) graph $\mathcal{G}$ with $\mathcal{E}_s$ (blue), $\mathcal{E}_t$ (red) and $\mathcal{G}_B$ (black)



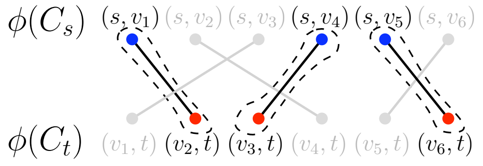
(s, v_1) (s, v_2) (s, v_3) (s, v_4) (s, v_5) (s, v_6)
(v_1, t) (v_2, t) (v_3, t) (v_4, t) (v_5, t) (v_6, t)

(b) graph $\mathcal{H}_\sigma$



$\phi(C_s)$
(s, v_1) (s, v_2) (s, v_3) (s, v_4) (s, v_5) (s, v_6)
$\phi(C_t)$
(v_1, t) (v_2, t) (v_3, t) (v_4, t) (v_5, t) (v_6, t)

(c) $h_\sigma(\phi(C)) = 5$ connected components



$\phi(C_s)$
(s, v_1) (s, v_2) (s, v_3) (s, v_4) (s, v_5) (s, v_6)
$\phi(C_t)$
(v_1, t) (v_2, t) (v_3, t) (v_4, t) (v_5, t) (v_6, t)

(d) balanced cut C: $h_\sigma(\phi(C)) = 3$ connected components

Figure 3.1. Graph for the reduction and examples for the definition of f_{bal} via ranks h_σ, with $n_B = 6$. In (c), $C_s = \{(s, v_1), (s, v_2)\}$ and $C_t = \{(v_3, t), (v_4, t), (v_5, t), (v_6, t)\}$; in (d), $C_s = \{(s, v_1), (s, v_4), (s, v_5)\}$ and $C_t = \{(v_2, t), (v_3, t), (v_6, t)\}$. Connected components are indicated by dashed lines.

Proof. To reduce Graph Bisection to MinCoopCut, we construct an auxiliary graph with two additional terminal nodes. The submodular edge weights on the edges adjacent to the terminal nodes will express the balance constraint $|V_1| = |V_2| = |\mathcal{V}_B|/2$. The edge weights on the instance for graph bisection remain unchanged. The proof involves three graphs: a given instance $\mathcal{G}_B$ of Graph Bisection, a graph $\mathcal{G}$ that has a cooperative cut cost function and represents the graph bisection instance $\mathcal{G}_B = (\mathcal{V}_B, \mathcal{E}_B)$, and a graph $\mathcal{H}_\sigma$ that defines the cost function on $\mathcal{G}$.

To form $\mathcal{G}$, we retain $\mathcal{G}_B$ with the modular costs on $\mathcal{E}_B$, add nodes s, t and connect those to every vertex in $\mathcal{G}_B$ with corresponding new edge sets $\mathcal{E}_s$ and $\mathcal{E}_t$. That means, $\mathcal{G} = (\mathcal{V}_B \cup \{s, t\}, \mathcal{E}_B \cup \mathcal{E}_s \cup \mathcal{E}_t)$, as Figure 3.1(a) shows. The cost of a cut in the auxiliary graph $\mathcal{G}$ is measured by the submodular function

$$f(C) = \sum_{e \in C \cap \mathcal{E}_B} w(e) + \beta f_{bal}(C \cap (\mathcal{E}_s \cup \mathcal{E}_t)), \tag{3.1}$$

where β is an appropriately large constant, and f_{bal} will be defined later. The cost f_{bal} on $\mathcal{E}_s \cup \mathcal{E}_t$ implements the equipartition constraint on $\mathcal{V}_B$. Obviously, any (s, t)-cut C must include at least $n_B = |\mathcal{V}_B|$ edges from $\mathcal{E}_s \cup \mathcal{E}_t$. A minimal cut assigns $v \in \mathcal{V}_B$ to t by cutting (s, v), and to s by cutting (v, t), and thus defines a partition of $\mathcal{V}_B$. As a result, the cardinality of $C_s = C \cap \mathcal{E}_s$ is the number of nodes in $\mathcal{V}_B$ assigned to t. An analogous equivalence holds for $C_t = C \cap \mathcal{E}_t$. In an equipartition, $|C_s| = |C_t| = n_B/2$.

We now implement the equipartition constraint by a submodular, nondecreasing cost on $\mathcal{E}_s \cup \mathcal{E}_t$. The function will be a sum of matroid rank functions h_σ. Each h_σ is based on a bipartite graph $\mathcal{H}_\sigma = (\mathcal{E}_s, \mathcal{E}_t, \mathcal{F}_\sigma)$ that has *nodes* $\mathcal{E}_s \cup \mathcal{E}_t$. Its edges $\mathcal{F}_\sigma$ form a derangement[1] σ between nodes from $\mathcal{E}_s$ and $\mathcal{E}_t$, as illustrated in Fig. 3.1(b). We denote by $\phi(C_s \cup C_t)$ the image of $C_s \cup C_t$ in the set of nodes of $\mathcal{H}_\sigma$. Let the rank function $h_\sigma : 2^{\phi(\mathcal{E}_s \cup \mathcal{E}_t)} \to \mathbb{N}_0$ count the number of connected components in the subgraph induced by the nodes $\phi(C_s \cup C_t)$. Figure 3.1 shows some examples. Each derangement on n_B items induces such a rank[2].

Let $\mathfrak{S}$ be the set of all derangements σ of n_B elements, i.e., all possible edge configurations in the graphs $\mathcal{H}_\sigma$. We define f_{bal} to be the expectation of h_σ if $\sigma \in \mathfrak{S}$ is drawn uniformly at random:

$$f_{bal}(C) = \mathbb{E}_\sigma[h_\sigma(\phi(C))] = |\mathfrak{S}|^{-1} \sum\nolimits_{\sigma \in \mathfrak{S}} h_\sigma(\phi(C)). \tag{3.2}$$

For a fixed derangement σ' and a fixed size $|C_s \cup C_t| = n_B$, the value $h_{\sigma'}(C_s \cup C_t)$ is minimal if the number of matched nodes is maximal. Then $\sigma'(C_s) = C_t$ and $|C_s| = |C_t|$.

To compute the rank $h_\sigma(C)$ for a fixed σ, we sum up all nodes $\phi(C_s \cup C_t) = |C_s| + |C_t|$. Then we subtract the number of matches, because those components were counted twice. To shorten notation, we denote the node (s, v_i) in $\mathcal{H}_\sigma$ by x_i, and its counterpart (v_i, t) by y_i. Formally, the rank is

$$h_\sigma(\phi(C_s) \cup \phi(C_t)) = |C_s| + |C_t| - \Big| \big\{(x_i, y_{\sigma(i)})\big\}_{i=1}^{n} \cap (\phi(C_s) \times \phi(C_t)) \Big|. \tag{3.3}$$

As an average of rank functions, f_{bal} is submodular and monotone. From Equation (3.3) it follows that the sum (3.2) consists of two terms:

$$\begin{aligned} \sum_{\sigma \in \mathfrak{S}} h_\sigma(C) &= |\mathfrak{S}|\big(|C_s| + |C_t|\big) - \sum_{\sigma \in \mathfrak{S}} \big|\{(x_i, y_{\sigma(i)})\}_{i=1}^{n} \cap (\phi(C_s) \times \phi(C_t))\big| && (3.4) \\ &= |\mathfrak{S}|\big(|C_s| + |C_t|\big) - \sum_{x_i \in \phi(C_s)} \sum_{\sigma \in \mathfrak{S}} \big|(x_i, y_{\sigma(i)}) \cap (\{x_i\} \times \phi(C_t))\big| && (3.5) \end{aligned}$$

That means we count the total number of matches as the sum of the number of matches for each x_i in $\phi(C_s)$. To count the matches of a fixed $x_i \in \phi(C_s)$, we calculate how many derangements map it to an element in $\phi(C_t)$ and yield a match.

When counting, we must regard that σ is a derangement, so there will never be an edge (x_i, y_i) in $\mathcal{H}_\sigma$. Let $C_{s \cap t} \triangleq \{(s, v) \mid \{(s, v), (v, t)\} \subseteq C\}$ be the set of s-edges whose counterpart on the t side is also contained in C. This set is nonempty if C cuts off a node from both s and t. Each element x_i in $\phi(C_s \setminus C_{s \cap t})$ can be mapped by σ to any element $y_k \in \phi(C_t)$. For each such (fixed) pairing (x_i, y_k), any of the

[1] A *derangement* is a permutation that maps no element to itself.

[2] In Section 2.3.3 we argue that this function is the rank of a partition matroid. Clearly, the edges in each derangement partition the set of nodes into sets of size 2.

remaining $n_B - 1$ elements x_j can be mapped to any y_ℓ with $j \neq \ell$. Moreover, the element x_k can be mapped to any remaining target in $\mathcal{E}_t$, since its counterpart y_k is already "taken" by x_i. Let $D'(n_B - 1)$ denote the number of permutations of $n_B - 1$ elements (pair (x_i, y_k), i.e., $\sigma(i) = k$, is fixed), where one specific element x_k can be mapped to any other of the $n_B - 1$ elements, and the remaining elements must not be mapped to their counterparts ($\sigma(j) \neq j$). Then there are $D'(n_B - 1)$ derangements σ realizing $\sigma(i) = k$, for each $y_k \in \phi(C_t)$. This makes $|C_t|D'(n_B - 1)$ matches for each x_i in $\phi(C_s \setminus C_{s\cap t})$, and so we count $|C_s \setminus C_{s\cap t}||C_t|D'(n_B - 1)$ matches in total for the $x_i \in C_s \setminus C_{s\cap t}$.

Each element x_i in the remaining $\phi(C_{s\cap t})$ can be mapped to $|C_t| - 1$ elements in $\phi(C_t)$, since its counterpart y_i is in $\phi(C_t)$. With a similar count as above, this leads to another $|C_{s\cap t}|(|C_t| - 1)D'(n_B - 1)$ matches. Let $D(n)$ be the number of derangements of n elements. In total, we get

$$\begin{aligned} D(n_B)f_{bal}(C) &= (|C_s| + |C_t|)D(n_B) \qquad (3.6)\\ &\quad - \sum_{x_i \in C_s \setminus C_{s\cap t}} \sum_{y_k \in C_t} D'(n_B - 1) - \sum_{x_i \in C_{s\cap t}} \sum_{y_k \in C_t, k \neq i} D'(n_B - 1) \\ &= (|C_s| + |C_t|)D(n_B) \qquad (3.7)\\ &\quad - \big(|C_s| - |C_{s\cap t}|\big)|C_t|D'(n_B - 1) - |C_{s\cap t}|(|C_t| - 1)D'(n_B - 1) \\ &= (|C_s| + |C_t|)D(n_B) - (|C_s||C_t| - |C_{s\cap t}|)D'(n_B - 1), \qquad (3.8) \end{aligned}$$

with $D(n) = |\mathfrak{S}| = n!\sum_{k=0}^{n}(-1)^k/k!$ [Stanley, 1997], and $D'(n-1) = \sum_{k=0}^{n-1}(n-2)!(n-1-k)!(-1)^k$ (derived in Appendix B.1). The derangements lead to the penalty $|C_{s\cap t}|$ for overlaps.

Given that $|C_s| + |C_t|$ must cut at least n_B edges and that f_{bal} is increasing, f_{bal} is minimized if $|C_s| = |C_t| = n_B/2$. In that case, $n_B/2$ nodes are assigned to s and $n_B/2$ to t. As a result, if β is large enough such that f_{bal} dominates the cost, then a minimum cooperative cut in $\mathcal{G}$ bisects the $\mathcal{G}_B$ subgraph of $\mathcal{G}$ optimally. □

3.3. Lower bound on the approximation factor

Beyond NP hardness, we show a lower bound on the approximation factor for MinCoopCut. The lower bound is information-theoretic and assumes oracle access to the cost function. For Theorem 3.1, in contrast, the cost function is completely known from the construction.

Theorem 3.2. *No polynomial-time algorithm can solve* MinCoopCut *with an approximation factor better than* $o(\sqrt{|\mathcal{V}|/\log|\mathcal{V}|})$.

Corollary 3.1. *For graphs where* $|\mathcal{E}| = \Theta(|\mathcal{V}|)$*, the lower bound can also be stated as* $\Omega(\sqrt{|\mathcal{E}|/\log|\mathcal{E}|})$.

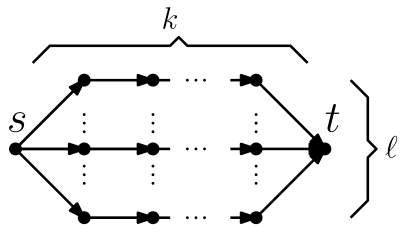


Figure 3.2. Graph for the proof of Theorem 3.2.

The proof relies on constructing two submodular cost functions f, h with different minima that are almost indistinguishable. In fact, with high probability they cannot be discriminated with a polynomial number of function queries. If the optima of h and f differ by a factor larger than α, then any solution for f within a factor α of the optimum would be enough evidence to discriminate f and h. As a result, a polynomial-time algorithm that guarantees an approximation factor α would lead to a contradiction. The proof technique is similar to that in [Goemans et al., 2009, Svitkina and Fleischer, 2008], and was first used by [Goemans et al., 2008].

One of the functions, f, depends on a hidden random set $R \subseteq E$ that will be its optimal cut. We will use the following Lemma that assumes f to depend on a random set R.

Lemma 3.1 (Svitkina and Fleischer [2008], Lemma 2.1)**.** *If for any set $Q \subseteq E$, chosen without knowledge of R, the probability of $f(Q) \neq h(Q)$ over the random R is $m^{-\omega(1)}$, then any algorithm that makes a polynomial number of oracle queries has probability at most $m^{-\omega(1)}$ of distinguishing f and h.*

Proof (Theorem 3.2). We will prove the bound in terms of the number $m = |\mathcal{E}|$ of edges in the graph. The graph we construct has $n = m - \ell$ nodes, and therefore the proof also shows the lower bound in terms of nodes.

Construct a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with ℓ parallel disjoint paths from s to t, where each path has k edges. Here, the random set $R \subset \mathcal{E}$ will always be a cut consisting of $|R| = \ell$ edges. The cut contains one edge from each path uniformly at random. We define $\beta = 8\ell/k < \ell$ (for $k > 8$), and, for any $Q \subseteq \mathcal{E}$,

$$h(Q) = \min\{|Q|, \ell\} \tag{3.9}$$

$$f(Q) = \min\{|Q \setminus R| + \min\{|Q \cap R|,\ \beta\},\ \ell\}. \tag{3.10}$$

The functions differ only for the relatively few sets Q with $|Q \cap R| > \beta$ and $|Q \setminus R| < \ell - \beta$. Define ε such that $\varepsilon^2 = \omega(\log m)$, and set $k = 8\sqrt{m}/\varepsilon$ and $\ell = \varepsilon\sqrt{m}$.

We compute the probability that f and h differ for a given query set Q. Probabilities are over the unknown R. Since $f \leq h$, the probability of difference is $P(f(Q) < h(Q))$. If $|Q| \leq \ell$, then $f(Q) < h(Q)$ only if $\beta < |Q \cap R|$, and the probability $P(f(Q) < h(Q)) = P(|Q \cap R| > \beta)$ increases as Q grows. If, on the other hand, $|Q| \geq \ell$, then the probability

$$P(f(Q) < h(Q)) = P(|Q \setminus R| + \min\{|Q \cap R|,\ \beta\} < \ell)$$

decreases as Q grows. Hence, the probability of difference is largest when $|Q| = \ell$.

So let $|Q| = \ell$. If Q spreads over $b \leq k$ edges of a path P, then the probability that Q includes the edge in $P \cap R$ is b/k. The expected overlap is the sum of hits on all paths, $\mathbb{E}[\ |Q \cap R|\ ] = |Q|/k = \ell/k$. Since the edges in R are independent across different paths, we bound the probability of a large intersection by a Chernoff bound, and Lemma 3.1 holds:

$$\begin{aligned} P\big(f(C) \neq h(C)\big) &\leq P\big(|C \cap R| \geq 8\ell/k\ \big) & (3.11) \\ &\leq 2^{-8\ell/k} = 2^{-\varepsilon^2} = 2^{-\omega(\log m)} = m^{-\omega(1)}. & (3.12) \end{aligned}$$

With this result, Lemma 3.1 applies. No polynomial-time algorithm can guarantee to distinguish f and h with high probability. A polynomial algorithm with approximation factor better than the ratio of optima $h(R)/f(R)$ would discriminate the two functions and thus lead to a contradiction. As a result, the lower bound is determined by the ratio of optima of h and f. The optimum of f is $f(R) = \beta$, and h has uniform cost ℓ for all minimal cuts. Hence, the ratio is $h(R)/f(R) = \ell/\beta = \sqrt{m}/\varepsilon = o(\sqrt{m/\log m})$.

For contradiction, assume there was an algorithm with approximation factor $\alpha = o(\sqrt{m/\log m})$. Set $\varepsilon = \sqrt{m}/(2\alpha)$, so $\varepsilon^2 = \omega(\log m)$ is satisfied. Given f for this ε, the algorithm would return a solution with cost at most $\alpha f(R) = \alpha\varepsilon^2 \leq \varepsilon\sqrt{m}/2 < \varepsilon\sqrt{m}$. If given the function h, it can only return a solution with strictly larger cost $\ell = \varepsilon\sqrt{m}$ and could thus distinguish f and h, contradicting Lemma 3.1. □

3.4. Discussion

The two theorems in this section prove that MinCoopCut is NP-hard and that it does not admit a constant-factor approximation in polynomial time. These results integrate well into the context of hardness results for related problems.

Strictly speaking, the lower bound in Theorem 3.2 implies that MinCoopCut cannot be solved exactly in polynomial time. Nevertheless, the reduction proving Theorem 3.1 employs submodularity in an unusual way, using large positive sums of partition matroids each defined based on derangements, and might therefore be of its own interest.

The bounds above are worst-case results, that means, there is an instance of MinCoopCut for which the approximation is not better than stated in Theorem 3.2. Nevertheless, the average case is better, as the empirical results in Section 4.4 will demonstrate. In fact, special cases of MinCoopCut are well-known to be polynomial-time solvable, such as the standard MINIMUM CUT problem. Another "easy" cost function is the bottleneck cost $f(C) = \max_{e \in C} w(e)$. Direct sums however can render simple problems hard. For instance, MINCOOPCUT is hard when using a sum of bottleneck cost functions defined by a partition $\{\mathcal{E}_i\}$, that is, $f(C) = \sum_i \max_{e \in C \cap E_i} w(e)$, as this includes MINIMUM LABEL CUT.

Apart from the simplicity of the cost function, the graph structure can play a role in the hardness. The simplest such example would be a graph that consists of one (s,t)-path. If all separators of the cost function are subsets of the neighborhoods δv of nodes $v \in \mathcal{V}$, then the approximation $\hat{f}_{pf}$ that will be introduced in Section 4.2.2 is exact and the cut problem is solvable as a polymatroidal network flow. Finally, COOPCUT is not NP-hard if the corresponding function on nodes, $g : 2^{\mathcal{V} \setminus t} \to \mathbb{R}_+$, $g(X) = f(\delta(X \cup s))$, remains a submodular function: any algorithm for general unconstrained submodular function minimization [Fujishige, 2005] applies. We explore the reverse side of this relation further in Chapter 7.

Chapter 4.

Approximation Algorithms

Given the NP-hardness of MinCoopCut, this chapter addresses algorithms for approximately solving the problem. After a literature review, we study approximation techniques and factors for cooperative cuts, and finally compare the derived algorithms empirically.

4.1. Techniques for approximations: an overview

Before focusing on cuts, we consider the general family of optimization problems

$$\min_{S \subseteq \mathcal{E}} \quad f(S) \quad \text{s.t. } S \in \mathcal{S}, \tag{4.1}$$

where f is a nondecreasing submodular function on subsets of $\mathcal{E}$, and $\mathcal{S} \subset 2^{\mathcal{E}}$ is a family of structures, such as all (s, t)-cuts, or all spanning trees, or all (s, t)-paths in a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Chapter 3 demonstrated that such problems are usually very hard. In contrast, if f is a modular function, Problem (4.1) is polynomial-time solvable or at least admits much better approximation factors. Similarly, unconstrained submodular minimization, where $\mathcal{S} = \mathcal{E}$, can be solved in polynomial time [Fujishige, 2005]. Thus, one could argue that the combination of a non-modular cost function and nontrivial constraints makes Problem (4.1) hard. We structure the approximation algorithms for this problem into two classes, depending on which of the two complications, cost function or constraints, they simplify. Constraints can be simplified by relaxations, or by re-formulating the problem. In addition to these categories, the structure of covering constraints suits greedy algorithms. Throughout this chapter, we will assume that f is a nonnegative, nondecreasing and normalized submodular function.

As a side note, a different approach is the center of [Svitkina and Fleischer, 2008]: they present sampling-based algorithms that fit in neither of the above categories. Their techniques apply, for instance, to size constraints of partitions.

Certain sub-classes of submodular functions admit better approximations than the general case. One example are the label cost functions defined in Section 2.5, where each item carries a label and the cost of a set of items S is the modular cost of the labels occurring on S. Algorithms for label costs use the specific structure and are outlined in Chapter 8 where we design an online algorithm for label cost problems. Related to the structure of label costs is that of multi-agent problems, where each agent has a simple cost function. These problems, like label costs, allow logarithmic instead of polynomial approximation factors for cover-type constraints [Goel et al., 2010]. Another special case is stochastic combinatorial optimization in the mean-risk model, with a cost function of the form $f(S) = \sum_{e \in S} w_1(e) + \sqrt{\sum_{e \in S} w_2(e)}$ that in many cases admits an FPTAS [Nikolova, 2010].

In the remainder of this chapter, we address algorithms for general nondecreasing submodular cost functions.

4.1.1. Approximations of the cost function

The first general option mentioned above is to replace the submodular function f by a "more tractable" function $\hat{f}$, and to solve

$$\min \hat{f}(S) \quad \text{s.t. } S \in \mathcal{S}. \tag{4.2}$$

This substitution is of course only useful if the resulting problem is easier to solve than Problem (4.1), i.e., in polynomial time. Let $\widehat{S}$ be a solution to Problem (4.2). If $\hat{f}$ is a "good" approximation, so that it does not deviate too much from f, then we can prove an approximation factor for $\widehat{S}$:

Lemma 4.1. *If for all $S \subseteq \mathcal{E}$, it holds that $f(S) \leq \hat{f}(S)$, and if for the optimal solution S^* to Problem (4.1), it holds that $\hat{f}(S^*) \leq \alpha f(S^*)$, and if $\widehat{S} = \operatorname{argmin}_{S \in \mathcal{S}} \hat{f}(S)$, then $\widehat{S}$ is at most by a factor α worse than S^* as a solution to Problem (4.1):*

$$f(\widehat{S}) \;\leq\; \alpha f(S^*).$$

Proof. Since $\hat{f}(\widehat{S}) \leq \hat{f}(S^*)$, it follows that $f(\widehat{S}) \leq \hat{f}(\widehat{S}) \leq \hat{f}(S^*) \leq \alpha f(S^*)$. □

The following two generic approximations $\hat{f}$ fill Lemma 4.1 with life. Specific approximations can use the special structure of the combinatorial problem; in Section 4.2.2, we derive such an approximation for cuts.

The simplest approximation

The simplest approximation $\hat{f}$ is the modular function

$$\hat{f}_{add}(S) = \sum_{e \in S} f(e). \tag{4.3}$$

By the subadditivity of the nondecreasing submodular function f, this approximation is an upper bound on f, that is, $\hat{f}_{add}(A) \geq f(A)$ for any $A \subseteq \mathcal{E}$. The approximation $\hat{f}_{add}$ is separable and therefore completely ignores any possible interaction between the elements. While this makes it an easy function to optimize, this simplicity strikes back via the factor α in Lemma 4.1, which can become rather large. This factor is

$$\alpha = \frac{\hat{f}(S^*)}{f(S^*)} = \frac{\sum_{e \in S^*} f(e)}{f(S^*)} \leq |S^*|, \tag{4.4}$$

and depends on how much the marginal cost $\rho(e|A)$ of any element $e \in S^*$ decreases as the reference set A grows. The upper bound $|S^*|$ refers to the worst case, where $f(e) = f(S^*)$ for all $e \in S^*$, that is, $\rho(S^* \setminus e|e) = 0$. In the best case, $\alpha = 1$. We will show a more detailed bound in Lemma 4.6.

Nevertheless, the approximation $\hat{f}_{add}$ leads to the best possible results for spanning trees and perfect matchings [Goel et al., 2009], and has also been used for submodular cover problems with submodular cost [Wan et al., 2010].

A generic approximation

The wide variance of α in the previous section raises the question whether $\hat{f}_{add}$ is the tightest possible approximation. This question has been studied by Goemans et al. [2009]. They show that if we require that $\hat{f}(A) \leq f(A) \leq \alpha \hat{f}(A)$ for all $A \subseteq \mathcal{E}$, then α is in general lower bounded as[1] $\Omega(\sqrt{m}/\log m)$. The paper also shows a generic approximation $\hat{f}_{ea}$ of a polymatroid rank function that is based on approximating a version of the submodular polyhedron by an ellipsoid. A nonnegative, nondecreasing submodular function is equivalent to a maximization over its polyhedron, i.e.,

$$f(S) = \max_{y \in P_f} y \cdot \chi_S. \tag{4.5}$$

Goemans et al. [2009] approximate a modified, symmetrized version of P_f by an ellipsoid E_f, and define the approximation

$$\hat{f}_{ea}(S) = \max_{y \in E_f} y \cdot \chi_S \tag{4.6}$$

$$= \sqrt{\sum\nolimits_{e \in S} w_f(e)}. \tag{4.7}$$

This approximation is a square root of a sum, which makes its square a simple function to optimize. Computing the weights w_f takes $O(m^3 \tau \log m)$ time, where τ is the time for finding a violated constraint, which is $O(m \log m)$ for a matroid rank function. The factor α is $O(\sqrt{m} \log m)$ in general, and $O(\sqrt{m})$ for matroid rank functions. We add that for an integer polymatroid rank function bounded by $M = \max_{e \in \mathcal{E}} f(e)$, the logarithmic factor can be replaced by a constant to yield

[1] Assuming f is given by an oracle.

$O(\sqrt{mM})$, if we approximate the matroid expansion[2] of the polymatroid instead of f directly.

4.1.2. Convex relaxation or re-formulation of constraints

Instead of simplifying the cost function, one can simplify the constraints. Perhaps the most straightforward option is to relax the discrete Problem (4.1) to a continuous problem with linear constraints. The crucial step for approximations is then the rounding of the optimal continuous solution to a discrete solution, i.e., a set. Alternatively, the problem can remain discrete, but the constraints are transformed to be more suitable for algorithms.

Relaxation using the Lovász extension

Instead of sets $S \subseteq \mathcal{E}$, we can equally use characteristic vectors $\chi_S \in \{0,1\}^{\mathcal{E}}$. With a slight abuse of notation, we define an equivalent cost function on $\{0,1\}^{\mathcal{E}}$ as $f(\chi_S) = f(S)$. For the relaxation, it must be possible that $\mathcal{S}$ can be described as a subspace of $\{0,1\}^{\mathcal{E}}$ by linear constraints $Ax \leq b$. (This is possible e.g. for (s,t)-cuts, simple paths or matchings.) In such case Problem 4.1 becomes

$$\min f(x) \quad \text{s.t. } Ax \leq b, \;\; x \in \{0,1\}^{\mathcal{E}}. \tag{4.8}$$

The Lovász extension helps relax this formulation to a continuous problem:

$$\min \tilde{f}(x) \quad \text{s.t. } Ax \leq b, \;\; x \in [0,1]^{\mathcal{E}}. \tag{4.9}$$

Such a relaxation has been used, for example, in [Iwata and Nagano, 2009, Chekuri and Ene, 2011a,b].

Problem (4.9) is a non-smooth convex minimization problem with linear constraints and can be solved by any suitable method. If merely polynomial time is the question, then we can use the ellipsoid method.

Given the optimal solution $x^* \in [0,1]^{\mathcal{E}}$ of Problem (4.9), the final task is to round x^* to a discrete $\hat{x} \in \{0,1\}^{\mathcal{E}}$ that is the characteristic vector of a set $S \in \mathcal{S}$. Iwata and Nagano [2009] exploit that their covering constraints are up-monotone, and find a threshold θ to select a feasible set $\widehat{S} = \{e \mid x^*(e) \geq \theta\}$ of all elements whose entries in x^* are at least θ.

Re-formulations

The following example illustrates an alternative to relaxing the integer constraints in Problem (4.8): reducing the problem to a different problem for which it is easier

[2]The expansion is described in Section 10.3 in [Narayanan, 1997]. In short, we replace each element e by a set $\hat{e}$ of $f(e)$ parallel elements. Thereby we extend f to a submodular function $\hat{f}$ on subsets of $\bigcup_i \hat{e}_i$. The desired rank function is now the convolution $r(\cdot) = \hat{f}(\cdot) * |\cdot|$ and it satisfies $f(S) = r(\bigcup_{e \in S} \hat{e})$.

to find an algorithm. The relaxation of a submodular-cost VERTEX COVER is special because it has a half-integral optimal solution [Iwata and Nagano, 2009, Goel et al., 2009, Hochbaum, 2010] which leads to a factor-2 approximation. After a transformation, this solution can be reached by a combinatorial algorithm too, in fact, a submodular minimization over a ring family.

Recall that VERTEX COVER seeks a minimum set of vertices such that each edge is incident with at least one vertex in this set. Iwata and Nagano [2009] first rephrase this as a vertex cover in a bipartite graph $\mathcal{G} = (\mathcal{V}^+, \mathcal{V}^-, \mathcal{F})$ that has two copies $\mathcal{V}^+, \mathcal{V}^-$ of the original set of nodes. For each original edge (v_i, v_j), there are two edges (v_i^+, v_j^-) and (v_j^+, v_i^-) in $\mathcal{F}$. If $(X^+, Y^-) \subseteq \mathcal{V}^+ \times \mathcal{V}^-$ is a vertex cover in $\mathcal{G}$, then $X \cup Y$ is a vertex cover in the original graph. The cost function on the bipartite graph is separable over the two copies of $\mathcal{V}$: $g(X^+, Y^-) = f(X) + f(Y)$. One can show that the minimum cost cover $(X^+, Y^-)^*$ in $\mathcal{G}$ corresponds to a solution of the relaxation, and therefore the union of these two parts is a factor-2 approximation for the original problem.

This observation is useful because a minimum cover in $\mathcal{G}$ can be found by minimizing the submodular function g over a ring family of closures[3], and this can be done with algorithms for unconstrained submodular minimization [Grötschel et al., 1988, Chapter 10].

Building on techniques for linear programs, Hochbaum [2010] extends the bipartite technique to general submodular minimization problems with constraints that use at most two variables, including complement of max clique, MINIMUM SATISFIABILITY, MIN-2SAT and the "submodular closure" problem. If the variables have the same sign, as for vertex cover, the result will be a 2-approximation; if they have differing signs, it will be exact. To generalize the constraints, she uses techniques for *monotonizing* (get opposite signs for variables in a constraint) and *binarizing* (get binary coefficients) the problem. Finally, she proves persistency of the formulation, that is, if a variable is integral in the optimal solution of the relaxation, then it will retain its value in an optimal solution to the original problem.

Another example for a re-formulation or reduction are the algorithms for label cost problems that additionally use the structure of the cost function.

4.1.3. Greedy approximations

Covering constraints in general have the simplifying property that they are *monotone*, i.e., their constraint polyhedron is *up-monotone*: if S is feasible, then any superset $T \supseteq S$ is also feasible. That means choosing more elements than necessary is not harmful for feasibility. This property not only enables the threshold rounding in [Iwata and Nagano, 2009], it also forms the basis for a greedy strategy. Let γ be the maximum number of variables occurring in any constraint. Koufogiannakis and Young [2009] present a greedy algorithm that yields a γ-approximation for any

[3]A ring family is a family of sets that is closed under union and intersection

algorithm	upper bound on approximation factor
simple approximation $\hat{f}_{add}$	$\frac{\lvert C^*\rvert}{1+(\lvert C^*\rvert-1)\nu(C^*)} \leq \lvert C^*\rvert = O(m)$
approximation by [Goemans et al., 2009]	$O(\sqrt{m}\log m)$
partition $\hat{f}_{pf}$	$\min\{\Delta_s, \Delta_t\} \leq n/2$
iterative approximation	$\frac{\lvert C^*\rvert}{1+(\lvert C^*\rvert-1)\nu(C^*)} \leq \lvert C^*\rvert = O(m)$
randomized greedy	$\lvert P_{\max}\rvert \leq n-1$
relaxation & thresholding	$\lvert P_{\max}\rvert \leq n-1$

Table 4.1. Overview of approximation factors for MinCoopCut in a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $n = |\mathcal{V}|$ nodes and $m = |\mathcal{E}|$ edges. The factor for the iterative approximation is function-dependent and varies between 1 and $|C^*|$.

submodular minimization problem with linear monotone constraints. For vertex cover, this leads to the same approximation factor 2 as above.

Their basic algorithm works with continuous variables; for strictly discrete problems, there is a randomized discrete equivalent. The algorithm considers the violated constraints one by one, and increases the value of the variables in the selected constraint by a fixed amount such that the constraint is less violated while the cost is not increased too much. Since the constraints are monotone, increasing a variable more than the minimum necessary will never lead to an infeasible solution.

4.2. Approximations for cuts: cost function

Backed by the summary of general approximation techniques, let us turn to approximations for MinCoopCut, which have not been addressed in the literature before. Both general strategies, simplifying the cost function as well as simplifying the constraints, apply to cuts, and the approximation algorithms in the sequel will be structured into these categories too. Table 4.1 summarizes the algorithms. Unless specifically stated, the algorithms apply to both directed and undirected graphs.

We provide three approximations of the cost function: (1) the generic ellipsoid-based function $\hat{f}_{ea}$ of Section 4.1.1; (2) an approximation that uses the structure of the problem and that could in this respect be seen closer conceptually to the bipartite graph reductions and $\hat{f}_{bip}$ in Section 4.1.2 than the generic approximations in Section 4.1.1; (3) an efficient iterative approximation that approximates a cooperative cut by a series of graph cut problems.

To apply the second strategy and simplify the constraint of S being a cut, we relate cuts to covers and thereby relax the cut polytope to be up-monotone. This relation leads to a rounding technique that is used with a convex relaxation, and to a reduction to cover problems and a greedy algorithm.

We begin with relaxing the cost function, and afterwards describe how to relax constraints. When talking about cuts specifically, we will refer to the feasible set as the family $\mathcal{C}$ of all minimal (s,t)-cuts $C \subseteq \mathcal{E}$, and to the optimal cut as C^*.

The simplest approximation to use with cuts is $\hat{f}_{add}$; however, the approximation factor can then become as large as $\alpha = |C^*| = O(m)$, which is $O(n^2)$ in very dense graphs. Section 4.4.2 will show an example graph where the approximation factor is indeed of order n^2.

4.2.1. Generic approximation

A more accurate approximation than $\hat{f}_{add}$ is the function $\hat{f}_{ea}$ defined in Equation (4.7) [Goemans et al., 2009]. When using $\hat{f}_{ea}$, we compute a minimum cut for the cost $\hat{f}_{ea}^2$ – a standard, efficiently computable minimum (s,t)-cut. In practice, the bottleneck therefore does not lie in computing the cut, but in computing the weights w_f for the approximation $\hat{f}_{ea}$.

Lemma 4.2. *Let $\widehat{C} = \operatorname{argmin}_{C\in\mathcal{C}} \hat{f}_{ea}(C)$ be the minimum cut for cost $\hat{f}_{ea}$, and $C^* = \operatorname{argmin}_{C\in\mathcal{C}} f(C)$. Then $f(\widehat{C}) = O(\sqrt{m}\log m)f(C^*)$. If f is integer-valued and we approximate its matroid expansion, then $f(\widehat{C}) = O(\sqrt{mM})f(C^*)$, where $M \leq \max_e f(e)$.*

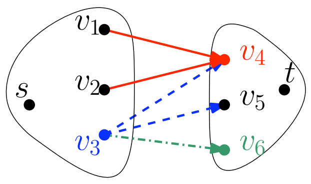
Lemma 4.2 follows from Lemma 4.1 and the discussion above. Given the lower bound in Theorem 3.2, for sparse graphs the bound in Lemma 4.2 is tight up to logarithmic factors.

4.2.2. A structural, locally exact approximation

Alternative to general approximations that might be expensive to compute, we can use the graph structure to define an approximation. We first address MinCoopCut in directed graphs. Note that Problem (4.1) is hard because f is globally non-separable: the cost of two edges e_1, e_2 can interact, $f(\{e_1, e_2\}) \ll f(e_1) + f(e_2)$, even if they are arbitrarily remotely located in the graph. In contrast, the cost function of the standard, efficiently solvable minimum cut is a *separable* sum of weights.

Therefore, we design $\hat{f}$ to be globally separable, but locally a tight approximation. To measure the cost of an edge set $C \subseteq \mathcal{E}$, we partition C into groups $\Pi(C) = \{C_v^\Pi\}_{v\in V}$, where the edges in C_v^Π must be incident to v (C_v^Π may be empty). That is, we assign each edge either to its head or to its tail node, as illustrated in Figure 4.1. Let $\mathcal{P}_C$ be the family of all such partitions (which vary in the head or tail assignment of each edge). We define an approximation

$$\hat{f}_{pf}(C) = \min_{\Pi(C)\in\mathcal{P}_C} \sum\nolimits_{v\in V} f(C_v^\Pi) \tag{4.10}$$



$$\begin{aligned}\hat{f}_{pf}(C) =& f(\{(v_1,v_4),(v_2,v_4)\})\\ &+ f(\{(v_3,v_4),(v_3,v_5)\})\\ &+ f(\{(v_3,v_6)\})\end{aligned}$$

Figure 4.1. Approximation of a cut cost. Red edges are in $C^{\Pi}_{v_4}$ (head), blue dashed edges in $C^{\Pi}_{v_3}$ (tail), and the green dash-dotted edge in $C^{\Pi}_{v_6}$ (head).

that decomposes across node neighborhoods, but is accurate within a group C^{Π}_v. Thanks to the subadditivity of f, the function $\hat{f}_{pf}$ is an upper bound on f. It always is the tightest approximation that is a direct sum over any partition in $\mathcal{P}(C)$. Instead of Problem (4.1), we now solve the optimization problem

$$\min \hat{f}_{pf}(C) \quad \text{s.t. } C \subseteq \mathcal{E} \ \text{ is a minimal } (s,t)\text{-cut.} \tag{4.11}$$

To solve Problem (4.11) exactly, we use its analogy to a generalized maximum flow problem. This analogy only holds for cuts, but that suffices here. We first introduce the flow problem.

Polymatroidal network flows

Polymatroidal network flows [Lawler and Martel, 1982, Hassin, 1982] generalize the capacity constraint of traditional flow problems. In the standard flow formulations, a function $\varphi : \mathcal{E} \to \mathbb{R}_+$ is a flow if the inflow at each node $v \in \mathcal{V} \setminus \{s,t\}$ equals the outflow, and if the flow on an edge does not exceed its capacity: $\varphi(e) \leq \text{cap}(e)$ for all $e \in \mathcal{E}$, given a capacity function $\text{cap} : \mathcal{E} \to \mathbb{R}_+$. Polymatroidal flows replace the usual additive capacities by submodular ones at each node v: cap^{in}_v for incoming edges, and $\text{cap}^{\text{out}}_v$ for outgoing edges. Let $\delta^- v$ be the incoming edges of v, and $\delta^+ v$ its outgoing edges. Then the capacity constraints at each $v \in \mathcal{V}$ are

$$\begin{aligned}\varphi(A) &\leq \text{cap}^{\text{in}}_v(A) \qquad \text{for all } A \subseteq \delta^- v,\\ \varphi(A) &\leq \text{cap}^{\text{out}}_v(A) \qquad \text{for all } A \subseteq \delta^+ v.\end{aligned}$$

Note that each edge (u,v) belongs to two neighborhoods, $\delta^+ u$ and $\delta^- v$. The maximum flow with such constraints is solved exactly in polynomial time by a layered augmenting paths algorithm [Tardos et al., 1986]. This algorithm involves submodular function minimization (SFM) only on the sets $\delta^+ v$, $\delta^- v$ that are in general much smaller than $\mathcal{E}$. It takes time $O(m^4 T)$, where T is the time for SFM on any $\delta^+ v$, $\delta^- v$.

Analogy

The next lemma relates Problem (4.11) to polymatroidal flows. We will use the restriction $f|_A$ of the function f to a subset A. For ease of notation, we explicitly write restrictions here, but drop them later. We assume throughout that the desired cut is minimal, as adding edges can only increase its cost.

Lemma 4.3. *Minimum (s,t)-cut with cost function $\hat{f}_{pf}$ is dual to a polymatroidal network flow with capacities* $\mathrm{cap}_v^{in} = f|_{\delta^- v}$ *and* $\mathrm{cap}_v^{out} = f|_{\delta^+ v}$ *at each node* $v \in \mathcal{V}$.

Proof. First, we state the dual of a polymatroidal flow. Let $\mathrm{cap}^{\mathrm{in}} : 2^{\mathcal{E}} \to \mathbb{R}_+$ be the joint incoming capacity, $\mathrm{cap}^{\mathrm{in}}(C) = \sum_{v \in V} \mathrm{cap}_v^{\mathrm{in}}(C \cap \delta^- v)$, and let equivalently $\mathrm{cap}^{\mathrm{out}}$ be the joint outgoing capacity. The dual of the polymatroidal maximum flow is a minimum cut problem whose cost is a convolution of edge capacities [Lovász, 1983]:

$$\mathrm{cap}(C) = (\mathrm{cap}^{\mathrm{in}} * \mathrm{cap}^{\mathrm{out}})(C) \triangleq \min_{A \subseteq C} \mathrm{cap}^{\mathrm{in}}(A) + \mathrm{cap}^{\mathrm{out}}(C \setminus A). \tag{4.12}$$

We will relate this dual to the approximation $\hat{f}_{pf}$. Given a minimal (s,t)-cut C, let $\Pi(C)$ be a partition of C, and $C_v^{\mathrm{in}} = C_v^{\Pi} \cap \delta_v^-$ and $C_v^{\mathrm{out}} = C_v^{\Pi} \cap \delta_v^+$. The cut C partitions the nodes into two sets $\mathcal{V}_s$ containing s and $\mathcal{V}_t$ containing t. Since C is a minimal directed cut, it contains only edges from the s side $\mathcal{V}_s$ to the t side $\mathcal{V}_t$ of the graph. In consequence, $C_v^{\mathrm{in}} = \emptyset$ if v is on the s side, and $C_v^{\mathrm{out}} = \emptyset$ otherwise. Hence, $C_v^{\mathrm{in}} \cup C_v^{\mathrm{out}}$ is equal to either C_v^{in} or C_v^{out}, and since $f(\emptyset) = 0$, it holds that $f(C_v^{\mathrm{in}} \cup C_v^{\mathrm{out}}) = f(C_v^{\mathrm{in}}) + f(C_v^{\mathrm{out}})$. Then, starting with the definition of $\hat{f}_{pf}$,

$$\begin{align}
\hat{f}_{pf}(C) &= \min_{\Pi(C) \in \mathcal{P}_C} \sum\nolimits_{v \in \mathcal{V}} f(C_v^{\Pi}) \tag{4.13}\\
&= \min_{\Pi(C) \in \mathcal{P}_C} \sum\nolimits_{v \in \mathcal{V}} f(C_v^{\mathrm{in}} \cup C_v^{\mathrm{out}}) \tag{4.14}\\
&= \min_{\Pi(C) \in \mathcal{P}_C} \sum\nolimits_{v \in \mathcal{V}} (f(C_v^{\mathrm{in}}) + f(C_v^{\mathrm{out}})) \tag{4.15}\\
&= \min_{\Pi(C) \in \mathcal{P}_C} \sum\nolimits_{v \in \mathcal{V}} (\mathrm{cap}_v^{\mathrm{in}}(C_v^{\mathrm{in}}) + \mathrm{cap}_v^{\mathrm{out}}(C_v^{\mathrm{out}})) \tag{4.16}\\
&= \min_{C^{\mathrm{in}}, C^{\mathrm{out}}} (\mathrm{cap}^{\mathrm{in}}(C^{\mathrm{in}}) + \mathrm{cap}^{\mathrm{out}}(C^{\mathrm{out}})) \tag{4.17}\\
&= \min_{C^{\mathrm{in}} \subseteq C} (\mathrm{cap}^{\mathrm{in}}(C^{\mathrm{in}}) + \mathrm{cap}^{\mathrm{out}}(C \setminus C^{\mathrm{in}})) \tag{4.18}\\
&= (\mathrm{cap}^{\mathrm{in}} * \mathrm{cap}^{\mathrm{out}})(C). \tag{4.19}
\end{align}$$

The minimum in Equation (4.15) is taken over all feasible partitions $\Pi(C)$ and their resulting intersections with the sets $\delta^+ v$,$\delta^- v$. Then we use the notation $C^{\mathrm{in}} = \bigcup_{v \in \mathcal{V}} C_v^{\mathrm{in}}$ for all edges assigned to their head nodes, and $C^{\mathrm{out}} = \bigcup_{v \in \mathcal{V}} C_v^{\mathrm{out}}$. The minima in Equations (4.17) and (4.18) are again taken over all partitions in $\mathcal{P}_C$. The final equality follows from the above definition of a convolution of submodular functions. □

Approximation factor

The previous subsection shows that Problem (4.11) can be solved exactly. With Lemma 4.1, we bound the approximation factor by a quantity that depends on the graph structure. Let C^* be the optimal cut for cost f. We define Δ_s to be the tail nodes of the edges in C^*: $\Delta_s = \{v \in \mathcal{V}_s \mid \exists (v,u) \in C^*\}$. These are still reachable from s if the edges is C^* are cut. Similarly, $\Delta_t = \{v \in \mathcal{V}_t \mid \exists (u,v) \in C^*\}$ contains all nodes on the t side that are the head of an edge in C^*.

Theorem 4.1. *Let $\widehat{C}$ be the minimum cut for cost $\hat{f}_{pf}$, and C^* the optimal cut for cost f. Then*

$$f(\widehat{C}) \;\leq\; \min\{|\Delta_s|, |\Delta_t|\}\, f(C^*) \;\leq\; \frac{|\mathcal{V}|}{2} f(C^*).$$

Proof. To apply Lemma 4.1, we need to show that $f(C) \leq \hat{f}_{pf}(C)$ for all $C \subseteq \mathcal{E}$, and find an α such that $\hat{f}_{pf}(C^*) \leq \alpha f(C^*)$. We already argued for the first condition using subadditivity. It remains to bound α. We do so by referring to the flow analogy with capacities set to f:

$$\begin{aligned}
\hat{f}_{pf}(C^*) &= (\mathrm{cap}^{\mathrm{in}} * \mathrm{cap}^{\mathrm{out}})(C^*) && (4.20)\\
&\leq \min\{\mathrm{cap}^{\mathrm{in}}(C^*),\ \mathrm{cap}^{\mathrm{out}}(C^*)\} && (4.21)\\
&\leq \min\Big\{ \textstyle\sum_{v \in \Delta_s} f(C^* \cap \delta^+ v),\ \textstyle\sum_{v \in \Delta_t} f(C^* \cap \delta^- v)\Big\} && (4.22)\\
&\leq \min\Big\{ |\Delta_s| \max_{v \in \Delta_s} f(C^* \cap \delta^+ v),\ |\Delta_t| \max_{v \in \Delta_t} f(C^* \cap \delta^- v)\Big\} && (4.23)\\
&\leq \min\big\{\, |\Delta_s|,\ |\Delta_t|\,\big\}\, f(C^*). && (4.24)
\end{aligned}$$

Thus, Lemma 4.1 implies an approximation bound $\alpha \leq \min\big\{\, |\Delta_s|,\ |\Delta_t|\,\big\} \leq |\mathcal{V}|/2$. □

Undirected graphs

Above, we solved a directed cooperative cut as a maximum polymatroidal network flow. Here, we argue that this construction is still correct for undirected graphs. An undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is first transformed into a directed graph $\mathcal{G}^{\pm} = (\mathcal{V}, \mathcal{E}^{\pm})$ by replacing each undirected edge e by two opposing directed edges e^+, e^- that are *parallel* with respect to the cost (the signs here are assigned arbitrarily). Let $f^{\pm}$ denote the cost equivalent of f on the directed edges. Edges e^+, e^- are *parallel* with respect to $f^{\pm}$ if

$$f^{\pm}(A \cup \{e^+\}) = f^{\pm}(A \cup \{e^-\}) = f^{\pm}(A \cup \{e^+, e^-\}) \qquad (4.25)$$

for all $A \subseteq \mathcal{E}$. The cost on the directed edges is equivalent to that on the undirected edges: parallelism yields that a set of directed edges has the same cost as the union of the undirected counterparts, regardless of whether both e^+ and e^- or only one of them is in the directed set. Such an $f^{\pm}$ can be defined as

$$f^{\pm}(A) = f(\{e \mid e^+ \in A \text{ or } e^- \in A\}). \qquad (4.26)$$

This function is a polymatroid rank function if f is, as can be shown e.g. via Proposition 2.1. If C is an undirected cut partitioning $\mathcal{V}$ into $\mathcal{V}_s$ and $\mathcal{V}_t$, then the cost of its directed counterpart is the same as $f(C)$: Let $C^+ = \delta^+(\mathcal{V}_s)$ and $C^- = \delta^-(\mathcal{V}_s)$ in the directed graph, then $f(C) = f^\pm(C^+) = f^\pm(C^-) = f^\pm(C^+ \cup C^-)$.

The remaining construction is as above. All capacity functions cap_v^{in}, $\text{cap}_v^{\text{out}}$ are set to $f^\pm$ restricted to the particular domain $\delta^- v$, $\delta^+ v$, respectively. The corresponding cap^{in}, cap^{out} decompose across neighborhood sets, and the convolution automatically assigns edges to their head or tail node to minimize the resulting cost, that is, to get the tightest approximation in the directed graph.

Doubts might arise because not every partition of directed edges retains the parallelism with respect to $f^\pm$. If e^+ and e^- are assigned to different neighborhood sets, then they are counted separately and thus doubly. Nevertheless, we claim the following.

Claim 4.1. *Minimizing $\hat{f}_{pf}^\pm$ via a* MAXIMUM POLYMATROIDAL NETWORK FLOW *is equivalent to finding an undirected minimum cut with respect to $\hat{f}_{pf}$.*

Proof. First, as argued above, the polymatroidal flow leads to a directed minimum cut with respect to $\hat{f}_{pf}^\pm$. The claim is true because $\hat{f}_{pf}$ and $\hat{f}_{pf}^\pm$ are equivalent on all cuts. The directed equivalent of a minimal undirected cut C is $C^+ = \delta^+\mathcal{V}_s$ for some $\mathcal{V}_s \subseteq \mathcal{V} \setminus t$. By definition, $\delta^+\mathcal{V}$ cannot contain both e^+ and e^-. Therefore, $\hat{f}_{pf}^\pm(C^+) = \hat{f}_{pf}(C)$ for all minimal cuts, and there is a one-to-one correspondence between directed and undirected cuts. This proves the claim. $\square$

For the optimal, maximal polymatroidal network flow, a similar observation to that in the proof holds. The flow on C^+ across the cut is maximal, and the back edges C^- are void [Lawler and Martel, 1982], analogously to a standard maximum flow. All edges in the corresponding minimum cut are full or blocking edges in the flow solution, and not void.

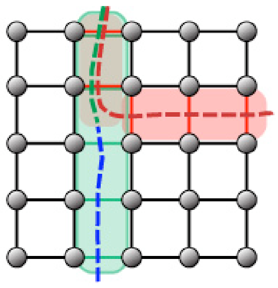
4.2.3. Iterative approximation

The simplest upper bound $\hat{f}_{add}$ on a submodular function ignores all coupling inherent in f. Here, we instead develop an *adjusting* bound that retains cooperation to some extent. As an advantage, both computing the bound and the corresponding minimum cut are very efficient. The bound requires a linear number of queries to the cost function, and the minimum cut is then a standard MINCUT computation.

The bound uses the *marginal cost* of an edge e with respect to a set $B \subseteq \mathcal{E}$, $\rho(e|B) = f(B \cup e) - f(B)$.

Lemma 4.4. *For a submodular function $f : 2^{\mathcal{E}} \to \mathbb{R}_+$, and an arbitrary $B \subseteq \mathcal{E}$, define $h_{B,f} : 2^{\mathcal{E}} \to \mathbb{R}_+$ as*

$$h_{B,f}(C) \triangleq f(B) + \sum_{e \in C \setminus B} \rho(e|B) - \sum_{e \in B \setminus C} \rho(e|\mathcal{E} \setminus e). \tag{4.27}$$



$$f(C) \leq f(B) + \sum_{e \in C \setminus B} \rho(e|B) - \sum_{e \in B \setminus C} \rho(e|\mathcal{E} \setminus e) =: \hat{f}_B(C)$$

Figure 4.2. Illustration of the adjusting upper bound. To estimate the cost of the red cut C from the given current green and blue cut B, we it subtract an estimate of the cost of the elements in $B \setminus C$ (blue) and add an estimate of the cost of $C \setminus B$ (red).

The function $h_{B,f}$ is a modular upper bound on f. The bound is tight at B, that means, $h_{B,f}(B) = f(B)$.

Proof. For any sets $C, B \subseteq \mathcal{E}$, it holds that [Nemhauser et al., 1978]

$$f(C) \leq f(B) + \sum_{e \in C \setminus B} \rho(e|B) - \sum_{e \in B \setminus C} \rho(e|(C \cup B) \setminus e). \tag{4.28}$$

From this inequality, Bound (4.27) follows because f has diminishing marginal costs: $\rho(e|\mathcal{E} \setminus e) \leq \rho(e|(B \cup C) \setminus e)$. Modularity and the tightness are immediate. □

This bound adds an upper bound on the cost of $C \setminus B$ and subtracts a lower bound on the cost of $B \setminus C$, as illustrated in Figure 4.2. Importantly, the cut cost $h_{B,f}$ is efficient to minimize using standard minimum cut, thanks to its modularity. For $\mathcal{G} = (\mathcal{V}, \mathcal{E}, f)$, define the graph $\mathcal{G}_B = (\mathcal{V}, \mathcal{E}, w_B)$ with the same structure but with additive edge weights

$$w_B(e) = \begin{cases} \rho(e|\mathcal{E} \setminus e) & \text{if } e \in B \\ \rho(e|B) & \text{otherwise.} \end{cases} \tag{4.29}$$

If f is nondecreasing, then the weights w_B are nonnegative.

Lemma 4.5. *If $\widehat{C} \in \operatorname{argmin}_{C \in \mathcal{C}} w_B(C)$ is a minimum (s,t)-cut in $\mathcal{G}_B$, then it is a minimizing cut, $\widehat{C} \in \operatorname{argmin}_{C \in \mathcal{C}} h_{B,f}(C)$, for the bound $h_{B,f}$.*

Proof. With weights w_B, the cost of a cut $C \subseteq \mathcal{E}$ in $\mathcal{G}_B$ is

$$\sum_{e \in C} w_B(e) = \sum_{e \in B \cap C} \rho(e|\mathcal{E} \setminus e) + \sum_{e \in C \setminus B} \rho(e|B) \tag{4.30}$$

$$= h_{B,f}(C) - f(B) + \sum_{e \in B} \rho(e|\mathcal{E} \setminus e). \tag{4.31}$$

Algorithm 1 Iterative bound minimization (ITB)

```
Input: $G = (\mathcal{V}, \mathcal{E})$; submodular cost $f$: $2^{\mathcal{E}} \to \mathbb{R}_0^+$; reference initialization set $\mathcal{I} = \{I_1, \dots, I_k\}$, $I_j \subseteq \mathcal{E}$; source and sink $s$, $t \in \mathcal{V}$
Output: cut $B \subseteq \mathcal{E}$
for $j = 1$ to $k$ do
    find $(s,t)$-mincut $C \in \operatorname{argmin}_{C \in \mathcal{C}} w_{I_j}(C)$ for edge weights $w_{I_j}$
    repeat
        $B_j = C$
        find $(s,t)$-mincut $C \in \operatorname{argmin}_{C \in \mathcal{C}} w_{B_j}(C)$ for edge weights $w_{B_j}$
    until $f(C) \geq f(B_j)$
end for
return $B = \operatorname{argmin}_{B_1,\dots,B_k} f(B_j)$
```

Since $f(B)$ and the sum over B are constant for a fixed B, it holds that $w_B(C) = h_{B,f}(C) + \mathit{const}$ for any edge set $C \subseteq \mathcal{E}$. □

Algorithm

As $h_{B,f}$ is adaptive, it can be used in an iterative minimization procedure (Algorithm 1). Given an initial reference set B, we find the minimum cut C with respect to $h_{B,f}$. Then we adjust the bound to be tight at C and repeat. Thus, $h_{B,f}$ is always tight at the currently best solution. The algorithm starts with an initial reference set $I_j \in \mathcal{I}$ from a set of initializations, the simplest case of which is $\mathcal{I} = \{\emptyset\}$. For further improvements, $\mathcal{I}$ could be set to the elements of a cut basis, e.g., the cuts induced by cutting edges of a spanning tree. The minimum cut basis in an undirected graph is the Gomory-Hu tree [Bunke et al., 2010]. For the experiments in Chapter 5, $\mathcal{I} = \{\emptyset\}$ was sufficient, and the algorithm converged in less than 10 iterations.

As a result of Lemma 4.5, the algorithm alternates between adjusting weights and computing a minimum cut. Implementation efficiency can be improved by noting that the marginal cost of an edge e depends only on edges that cooperate with e. The weights w_B show how $h_{B,f}$ captures the cost-reducing effect of f: $\rho(e|B) < f(e)$ if e cooperates with B. For a modular function $f = f_m$, the marginal cost $\rho(e|B) = f_m(e)$ never decreases and Algorithm 1 becomes a standard minimum cut[4] for edge weights $w(e) = f_m(e)$.

[4]Strictly speaking, the algorithm as displayed will run two iterations to ensure that the cost does not change. One iteration could be saved by checking whether the edge weights change at all.

Approximation bound

Lemma 4.6 gives an approximation bound for the case that $\emptyset \in \mathcal{I}$. It holds for the initial solution $C_\emptyset$ for $h_{\emptyset,f}$, which improves in subsequent iterations. The *total curvature* of a nondecreasing submodular function is defined as [Conforti and Cornuéjols, 1984]

$$\kappa_f = \max_{e \in \mathcal{E}} \frac{f(e) - \rho_f(e|\mathcal{E} \setminus e)}{f(e)}.$$

It indicates how close to modular a function is, with $0 \leq \kappa \leq 1$, and $\kappa = 0$ if and only if f is modular. We define the total curvature with respect to a set $A \subseteq \mathcal{E}$ as

$$\kappa_f(A) = \max_{e \in A} \frac{f(e) - \rho_f(e|A \setminus e)}{f(e)} \leq \kappa_f.$$

Lemma 4.6. *Let $C_\emptyset \in \operatorname{argmin}\{h_{\emptyset,f}(C) \mid C \subseteq \mathcal{E}$ an (s,t)-cut $\}$ be a minimum cut for $h_{\emptyset,f}$, and $C^* \in \operatorname{argmin}\{f(C) \mid C \subseteq \mathcal{E}$ an (s,t)-cut $\}$ an optimal solution. Let $\nu(C^*) = \min_{e \in C^*} \rho(e|C^* \setminus e) / \max_{e \in C^*} f(e)$. Then*

$$f(C_\emptyset) \leq \frac{|C^*|}{1 + (|C^*| - 1)\nu(C^*)} f(C^*) \quad \leq \quad |C^*| f(C^*). \tag{4.32}$$

Alternatively, the bound can be stated in terms of the curvature and $h_{\emptyset,f}(C^*)$ as

$$f(C_\emptyset) \leq \frac{f(e) + h_{\emptyset,f}(C^* \setminus e)}{f(e) + (1 - \kappa_f(C^*))\, h_{\emptyset,f}(C^* \setminus e)} f(C^*) \quad \leq \quad |C^*| f(C^*). \tag{4.33}$$

for any $e \in C^*$, in particular for $e' \in \operatorname{argmax}_{e \in C^*} f(e)$. If $\kappa_f(C^*) = 0$, then the approximation factor is 1, and if $\kappa_f(C^*) = 1$, then it is bounded by $|C^*|$.

Both approximation bounds in the lemma vary between 1 and $|C^*|$. The parameter ν is closely related to the "submodularity ratio" in [Das and Kempe, 2011] and also to curvature, which has been used to analyze the algorithms for maximizing a polymatroid rank function over a matroid [Conforti and Cornuéjols, 1984, Vondrák, 2008b].

Proof. We begin with the first statement and first bound $f(C_\emptyset)$ from above. Let $e' \in \operatorname{argmax}_{e \in C^*} f(e)$. The cost function f is subadditive, thanks to its submodularity and nonnegativity, and thus

$$f(C_\emptyset) \;\leq\; \sum_{e \in C_\emptyset} f(e) = h_{\emptyset,f}(C_\emptyset) \leq\; h_{\emptyset,f}(C^*). \tag{4.34}$$

The second inequality holds by the optimality of $C_\emptyset$ for $h_{\emptyset,f}$. Now, again using subadditivity, we bound

$$h_{\emptyset,f}(C^*) \leq \sum_{e \in C^*} f(e) \leq |C^*| f(e'). \tag{4.35}$$

Having the resulting upper bound $f(C_\emptyset) \leq |C^*|f(e')$, we derive a lower bound on $f(C^*)$. Diminishing marginal costs imply that

$$\begin{aligned} f(C^*) &\geq f(e') + \sum_{e \in C^* \setminus \{e'\}} \rho(e|C^* \setminus e) \\ &\geq f(e') + (|C^*| - 1) \min_{e \in C^*} \rho(e|C^* \setminus e). \end{aligned} \tag{4.36}$$

Using the upper bound on $f(C_\emptyset)$ and the lower bound on $f(C^*)$, we get

$$\frac{f(C_\emptyset)}{f(C^*)} \leq \frac{|C^*|f(e')}{f(e') + (|C^*| - 1) \min_{e \in C^*} \rho(e|C^* \setminus e)}. \tag{4.37}$$

Dividing by $f(e')$ yields the lemma:

$$f(C_\emptyset) \leq \frac{|C^*|}{1 + (|C^*| - 1) \min_{e \in C^*} \rho(e|C^* \setminus e)/f(e')}. \tag{4.38}$$

For the second statement of the approximation bound, we apply Lemma 4.1. First, observe that $\kappa_f(C^*) \geq 1 - \rho_f(e|C^*)/f(e)$ for all $e \in C^*$, and therefore

$$\rho_f(e|B) \geq (1 - \kappa_f(C^*))f(e) \tag{4.39}$$

for all $B \subseteq C^* \setminus e$ and $e \in C^*$. This inequality implies that

$$f(C^*) = f(e) + \rho_f(C^* \setminus e|e) \geq f(e) + (1 - \kappa_f(C^*))h_{\emptyset,f}(C^* \setminus e), \tag{4.40}$$

which, using $h_{\emptyset,f}(C^*) = f(e) + h_{\emptyset,f}(C^* \setminus e)$, implies the bound (4.33). If $\kappa_f(C^*) = 1$, then the factor is, using e',

$$h_{\emptyset,f}(C^*)/f(e') = \sum_{e \in C^*} f(e)/f(e') \leq |C^*|. \qquad \square$$

For the functions we use in Chapter 5, the term $\nu(C^*)$ is always nonzero and the second inequality is strict. Lemma 4.6 is a worst case bound and holds for *any* nondecreasing submodular f. We will see in Section 4.4 and Chapter 6 that in practice, the algorithm usually performs much better.

4.3. Approximations for cuts: Simplifying the constraints

Instead of relaxing the cost function, we can retain the cost and relax the constraints. We either relate the constraints to a simpler problem, here, a cover, or solve a continuous problem in place of a discrete one.

Algorithm 2 Greedy randomized path cover

```
**Input:** graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, terminal nodes $s, t \in \mathcal{V}$, cost function $f : 2^{\mathcal{E}} \to \mathbb{R}_+$.
$C = \emptyset$, $x = 0$
**while** $\sum_{e \in P_{\min}} x(e) < 1$ for the shortest path $P_{\min}$ **do**
  let $\beta \in (0, \min_{e \in P_{\min}} \rho(e|C)]$
  **for** $e$ in $P_{\min}$ **do**
    with probability $\beta / \rho(e|C)$, set $C = C \cup \{e\}$, $x(e) = 1$.
  **end for**
**end while**
prune $C$ to $C'$ and return $C'$
```

4.3.1. Greedy covering

First, we reduce a minimum (s, t)-cut to a cover problem, and then adapt an algorithm by Koufogiannakis and Young [2009]. An (s, t)-cut can equivalently be defined as a hitting set: a set of edges is a cut if and only if it "hits", i.e., intersects, each (s, t)-path in the graph. Equivalently, we can define that an edge set "covers" a path if it intersects it. Therefore, we write the MinCoopCut problem as

$$\begin{aligned} \min\ & f(x) \\ \text{s.t.}\ & \sum\nolimits_{e \in P} x(e) \geq 1 \quad \text{for all } (s,t)\text{-paths } P \subseteq \mathcal{E} \\ & x \in \{0,1\}^{\mathcal{E}}. \end{aligned} \tag{4.41}$$

Here, with a little abuse of notation, we use the same notation f for the set function and the equivalent pseudo-boolean function on binary indicator vectors, $f(\chi_A) = f(A)$. Note that the constraints in (4.41) do not require the cut to be *minimal*: a set is feasible if it has a subset that is a minimal cut. (Adding the cardinality $\epsilon|S|$ for very small ϵ to the cost function will however ensure that any optimal solution is minimal.) Relaxing this minimality constraint makes the feasible set monotone, and this property will be key to the algorithm. In fact, the constraints imply that Problem (4.41) is a minimum-cost cover problem.

Since a graph can have exponentially many (s, t)-paths, there can be exponentially many constraints. Luckily, all that will be required is finding a violated constraint, and this is possible in polynomial time. We compute the shortest path $P_{\min}$, using x as the (additive) edge lengths. If $P_{\min}$ is longer than one, then x is feasible. Otherwise, $P_{\min}$ defines a violated constraint.

Owing to the form of the constraints, we can adapt a randomized greedy cover algorithm [Koufogiannakis and Young, 2009] to Problem (4.41) and obtain Algorithm 2. In each step, we compute the shortest path with weights x to find a possibly uncovered path. Ties are resolved arbitrarily. To cover the path, we randomly pick edges from $P_{\min}$. The probability of picking edge e is inversely proportional to the marginal cost $\rho_f(e|C)$ of adding e to the current selection of edges[5]. We must also specify an appropriate β. With the maximum possible β we select the cheapest edge deterministically, and others randomly. Since C grows by at least one edge in each iteration, the algorithm terminates after at most m iterations.

Finally, the algorithm may return a set $\widehat{C}$ that is feasible but not a minimal cut. Then we prune $\widehat{C}$ to a minimal cut $\widehat{C}' \subseteq \widehat{C}$. Since f is nondecreasing, $f(\widehat{C}') \leq f(\widehat{C})$. Pruning can for example be done via breadth-first search. Let $\mathcal{V}_s$ be the set of nodes reachable from s after the edges in $\widehat{C}$ have been removed. Then we set $\widehat{C}' = \delta(\mathcal{V}_s)$. The set $\widehat{C}'$ must be a subset of $\widehat{C}$ because if there was an edge $(u, v) \in \widehat{C}' \setminus \widehat{C}$, then v would also be in $\mathcal{V}_s$, and then (u, v) cannot be in $\widehat{C}'$, a contradiction.

The last important question is the approximation bound. Lemma 4.7 implies that in expectation, Algorithm 2 returns at least an $O(n)$-approximation, because the longest path spans at most $|\mathcal{V}| - 1$ edges.

Lemma 4.7. *In expectation (over the probability of sampling edges), Algorithm 2 returns a solution $\widehat{C}'$ with $\mathbb{E}[f(\widehat{C}')] \leq |P_{\max}| f(C^*)$, where $P_{\max}$ is the longest simple (s, t)-path in $\mathcal{G}$.*

Proof. First, as f is nondecreasing, the pruned $\widehat{C}' \subseteq \widehat{C}$ can only be better than its superset $\widehat{C}$, that means, $f(\widehat{C}') \leq f(\widehat{C})$. By Theorem 7 in [Koufogiannakis and Young, 2009], a greedy randomized procedure like Algorithm 2 gives in expectation a γ-approximation for a cover, where γ is the maximum number of variables in any constraint. In Problem (4.41), γ is the maximum number of edges in any simple path, i.e., the length of the longest path. This implies that $\mathbb{E}[f(\widehat{C}')] \leq \mathbb{E}[f(\widehat{C})] \leq |P_{\max}| f(C^*)$. $\square$

The perils of being too greedy

Instead of sampling edges from each path, one could deterministically greedily pick one edge of minimum marginal cost from the uncovered path. A comparison to such an algorithm reveals the benefits of sampling: averaging over sampling identifies structure in the graph. If there is, for instance, a bottleneck edge that occurs in many paths, then this edge is more likely to be sampled for at least one of the paths, and it covers all of them together. The example in Section 4.4.2 illustrates that in such a case, focusing on the minimum-weight edges results in a cut that is by a factor $\frac{(n-1)^2}{4}(1 - \epsilon')$ more costly than the optimal cut, for any $\epsilon' > 0$. This factor is the cardinality of the returned cut, and asymptotically by a factor n worse than the length $n - 1$ of the longest path. The example shows that the following Lemma is tight.

Lemma 4.8. *For the solution $\widehat{C}$ returned by the greedy deterministic heuristic, it holds that $f(\widehat{C}) \leq |\widehat{C}| f(C^*)$.*

[5]If $\min_{e \in P_{\min}} \rho(e|C) = 0$, then we greedily pick all edges with zero marginal cost, because they do not increase the cost. Otherwise we sample as indicated in the algorithm.

Proof. To each edge $e \in \widehat{C}$ assign the path $P(e)$ which it was chosen to cover. By the nature of the algorithm, it must hold that $f(e) \leq f(C^* \cap P(e))$, because otherwise an edge in $C^* \cap P(e)$ would have been chosen. As C^* is a cut, the set $C^* \cap P(e)$ must be non-empty. These observations imply that

$$f(\widehat{C}) \leq \sum_{e \in \widehat{C}} f(e) \leq \sum_{e \in \widehat{C}} f(C^* \cap P(e)) \leq |\widehat{C}| \max_{e \in \widehat{C}} f(C^* \cap P(e)) \leq |\widehat{C}| f(C^*). \tag{4.42}$$

□

4.3.2. Relaxation

An alternative to the greedy randomized algorithm is to solve a relaxation of Problem (4.41). For the relaxation, we need to extend f from a set function to a function on a continuous domain. To do so, we view f as a function on binary indicator vectors, $\{0,1\}^{\mathcal{E}}$, and extend it to $[0,1]^{\mathcal{E}}$ via its *Lovász extension* $\tilde{f} : [0,1]^{\mathcal{E}} \to \mathbb{R}_+$,

$$\tilde{f}(x) = \max\nolimits_{y \in P_f} y \cdot x. \tag{4.43}$$

The maximization over the submodular polyhedron P_f takes $O(m \log m)$ time [Edmonds, 1970]. The Lovász extension is convex and piecewise linear. We substitute $\tilde{f}$ for f in Program (4.41), and allow $x \in [0,1]^{\mathcal{E}}$. The result is a non-smooth convex program with exponentially many constraints. These constraints can be summarized by the $m+1$ constraints of a standard linear program for minimum cut[6] [Papadimitriou and Steiglitz, 1998]:

$$\begin{aligned} \min \quad & \tilde{f}(x) \\ \text{s.t.} \quad & x(e) \geq \pi(v) - \pi(u) \;\; \forall (u,v) \in \mathcal{E} \\ & \pi(t) - \pi(s) \geq 1 \\ & \pi \in [0,1]^{\mathcal{V}},\; x \in [0,1]^{\mathcal{E}} \end{aligned} \tag{4.44}$$

The additional node variables π indicate membership of a node in the s side $\mathcal{V}_s$ ($\pi(v) = 0$) or t side $\mathcal{V}_t$ ($\pi(v) = 1$) of the cut. The constraints demand that an edge e from a label-zero node to a label-one node should be selected via $x(e) = 1$. These edges will eventually make up the cut. On closer inspection, the label $\pi(v)$ indicates the length of the shortest path from s to v, measured by additive distances x. Program (4.44) can be solved using any solver for non-smooth convex problems, or by adapting the approach in [Chudak and Nagano, 2007].

[6]Compared to the program in Section 2.4, the constraints here are inverted to have s carry label zero and t label one, as it fits the context better. The constraint sets are equivalent.

Algorithm 3 Rounding procedure given x^*

```
order $\mathcal{E}$ such that $x^*(e_1) \geq x^*(e_2) \geq \ldots \geq x^*(e_m)$
for $i = 1, \ldots, m$ do
    let $C_i = \{e_j \mid x^*(e_j) \geq x^*(e_i)\}$
    if $C_i$ is a cut then
        prune $C_i$ to $\widehat{C}$ and return $\widehat{C}$
    end if
end for
```

Rounding

The optimal solution to the nonlinear Program (4.44) is usually not integral, and must therefore be rounded to a discrete cut. The rounding procedure, shown in Algorithm 3, will determine the approximation guarantee. Let x^* be the optimal solution of Program (4.44). We use the values of $x^*(e)$ as a set of test thresholds θ_i in decreasing order (or we use binary search). If the set C_i of edges e with $x^*(e) \geq \theta_i$ contains a cut, we stop and prune C_i to a minimal cut. The pruning is exactly the same as for the greedy method in Section 4.3.1.

A faster, cruder rounding uses a threshold that is at most as large as the inverse of the length of the longest path in the graph (threshold $(n-1)^{-1}$ always works). The reason for this quantity becomes clear in the proof of the following bound.

Lemma 4.9. *Let $\widehat{C}$ be the rounded solution returned by Algorithm 3, $\theta = \theta_i$ the threshold at the last iteration i, and C^* the optimal cut. Then*

$$f(\widehat{C}) \;\leq\; \frac{1}{\theta} f(C^*) \;\leq\; |P_{\max}| f(C^*) \;\leq\; (n-1) f(C^*),$$

where $P_{\max}$ is the longest simple path in the graph.

Proof. Program (4.41) is a submodular program with covering constraints, which are up-monotone. Thus, thresholded rounding is possible, analogously to the case of cover problems [Iwata and Nagano, 2009], with an analogous approximation bound. Let θ be the rounding threshold that implied the final C_i. In the worst case, x^* is uniformly distributed along the longest path, i.e., $x^*(e) = |P_{\max}|^{-1}$ for all $e \in P_{\max}$ as x^* must sum to at least one along each path. Then θ must be $|P_{\max}|^{-1}$ to include at least one of the edges in $P_{\max}$. Since $\tilde{f}$ is nondecreasing like f and also positively homogeneous, it holds that

$$f(\widehat{C}) \leq f(C_i) = \tilde{f}(\chi_{C_i}) \;\leq \tilde{f}(\theta^{-1} x^*) = \theta^{-1} \tilde{f}(x^*) \leq \theta^{-1} \tilde{f}(\chi_{C^*}) = \theta^{-1} f(C^*). \tag{4.45}$$

The first inequality follows from monotonicity of f and the fact that $\widehat{C} \subseteq C_i$. Similarly, the relation between $\tilde{f}(\chi_{C_i})$ and $\tilde{f}(\theta^{-1} x^*)$ holds because $\tilde{f}$ is nondecreasing: by construction, $x^*(e) \geq \theta \chi_{C_i}(e)$ for all $e \in E$, and hence $\chi_{C_i}(e) \leq \theta^{-1} x^*(e)$. Finally, we use the optimality of x^* to relate the cost to $f(C^*)$; the vector χ_{C^*} is also feasible, but x^* optimal. The lemma follows since $\theta^{-1} \leq |P_{\max}|$. □

A note on the dual problem

Program (4.44) can equivalently be written by replacing the Lovász extension by the dual problem to the maximization in its definition (4.43). This dual is described e.g. in [Edmonds, 1970, Lovász, 1983], and contains a variable y_T for each subset $T \subseteq \mathcal{E}$ for the constraint $\sum_{e \in T} x(e) \leq f(T)$.

$$\begin{aligned}
\min_{x,\pi} \min_{y} \quad & \sum_{T \subseteq \mathcal{E}} f(T) y_T && (4.46)\\
\text{s.t.} \quad & \sum_{T: e \in T} y_T = x(e) \quad \forall e \in \mathcal{E} && (4.47)\\
\pi(u) - \pi(v) + x(e) &\geq 0 \quad \forall e = (u,v) \in \mathcal{E}\\
-\pi(s) + \pi(t) &\geq 1\\
0 \leq y &\leq 1\\
x &\geq 0
\end{aligned}$$

The dual (flow program) to this program is:

$$\begin{aligned}
\max_{\nu,\varphi} \quad & \nu && (4.48)\\
\text{s.t.} \quad & \varphi(T) \leq f(T) \quad \text{for all } T \subseteq \mathcal{E} && (4.49)\\
\sum_{e \in \delta^+ u} \varphi(e) - \sum_{e' \in \delta^- u} \varphi(e') &= d(u)\nu \quad \text{for all } u \in \mathcal{V}\\
\varphi &\geq 0.
\end{aligned}$$

The variables are $x \in \mathbb{R}^{\mathcal{E}}$ and $y \in R^{2^{\mathcal{E}}}$, $\nu \in \mathbb{R}$, $\varphi \in \mathbb{R}^{\mathcal{E}}$ and the constant $d(u) = 1$ if $u = s$, $d(u) = -1$ if $u = t$, and $d(u) = 0$ otherwise. The constraints involving x and π define x to be a cut. Constraint (4.47) transfers this to the chosen sets T. When comparing this pair of programs to the standard MaxFlow - MinCut pair in Section 2.4, one notes that the submodular cut cost transfers into the capacity constraints for the flow program. The flow problem closest to the non-relaxed MinCoopCut problem would be a maximum flow with capacity constraints $\varphi(\delta^+(T \cup s)) \leq f(\delta^+(T \cup s))$ for all $T \subseteq \mathcal{V} \setminus \{s,t\}$. While polymatroidal flows restrict the capacity constraints to local neighborhoods, the relaxation extends them to all sets, in particular also along paths.

4.4. An empirical comparison and worst cases

Having introduced a range of algorithms, we complement their theoretical analysis by an empirical comparison. The first data set is a benchmark of possible average-case cost functions. To our knowledge, no standard benchmark exists yet for submodular minimization and therefore the benchmark might be of interest on its own. The second set of experiments illustrates worst-case examples. These examples are specifically designed to test the limits of algorithms that approximate the cost by a modular function. In particular, they demonstrate the tightness of Lemma 4.6.

The task in the sequel is to find a minimum cooperative cut in an undirected graph. This problem can be solved directly or via $n-1$ minimum (s,t)-cuts. In most cases, the algorithms solve the (s,t) version. Note that the approximation bounds still apply, as the minimum cut is the minimum (s,t)-cut for at least one pair of source and sink. In general, the algorithms perform well, and much better than their theoretical worst-case bounds. Which algorithm is best depends on the cost function and graph at hand.

Algorithms and baseline methods

The experiments compare variants of the algorithms proposed in this chapter, and additional heuristics. All tested methods are listed in Table 4.2.

As a baseline, we show results for a minimum cut with the simplest approximation $\hat{f}_{add}$ (MC) defined in Equation (4.3), and for computing the minimum cut basis $\mathcal{C} = \{C_1, \dots, C_{n-1}\}$ and selecting $\widehat{C} = \operatorname{argmin}_{C_i \in \mathcal{C}} f(C_i)$ (MB). The minimum cut basis can be computed via a Gomory-Hu tree [Bunke et al., 2010]. None of these takes any coupling of edges into account when computing the cut. Comparing the iterative method to these baselines illustrates the effect of the adjusting upper bound, which does consider parts of the coupling. To test the effect of initializations, we initialize the iterative method with a random cut basis (RI) and the minimum-weight cut basis (MI). Given an initial cut, we compute the upper bound and a minimum cut with respect to this bound, and iterate. Together, these methods illustrate the effect of the iterative bounds, and the effect of initialization. Since the random basis of RI does not necessarily contain the minimum cut with respect to $\hat{f}_{add}$, Lemma 4.6 does not hold for this method.

Besides the iterative method, we test other approximations to the cost function, including the generic approximation by Goemans et al. [2009] (EA) followed by a minimum cut, and the local approximation solved via a polymatroidal maximum flow problem (PF).

The other presented algorithms address the constraints. We implemented the convex relaxation (CR) using Matlab's `fmincon` function. For the greedy covering algorithm, we include three variants. First, we test two different factors β (GM, GA). A larger β is more likely to sample more edges; the largest β used in GM always leads to including all minimum-weight edges of a path. We also test the deterministic variant that picks one minimum-cost edge from a non-covered path in each iteration and nothing else (GH). Non-minimal solutions are always pruned by computing a minimum cut.

Apart from the specific algorithms, we run another method that solves minimum cut for modular costs f. Queyranne's algorithm (QU) [Queyranne, 1998] minimizes symmetric submodular functions without constraints in $O(n^3)$ time. We apply

	approximating f
MC	Simple approximation $\hat{f}_{add}$ with standard **m**inimum **c**ut
RI	**I**terative approximation, initialized by a **r**andom cut basis (§4.2.3)
MI	**I**terative approximation, initialized by the **m**inimum cut basis with respect to $\hat{f}_{add}$ (§4.2.3)
EA	**E**llipsoid-based **a**pproximation $\hat{f}_{ea}$ of f (§4.2.1)
PF	Approximation $\hat{f}_{pf}$ of f via **p**olymatroidal network **f**lows (§4.2.2)
	simplifying the constraints
CR	**C**onvex **r**elaxation and rounding (§4.3.2)
GM	**G**reedy cover with **m**aximum β (§4.3.1)
GA	**G**reedy cover with **a**lmost maximum $\beta = 0.9\beta_{\max}$ (§4.3.1)
GH	**G**reedy **h**euristic: always pick a minimum marginal-weight edge
	comparison methods
QU	**Qu**eyranne's algorithm for minimizing symmetric submodular functions
MB	**M**inimum cut **b**asis for a modular approximation, no iterations

Table 4.2. Acronyms for the algorithms used in the experiments.

this algorithm to the induced function $g : 2^{\mathcal{V}} \to \mathbb{R}_+$ on *nodes*, $g(X) = f(\delta X)$. The function g is symmetric, but, as will be shown in Chapter 5, it is not in general submodular. Thus, Queyranne's algorithm can here at most be viewed as a heuristic that is not guaranteed to find the optimal solution. In fact, Section 4.4.2 reveals that it is impossible to provide any bounds on the solution returned by this heuristic. Nevertheless, the algorithm often does find low-cost solutions, possibly because the function g may often be close to submodular.

All algorithms were implemented in Matlab, with the help of a graph cut toolbox [Bagon, 2006, Boykov and Kolmogorov, 2004], and a toolbox for submodular function optimization [Krause, 2009].

4.4.1. Benchmark data for average cases

The benchmark data for cooperative cuts consists of two ingredients: (i) random graphs and (ii) a range of nondecreasing submodular cost functions. These functions by themselves can also serve as a benchmark for general submodular minimization. They can be turned into nondecreasing submodular functions by adding an appropriately scaled negative modular function.

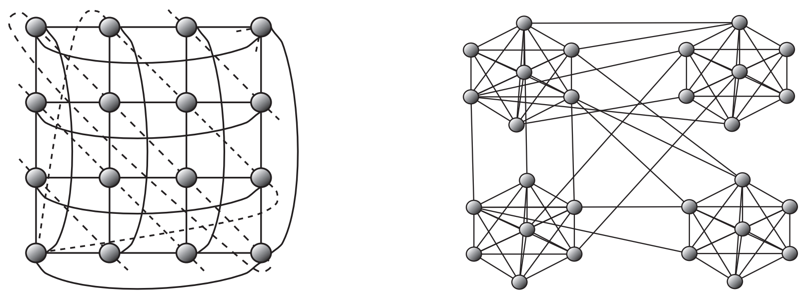

(a) Grids I and II (b) Clustered graph

Figure 4.3. Examples of our test graphs. The grid (a) was used with and without the dashed diagonal edges, and also with a variation of the connections in the first and last row. The clustered graphs were similar to the example shown in (b).

Graphs

We use two types of graph structures, regular graphs and clustered graphs.

Grid graphs. Three variants of regular grid graphs of degree four or six are used. Type I is a plane grid with horizontal and vertical edges displayed as solid edges in Figure 4.3(a). Type II is similar, but has additional diagonal edges (dashed in Figure 4.3(a)). Type III is a cube with plane square grids on four faces (sparing the top and bottom faces). Different from Type I, the nodes in the top row are connected to their counterparts on the opposite side of the cube. The connections of the bottom nodes are analogous.

Clustered graphs. The clustered graphs consist of a number of cliques that are connected to each other by few edges, as depicted in Figure 4.3(b).

We use the three grid types with 25 or 32 nodes and four random clustered graphs with 30 nodes and 90 edges. For each graph, we generate five random instances of each cost function.

Cost functions

The cost functions are listed in Table 4.3 and 4.4. The benchmark includes four families of functions. The first group are matroid rank functions or sums of three such functions. The functions used here are either based on matrix rank or ranks of partition matroids. We refer to those functions as *rank-like* costs.

Second, we use two variants of discounted price functions, one with a logarithm and one with a square root. These functions too are designed to favor a certain random optimal cut. This construction ensures that the minimum cut will not be one that separates out a single node, but one that cuts several edges.

The third family is constructed particularly to make a cut optimal that has many edges and that is therefore different from the cut that uses fewest edges. For such a cut, we expect $\hat{f}_{add}$ to yield relatively poor solutions.

name	description
matrix rank I	Each element $e \in \mathcal{E}$ indexes a column in matrix $\mathbf{X}$. The cost of $A \subseteq \mathcal{E}$ is the rank of the sub-matrix $\mathbf{X}_A$ of the columns indexed by the $e \in A$: $f_{\text{mrI}}(A) = \text{rank}(\mathbf{X}_A)$. The matrix $\mathbf{X}$ is of the form $[\, \mathbf{I}' \;\; \mathbf{R} \,]$, where $\mathbf{R} \in \{0,1\}^{d \times (m-d)}$ is a random binary matrix with $d = 0.9\sqrt{m}$, and $\mathbf{I}'$ is a column-wise permutation of the identity matrix.
(poly-)matrix rank II	$f_{\text{mrII}}(A) = 0.33 \sum_{i=1}^{3} f_{\text{mrI}}^{(i)}(A)$ sums up three functions $f_{\text{mrI}}^{(i)}$ of type *matrix rank I* with different random matrices $\mathbf{X}$.
labels I	$f_{\ell\text{I}}(A) = \lvert \bigcup_{e \in A} \ell(e) \rvert$. Each element e is assigned a random label $\ell(e)$ from a set of $0.8\sqrt{m}$ possible labels. The cost counts the number of labels in A.
labels II	$f_{\ell\text{II}}(A) = 0.33 \sum_{i=1}^{3} f_{\ell\text{I}}^{(i)}(A)$ is the sum of three functions of type *labels I* with different random labels.
discounted price function I	$f_{\text{dpI}}(A) = \log \sum_{e \in A} w(e)$, where weights $w(e)$ are chosen randomly as follows. Sample an $X \subset V$ with $\lvert X \rvert = 0.4n$, and set $w(e) = 1.001$ for all $e \in \delta X$. Then randomly assign some "heavy" weights in $[n/2, n^2/4]$ to some edges not in δX, so that each node is incident to one or two heavy edges. The remaining edges get random (mostly integer) weights between 1.001 and $n^2/4 - n + 1$.
discounted price function II	$f_{\text{dpII}}(A) = \sqrt{\sum_{e \in A} w(e)}$ with weights assigned as for "discounted price function I".
bestcut I	We randomly pick a connected subset $X^* \subseteq \mathcal{V}$ of size $0.4n$ and define the cost $f_{\text{bcI}}(A) = \mathbf{1}[\lvert A \cap \delta X^* \rvert \geq 1] + \sum_{e \in A \setminus \delta X^*} w(e)$. The edges in $\mathcal{E} \setminus \delta X^*$ are assigned random weights $w(e) \in [1.5, 2]$. If there is still a cut $C \neq \delta X^*$ with cost one or lower, we correct w by increasing the weight of one $e \in C$ to $w(e) = 2$. The optimal cut is then δX^*, but it is usually not the one with fewest edges.
bestcut II	Similar to *bestcut I* (δX^* is again optimal), but with submodularity on all edges: $\mathcal{E}$ is partitioned into three sets, $E = (\delta X^*) \cup B \cup C$. Then $f_{\text{bcII}}(A) = \mathbf{1}[\lvert A \cap \delta X^* \rvert \geq 1] + \sum_{e \in A \cap (B \cup C)} w(e) + \max_{e \in A \cap B} w(e) + \max_{e \in A \cap C} w(e)$. The weights of two edges in B and two edges in C are set to $w(e) \in (2.1, 2.2)$.

Table 4.3. Cost functions for the experiments in Section 4.4.1 (part I), $n = \lvert \mathcal{V} \rvert$, $m = \lvert \mathcal{E} \rvert$. "Matrix rank I, II" and "labels I, II" are summarized as "rank-like" costs in the results.

name	description
truncated rank	This function is similar to the truncated rank in the proof of the lower bound (Theorem 3.2). Sample a connected $X \subseteq \mathcal{V}$ with $\vert X\vert = 0.3\vert\mathcal{V}\vert$ and set $R = \delta X$. The cost is $f_{\text{tr}}(A) = \min\{\vert A \cap \overline{R}\vert + \min\{\vert A \cap R\vert, \lambda_1\},\ \lambda_2\}$ for $\lambda_1 = \sqrt{\vert R\vert}$ and $\lambda_2 = 2\vert R\vert$. Here, R is not necessarily the optimal cut.

Table 4.4. Cost functions for the experiments in Section 4.4.1 (part II), $n = |\mathcal{V}|$, $m = |\mathcal{E}|$. The indicator function is denoted by $\mathbf{1}[\cdot]$.

The fourth set of function is inspired by the difficult truncated functions that can be used to establish lower bounds on approximation factors. These functions “hide” an optimal set, and interactions are only visible when guessing a large enough part of this hidden set.

To estimate the approximation factor on one problem instance (one graph and one cost function), we divide by the cost of the best solution found by any of the algorithms, unless the optimal solution is known (this is the case for bestcut I and II).

Results

Figure 4.4 shows average empirical approximation factors and also the worst observed factors. The first observation is that all algorithms remain well below their theoretical approximation bounds[7]. That means, these bounds are really worst-case results. For several instances we obtain optimal solutions.

The performance of single algorithms varies across graphs and cost functions. Overall, the greedy algorithm GA performs well, as does the convex relaxation. Despite its varying bound, the iterative algorithm is competitive too. In particular, the adjusting bound improves on average over merely using $\hat{f}_{add}$ (MB, MC). The basis used for initialization, however, does not have a big effect.

As expected, a modular-cost estimate of the minimum cut via $\hat{f}_{add}$ performs relatively poorly for the *bestcut* functions. In addition, the *truncated rank* family appears to be the most challenging in that no algorithm always finds the optimal solution.

The generic ellipsoid-based approximation performs best for the square root *discounted price function*, because this function exactly fits the form of $\hat{f}_{ea}$. The generic ellipsoidal approximation couples elements uniformly via a square root of a sum, and therefore appears to be less suitable to capture the interaction of specific elements that occurs in the other cost functions.

[7]Most of the bounds proved above are absolute, and not asymptotic. The only exception is $\hat{f}_{ea}$. For simplicity, it is here treated as an absolute bound.

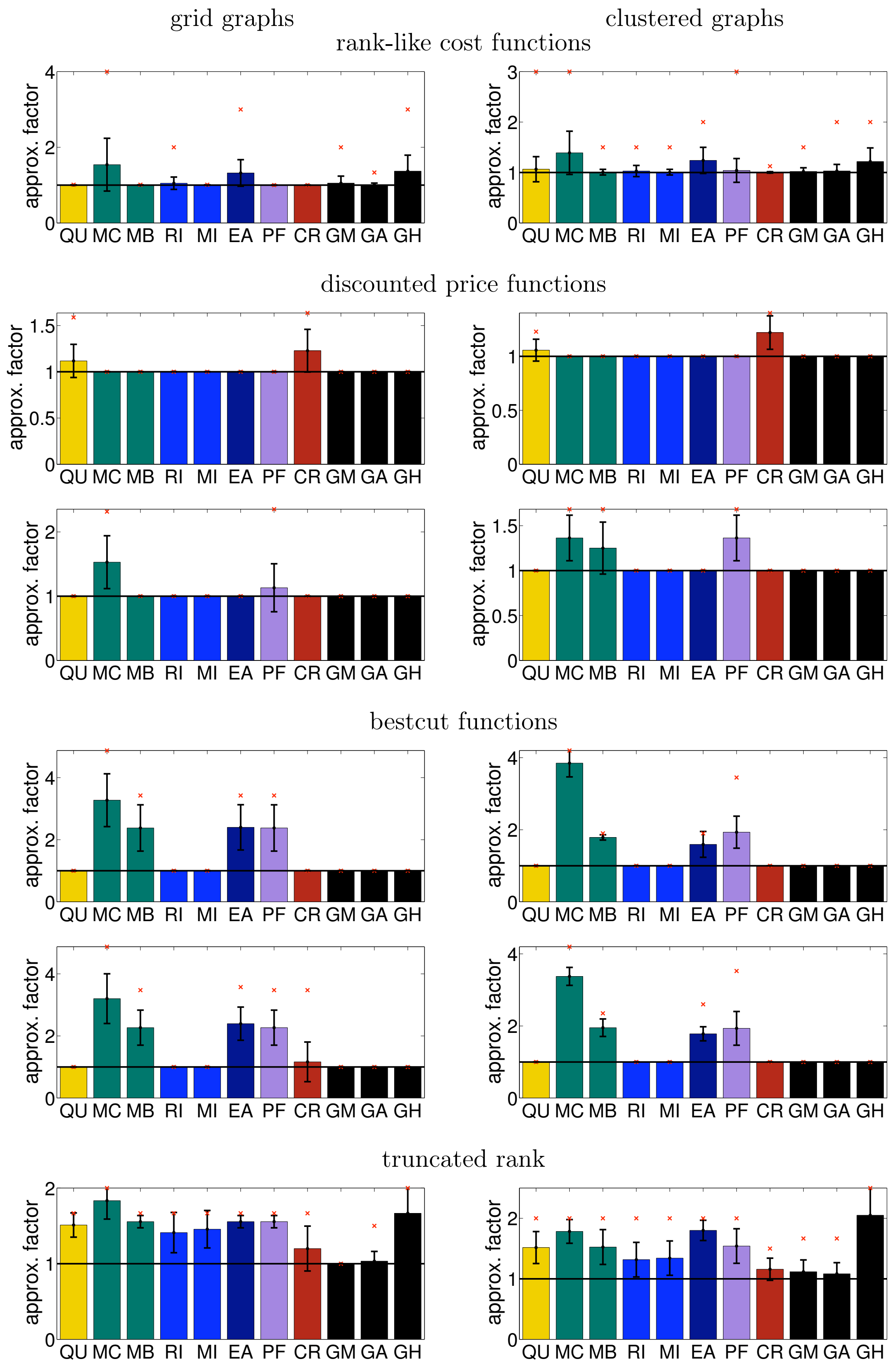


Figure 4.4. Results for the experiments of Section 4.4.1. The bars show the mean empirical approximation factors, and red crosses mark the maximum observed empirical approximation factor. The left column refers to grid graphs, the right column to clustered graphs.

Not surprisingly, selecting the maximal β for the randomized greedy algorithm is very similar to deterministically choosing the minimum-cost edge from each uncovered path. The difference is that the randomized algorithm can pick more edges, and, if there are multiple edges with minimum marginal cost, it will pick all. This makes a difference for the difficult truncated rank function, where interactions can only be discovered if a sufficient part of the random set is selected. In general, there is no big difference between the maximum β and a slightly smaller one. An exception are the *ranklike* functions, where many edges have the same marginal cost.

Even though it is only a heuristic here, Queyranne's algorithm performs quite well on average. This observation might indicate that several of the induced symmetric functions g still closely resemble submodular functions. The worst-case instances in the next section, however, will demonstrate the limits of this heuristic.

4.4.2. Worst-case examples

The previous section explored results for average-case instances. In this section, we study two sets of problem instances that are specifically crafted to mislead particular algorithms. The graphs in the sequel are again undirected.

Type I: difficult instances for purely modular approximations

The first example exploits the weakness of the modular approximation $\hat{f}_{add}(A) = \sum_{e \in A} f(e)$ to ignore the interaction of edges, that is, the subadditivity of f resulting from diminishing marginal costs, $f(A) \ll \hat{f}_{add}(A)$, for *certain* sets $A \subseteq \mathcal{E}$. We study two variants, (a) and (b).

In version (a), the modular cost of the true optimum C^* is $n^2/4$ times higher than its submodular cost, that is, $\hat{f}_{add}(C^*) = \frac{n^2}{4} f(C^*)$. Furthermore, we construct the cost f such that for the minimum cut $\widehat{C}$ for cost $\hat{f}_{add}$, in contrast, the true cost f is not much smaller than the estimate $\hat{f}_{add}$: $f(\widehat{C}) = \hat{f}_{add}(\widehat{C}) - \frac{n}{2} - 1$. As a result, $\hat{f}_{add}$ is not a good estimate for the relative costs $f(C^*)$ and $f(\widehat{C})$.

The graph structure, shown in Figure 4.5, completes the instance. The graph is a clique with three types of edges, marked by different colors. Let $\mathcal{E}_k$, $\mathcal{E}_b$, and $\mathcal{E}_r$ be the set of black, blue and red edges, respectively. The cost f_{Ia} is the direct sum of the cost functions f_k, f_b, f_r on these sets,

$$f_{Ia}(A) = f_k(A \cap \mathcal{E}_k) + f_b(A \cap \mathcal{E}_b) + f_r(A \cap \mathcal{E}_r) \tag{4.50}$$

with
$$\begin{aligned} f_k(A) &= 1 && \text{for all } A \subseteq \mathcal{E}_k; \\ f_b(A) &= \frac{n}{2}|A| && \text{for all } A \subseteq \mathcal{E}_b; \\ f_r(A) &= |A| \left(\frac{n}{2} - \frac{\epsilon}{n/2 - 1} \right) && \text{for all } A \subseteq \mathcal{E}_r \end{aligned}$$

for a small $\epsilon > 0$. The optimal cut is $C^* = \mathcal{E}_k$ and relies on the only but strongly submodular part of the cost function, f_k. The optimal cut for $\hat{f}_{add}$ separates out node $v_{n/2+1}$, cutting all red edges and the black edges adjacent to $v_{n/2+1}$, and has true cost $f_{Ia}(\delta v_{n/2+1}) = n^2/4 - n/2 + 1 - \epsilon$. Thus, the approximation factor for this cut grows as $n^2/4$, and follows the order $O(m)$ of the theoretical worst-case bound.

Figure 4.5 shows the results of the tested algorithms. Both the minimum cut with weights $\hat{f}_{add}$ (MC) and the minimum cut basis (MB) return the cut $\delta v_{n/2+1}$. All other algorithms that take submodularity into account find the optimal solution.

Version (b) of the problem instance has the same graph structure but a modified cost function. This cost function is truncated and thereby renders the adjusting bounds ineffective. The modified cost function is

$$f_{Ib}(A) = \min\{f_{Ia}(A) + \epsilon'|A \cap \mathcal{E}_k|,\ \gamma\} \tag{4.51}$$

for a truncation threshold $\gamma = f_r(\mathcal{E}_r) + f_b(\mathcal{E}_b) - (n/2 - \epsilon(n/2-1)^{-1}) + 1$ and a small $\epsilon' > 0$. The truncation makes the marginal costs and thus the weight $w_B(e) = \rho(e|\mathcal{E} \setminus e)$ in the iterative bounds zero for all edges in the current cut. However, the marginal cost $\rho(e|B)$ that is used for all other edges is not zero, so Algorithm 1 will never move away from a starting point, and will thus only find the optimal solution if it is given as an initialization. Since C^* has the maximum possible number of edges, it is not in the minimum cut basis with respect to $\hat{f}_{add}$. The result is obvious in Figure 4.5(b): the advantage of the iterative minimization is gone, and the iterative algorithm too only returns the quadratically worse second-best cut. In comparison, in version (a), the weights $w_B(e)$ are zero for the black edges given any cut B, since every cut must contain a black edge.

We remark that Instance I(b) is theoretically useful, because it shows that Lemma 4.6 is tight as a worst-case bound. However, the instance is not very realistic.

Type II: difficult instances for modular approximations and Queyranne's algorithm

The first few instances of the second group, II(a)-(c), again challenge modular approximations. Finally, instance II(d) demonstrates the benefit of theoretical approximation guarantees: there is no upper bound on the approximation factor for Queyranne's algorithm. On this instance, the heuristic can indeed perform arbitrarily badly, whereas the other algorithms are saved by upper bounds on their approximation factors.

The graph for examples II is again a clique, but its edges are partitioned into $n/2$ sets, indicated by colors in Figure 4.6. The black set $\mathcal{E}_k$ is, as for Graph I, the cut with the maximum number of edges. The remaining sets are constructed node-wise as

$$\mathcal{E}_i = \big\{(v_i, v_j) \in \mathcal{E} \mid i < j \le n/2\big\} \cup \big\{(v_{n/2+i}, v_j) \in \mathcal{E} \mid n/2 + i < j \le n\big\} \tag{4.52}$$

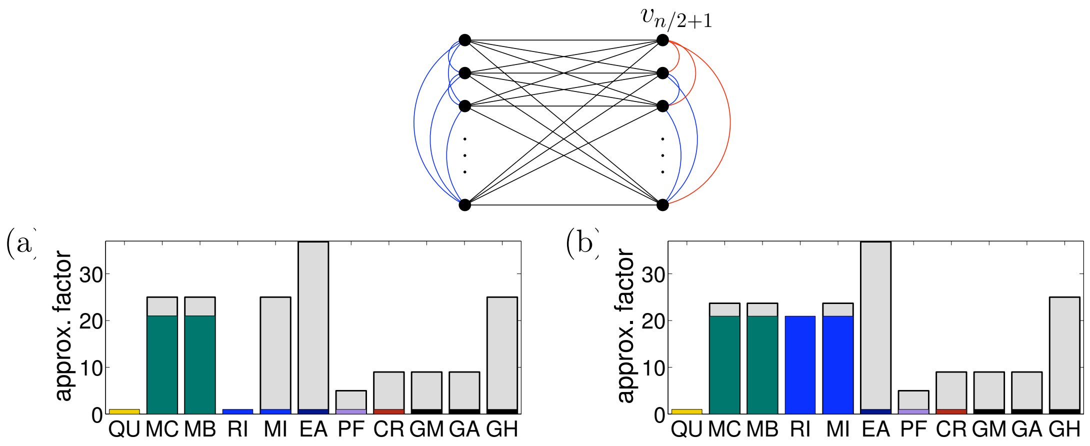


Figure 4.5. Graph I and empirical approximation factors with $n = 10$ nodes, so $n^2/4 - n/2+1 = 21$. Where applicable, gray bars illustrate the theoretical approximation bound.

for each $1 \leq i < n/2$. In Figure 4.6, set $\mathcal{E}_1$ is red, set $\mathcal{E}_2$ is blue, and so on. The cost function adds cost b for any set $\mathcal{E}_i$ intersecting the cut, and cost 1 if any black edge is in the cut:

$$f_{IIa}(A) = \mathbf{1}[|A \cap \mathcal{E}_k| \geq 1] + \sum_{i=1}^{n/2-1} b \cdot \mathbf{1}[|A \cap \mathcal{E}_i| \geq 1], \tag{4.53}$$

with $b = n/2$ for versions II(a) to II(c). As before, $\mathbf{1}[\cdot]$ denotes the indicator function. The optimal solution is again $C^* = \mathcal{E}_k$ with $f_{IIa}(C^*) = 1$. The results for the different algorithms are illustrated in Figure 4.6.

For II(a), the iterations with adaptive bounds still help to find the optimal solution. With the next two examples II(b) and II(c), this benefit vanishes thanks to two modifications: a truncation in II(b) and the addition of a tiny modular cost in II(c) tint the look through the glasses of marginal costs with respect to initializing cuts. The cost functions II(b) and II(c) are

$$f_{IIb}(A) = \min\{f_{IIa}(A),\ n\} \tag{4.54}$$
$$f_{IIc}(A) = f_{IIa}(A) + \epsilon|A \cap \mathcal{E}_k|. \tag{4.55}$$

Finally, a small modification of function f_{IIa} demonstrates the benefit of bounded factor approximations. We increase the constant b in function f_{IIa}. For any $b > n/2$, any solution other than C^* is more than $n^2/4 = |C^*| > n$ times worse than the optimal solution. Thanks to the upper bounds on their approximation factors, all algorithms except for QU find the optimal solution. The result of the latter depends on how it selects a minimizer of $f(B \cup e) - f(e)$ in the search for a pendent pair; this quantity often has several minimizers here. Some of those minimizers will lead to a good solution and some to a bad one. Versions II(a) to II(c) show lucky cases. Version II(d) is like II(a), but uses a different sequence, that is, permuted node

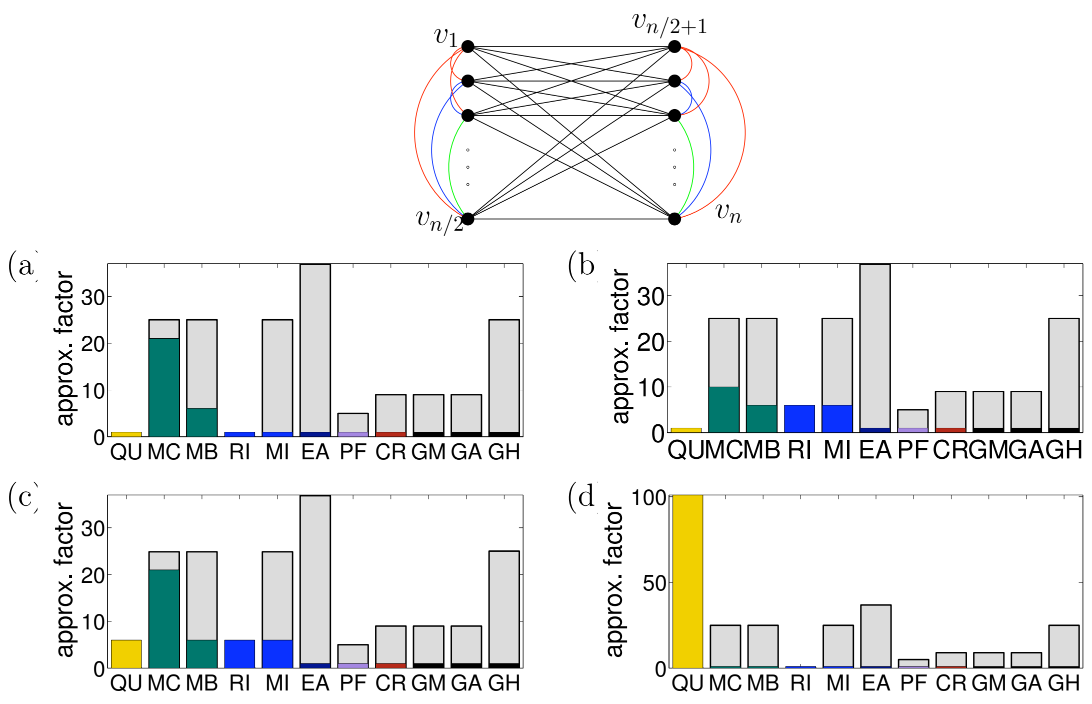


Figure 4.6. Graph II and empirical approximation factors with $n = 10$ nodes. Gray bars illustrate theoretical approximation bounds where applicable. In (a) and (b), cutting off v_1 costs $f(\delta v_1) = n/2 + 1 = 6$ and is the second-best cut. Cutting off v_n costs $f(\delta v_n) = n/2(n/2 - 1) + 1 = 21$, the worst cut that cuts off a single node. For (b), the maximum cost of a cut is $n = 10$. In (d), the second-best cut δv_1 has cost $f(\delta v_1) = b + 1 = 101 \gg \max\{|C^*|, n, \sqrt{m} \log m\}$.

labels, and $b = n^2 = 100$. For the permutation in (d), QU will always return the same solution δv_1 with cost $b + 1$, no matter how large b is. Therefore, its solution can become arbitrarily poor.

Type III: a difficult instance for the greedy heuristic

The worst input for the algorithm GH is in fact an instance of standard Minimum Cut with a modular cost function. The iterative algorithm, the approximation via polymatroidal flows and the approximation by $\hat{f}_{add}$ find the exact minimum for such instances.

Here, we again consider an (s, t)-cut (the worst-case examples above also work as (s, t)-cuts, with the exception that Queyranne's algorithm cannot be applied any more). Construct a clique of $(n - 1)$ nodes, where s is one of these nodes, and connect t with a single edge to some $v' \neq s$. The edge (v', t) has weight γ for a very small ϵ. Pick a subset X of $(n - 1)/2 - 1$ nodes excluding v'. All edges in $\delta(X \cup s)$ are assigned weight $\gamma - \epsilon$ for any small $\epsilon > 0$, and all other edges in the clique have weight $\gamma - \epsilon/2$. The minimum (s, t)-cut $C^* = \{v', t\}$ in this graph has therefore cost $f(C^*) = \gamma$. But the greedy heuristic will never select the optimal

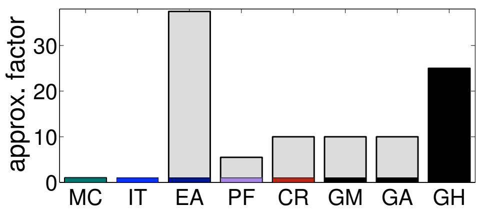


Figure 4.7. Results for Graph III with $n = 11$ nodes, $n^2/4 = 25$ and $\gamma = 50$, $\epsilon = 10^{-4}$. This instance was solved as an (s,t)-cut by applicable methods. Algorithm IT is the iterative algorithm with $\mathcal{I} = \{\emptyset\}$. By Lemma 4.6, both IT and MC are guaranteed to find an optimal solution here.

edge, and instead return the cut $\widehat{C} = \delta(X \cup s)$, with $f(\widehat{C}) = \frac{(n-1)^2}{4}(\gamma - \epsilon)$. This is by a quadratic factor $\frac{(n-1)^2}{4}(1 - \epsilon')$ worse than the optimum. A similar example gives a linear factor in n for a minimum cut without terminals. The randomized algorithms will behave differently: as the optimal edge is a bottleneck and occurs in all paths, and as its (marginal) weight only differs slightly from that of the other edges, it will be selected by the randomized algorithms with high probability. Figure 4.7 shows the empirical approximation factors for all methods. In fact, MC, the iterative approximation, the partition $\hat{f}_{pf}$ and the convex relaxation are always exact if the cost function is modular.

4.5. Summary and discussion

This chapter introduced approximation algorithms for CoopCut that either relax the cost function or simplify the constraints. When approximating the cost function, it is important that the approximation retains as much as possible of the relevant coupling, while at the same time it must be simple enough to admit polynomial-time algorithms. We compare two generic approximations from the literature, $\hat{f}_{add}$ and $\hat{f}_{ea}$, to two new functions, $\hat{f}_{pf}$ and an adjusting upper bound (ITB). The function $\hat{f}_{pf}$ takes the minimum over a family of graph-specific partitions and we show that its dual corresponds to a problem for which polynomial-time algorithms exist. On dense graphs, its approximation bound is better than that of the generic functions. The adjusting bounds yield good empirical results. Even though we leave it as an open problem whether the iterative Algorithm 1 always terminates in a polynomial number of iterations, in the experiments it finished after few iterations, and its average results in Section 4.4.1 are better than those for $\hat{f}_{add}$ and $\hat{f}_{ea}$.

When taking the second route of simplifying the constraints, one must ensure that the new problem is easier to optimize and that its solution can be transformed into a similarly good solution that is a cut. Our algorithms rely on the insight that graph cut problems are related to cover problems, if we allow a cut to include additional edges. We then transfer existing methods for covers to cuts and derive a rounding method for relaxations as well as greedy algorithms. The covers resulting from cuts have an exponential number of constraints, but we show that these constraints can still be handled. In particular the randomized greedy algorithm is easy to implement and performs very well in the experiments. Its theoretical bound refers to the unpruned solution that can include many additional edges. The pruning step may not always find the best solution within the set of selected edges. Designing a worst-case example for GM and GA is a question for future exploration.

We prove upper bounds on the approximation factors of all algorithms. Chapter 3 shows that MinCoopCut does not admit constant-factor approximations, only polynomial ones. In that respect, the bounds shown in Lemmas 4.2, 4.6, 4.7 and 4.9 and Theorem 4.1 are optimal. While we can state these bounds very generally as $O(n)$, $O(\sqrt{m}\log m)$ or $O(m)$, almost all of them depend on the graph structure, such as the longest path or the vertex boundary of the minimum cut. The exact statement of the factors can make a significant difference: Consider a graph that is a chain of $\sqrt{n}$ cliques between s and t. Each clique has $\sqrt{n}$ nodes, and two adjacent cliques share one node. The longest path has length $n-1$, whereas $|\Delta_s| \leq \sqrt{n}$, and $\sqrt{m} \approx n^{3/4}$. Furthermore, for standard MinCut with additive edge weights, the approximations $\hat{f}_{add}$, $\hat{f}_{pf}$, the adjusting bounds and the convex relaxation always find the optimal solution. This is not necessarily the case for the greedy algorithms.

The empirical results complement the theoretical analysis and show that the solutions returned by the approximation algorithms are in general better than the worst-case approximation bounds. Finally, Section 4.4.2 demonstrates that even non-constant approximation bounds can be beneficial, as they prevent arbitrarily poor solutions. The worst-case examples moreover show some of the proven bounds to be tight.

Chapter 5.

Cooperative Cuts and Energy Minimization

In this chapter, we explore the relations between cooperative cuts and probabilistic inference. The relations draw from a crucial observation: wherever standard graph cuts are applied but they do not express sufficient coupling, cooperative cuts can be used instead. In particular, many applications arise from representing functions via graph cuts. Our motivation and applications here come from computer vision, but the introduced model can be applied in any field where similar graphical models are used.

To discriminate between binary vectors $\boldsymbol{x}$ and their entries x_i, we will denote binary vectors by bold letters in this chapter.

5.1. Graph cuts, probabilistic models and inference

Probabilistic models are used in a wide range of settings. In several cases, inference reduces to a Minimum Cut problem in an appropriate graph. While this observation, which we describe in detail in the sequel, has found many successful applications, the graph cut framework implies certain independence assumptions that set limits to the associated class of models. In particular, graph cuts can implement only pairwise, submodular potentials. Figure 5.1 shows two examples from a common application, image segmentation, where standard graph-based methods fail. In this chapter, we address the question of how cooperative cuts can help overcome some of the limitations of commonly used tractable probabilistic models.

Probabilistic models and inference

To understand how cooperative cuts integrate in an inference framework, we begin with the basics of tractable inference in graphical models. Our general interest is in the *most probable explanation* (MPE) problem [Pearl, 1988]: given a probability distribution $p(\boldsymbol{x}|\bar{\boldsymbol{z}}) = \frac{1}{Z}\exp(-E(\boldsymbol{x}))$ for variables $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \mathcal{L}^n$ in some discrete domain $\mathcal{L}$ (the set $\bar{\boldsymbol{z}}$ of observed variables may be empty), find an assignment that maximizes the probability of the unobserved variables,

$$\boldsymbol{x}^* \in \operatorname*{argmax}_{\boldsymbol{x}} p(\boldsymbol{x}).$$

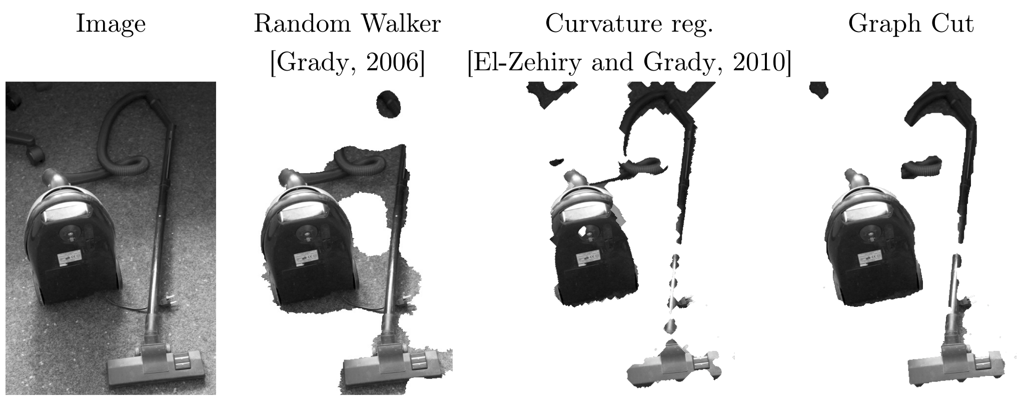


Image
Random Walker [Grady, 2006]
Curvature reg. [El-Zehiry and Grady, 2010]
Graph Cut

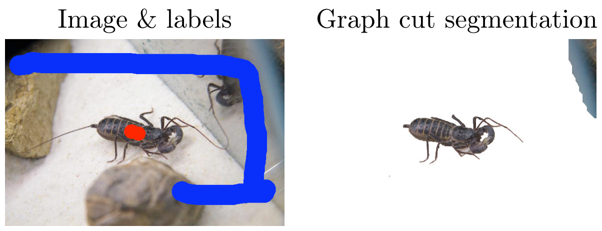


Figure 5.1. Examples from interactive image segmentation illustrating the limits of local probabilistic models and of state-of-the-art algorithms involving graphs. Here, a user provides initial partial labels of object and background, and the algorithm completes the figure-ground segmentation. For low-contrast regions or objects with long, fine segments, the shown local methods fail to identify the correct object boundaries. They tend to "shortcut" long or weak boundaries, an effect known as *shrinking bias*. Chapter 6 will demonstrate non-local methods that can handle these difficulties better.

The term Z is a normalizing factor. The probability is determined by an *energy* function[1] $E(\boldsymbol{x})$, so that a maximizing assignment $\boldsymbol{x}^*$ equivalently minimizes the energy:

$$\boldsymbol{x}^* \in \operatorname*{argmin}_{\boldsymbol{x}} E(\boldsymbol{x}).$$

First, for simplicity, we assume all variables to be binary, i.e., $\mathcal{L} = \{0,1\}$. Then $E : \{0,1\}^n \to \mathbb{R}_+$ is a (pseudo-boolean) function on binary vectors. The distinction between MPE and *Maximum a posteriori* (MAP) inference is not always made clearly in the literature. We use the definition that MAP is a case of MPE where $\bar{\boldsymbol{z}}$ is non-empty. In any case, important is the form of the energy function.

Without any restrictions placed on E, it is easy to see that there is not much hope for efficient inference in general, even if we consider bounded approximations. For example, assume that E is given by an oracle, and let $y \in \{0,1\}^n$ be an unknown vector. Consider the energy $E(\boldsymbol{x}) = 1$ if $\boldsymbol{x} = \mathbf{y}$, and $E(\boldsymbol{x}) = \gamma(n)$ otherwise, where $\gamma(n) > 1$ could be any (polynomial-time) computable function of n. With only polynomially many queries to E, it is exponentially unlikely to identify $\mathbf{y}$,

[1]We implicitly include the common case that the energy is also a function of $\bar{\boldsymbol{z}}$.

and since $\gamma(n)$ is almost arbitrary, no approximation guarantee of any form is possible in polynomial time. The exponential difficulty of approximate inference in such unrestricted models, therefore, is worse than that implied by the well known fact that MPE is NP-hard and not constant-factor approximable [Abdelbar and Hedetniemi, 1998].

Thus, model restrictions are often applied to allow for exact or good approximate inference in polynomial time. These are either structural, such as treewidth or factor size, or functional, such as submodularity. There can be problems with such restrictions, however, such as the well known drawbacks of local pairwise random fields in computer vision, illustrated in Figure 5.1. Thus, the question arises whether there are other combinatorial structures that go beyond the previous restrictions but still, as opposed to the introductory example, enable inference with a bounded approximation factor.

Structural restrictions

The common structural restrictions for tractability correspond to factorizations of p. Let p factor with respect to a graphical model $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ comprising $n = |\mathsf{V}|$ nodes and edge set E. That means there is a node $\mathsf{v}_i \in \mathsf{V}$ for each variable x_i, and

$$p(\boldsymbol{x}) \propto \prod_{(\mathsf{v}_i, \mathsf{v}_j) \in \mathsf{E}} \exp(\psi_{ij}(x_i, x_j)) \prod_i \exp(\psi_i(x_i)). \tag{5.1}$$

This is equivalent to

$$E(\boldsymbol{x}) = \sum_{(\mathsf{v}_i, \mathsf{v}_j) \in \mathsf{E}} \psi_{i,j}(x_i, x_j) + \sum_i \psi_i(x_i). \tag{5.2}$$

In particular, the graph G indicates conditional independence relations that the distribution p must satisfy.
The decisive parameter indicating the complexity of MPE in G is the *treewidth* of G [Chandrasekaran et al., 2008]. The treewidth [Kloks, 1994] is one less than the size of the maximum clique in a minimum triangulation. Generally, finding the MPE takes time exponential in the treewidth when it is known.

In general, we write $E(\boldsymbol{x}) = \sum_{\phi \in \Phi} \psi_\phi(\boldsymbol{x}_\phi)$ where Φ corresponds to the set of factors comprising the distribution. Viewed as a bipartite (factor) graph, each $\phi \in \Phi$ is the subset of nodes $\phi \subseteq \mathsf{V}$ involved in a factor. Many approximate inference algorithms rely on $\max_{\phi \in \Phi} |\phi|$ being small. For example, the cost of sending messages even in loopy belief propagation is exponential in $|\phi|$. Therefore, $\max_{\phi \in \Phi} |\phi|$ (which we call the *factorwidth*) may also be seen as a complexity parameter for certain approximate inference algorithms.

Functional restrictions

Nevertheless, treewidth and factorwidth are not the only characterizations of tractability. In fact, exact polynomial-time MPE is possible even with maximum treewidth and factorwidth if E is restricted in other ways. A recent class of energy functions having received attention in the vision community is that of submodular functions. To map between pseudo-boolean energy functions and set functions, we introduce some notation. From energy functions to set functions, the correspondence is via characteristic vectors. Conversely, given a set function $g : 2^{\mathcal{V}} \to \mathbb{R}_+$ defined on a ground set $\mathcal{V}$ of n elements, an equivalent energy function $E : \{0,1\}^n$ over variables x_i, one for each $v_i \in \mathcal{V}$, is defined as follows. Let $X(\boldsymbol{x}) \subseteq \mathcal{V}$ be the set of elements $v_i \in \mathcal{V}$ whose corresponding variable x_i is one, i.e., $X(\boldsymbol{x}) = \{v_i \in \mathcal{V} : x_i = 1\}$. As a result, $\boldsymbol{x} = \chi_{X(\boldsymbol{x})}$ is the characteristic vector of $X(\boldsymbol{x})$, and the energy equivalent to g is $E(\boldsymbol{x}) \triangleq g(X(\boldsymbol{x}))$.

Finding an assignment that minimizes the energy is equivalent to finding the subset $X \subseteq \mathcal{V}$ that minimizes g. When g is submodular, this can be done in polynomial time [Fujishige, 2005]. As an example of a submodular g that places restrictions neither on treewidth nor factorwidth, consider the submodular function $g(S) = -\sum_i \prod_{v \in S} w_{i,v}$, where $0 < w_{i,v} \leq 1$ is a set of coefficients for all i and $v \in \mathcal{V}$.

However, submodular function minimization is not currently a low-order polynomial time algorithm. In consequence, areas such as computer vision and constraint satisfaction have seen a lot of interest in special subclasses of submodular functions that admit faster optimization, in particular, functions representable as graph cuts.

Graph cut representations

If an energy function has order two and each term is a submodular function of two variables, then MPE reduces to a minimum (s,t)-cut [Greig et al., 1989, Boykov and Jolly, 2001, Kolmogorov and Zabih, 2004] on a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. We call $\mathcal{G}$ the *structure graph* to clearly distinguish it from the graphical model. The graph $\mathcal{G}$ has terminal nodes s, t, and a node v_i for each variable x_i. A labeling $\boldsymbol{x}$ induces a partition of $\mathcal{V}$ and thereby an (s,t)-cut $\Gamma(\boldsymbol{x}) = \delta(X(\boldsymbol{x}))$. We call $\Gamma(\boldsymbol{x})$ also the *boundary* of $\boldsymbol{x}$. The graph $\mathcal{G}$ has weights $w : \mathcal{E} \to \mathbb{R}_+$ and is designed such that its cut equals the energy:

$$E(\boldsymbol{x}) = \sum_{e \in \delta(X(\boldsymbol{x}))} w(e) \triangleq w(\delta(X(\boldsymbol{x}))). \tag{5.3}$$

Let $C^* \subseteq \mathcal{E}$ be the minimum cut, and X^* the nodes reachable from s after removal of C^*. Then $C^* = \delta(X^*)$ and the optimal assignment $\boldsymbol{x}^*$ is determined as $x_i^* = 1$ if and only if $v_i \in X^*$. To achieve efficiency by using graph cuts, the construction must be limited to pairwise energies, that is, a factorwidth of 2. Higher order models may be obtained by adding auxiliary variables. Section 7.2.1 will define graph representability of set functions (and therefore also energy functions) in greater detail.

Common models in computer vision are pairwise, grid-structured Markov random fields, whose energy function (given an observed image $\bar{\boldsymbol{z}}$) contains unary and

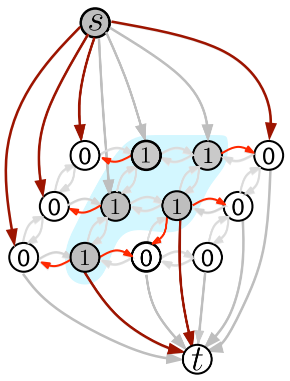


Figure 5.2 Commonly used graph cut representation of a local pair-wise Markov random field. The energy of the labeling $\boldsymbol{x}$ indicated in the nodes of the graph is equivalent (up to constants) to the sum of weights of the edges in $\Gamma(\boldsymbol{x})$, i.e., the edges from label-one to label-zero nodes.

pairwise terms. The pairwise terms refer to edges in the random field, which are often denoted by a neighborhood relation $\mathcal{N}$:

$$E(\boldsymbol{x}; \bar{\boldsymbol{z}}) = \sum_i \psi_i(x_i) + \sum_{(i,j)\in\mathcal{N}} \psi_{ij}(x_i, x_j). \tag{5.4}$$

The corresponding structure graph is depicted in Figure 5.2. Its *terminal edges* (s, v) and (v, t) for all $v \in \mathcal{V}$ model the unary potentials, and the remaining grid edges represent the pairwise terms.

Inspired by the efficiency and wide applicability of graph cut representations, a principal goal has become identifying the most general classes of energies that can be exactly optimized either directly or indirectly via graph cuts. For example, while some binary pairwise potential functions can be solved exactly using graph cuts, in many cases higher order (e.g., k-ary) potential functions [Zalesky, 2003, Kolmogorov and Zabih, 2004, Freedman and Drineas, 2005, Ramalingam et al., 2008, Živný and Jeavons, 2010, Ramalingam et al., 2011] and potentials functions over non-binary variables [Boykov et al., 2001] can also be solved efficiently.

Unfortunately, there are critical deficiencies when graph cuts are used in practice, partly stemming from their inability to represent more than only a limited class of energies efficiently (see also Section 7.2.1). The core issue is that graph cuts model an energy that decomposes into *pairwise* terms with *nonnegative* weights. The direct use of such energies can cause insurmountable over-smoothing in image segmentation. Some higher-order energies are graph-representable, but this representation might regrettably require additional variables which can impair computational efficiency [Živný and Jeavons, 2010, Ramalingam et al., 2011]. Recent research, therefore, has aimed to identify practically manageable higher order energies [Ishikawa, 2009, Kohli and Kumar, 2010, Kohli et al., 2007, Komodakis and Paragios, 2009], and to develop efficient optimization methods for non-submodular potentials [Kolmogorov and Rother, 2007].

5.1.1. Multi-label energies and move-making algorithms

The above sections address binary probabilistic models. To complement this, we briefly review a family of common methods for inference in probabilistic models where variables can take multiple values. These methods have been used primarily in computer vision. Extending Markov Random Fields to the multi-label case allows each variable x_i to take labels from a set $\mathcal{L}$ of discrete labels. The unary potentials ψ_i are then defined over $\mathcal{L}$, and the pairwise potentials ψ_{ij} for values in $\mathcal{L} \times \mathcal{L}$. Whereas binary segmentation for regular potentials is polynomial-time solvable, the multi-label version with $|\mathcal{L}| > 2$ becomes NP-complete [Boykov et al., 2001]. Multi-label segmentation relates to multi-way cuts, but the approximation factor for multi-way cut does not transfer to the inference problem [Boykov et al., 2001]. One approach to reduce the multi-label case to a sequence of binary cases are *move-making algorithms*. Other algorithms apply too, but we limit ourselves here to move-making algorithms as those become relevant later in this chapter.

Move-making algorithms [Boykov et al., 2001, Milis, 1996] can be viewed as local search algorithms. Given a current labeling $\boldsymbol{x}_t \in \mathcal{L}^n$, we seek the optimal re-labeling

$$\boldsymbol{x}_{t+1} \in \operatorname{argmin} E(\boldsymbol{x}) \text{ s.t. } \boldsymbol{x} \in \mathcal{X}(\boldsymbol{x}_t, \mathfrak{a}_t) \tag{5.5}$$

from a limited set $\mathcal{X}(\boldsymbol{x}_t, \mathfrak{a}_t)$. This search space consists of all labelings that are reachable within one "move" and is parameterized by one or more labels $\mathfrak{a}, \mathfrak{b}, \mathfrak{c} \in \mathcal{L}$.

Having found the optimal $\boldsymbol{x}_{t+1}$, the typical algorithm defines a new search space $\mathcal{X}(\boldsymbol{x}_{t+1}, \mathfrak{a}_{t+1})$ around this new labeling and finds the best labeling in the new space. It iterates until no more candidate search space around the current labeling holds a point that lowers the energy further. The algorithms differ in the type of move they allow, that means, in the definition of the local search neighborhood. Commonly used are *expansion moves* and *swap moves* introduced by Boykov et al. [2001], Milis [1996]. A more recent variant are *fusion moves* that allow to merge two labelings using a graph cut with negative edge weights [Lempitsky et al., 2010]. In the sequel, we focus on the former two that use graph cuts with nonnegative weights.

Expansion moves

Expansion moves allow to re-label each x_i to a pre-determined label $\mathfrak{a}$, or to keep its current label:

$$\mathcal{X}(\boldsymbol{x}', \mathfrak{a}) = \{\boldsymbol{x} \mid x_i = x'_i \text{ or } x_i = \mathfrak{a}\ \forall i\}. \tag{5.6}$$

Boykov et al. [2001] show how to find the energy-minimizing labeling in $\mathcal{X}(\boldsymbol{x}_t, \mathfrak{a})$ by a graph cut if the potentials are at most pairwise, symmetric and metric. That is, for any labels $\mathfrak{a}, \mathfrak{b}, \mathfrak{c} \in \mathcal{L}$, it must hold that

$$\psi_{ij}(\mathfrak{a}, \mathfrak{b}) = 0 \quad \text{if and only if } \mathfrak{a} = \mathfrak{b} \tag{5.7}$$

$$\psi_{ij}(\mathfrak{a}, \mathfrak{b}) = \psi_{ij}(\mathfrak{b}, \mathfrak{a}) \geq 0 \tag{5.8}$$

$$\psi_{ij}(\mathfrak{a}, \mathfrak{b}) \leq \psi_{ij}(\mathfrak{a}, \mathfrak{c}) + \psi_{ij}(\mathfrak{c}, \mathfrak{b}). \tag{5.9}$$

Semi-metrics, for which the triangle inequality does not hold, can be approximated (at some loss of accuracy) by Potts potentials[2].

The algorithm proceeds by alternating the label $\mathfrak{a}$ for each move, until the labeling does not change for any possible label $\mathfrak{a}$ in the move. Let $\hat{\boldsymbol{x}}$ be the solution at convergence of the algorithm, that means, $\hat{\boldsymbol{x}} \in \operatorname{argmin}_{\boldsymbol{x} \in \mathcal{X}(\hat{\boldsymbol{x}},\mathfrak{a})} E(\boldsymbol{x})$ for all possible $\mathfrak{a} \in \mathcal{L}$. Boykov et al. [2001] prove that then $\hat{\boldsymbol{x}}$ is at most by a bounded approximation factor worse that the optimal solution $\boldsymbol{x}^*$: $E(\hat{\boldsymbol{x}}) \leq 2\nu E(\boldsymbol{x}^*)$, where $\nu = \max_{i,j} \max_{\mathfrak{a},\mathfrak{b} \in \mathcal{L}} \psi_{ij}(\mathfrak{a},\mathfrak{b}) / \min_{\mathfrak{a}',\mathfrak{b}' \in \mathcal{L}} \psi_{ij}(\mathfrak{a}',\mathfrak{b}')$.

The *metric labeling problem* defined by metric pairwise potentials has also stipulated further theoretical work, e.g., [Kleinberg and Tardos, 1999, Chuzhoy and Naor, 2004].

Swap moves

Swap moves apply to pairwise potentials if those are semi-metrics [Boykov et al., 2001], but no guarantees have been proven about their solution at convergence. For those moves, we pick two labels $\mathfrak{a}$ and $\mathfrak{b}$, and only allow to re-label any variable that currently carries any of those two labels. For those variables, we are allowed to swap labels $\mathfrak{a}$ and $\mathfrak{b}$. All other labels remain untouched:

$$\mathcal{X}(\boldsymbol{x}', \mathfrak{a}, \mathfrak{b}) = \{\boldsymbol{x} \mid x_i = x_i' \text{ or both } x_i' \in \{\mathfrak{a},\mathfrak{b}\} \text{ and } x_i \in \{\mathfrak{a},\mathfrak{b}\}\ \forall i\}. \tag{5.10}$$

Both expansion and swap moves admit to optimize over their search space via a graph cut.

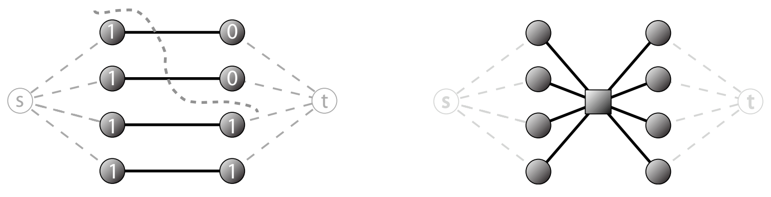
5.2. Energy functions induced by cooperative cuts

We saw that graph cut representations for binary energies are efficient and practical, but also that efficient representations are limited to a subset of functions. Importantly, we can view a representable function either as a pseudo-boolean potential function or as a graph cut function. In the sequel, we build on the graph cut viewpoint and investigate the energy functions induced not only by standard graph cuts, but by cooperative cuts. Analogously to Equation (5.3), we define a cooperative cut energy that is based on a structure graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ which contains a node $v_i \in \mathcal{V}$ for each variable x_i.

Definition 5.1 (Cooperative cut energy function)**.** An energy function $E : \{0,1\}^n \to \mathbb{R}_+$ is a *cooperative cut energy function* if there exists a structure graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with nodes $\mathcal{V} = \{v_i\}_{i=1}^n \cup \{s,t\}$ and a nondecreasing submodular function $f : 2^{\mathcal{E}} \to \mathbb{R}_+$ such that for all $\boldsymbol{x} \in \{0,1\}^n$, it holds that

$$E(\boldsymbol{x}) = E_f(\boldsymbol{x}) \triangleq f(\delta(X(\boldsymbol{x}))) = f(\Gamma \boldsymbol{x}). \tag{5.11}$$

[2]Potts potentials consist of pairwise terms $\psi_{ij}(x_i, x_j) = \gamma \mathbf{1}[x_i \neq x_j]$ that ignore the actual labels and merely detect disagreements of labels.



(a) structure graph (b) factor graph

Figure 5.3. Structure graph and corresponding factor graph. Let the edge cost be $f(S) = \min\{|S|, 1\}$ on the solid edges. Then any cut through the solid edges costs one, and the corresponding potential is one whenever there is *any* (1,0) pair among the variable pairs induced by those edges. Thus, there must be a factor that includes all of the nodes incident to any solid edge.

We denote the family of all cooperative cut energy functions by $\mathcal{F}_{coop}$.

That means we replace the sum of edge weights by a nonnegative submodular function $f : 2^{\mathcal{E}} \to \mathbb{R}_+$ on *edges* of the structure graph. Resulting from the equivalence between cuts and energy functions, MAP inference with cooperative cut energies corresponds to solving an instance of MinCoopCut. Any of the algorithms from Chapter 4 applies for approximate inference and thereby shows that the energies in $\mathcal{F}_{coop}$ do not fall in the class of completely inapproximable energies.

Next, we outline some properties of the new family $\mathcal{F}_{coop}$. Chapter 3 already implies results on the hardness of inference for E_f in general. Recall, however, that inference with arbitrary energy functions does not even admit any approximations at all.

5.2.1. Properties of cooperative cut energies

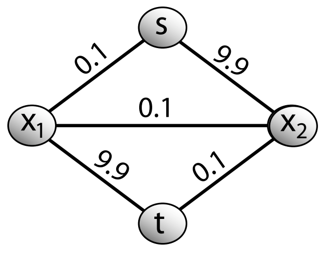
The first question arising is whether the energies in $\mathcal{F}_{coop}$ satisfy any of the simplifying properties summarized above, such as limited tree- or factorwidth, locality, or submodularity (regularity).

Treewidth and factorwidth

In general, the submodular function f does not decompose as a sum of weights over single elements. Instead, it is sensitive to the entirety, or composition, of elements in its set argument. Thus, it may *couple* edges from anywhere in the structure graph. Since edges map to pairwise potentials in graph cuts, cooperative cuts formally couple pairwise potentials:

$$E_f(\boldsymbol{x}) = f(\{(v_i, v_j) \in \mathcal{N} \mid x_i \neq x_j\}). \tag{5.12}$$

The energy only decomposes further into smaller terms if f does. As a result, when re-writing a cooperative cut energy as a pseudo-boolean polynomial in $\boldsymbol{x}$, all



$$f(S) = \sqrt{\sum_{e \in S} w(e)}$$

$$E_f(0,0) + E_f(1,1) = \sqrt{10} + \sqrt{10} \approx 6.32$$
$$E_f(0,1) + E_f(1,0) = \sqrt{0.3} + \sqrt{19.9} \approx 5.01$$

Figure 5.4. A cooperative cut energy function E_f that is not submodular. Its cost function is a square root of the sum of weights in the cut. The same example works for a directed graph too.

nodes $v \in e$ adjacent to any edge e in a group of coupled edges must occur in the same term and are coupled, too. Figure 5.3 demonstrates that the factor graph corresponding to a cooperative cut energy can have arbitrarily large factors. Thus, the graphical model corresponding to $E_f \in \mathcal{F}_{coop}$ may have unrestricted treewidth or factorwidth, without any locality restrictions. In other words, the energy E_f is a higher-order energy of order up to n.

Regularity, submodularity and subadditivity

Having ruled out any algorithm that relies on treewidth for tractable inference, we remain with submodularity as a candidate common simplifying property. Recall that an energy of two variables is regular if

$$E(0,0) + E(1,1) \leq E(0,1) + E(1,0). \tag{5.13}$$

This definition extends to more than two variables via projections, and is then equivalent to submodularity (Section 2.3).

Lemma 5.1. *A cooperative cut energy function can be non-submodular.*

The example in Figure 5.4 proves that some functions in $\mathcal{F}_{coop}$ violate Condition (5.13), and therefore the members of $\mathcal{F}_{coop}$ are not in general submodular. In fact, submodular energy functions are only a strict subset of $\mathcal{F}_{coop}$. They are the instances where the cut cost f is a sum of weights, or those where the graph structure and f have very regular properties – some examples are shown in Section 5.3, and also in Chapter 7.

Nevertheless, any function $E_f \in \mathcal{F}_{coop}$, or the equivalent set function $g(X) = f(\delta(X))$ on sets of nodes, is subadditive:

Lemma 5.2. *The function $g : 2^{\mathcal{V}} \to \mathbb{R}_+$ induced by a cooperative cut as $g(X) = f(\delta X)$ is subadditive if f is nondecreasing submodular. Thus, any cooperative cut energy is subadditive.*

Proof. Let $\mathcal{E}(X,Y) = \{(u,v) \in \mathcal{E} \mid u \in X, v \in Y\}$ denote the set of all edges between $X \subseteq \mathcal{V}$ and $Y \subseteq \mathcal{V}$. For any $X, Y \subseteq \mathcal{V}$, it holds that $\delta(X \cup Y) \subseteq \delta X \cup \delta Y$:

$$\delta(X \cup Y) = (\delta X \setminus \mathcal{E}(X,Y)) \cup (\delta Y \setminus \mathcal{E}(Y,X)) \subseteq (\delta X \cup \delta Y). \tag{5.14}$$

Using (5.14) and the submodularity, monotonicity and nonnegativity of f, we conclude that for any $X, Y \subseteq \mathcal{V}$, the function g satisfies the conditions of subadditivity:

$$\begin{aligned} g(X) + g(Y) &= f(\delta X) + f(\delta Y) && (5.15)\\ &\geq f(\delta X \cup \delta Y) + f(\delta X \cap \delta Y) && (5.16)\\ &\geq f(\delta X \cup \delta Y) && (5.17)\\ &\geq f(\delta(X \cup Y)) && (5.18)\\ &= g(X \cup Y) && \square \end{aligned}$$

Subadditivity, however, is not a strong property for simplifying optimization. As an illustrating example, consider the subadditive function[3] $g : 2^{\mathcal{V}} \to \mathbb{R}_+$,

$$g(X) = \begin{cases} 0 & \text{if } X = \emptyset \text{ or } X = \mathcal{V} \\ 1 & \text{if } X = R \\ \gamma(n) & \text{otherwise,} \end{cases} \tag{5.19}$$

for a large, arbitrary function $\gamma(n)$. Without knowledge of R, it is impossible to find the optimum $Y^* = \operatorname{argmin}_{Y \subset \mathcal{V}, Y \neq \emptyset} g(Y)$ in polynomial time. That means if g is given as an oracle and if we exclude the full or empty set as solutions, then no polynomial-time algorithm can minimize g to any approximation factor smaller than $\gamma(n)$. This difficult function is equivalent to a cooperative cut energy: define a connected graph on $\mathcal{V}$, with edge weights

$$w(e) = \begin{cases} 1 & \text{if } e \in \delta(R \cup s) \\ \gamma(n) & \text{otherwise.} \end{cases}$$

Then $g(X) = f(\delta X)$ for a submodular function $f(S) = \max_{e \in S} w(e)$. If the graph structure, the functional form of f and the edge weights are known, then optimizing g becomes much simpler, and polynomial-time solvable (Section 6.2.4).

Implications

Summarizing the above thoughts, cooperative cut energies do not satisfy any of the common properties that lead to tractable (approximate) inference. They neither have restrictions on their order or treewidth, nor are they submodular. Given just an oracle of the energy, energy minimization may not even be approximable in polynomial time, as the subadditive example demonstrates. However, the mere

[3]We thank Jens Vygen for this example of a difficult subadditive function.

assumption of knowing a structure graph $\mathcal{G}$ and being able to query a function over edges changes the picture. Then, the algorithms in Chapter 4 enable approximate inference with bounded approximation factors. Hence, $\mathcal{F}_{coop}$ defines a new family of energy functions that are very general but still admit approximate optimization.

Furthermore, in Chapter 7 we will see that $\mathcal{F}_{coop}$ is a strict superset of the set of all normalized submodular functions: that is, *any* normalized submodular function can be represented by a cooperative cut.

5.3. Expressive power: Cooperative cuts and models in computer vision

Section 5.2 states that cooperative cut energies can have arbitrarily high order. This observation raises the question whether any of the recent higher-order energy functions that have been suggested in the computer vision literature can be reduced to cooperative cuts. We begin with general nondecreasing submodular functions and then proceed with specific recent higher-order potentials. Some of these admit exact algorithms and specialized methods [Kohli et al., 2009a,b, Delong et al., 2011, Ladický et al., 2010]. In practice, of course, a specialized exact algorithm is preferable if it exists and if it is efficient. In any case, the cooperative cut viewpoint is a conceptual enhancement that can enable natural extensions of the models discussed below.

This section addresses mostly discrete potentials. Further models, some of them not submodular, will be described in Section 6.2. Chapter 7 is dedicated to the representation of general submodular functions. A selection of relations and representations is summarized in Table 5.1.

5.3.1. Nonnegative nondecreasing submodular functions

Nondecreasing submodular energy functions are usually considered together with other functions, such as graph cuts or negative modular functions. Otherwise, the minimum is trivially the empty set, i.e., the all-zeroes vector.

Lemma 5.3. *Cooperative cut energies $\mathcal{F}_{coop}$ include all nonnegative nondecreasing submodular functions.*

Proof. To represent the submodular function g (which is equivalent to an energy function $E(\boldsymbol{x}) = g(X(\boldsymbol{x}))$), we construct a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with a node v_i for each variable or element x_i, and additional terminal nodes s, t. The set of edges $\mathcal{E}$ consists of all edges (v,t) from a node $v \neq s$ to t. Then we define a function $f : 2^{\mathcal{E}} \to \mathbb{R}_+$ on edges that is equivalent to g: $f(S) = g(\{v \mid (v,t) \in S\})$. Then $f(\delta(X(\boldsymbol{x}) \cup s)) = g(X(\boldsymbol{x})) = E(\boldsymbol{x})$. $\square$

Adding a modular function to g then merely corresponds to adding modular-cost terminal edges, and adding a standard graph cut function amounts to adding modular-cost edges between nodes in $\mathcal{V}$.

5.3.2. P^n functions

Kohli et al. [2009a] introduce a general family of potential functions to which move-making algorithms [Boykov et al., 2001] can be applied. For a clique $\phi \subseteq \mathcal{V}$ corresponding to variables $\boldsymbol{x}_\phi$, these potentials are of the form

$$\psi_\phi(\boldsymbol{x}_\phi) = g\Big(\sum_{i,j\in\phi} \tilde{\psi}_\phi(x_i, x_j)\Big), \tag{5.20}$$

where $g : \mathbb{R} \to \mathbb{R}$ is concave non-decreasing and $\tilde{\psi}_\phi$ is a symmetric pairwise potential satisfying $\tilde{\psi}_\phi(\mathfrak{a}, \mathfrak{b}) \geq \tilde{\psi}_\phi(\mathfrak{c}, \mathfrak{c})$ for all labels $\mathfrak{a}, \mathfrak{b}, \mathfrak{c} \in \mathcal{L}$. (This implies also that $\tilde{\psi}_\phi(\mathfrak{a}, \mathfrak{a}) = \tilde{\psi}_\phi(\mathfrak{b}, \mathfrak{b})$ even for $\mathfrak{a} \neq \mathfrak{b}$.) We further assume that g is nonnegative. When taking g to be the identity function, their proofs immediately imply that the argument of g, the sum of potentials, is amenable to swap moves via graph cuts, and, if $\tilde{\psi}$ is a metric, also to expansion moves.

Lemma 5.4. *If the swap (expansion) move for the potential $\sum_{i,j\in\phi} \tilde{\psi}_\phi(x_i, x_j)$ can be solved as a graph cut, then the swap (expansion) move for ψ_ϕ can be solved as a Minimum Cooperative Cut.*

Lemma 5.4 allows g to be any nondecreasing concave function and thus provides an extension to graph cuts: with standard graph cuts, expansion moves are only possible if g is a linear function.

Proof. We start with the structure graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ whose minimum cut yields the optimal move if g is the identity. Then, for any elementary cut $\Gamma\boldsymbol{x}$ that can be reached by a move, it holds that $w(\Gamma\boldsymbol{x}) = w(\delta(X(\boldsymbol{x}_\phi))) = \sum_{i,j\in\phi} \tilde{\psi}_\phi(x_i, x_j) + c$ for some constant c (this constant may be zero) [Boykov et al., 2001]. Now we define a submodular function $f : 2^{\mathcal{E}} \to \mathbb{R}$ on the edges of $\mathcal{G}$ as $f(S) = g(\sum_{e\in S} w(e) - c)$. This function is submodular on $2^{\mathcal{E}}$ because g is concave and nondecreasing. It follows that

$$f(\delta(X(\boldsymbol{x}_\phi))) = g(w(\delta(X(\boldsymbol{x}_\phi))) - c) = \psi_\phi(\boldsymbol{x}_\phi). \qquad \square$$

As an illustration, we show the cut cost in the binary label case. Let $\nu_1 = \tilde{\psi}_\phi(1, 1)$, $\nu_0 = \tilde{\psi}_\phi(0, 0)$ and $\gamma = \tilde{\psi}_\phi(1, 0) \geq \nu_0, \nu_1$. We define a graph $\mathcal{G} = (\mathcal{V} \cup \{ s, t\}, \mathcal{E})$. For every $i \in \phi$, this graph has a node $v_i \in \mathcal{V}$ and terminal edges (s, v_i) and (v_i, t) with weights $(|\phi| - 1)\nu_0$ and $(|\phi| - 1)\nu_1$, respectively. That means a terminal weight $w(s, v)$ has one "unit" ν_0 for each possible pairing of v. The remaining edges complete a clique over ϕ and have weight $2\gamma - \nu_0 - \nu_1$. Let $\mathcal{E}_t$ denote the edges connected to t, $\mathcal{E}_s$ the edges connected to s, and let $\mathcal{E}_n = \mathcal{E} \setminus (\mathcal{E}_s \cup \mathcal{E}_t)$. Then, denoting by $\Gamma = \Gamma(\boldsymbol{x}_\phi)$ the cut corresponding to the labeling $\boldsymbol{x}_\phi$,

$$
\begin{aligned}
& f(\Gamma(\boldsymbol{x}_\phi)) && (5.21)\\
&= g\big((|\phi|-1)(|\Gamma \cap \mathcal{E}_s|\nu_0 + |\Gamma \cap \mathcal{E}_t|\nu_1) + |\Gamma \cap \mathcal{E}_n|(2\gamma - \nu_1 - \nu_0)\big) && (5.22)\\
&= g\Big(\sum_{i\in\phi, x_i=0} \nu_0(|\phi|-1) + \sum_{j\in\phi, x_j=1} \nu_1(|\phi|-1) + \sum_{\substack{i,j\in\phi,\\ x_i=1, x_j=0}} (2\gamma - \nu_1 - \nu_0)\Big) && (5.23)\\
&= g\Big(\sum_{\substack{i\in\phi,\\ x_i=1}} \sum_{\substack{j\neq i\in\phi,\\ x_j=1}} \nu_1 + \sum_{\substack{i\in\phi,\\ x_i=1}} \sum_{\substack{j\in\phi,\\ x_j=0}} (\nu_1 + 2\gamma - \nu_1 - \nu_0) && (5.24)\\
&\qquad + \sum_{\substack{i\in\phi,\\ x_i=0}} \sum_{\substack{j\neq i\in\phi,\\ x_j=0}} \nu_0 + \sum_{\substack{i\in\phi,\\ x_i=0}} \sum_{\substack{j\neq i\in\phi,\\ x_j=1}} \nu_0 \Big)\\
&= g\Big(\sum_{\substack{i\neq j\in\phi,\\ x_i=x_j=1}} \nu_1 + \sum_{\substack{i\neq j\in\phi,\\ x_i=x_j=0}} \nu_0 + \sum_{\substack{i,j\in\phi,\\ x_i>x_j}} (2\gamma - \nu_1 - \nu_0 + \nu_1 + \nu_0)\Big) && (5.25)\\
&= g\Big(\sum_{\substack{i\neq j\in\phi,\\ x_i=x_j=1}} \nu_1 + \sum_{\substack{i\neq j\in\phi,\\ x_i=x_j=0}} \nu_0 + \sum_{\substack{i,j\in\phi,\\ x_i\neq x_j}} \gamma\Big) && (5.26)\\
&= \psi_\phi(\boldsymbol{x}_\phi). && (5.27)
\end{aligned}
$$

5.3.3. P^n Potts model

The P^n Potts potential, defined in [Kohli et al., 2009a], is a generalization of the pairwise Potts potential

$$\psi(x_i, x_j) = \begin{cases} \gamma & \text{if } x_i \neq x_j \\ 0 & \text{otherwise.} \end{cases} \tag{5.28}$$

The P^n potential of a clique ϕ is, for constants $\gamma_{\max} > \gamma_k$,

$$\psi_\phi(\boldsymbol{x}_\phi) = \begin{cases} \gamma_k & \text{if } x_i = k \text{ for all } i \in \phi \\ \gamma_{\max} & \text{otherwise.} \end{cases} \tag{5.29}$$

The P^n Potts model is generally used with more than two labels, so we address swap and expansion moves here. Swap moves reduce to the binary label case, because it is impossible to change the cost $\gamma_{\max}$ if any variable in ϕ has a label $x_i' \notin \{\mathfrak{a}, \mathfrak{b}\}$. To reduce a binary P^n potential to a cooperative cut potential, we construct a complete graph with a node v_i for each i in ϕ. The intra-clique edges all carry weight $\gamma_{\max}$. Then we add terminal nodes s and t, and edges (s, v_i) and (v_i, t) for each node v_i, with weight $w(s, v_i) = \gamma_{\mathfrak{a}}$, and $w(v_i, t) = \gamma_{\mathfrak{b}}$. As the submodular cut cost function, we define the function $f : 2^{\mathcal{E}} \to \mathbb{R}_+$,

$$f(S) = \max_{e\in S} w(e). \tag{5.30}$$

Given a minimum cooperative cut C, we recover a labeling $\boldsymbol{x}$ as follows: all nodes v_i for which $(s, v_i) \in C$ receive label $x_i = \mathfrak{a}$, all others $x_j = \mathfrak{b}$. If there is a pair v_i, v_j where C separates v_i from s (thus, $x_i = \mathfrak{a}$) and v_j from t (thus, $x_j = \mathfrak{b}$), then C must also cut an edge between x_i and x_j with weight $\gamma_{\max}$, and then $f(\Gamma(\boldsymbol{x})) = \gamma_{\max}$.

A reduction for expansion moves with respect to a label $\mathfrak{a}$ can be constructed analogously. The only change is that the weight of all t-edges is $\gamma_{\mathfrak{b}}$ if initially, all nodes in the clique are labeled $\mathfrak{b} \neq \mathfrak{a}$, and $\gamma_{\max}$ otherwise.

5.3.4. Robust P^n potentials

The P^n Potts potential sharply penalizes even a single deviating label in an otherwise uniformly labeled clique. To partially relax this penalty and take into account the number of deviating labels, Kohli et al. [2009b] define *robust P^n potentials.*

Let $N(\boldsymbol{x}_\phi)$ be the number of deviating labels in a clique ϕ, i.e., the number of nodes taking the "minority label", and let $q \leq |\phi|/2$. The robust P^n potential is defined as

$$\psi_\phi(\boldsymbol{x}) = \begin{cases} N(\boldsymbol{x}_\phi)\gamma_{\max}/q & \text{if } N(\boldsymbol{x}_\phi) \leq q \\ \gamma_{\max} & \text{otherwise.} \end{cases} \tag{5.31}$$

To reduce the binary label version of ψ_ϕ to a cooperative cut, we introduce coupling between terminal edges. We construct a graph $\mathcal{G} = (\mathcal{V} \cup \{s, t\}, \mathcal{E})$ with one $v \in \mathcal{V}$ for each member of the clique. Each v is connected to the terminal nodes s, t by edges (s, v), (v, t). Let S_1 be the group of all edges (v_i, t) for $i \in \phi$, and S_2 the group of all edges (s, v_i), and define two edge cost functions

$$f_j(C) = \min\{|C \cap S_j|,\ q\}\gamma_{\max}/q \tag{5.32}$$

for $j = 1, 2$. The thresholded functions f_j are submodular, since they can be viewed as a concave function of a sum of weights, where each edge has weight $\gamma_{\max}/q$. The overall cut cost is then

$$f(C) = f_1(C) + f_2(C), \tag{5.33}$$

and thus, it follows that

$$\begin{aligned} f(\delta(X(\boldsymbol{x}_\phi))) &= f_1(\delta(X(\boldsymbol{x}_\phi))) + f_2(\delta(X(\boldsymbol{x}_\phi))) && (5.34)\\ &= \min\big\{\, |\{i \,|\, x_i = 1\}|,\ q\big\}\gamma_{\max}/q + \min\big\{\, |\{i \,|\, x_i = 0\}|,\ q\big\}\gamma_{\max}/q && (5.35)\\ &= \gamma_{\max} + \min\{N(\boldsymbol{x}_\phi),\ q\}\gamma_{\max}/q && (5.36)\\ &= \psi_\phi(\boldsymbol{x}) + \gamma_{\max}. && (5.37) \end{aligned}$$

Since $\gamma_{\max}$ is constant, the potential ψ_ϕ is equivalent to a cooperative cut.

5.3.5. Co-occurrences of object labels

In class-based image segmentation, each pixel must be labeled to belong to one of many object classes. Ladický et al. [2010] propose a global potential $g_{\mathcal{L}}(L(\boldsymbol{x}))$ on the set of class labels $L(\boldsymbol{x})$ used in the labeling $\boldsymbol{x}$. This potential considers how often certain groups of labels co-occur.

Assume $g_{\mathcal{L}} : 2^{\mathcal{L}} \to \mathbb{R}_+$ is a nondecreasing and submodular function defined on sets of class labels. Then an expansion move for this label potential can be computed as a cooperative cut, as we show next.

When expanding label $\mathfrak{a}$, we create a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ that contains a node v_i for each $x'_i \neq \mathfrak{a}$ and a source and sink node. Figure 5.5 shows the construction. In the figure, each label is indicated by a color, and nodes are colored according to their initial labeling. We connect the source s to each v_i by an edge labeled $\mathfrak{a}$, and we connect each v_i to t by an edge labeled x'_i. That is, we transfer the labels from nodes to edges. If the minimum cooperative cut severs edge (s, v_i), then we set $x_i = \mathfrak{a}$, otherwise edge (v_i, t) is cut and x_i retains its old label. An (s,t)-cut must cut one of those terminal edges. Since we will define a nondecreasing function, there is an optimal cut that cuts only one of the edges.

Let $L(C)$ be the set of labels on the edges in $C \subseteq \mathcal{E}$. If the current labeling does not use the label $\mathfrak{a}$, then we set the submodular cost function on edges to

$$f(C) = g_{\mathcal{L}}(L(C)). \tag{5.38}$$

This is an induced submodular function on subsets of edges. Submodularity follows from Proposition 2.1, if we let $\mathcal{V}_1 = \mathcal{E}$ and $\mathcal{V}_2 = \mathcal{L}$, and if we connect each node in $\mathcal{V}_1$ (corresponding to an edge in $\mathcal{E}$) to its label in $\mathcal{V}_2$.

If the current labeling $\boldsymbol{x}'$ already uses the label $\mathfrak{a}$, i.e., there is an i with $x'_i = \mathfrak{a}$, then we set the submodular cost function on edges to

$$f(C) = g_{\mathcal{L}}(L(C) \cup \{\mathfrak{a}\}). \tag{5.39}$$

Submodularity follows as above. This construction is similar to the node-based construction in [Ladický et al., 2010]. If the minimum cooperative cut C cuts edge (s, v_i), then $\mathfrak{a} \in L(C)$, and we set $x_i = \mathfrak{a}$. Otherwise, C cuts edge (v_i, t), and then $x_i = x'_i \in L(C)$. Thus, the cut cost is equivalent to the resulting labeling of nodes.

The construction for a swap move is very similar and uses only nodes that are initially labeled $\mathfrak{a}$ or $\mathfrak{b}$; Figure 5.5 illustrates the graph.

The function $g_{\mathcal{L}}$ defined by Ladický et al. [2010], however, is not submodular. We propose an alternative submodular function. Let $I_1, \ldots, I_M$ be a set of training images, and $L_I(I_k)$ the set of labels occurring in the correct labeling of image I_k. We define $g_{\mathcal{L}}$ to count the number of images that do not contain all labels in L together:

$$g_{\mathcal{L}}(L) = \sum_{k=1}^{M} \mathbf{1}[L \nsubseteq L_I(I_k)], \tag{5.40}$$

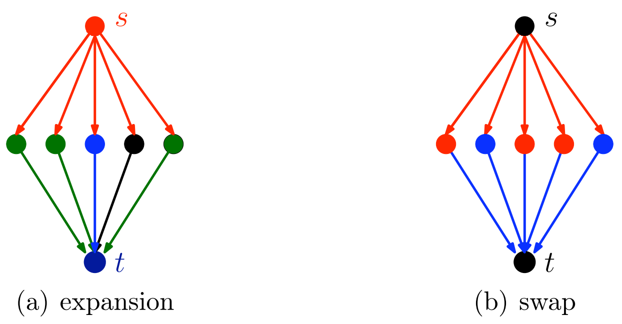


Figure 5.5. Graph construction for expansion and swap moves for class co-occurrence costs. The nodes v_i are aligned in the middle. Here, colors denote labels. Each node v_i is colored according to its initial label x'_i. If edge (s, v_i) is cut, then we set $x_i = \mathfrak{a}$ (red label), otherwise we (a) keep the old label x'_i or (b) set $x_i = \mathfrak{b}$ (blue label).

This is a submodular function, again by Proposition 2.1: Let $\mathcal{V}_1$ be all labels in $\mathcal{L}$, and let $\mathcal{V}_2$ consist of all images. Now, connect all labels to the images in which they do not occur. Then $g_{\mathcal{L}}(L)$ is the size of the neighborhood of L in the bipartite graph.

As another example, the label cost part of energy $(\star)$ in [Delong et al., 2011] is a submodular function on subsets of labels. Given label sets $S \subseteq \mathcal{L}$, and constant subset costs $h_S \geq 0$, their function $g_{\mathcal{L}}$ is defined as

$$g_{\mathcal{L}}(L) = \sum_{S} h_S \mathbf{1}[L \cap S \neq \emptyset]. \tag{5.41}$$

Submodularity again follows from Proposition 2.1, by setting $\mathcal{V}_1 = \mathcal{L}$ and $\mathcal{V}_2$ to the set of all S for which $h_S > 0$. The edges in the bipartite graph connect each label $\ell \in \mathcal{L}$ to the subsets $S \ni \ell$ that include it. Delong et al. [2011] discuss optimization methods for and applications of this energy, and relations to uncapacitated facility location. Note that the constructions with cooperative cut neither add additional nodes nor dense cliques to the graph; this can be an advantage for higher-order label interactions.

5.4. Multi-label cooperative cut energies

Motivated by the success of graph cuts for multi-label problems, we investigate multi-label models for cooperative cut energies and design move-making algorithms for those energies.

5.4.1. Models of multi-label cooperative cut energies

In the sequel, we assume the typical grid graph structure that is used for images. We limit the presentation to graphs that have modular costs on terminal edges and

potential	original ref.	edges	$f(C)$
Graph Cut	Boykov and Jolly [2001]	$\mathcal{E}_n$	$w(C)$
congruent boundaries	Chapter 6	groups of grid	$\sum_S g_\theta(w(C \cap S))$
(binary) P^n fct.	Kohli et al. [2007]	$\mathcal{E}_n$	$g(\lvert C\rvert)$
P^n Potts	Kohli et al. [2007]	$\mathcal{E}$	$\max_{e \in C} w(e)$
robust P^n	Kohli et al. [2009b]	$\mathcal{E}_t$	$\sum_j \min\{\lvert C \cap S_j\rvert,\ q\}\gamma/q$
random walker (discretized)	Grady [2006]	$\mathcal{E}$	$\sqrt{w^2(C)}$
ℓ_∞	Sinop and Grady [2007]	$\mathcal{E}_n$	$\max_{e \in C \cap \mathcal{E}_n} w(e)$
class labels	Ladický et al. [2010], Delong et al. [2011]	$\mathcal{E}_t$	$g_{\mathcal{L}}(\bigcup_{e \in C} \ell(e))$
general submodular	Chapter 7	$\mathcal{E}_t$	polymatroid part of energy

Table 5.1. Examples of cooperative cuts explained in Sections 5.3 and 6.2; $\ell(e)$ is the label of edge e, and $w(C)$ ($w^2(C)$) the sum of (squared) weights. The third column indicates the domain of the function in the fourth column. The edges $\mathcal{E}_t$ are those adjacent to a terminal node, and $\mathcal{E}_n = \mathcal{E} \setminus \mathcal{E}_t$. With the exception of standard graph cuts, these edges are coupled by a non-modular function.

submodular costs on other edges. This limitation is in conformity with the model that will be introduced in Chapter 6. The theory here extends to more general models too.

If binary cooperative cut energies are defined by a cut in a directed graph, then the energy equals the cost of the set of edges going from a label-1 node in $X_1 = \{v_i \mid x_i = 1\} \cup \{s\}$ to a label-zero node in $X_0 = \{v_i \mid x_i = 0\} \cup \{t\}$. We mention two possible generalizations to more than two labels. In each case, we assume to have a structure graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$.

M1 We define a cooperative multi-cut, and allow coupling between all cut edges. For a labeling $\boldsymbol{x} \in \mathcal{L}^n$, let the corresponding (undirected) multiway cut be

$$\Gamma(\boldsymbol{x}) = \{e = (v_i, v_j) \mid x_i \neq x_j\}.$$

For a nondecreasing submodular function $f : 2^{\mathcal{E}} \to \mathbb{R}_+$, this cut induces an energy

$$E_f(\boldsymbol{x}) = \sum_i \psi_i(x_i) + f(\Gamma(\boldsymbol{x})). \tag{5.42}$$

M2 The boundary of one label is independent of the boundary of other labels (objects). In a directed structure graph $\mathcal{G}$, we define the sets $X_{\mathfrak{a}} = \{v_i \in \mathcal{V} \mid x_i = \mathfrak{a}\}$ of nodes with label $\mathfrak{a}$. Then the *boundary* of label $\mathfrak{a}$ is

$$\Gamma_{\mathfrak{a}}(\boldsymbol{x}) = \delta(X_{\mathfrak{a}}) = \{(v_i, v_j) \in \mathcal{E} \mid x_i = \mathfrak{a}, x_j \neq \mathfrak{a}\}.$$

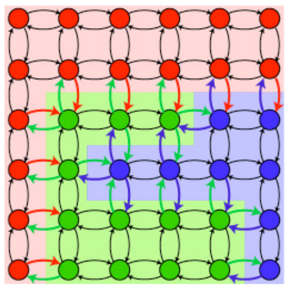


Figure 5.6 Illustration of the boundaries used in M2: $\Gamma_{\text{blue}}(\boldsymbol{x})$, $\Gamma_{\text{red}}(\boldsymbol{x})$ and $\Gamma_{\text{green}}(\boldsymbol{x})$.

For $|\mathcal{L}|$ (identical or different) nondecreasing submodular functions, we define a label-sensitive energy

$$E_f(\boldsymbol{x}) = \sum_i \psi_i(x_i) + \sum_{\mathfrak{a} \in \mathcal{L}} f_{\mathfrak{a}}(\Gamma_{\mathfrak{a}}(\boldsymbol{x})). \tag{5.43}$$

For the algorithms in the sequel, we focus on M2, but analogous results hold for M1. Similarly, we only discuss expansion moves, but swaps are also possible. The model M2 implicitly favors fewer labels: the joint cost of edges coupled by one function $f_{\mathfrak{a}}$ is lower than the sum of their individual costs, and coupling is only possible for edges that are in the boundary of the same label. At least if all $f_{\mathfrak{a}}$ are the same, then using few labels means that the edges in the cut are distributed over fewer boundaries $\Gamma_{\mathfrak{a}}(\boldsymbol{x})$, and more coupling and resulting discounts are possible.

We can take two routes to optimize these energies. One option is to derive an adaptive approximation as in Section 4.2.3, which results in a sequence of approximate pairwise energy function E_h and leads to an iterative algorithm. We show this approximation in Section 5.4.2. Any algorithm for multi-label pairwise MRFs applies to minimize E_h. In particular, we show expansion moves and an approximation bound. Alternatively, we can directly build on the cut formulation and move-making algorithms [Boykov et al., 2001]. In Section 5.4.4 we show that the best expansion move can be found as a minimum cooperative cut. Hence, any approximation algorithm for cooperative cut finds a move with an approximation guarantee.

5.4.2. A pairwise approximation

Pursuing the first option, we derive an approximation $\widehat{E}$ of the energy E by applying Lemma 4.4 to each $f_{\mathfrak{a}}$ separately. Let $\boldsymbol{x}'$ be the current labeling, and define the shorthand $\Gamma'_{\mathfrak{a}} = \Gamma_{\mathfrak{a}}(\boldsymbol{x}') = \delta(X'_{\mathfrak{a}})$. We use the marginal costs $\rho^{\mathfrak{a}}(e|A) = f_{\mathfrak{a}}(A \cup \{e\}) - f_{\mathfrak{a}}(A)$ for each function $f_{\mathfrak{a}}$ and all sets $A \subseteq \mathcal{E}$. Analogously to the binary case, we obtain upper bounds for each possible re-labeling $\boldsymbol{x}$ and the associated boundaries $\Gamma_{\mathfrak{a}} = \Gamma_{\mathfrak{a}}(\boldsymbol{x})$:

$$\begin{aligned}
f_{\mathfrak{a}}(\Gamma_{\mathfrak{a}}) &\leq h_{\mathfrak{a},\Gamma'}(\Gamma_{\mathfrak{a}}) && (5.44)\\
&= f_{\mathfrak{a}}(\Gamma'_{\mathfrak{a}}) + \sum_{e\in\Gamma_{\mathfrak{a}}\setminus\Gamma'_{\mathfrak{a}}} \rho^{\mathfrak{a}}(e|\Gamma'_{\mathfrak{a}}) - \sum_{e\in\Gamma'_{\mathfrak{a}}\setminus\Gamma_{\mathfrak{a}}} \rho^{\mathfrak{a}}(e|\mathcal{E}\setminus e) && (5.45)\\
&= \underbrace{f_{\mathfrak{a}}(\Gamma'_{\mathfrak{a}})}_{\text{const.}} + \sum_{e\in\Gamma_{\mathfrak{a}}\setminus\Gamma'_{\mathfrak{a}}} \rho^{\mathfrak{a}}(e|\Gamma'_{\mathfrak{a}}) \underbrace{- \sum_{e\in\Gamma'_{\mathfrak{a}}} \rho^{\mathfrak{a}}(e|\mathcal{E}\setminus e)}_{\text{const.}} + \sum_{e\in\Gamma'_{\mathfrak{a}}\cap\Gamma_{\mathfrak{a}}} \rho^{\mathfrak{a}}(e|\mathcal{E}\setminus e) && (5.46)\\
&= const + \sum_{e\in\Gamma_{\mathfrak{a}}\setminus\Gamma'_{\mathfrak{a}}} \rho^{\mathfrak{a}}(e|\Gamma'_{\mathfrak{a}}) + \sum_{e\in\Gamma'_{\mathfrak{a}}\cap\Gamma_{\mathfrak{a}}} \rho^{\mathfrak{a}}(e|\mathcal{E}\setminus e) && (5.47)\\
&= const + \sum_{e\in\Gamma_{\mathfrak{a}}} w_{\mathfrak{a}}(e). && (5.48)
\end{aligned}$$

The last sum shows that $h_{\mathfrak{a},\Gamma'}$ is a sum of weights over the edges in the boundary $\Gamma_{\mathfrak{a}}(\boldsymbol{x})$. Each $f_{\mathfrak{a}}$ defines its own edge weights $w_{\mathfrak{a}}$, analogously to the weights in Equation (4.29). If $f_{\mathfrak{a}}$ is nondecreasing, then $w_{\mathfrak{a}} \geq 0$ for all labels $\mathfrak{a}$ and all edges. The functions $h_{\mathfrak{a},\Gamma'}$ define an approximation of the cooperative cut energy E_f:

$$\begin{aligned}
\widehat{E}_{\Gamma'}(\boldsymbol{x}) &= \sum_i \psi_i(x_i) + \sum_{\mathfrak{a}} h_{\mathfrak{a},\Gamma'}(\Gamma_{\mathfrak{a}}(\boldsymbol{x})) - const && (5.49)\\
&= \sum_i \psi_i(x_i) + \sum_{i,j\in\mathcal{N}} \psi_{ij,\Gamma'}(x_i, x_j) && (5.50)
\end{aligned}$$

with pairwise potentials

$$\psi_{ij,\Gamma'}(x_i, x_j) = \begin{cases} 0 & \text{if } x_i = x_j,\\ w_{\mathfrak{a}}((v_i, v_j)) + w_{\mathfrak{b}}((v_j, v_i)) & \text{if } x_i = \mathfrak{a}, x_j = \mathfrak{b}. \end{cases} \qquad (5.51)$$

The approximation $\widehat{E}_{\Gamma'}$ is an asymmetric, pairwise energy that can be represented by a directed graph cut with nonnegative edge weights. Therefore, it is a submodular energy. We use these observations to construct expansion moves for the approximation $\widehat{E}_{\Gamma'}$. In the sequel, we drop the subscript Γ' as the approximation always refers to the current labeling $\boldsymbol{x}'$.

5.4.3. Expansion moves for the approximation $\widehat{E}_{\Gamma'}$

To construct expansion moves for multi-label cooperative cut energies, we build on the work by Boykov et al. [2001]. Their construction does not suit $\widehat{E}$ directly because $\widehat{E}$ is not symmetric. We therefore modify their construction and, for M2, we propose a simpler construction that uses only $n + 2$ nodes. The original construction needs an auxiliary node for each pair of variables $x_i \neq x_j$ for which there is a pairwise potential.

Recall that an expansion move with respect to a label $\mathfrak{a}$ and a given current assignment $\boldsymbol{x}'$ finds the best relabeling

$$\boldsymbol{x}^{\mathfrak{a}} \in \operatorname{argmin} \widehat{E}_{\Gamma'}(\boldsymbol{x}) \quad \text{s.t. } \boldsymbol{x} \in \mathcal{X}(\boldsymbol{x}', \mathfrak{a}). \tag{5.52}$$

Lemma 5.5. *Given a current assignment $x' \in \mathcal{L}^n$ and any label $\mathfrak{a}$, a minimizer $\boldsymbol{x}^{\mathfrak{a}} \in \mathcal{X}(\boldsymbol{x}', \mathfrak{a})$ of $\widehat{E}_{\Gamma'}$ can be found via a minimum cut in a graph.*

We prove Lemma 5.5 for the general case in Appendix B.2. The construction in the appendix generalizes the widely used construction by Boykov et al. [2001] to non-symmetric potentials. Here, we focus on M2 and observe that the potentials (5.51) of $\widehat{E}$ for M2 have a particular composite form. They are a sum of two edge weights, and any such weight $w_{\mathfrak{a}}((v_i, v_j))$ only depends on the edge (v_i, v_j) and on the label $x_i = \mathfrak{a}$ of the tail node. To emphasize the independence of $w_{\mathfrak{a}}((v_i, v_j))$ from x_j, we denote it by $w_{\mathfrak{a}}(i, j)$. This form admits a more efficient construction than the classical one[4].

Lemma 5.6. *If $E : \mathcal{L}^n \to \mathbb{R}_+$ is an energy function of order two with pairwise potentials of the form*

$$\psi_{ij}(x_i, x_j) = \begin{cases} 0 & \text{if } x_i = x_j, \\ w_{\mathfrak{a}}(i,j) + w_{\mathfrak{b}}(j,i) & \text{if } x_i = \mathfrak{a}, x_j = \mathfrak{b} \ (\mathfrak{a} \neq \mathfrak{b}), \end{cases} \tag{5.53}$$

with $w_{\mathfrak{a}} \geq 0$ for all $\mathfrak{a} \in \mathcal{L}$, then the optimal expansion move can be found by solving a minimum cut in a graph with only $n+2$ nodes.

To prove Lemmas 5.5 and 5.6, we construct an auxiliary graph G and show the following two claims. The Lemma follows as a corollary.

Claim 5.1. *Any minimal cut in G corresponds to a labeling that is within one expansion (with respect to $\mathfrak{a}$) of the current labeling $\boldsymbol{x}'$.*

Claim 5.2. *The cost of any such cut is equal to the energy of the associated labeling.*

We construct G to satisfy an equivalent to Property 4.2 in [Boykov et al., 2001].

Property 5.1. *For a minimal cut C in G, the following holds:*

1. *if $(s, \mathsf{v}_i), (s, \mathsf{v}_j) \in C$, then $(\mathsf{v}_i, \mathsf{v}_j), (\mathsf{v}_j, \mathsf{v}_i) \notin C$;*
2. *if $(\mathsf{v}_i, t), (\mathsf{v}_j, t) \in C$, then $(\mathsf{v}_i, \mathsf{v}_j), (\mathsf{v}_j, \mathsf{v}_i) \notin C$;*
3. *if $(s, \mathsf{v}_i), (\mathsf{v}_j, t) \in C$, then $\mathcal{E} \cap \{ (\mathsf{v}_j, \mathsf{v}_i)\} \subseteq C$, $(\mathsf{v}_i, \mathsf{v}_j) \notin C$;*
4. *for each v_i, either $(s, \mathsf{v}_i) \in C$ or $(\mathsf{v}_i, t) \in C$.*

[4]It turns out that a related symmetric version of our result, which corresponds to the special case that $w_{\mathfrak{a}}$ is independent of the label $\mathfrak{a}$, has been known for long [Milis, 1996], as was recently brought to our notice by R. Zabih.

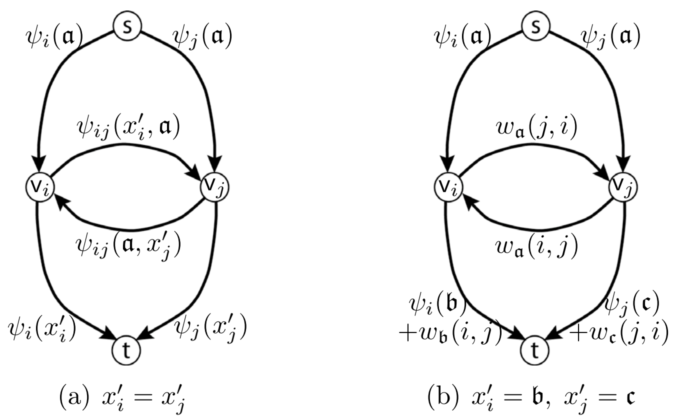


Figure 5.7. Graph construction for expanding label $\mathfrak{a}$ for the approximate energy $\widehat{E}_{\Gamma'}$. If $x_i' = \mathfrak{a}$, we set the weight of edge (v_i, t) to ∞ rather than to $\psi_i(\mathfrak{a})$, and equally for $x_j' = \mathfrak{a}$.

Proof (Lemma 5.6). We can assume that the unary potentials are nonnegative, as they can be shifted without affecting optima. Given a structure graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we construct an auxiliary graph G with terminal nodes s and t, and one node v_i for each variable x_i. Let $\boldsymbol{x}'$ be the current assignment, and $\mathfrak{a}$ be the label we expand.

For each $v_i \in \mathcal{V}$, we introduce edges (s, v_i) with weight $\psi_i(\mathfrak{a})$ and (v_i, t) with weight $\psi_i(x_i')$. If a variable already has label $\mathfrak{a}$, $x_i' = \mathfrak{a}$, then we set the weight of edge (v_i, t) to infinity, so that node v_i is always in the partition belonging to t. We infer a labeling from an (s,t)-cut as follows: if edge (s, v_i) is cut, then we set $x_i = \mathfrak{a}$, otherwise edge (v_i, t) is cut, and we retain the label $x_i = x_i'$. If the weight $w(\mathsf{v}_i, t) = \infty$, then any minimum (s,t)-cut assigns v_i to t and yields $x_i = \mathfrak{a}$. This implies a bijection between minimal cuts with finite weight and assignments in $\mathcal{X}(\boldsymbol{x}', \mathfrak{a})$, and therefore Claim 5.1.

Now we set the remaining edges such that they implement the potentials ψ_{ij}. If the current labeling is $x_i' = x_j'$, then we introduce edges $(\mathsf{v}_i, \mathsf{v}_j)$ with weight $\psi_{ij}(x_i', \mathfrak{a})$ and $(\mathsf{v}_j, \mathsf{v}_i)$ with weight $\psi_{ij}(\mathfrak{a}, x_j')$, as shown in Figure 5.7(a). If then later both v_i and v_j are assigned to the same partition, i.e., $x_i = x_j$, then none of the two edges is cut, in conformity with $\psi_{ij}(x_i, x_j) = 0$ if $x_i = x_j$. If $x_i = \mathfrak{a}$ and $x_j = x_j'$, i.e., (s, v_i) is cut, then edge $(\mathsf{v}_j, \mathsf{v}_i)$ must be cut as well, contributing the cost $\psi_{ij}(\mathfrak{a}, x_j')$. Conversely, if $x_i = x_i'$ and $x_j = \mathfrak{a}$, then the reverse edge contributes $\psi_{ij}(x_i', \mathfrak{a})$ to the cut.

If $x_i' \neq x_j'$, then we represent the term ψ_{ij} by the sub-graph shown in Figure 5.7(b). Let $\mathfrak{b} = x_i'$ and $\mathfrak{c} = x_j'$ be the current labels. We add $w_\mathfrak{b}(i,j)$ to the weight of edge (v_i, t) (whose cutting corresponds to assigning $x_i = x_i' = \mathfrak{b}$), and $w_\mathfrak{c}(j,i)$ to the weight of edge (v_j, t). Note that these weights must be added to the

Algorithm 4 Expansion moves for multi-label cooperative cuts

```
**Input:** $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $f_{\mathfrak{a}}$ for all $\mathfrak{a} \in \mathcal{L}$.
**Output:** labeling $\hat{\boldsymbol{x}} \in \mathcal{L}^n$.
set initial $\boldsymbol{x}_0 = \{\mathfrak{b}\}^n$, $i = 0$
**repeat**
  define $\widehat{E}_{\Gamma_i}$ for $\Gamma_i = \Gamma(\boldsymbol{x}_i)$
  $\boldsymbol{x}_{i+1} = \text{Expansions}(\boldsymbol{x}_i, \widehat{E}_{\Gamma_i})$ (Algorithm 5)
  set $i = i + 1$
**until** $\boldsymbol{x}_i = \boldsymbol{x}_{i-1}$ or $E(\boldsymbol{x}_i) \geq E(\boldsymbol{x}_{i-1})$
return $\hat{\boldsymbol{x}} = \boldsymbol{x}_i$
```

terminal edges for each ψ_{ij} whose current arguments differ. Furthermore, we add edges $(\mathsf{v}_i, \mathsf{v}_j)$ with weight $w_{\mathfrak{a}}(j, i)$, and $(\mathsf{v}_j, \mathsf{v}_i)$ with weight $w_{\mathfrak{a}}(i, j)$.

The Lemma now follows from Property 5.1. Enumeration shows Claim 5.2: the weight of each minimal cut is the same as the energy $\widehat{E}(\boldsymbol{x})$ for the associated assignment $\boldsymbol{x}$ where $x_i = \mathfrak{a}$ if (s, v_i) is cut, and $x_i = x_i'$ otherwise. Claims 5.1 and 5.2 imply that the minimum energy labelings in $\mathcal{X}(\boldsymbol{x}', \mathfrak{a})$ correspond to the minimum cuts in G. □

Approximation factor

Equipped with $\widehat{E}$ and expansion moves, we can now put pieces together to build an algorithm for approximately minimizing multi-label cooperative cut energies. The result is Algorithm 4.

We analyze this algorithm in two steps. Lemma 5.7 addresses the expansion moves (Algorithm 5), and Lemma 5.8 combines this with the impact of $\widehat{E}$, completing the bound for Algorithm 4. Edges here refer to the structure graph that defines the cooperative cut energy.

Lemma 5.7. *Let $\hat{\boldsymbol{x}}$ be a local minimum reached via a sequence of expansion moves, i.e., $\hat{\boldsymbol{x}} \in \operatorname{argmin}_{\boldsymbol{x} \in \mathcal{X}(\hat{\boldsymbol{x}}, \mathfrak{a})} \widehat{E}(\boldsymbol{x})$ for all $\mathfrak{a} \in \mathcal{L}$. Let also $\boldsymbol{x}^i = \operatorname{argmin}_{\boldsymbol{x} \in \mathcal{L}^n} \widehat{E}(\boldsymbol{x})$. Then*

$$\widehat{E}(\hat{\boldsymbol{x}}) \leq (1 + \gamma) \widehat{E}(\boldsymbol{x}^i),$$

$$\textit{where} \qquad \gamma = \max_{e \in \mathcal{E}, \mathfrak{c}, \mathfrak{b} \in \mathcal{L}} \frac{w_{\mathfrak{c}}(e)}{w_{\mathfrak{b}}(e)} \leq \max_{e, \mathfrak{c}, \mathfrak{b}} \frac{\rho^{\mathfrak{c}}(e | \Gamma_i)}{\rho^{\mathfrak{b}}(e | \mathcal{E} \setminus e)}.$$

For the first iteration of Algorithm 4, it holds that $\gamma \leq \max_{e, \mathfrak{b}, \mathfrak{c}} f_{\mathfrak{b}}(e)/f_{\mathfrak{c}}(e)$. If $f_{\mathfrak{a}} = f_{\mathfrak{b}}$ for all $\mathfrak{a}, \mathfrak{b} \in \mathcal{L}$, then $\gamma = 1$.

The bound is nontrivial for Algorithm 4, because the cut in the first iteration uses no coupled edges (only terminal edges) $\Gamma_i \subseteq \mathcal{E}_t$, and then $w_{\mathfrak{b}}(e) = f_{\mathfrak{b}}(e) > 0$ for all edges and labels $\mathfrak{b}$. This initial approximation bound transfers to later iterations of Algorithm 4, where successive iterations only decrease the energy of

Algorithm 5 Expansions

```
Input: labeling $\boldsymbol{x}_{in} \in \mathcal{L}^n$ and pairwise energy $\widehat{E}$
Output: labeling $\boldsymbol{x}$
set initial $\boldsymbol{x}_0 = \boldsymbol{x}_{in}$, $i = 0$
repeat
  set $j = 0$, $\boldsymbol{x}_j = \boldsymbol{x}_i$ and $i = i + 1$
  for $\mathfrak{a} \in \mathcal{L}$ do
    find $\boldsymbol{x}_{j+1} \in \operatorname{argmin}\{\widehat{E}(\boldsymbol{x}_j) \mid \boldsymbol{x} \in \mathcal{X}(\boldsymbol{x}_j, \mathfrak{a})\}$
    $j = j + 1$
  end for
  set $\boldsymbol{x}_i = \boldsymbol{x}_j$
until $\boldsymbol{x}_i = \boldsymbol{x}_{i-1}$ or $\widehat{E}(\boldsymbol{x}_i) \geq \widehat{E}(\boldsymbol{x}_{i-1})$
return $\boldsymbol{x} = \boldsymbol{x}_i$
```

the initial labeling. The proof of Lemma 5.7 relies on a similar strategy as the proof of Theorem 6.1 in [Boykov et al., 2001].

Proof. First, we recall that $\widehat{E}$ can be written as a sum over label boundaries,

$$\widehat{E}(\boldsymbol{x}) = \sum_{\mathfrak{a} \in \mathcal{L}} \sum_{e \in \Gamma_{\mathfrak{a}}(\boldsymbol{x})} w_{\mathfrak{a}}(e) \triangleq \sum_{\mathfrak{a} \in \mathcal{L}} w_{\mathfrak{a}}(\Gamma_{\mathfrak{a}}(\boldsymbol{x})). \tag{5.54}$$

Let for all labels $\mathfrak{a} \in \mathcal{L}$ the set $X^*_{\mathfrak{a}} = \{v_j \mid x^i_j = \mathfrak{a}\}$ contain the nodes labeled $\mathfrak{a}$ in the minimizer of $\widehat{E}$. Define the edge sets

$$B_{\mathfrak{a},\mathfrak{b}} = (X^*_{\mathfrak{a}} \times X^*_{\mathfrak{b}}) \cap \mathcal{E} \tag{5.55}$$
$$O_{\mathfrak{a}} = ((\mathcal{V} \setminus X^*_{\mathfrak{a}}) \times (\mathcal{V} \setminus X^*_{\mathfrak{a}})) \cap \mathcal{E} \tag{5.56}$$
$$I_{\mathfrak{a}} = (X^*_{\mathfrak{a}} \times X^*_{\mathfrak{a}}) \cap \mathcal{E}. \tag{5.57}$$

Furthermore, we denote restrictions of unary terms to a set $Y \subseteq \mathcal{V}$ of modes and of pairwise terms to a set $A \subseteq \mathcal{E}$ of edges by

$$\widehat{E}^u|_Y(\boldsymbol{x}) = \sum_{v_j \in Y} \psi_j(x_j) \tag{5.58}$$
$$\widehat{E}|_A(\boldsymbol{x}) = \sum_{\mathfrak{a} \in \mathcal{L}} \sum_{e \in \Gamma_{\mathfrak{a}}(\boldsymbol{x}) \cap A} w_{\mathfrak{a}}(e). \tag{5.59}$$

Fix an arbitrary label $\mathfrak{a}$, and define an assignment $\boldsymbol{x}^{\mathfrak{a}}$ that agrees with the optimal assignment for $\mathfrak{a}$, and otherwise with $\hat{\boldsymbol{x}}$:

$$x^{\mathfrak{a}}_j = \begin{cases} \mathfrak{a} & \text{if } x^i_j = \mathfrak{a}, \\ \hat{x}_i & \text{otherwise.} \end{cases} \tag{5.60}$$

This assignment lies in $\mathcal{X}(\hat{\boldsymbol{x}}, \mathfrak{a})$, i.e., it can be reached within one expansion move with respect to $\mathfrak{a}$. Since by definition $\hat{\boldsymbol{x}}$ is a minimizer of $\widehat{E}$ in $\mathcal{X}(\hat{\boldsymbol{x}}, \mathfrak{a})$, it holds that $\widehat{E}(\hat{\boldsymbol{x}}) \leq \widehat{E}(\boldsymbol{x}^{\mathfrak{a}})$. This implies that

$$\begin{aligned}
\widehat{E}(\hat{\boldsymbol{x}}) &= \widehat{E}^u|_{X^*_{\mathfrak{a}}}(\hat{\boldsymbol{x}}) + \widehat{E}^u|_{\bar{X}^*_{\mathfrak{a}}}(\hat{\boldsymbol{x}}) + \widehat{E}|_{O_{\mathfrak{a}}}(\hat{\boldsymbol{x}}) + \widehat{E}|_{I_{\mathfrak{a}}}(\hat{\boldsymbol{x}}) \\
&\qquad + \sum_{\mathfrak{b}\neq\mathfrak{a}} \sum_{\mathfrak{c}\in\mathcal{L}} w_{\mathfrak{c}}(\Gamma_{\mathfrak{c}}(\hat{\boldsymbol{x}}) \cap B_{\mathfrak{a},\mathfrak{b}}) + \sum_{\mathfrak{b}\neq\mathfrak{a}} \sum_{\mathfrak{c}\in\mathcal{L}} w_{\mathfrak{c}}(\Gamma_{\mathfrak{c}}(\hat{\boldsymbol{x}}) \cap B_{\mathfrak{b},\mathfrak{a}}) \qquad (5.61) \\
&\leq \widehat{E}^u|_{X^*_{\mathfrak{a}}}(\boldsymbol{x}^{\mathfrak{a}}) + \widehat{E}^u|_{\bar{X}^*_{\mathfrak{a}}}(\boldsymbol{x}^{\mathfrak{a}}) + \widehat{E}|_{O_{\mathfrak{a}}}(\boldsymbol{x}^{\mathfrak{a}}) + \widehat{E}|_{I_{\mathfrak{a}}}(\boldsymbol{x}^{\mathfrak{a}}) \\
&\qquad + \sum_{\mathfrak{b}\neq\mathfrak{a}} \sum_{\mathfrak{c}\in\mathcal{L}} w_{\mathfrak{c}}(\Gamma_{\mathfrak{c}}(\boldsymbol{x}^{\mathfrak{a}}) \cap B_{\mathfrak{a},\mathfrak{b}}) + \sum_{\mathfrak{b}\neq\mathfrak{a}} \sum_{\mathfrak{c}\in\mathcal{L}} w_{\mathfrak{c}}(\Gamma_{\mathfrak{c}}(\boldsymbol{x}^{\mathfrak{a}}) \cap B_{\mathfrak{b},\mathfrak{a}}) \qquad (5.62) \\
&= \widehat{E}^u|_{X^*_{\mathfrak{a}}}(\boldsymbol{x}^{i}) + \widehat{E}^u|_{\bar{X}^*_{\mathfrak{a}}}(\hat{\boldsymbol{x}}) + \widehat{E}|_{O_{\mathfrak{a}}}(\hat{\boldsymbol{x}}) + \widehat{E}|_{I_{\mathfrak{a}}}(\boldsymbol{x}^{i}) \\
&\qquad + \sum_{\mathfrak{b}\neq\mathfrak{a}} w_{\mathfrak{a}}(\Gamma_{\mathfrak{a}}(\boldsymbol{x}^{\mathfrak{a}}) \cap B_{\mathfrak{a},\mathfrak{b}}) + \sum_{\mathfrak{b}\neq\mathfrak{a}} \sum_{\mathfrak{c}\in\mathcal{L}} w_{\mathfrak{c}}(\Gamma_{\mathfrak{c}}(\boldsymbol{x}^{\mathfrak{a}}) \cap B_{\mathfrak{b},\mathfrak{a}}). \qquad (5.63)
\end{aligned}$$

Here, we used the notation $\bar{X}^*_{\mathfrak{a}} = \mathcal{V} \setminus X^*_{\mathfrak{a}}$. The last equality relies on $x^{\mathfrak{a}}_j = x^i_j$ for all $j \in X^*_{\mathfrak{a}}$, and $x^{\mathfrak{a}}_j = \hat{x}_j$ for all $j \in \bar{X}^*_{\mathfrak{a}}$. These equalities imply that $\widehat{E}^u|_{X^*_{\mathfrak{a}}}(\boldsymbol{x}^{\mathfrak{a}}) = \widehat{E}^u|_{X^*_{\mathfrak{a}}}(\boldsymbol{x}^{i})$, $\widehat{E}|_{I_{\mathfrak{a}}}(\boldsymbol{x}^{\mathfrak{a}}) = \widehat{E}|_{I_{\mathfrak{a}}}(\boldsymbol{x}^{i})$, $\widehat{E}^u|_{\bar{X}^*_{\mathfrak{a}}}(\boldsymbol{x}^{\mathfrak{a}}) = \widehat{E}^u|_{\bar{X}^*_{\mathfrak{a}}}(\hat{\boldsymbol{x}})$ and $\widehat{E}|_{O_{\mathfrak{a}}}(\boldsymbol{x}^{\mathfrak{a}}) = \widehat{E}|_{O_{\mathfrak{a}}}(\hat{\boldsymbol{x}})$. They also imply that $\Gamma_{\mathfrak{c}}(\boldsymbol{x}^{\mathfrak{a}}) \cap B_{\mathfrak{a},\mathfrak{b}} = \emptyset$ for $\mathfrak{c} \neq \mathfrak{a}$ and any $\mathfrak{b} \in \mathcal{L}$. In summary, the result (5.63) shows that, for any $\mathfrak{a} \in \mathcal{L}$,

$$\begin{aligned}
&\widehat{E}^u|_{X^*_{\mathfrak{a}}}(\hat{\boldsymbol{x}}) + \widehat{E}|_{I_{\mathfrak{a}}}(\hat{\boldsymbol{x}}) + \sum_{\mathfrak{b},\mathfrak{c}\in\mathcal{L}} w_{\mathfrak{c}}(\Gamma_{\mathfrak{c}}(\hat{\boldsymbol{x}}) \cap B_{\mathfrak{a},\mathfrak{b}}) + \sum_{\mathfrak{b},\mathfrak{c}\in\mathcal{L}} w_{\mathfrak{c}}(\Gamma_{\mathfrak{c}}(\hat{\boldsymbol{x}}) \cap B_{\mathfrak{b},\mathfrak{a}}) \\
&\quad \leq \widehat{E}^u|_{X^*_{\mathfrak{a}}}(\boldsymbol{x}^{i}) + \widehat{E}|_{I_{\mathfrak{a}}}(\boldsymbol{x}^{i}) + \sum_{\mathfrak{b}\neq\mathfrak{a}} w_{\mathfrak{a}}(\Gamma_{\mathfrak{a}}(\boldsymbol{x}^{\mathfrak{a}}) \cap B_{\mathfrak{a},\mathfrak{b}}) + \sum_{\mathfrak{b},\mathfrak{c}\in\mathcal{L}} w_{\mathfrak{c}}(\Gamma_{\mathfrak{c}}(\boldsymbol{x}^{\mathfrak{a}}) \cap B_{\mathfrak{b},\mathfrak{a}}).
\end{aligned} \qquad (5.64)$$

Furthermore, we can bound

$$w_{\mathfrak{a}}(\Gamma_{\mathfrak{a}}(\boldsymbol{x}^{\mathfrak{a}}) \cap B_{\mathfrak{a},\mathfrak{b}}) \leq w_{\mathfrak{a}}(\Gamma_{\mathfrak{a}}(\boldsymbol{x}^{i}) \cap B_{\mathfrak{a},\mathfrak{b}}) \qquad (5.65)$$

for all $\mathfrak{b} \neq \mathfrak{a}$, because $\Gamma_{\mathfrak{a}}(\boldsymbol{x}^{\mathfrak{a}}) \cap B_{\mathfrak{a},\mathfrak{b}} \subseteq B_{\mathfrak{a},\mathfrak{b}} = \Gamma_{\mathfrak{a}}(\boldsymbol{x}^{i}) \cap B_{\mathfrak{a},\mathfrak{b}}$. Now we sum Inequality (5.64) for all labels $\mathfrak{a} \in \mathcal{L}$, and then apply Inequality (5.65):

$$\begin{aligned}
&\sum_{\mathfrak{a}} \Big(\underbrace{\widehat{E}^u|_{X^*_{\mathfrak{a}}}(\hat{\boldsymbol{x}}) + \widehat{E}|_{I_{\mathfrak{a}}}(\hat{\boldsymbol{x}}) + \sum_{\mathfrak{b}\neq\mathfrak{a}} w_{\mathfrak{a}}(\Gamma_{\mathfrak{a}}(\hat{\boldsymbol{x}}) \cap B_{\mathfrak{a},\mathfrak{b}}) + \sum_{\mathfrak{b}\neq\mathfrak{a}} \sum_{\mathfrak{c}\neq\mathfrak{a}} w_{\mathfrak{c}}(\Gamma_{\mathfrak{c}}(\hat{\boldsymbol{x}}) \cap B_{\mathfrak{a},\mathfrak{b}})}_{\widehat{E}(\hat{\boldsymbol{x}}) \text{ (after summing over } \mathfrak{a})} \\
&\qquad + \sum_{\mathfrak{b}\neq\mathfrak{a}} \sum_{\mathfrak{c}} w_{\mathfrak{c}}(\Gamma_{\mathfrak{c}}(\hat{\boldsymbol{x}}) \cap B_{\mathfrak{b},\mathfrak{a}}) \Big) \\
&\leq \sum_{\mathfrak{a}} \Big(\widehat{E}^u|_{X^*_{\mathfrak{a}}}(\boldsymbol{x}^{i}) + \widehat{E}|_{I_{\mathfrak{a}}}(\boldsymbol{x}^{i}) + \sum_{\mathfrak{b}\neq\mathfrak{a}} w_{\mathfrak{a}}(\Gamma_{\mathfrak{a}}(\boldsymbol{x}^{\mathfrak{a}}) \cap B_{\mathfrak{a},\mathfrak{b}}) + \sum_{\mathfrak{b}\neq\mathfrak{a};\mathfrak{c}} w_{\mathfrak{c}}(\Gamma_{\mathfrak{c}}(\boldsymbol{x}^{\mathfrak{a}}) \cap B_{\mathfrak{b},\mathfrak{a}}) \Big)
\end{aligned}$$

$$\leq \sum_{\mathfrak{a}} \Big(\widehat{E}^u|_{X^*_{\mathfrak{a}}}(\boldsymbol{x}^i) + \widehat{E}|_{I_{\mathfrak{a}}}(\boldsymbol{x}^i) + \sum_{\mathfrak{b}\neq\mathfrak{a}} w_{\mathfrak{a}}(\Gamma_{\mathfrak{a}}(\boldsymbol{x}^i) \cap B_{\mathfrak{a},\mathfrak{b}}) + \sum_{\mathfrak{b}\neq\mathfrak{a};\mathfrak{c}} w_{\mathfrak{c}}(\Gamma_{\mathfrak{c}}(\boldsymbol{x}^{\mathfrak{a}}) \cap B_{\mathfrak{b},\mathfrak{a}})\Big)$$
$$= \widehat{E}(\boldsymbol{x}^i) + \sum_{\mathfrak{a}} \sum_{\mathfrak{b}\neq\mathfrak{a};\mathfrak{c}} w_{\mathfrak{c}}(\Gamma_{\mathfrak{c}}(\hat{\boldsymbol{x}}) \cap B_{\mathfrak{b},\mathfrak{a}}) + \sum_{\mathfrak{a},\mathfrak{b}\neq\mathfrak{a};\mathfrak{c}} w_{\mathfrak{c}}(\Gamma_{\mathfrak{c}}(\boldsymbol{x}^{\mathfrak{a}}) \cap (B_{\mathfrak{b},\mathfrak{a}} \setminus \Gamma(\hat{\boldsymbol{x}}))). \tag{5.66}$$

Now, note that $\bigcup_{\mathfrak{a};\mathfrak{b}\neq\mathfrak{a}} B_{\mathfrak{b},\mathfrak{a}} \subseteq \Gamma(\boldsymbol{x}^i)$. Furthermore, $\Gamma_{\mathfrak{c}}(\boldsymbol{x}^{\mathfrak{a}}) \cap B_{\mathfrak{b},\mathfrak{a}}$ contains the edges (v_k, v_j) with $x^i_j = x^{\mathfrak{a}}_j = \mathfrak{a}$ and $x^i_k = \mathfrak{b}$, $x^{\mathfrak{a}}_k = \mathfrak{c}$. Therefore,

$$\bigcup_{\mathfrak{c}\neq\mathfrak{a}} \Gamma_{\mathfrak{c}}(\boldsymbol{x}^{\mathfrak{a}}) \cap B_{\mathfrak{b},\mathfrak{a}} \;\subseteq\; B_{\mathfrak{b},\mathfrak{a}} \;=\; \Gamma_{\mathfrak{b}}(\boldsymbol{x}^i) \cap B_{\mathfrak{b},\mathfrak{a}}, \tag{5.67}$$

and thus Inequality (5.66) implies that

$$\widehat{E}(\hat{\boldsymbol{x}}) \leq \widehat{E}(\boldsymbol{x}^i) + \sum_{\mathfrak{a},\mathfrak{c}} \sum_{\mathfrak{b}\neq\mathfrak{a}} w_{\mathfrak{c}}(\Gamma_{\mathfrak{c}}(\boldsymbol{x}^{\mathfrak{a}}) \cap (B_{\mathfrak{b},\mathfrak{a}} \setminus \Gamma(\hat{\boldsymbol{x}}))) \tag{5.68}$$
$$\leq \widehat{E}(\boldsymbol{x}^i) + \gamma \sum_{\mathfrak{a}} \sum_{\mathfrak{b}\neq\mathfrak{a}} w_{\mathfrak{b}}(\Gamma_{\mathfrak{b}}(\boldsymbol{x}^i) \cap (B_{\mathfrak{b},\mathfrak{a}} \setminus \Gamma(\hat{\boldsymbol{x}}))) \tag{5.69}$$
$$\leq (1+\gamma)\widehat{E}(\boldsymbol{x}^i). \qquad \square$$

By Definition (5.50), $\widehat{E}$ differs by an additive constant from the approximation

$$\widehat{E}_h(\boldsymbol{x}) = \sum_j \psi_j(x_j) + \sum_{\mathfrak{a}\in\mathcal{L}} h_{\mathfrak{a},\Gamma_i}(\Gamma_{\mathfrak{a}}(\boldsymbol{x})),$$

and this constant is nonnegative:

$$c = \widehat{E}_h(\boldsymbol{x}) - \widehat{E}_{\Gamma_i}(\boldsymbol{x}) = \sum_{\mathfrak{a}\in\mathcal{L}} f_{\mathfrak{a}}(\Gamma_{\mathfrak{a}}(\boldsymbol{x}^i)) - \sum_{\mathfrak{a}\in\mathcal{L}} \sum_{e\in\Gamma_{\mathfrak{a}}(\boldsymbol{x}^i)} \rho^{\mathfrak{a}}(e|\mathcal{E}\setminus e) \geq 0,$$

because the $f_{\mathfrak{a}}$ are submodular and satisfy diminishing marginal costs. Due to this nonnegativity, the approximation bound from Lemma 5.7 also holds for $\widehat{E}_h$:

$$\widehat{E}(\hat{\boldsymbol{x}}) = \widehat{E}_{\Gamma_i}(\hat{\boldsymbol{x}}) + c \tag{5.70}$$
$$\leq (1+\gamma)\widehat{E}_{\Gamma_i}(\boldsymbol{x}^i) + c \tag{5.71}$$
$$\leq (1+\gamma)(\widehat{E}_{\Gamma_i}(\boldsymbol{x}^i) + c) = (1+\gamma)\widehat{E}_h(\boldsymbol{x}^i). \tag{5.72}$$

From these observations, we derive a complete approximation guarantee for Algorithm 4 in terms of the energy.

Lemma 5.8. *Let $\boldsymbol{x}^* \in \operatorname{argmin}_{\boldsymbol{x}\in\mathcal{L}^n} E(\boldsymbol{x})$. If the unary potentials in E are all nonnegative and if $\widehat{E}_{\Gamma_0} = \widehat{E}_\emptyset$ in Algorithm 4, then the solution $\hat{\boldsymbol{x}}$ returned by Algorithm 4 satisfies*

$$E(\hat{\boldsymbol{x}}) \leq \alpha(1+\gamma)E(\boldsymbol{x}^*)$$

for $\alpha = \max_{\mathfrak{a}\in\mathcal{L}} \frac{|\Gamma_{\mathfrak{a}}(\boldsymbol{x}^*)|}{1+(|\Gamma_{\mathfrak{a}}(\boldsymbol{x}^*)|-1)\nu(\mathfrak{a})}$ *and* $\nu(\mathfrak{a}) = \frac{\min_{e\in\Gamma_{\mathfrak{a}}(\boldsymbol{x}^*)} \rho^{\mathfrak{a}}(e|\Gamma_{\mathfrak{a}}(\boldsymbol{x}^*)\setminus e)}{\max_{e\in\Gamma_{\mathfrak{a}}(\boldsymbol{x}^*)} f_{\mathfrak{a}}(e)}$.

Corollary 5.1. *If $f_{\mathfrak{a}} = f_{\mathfrak{b}}$ for all $\mathfrak{a}, \mathfrak{b} \in \mathcal{L}$ and if the conditions of Lemma 5.8 hold, then the solution returned by Algorithm 4 satisfies*

$$E(\hat{\boldsymbol{x}}) \leq 2\alpha E(\boldsymbol{x}^*)$$

for α as in Lemma 5.8.

The constant $\nu(\mathfrak{a})$ in Lemma 5.8 corresponds to the ν in Lemma 4.6. Analogously to Lemma 4.6, Lemma 5.8 can be stated in terms of the curvature. Then the corollary includes the factor-two approximation by Boykov et al. [2001] as a special case. To prove the Lemma 5.8, we use a simple proposition:

Proposition 5.1. *Let $\{a_i\}_i$, $\{b_i\}_i$ be two series of n nonnegative numbers and $b_i > 0$ for all i. Then*

$$\frac{\sum_{i=1}^n a_i}{\sum_{i=1}^n b_i} \leq \max_{1 \leq j \leq n} \frac{a_j}{b_j}.$$

Proof. It holds that

$$\sum_{i=1}^n b_i \Big(\max_{1 \leq j \leq n} \frac{a_j}{b_j} \Big) \geq \sum_{i=1}^n b_i \frac{a_i}{b_i} = \sum_{i=1}^n a_i. \tag{5.73}$$

Dividing both sides by the sum of b_i proves the proposition. □

Proof (Lemma 5.8). First, recall that any assignment $\hat{\boldsymbol{x}}$ returned by Algorithm 4 can only be better than the assignment $\hat{\boldsymbol{x}}_0$ after the first iteration, where $\widehat{E}_h = \widehat{E}_{\Gamma_0} = \widehat{E}_{\emptyset}$. Let $\boldsymbol{x}_0 \in \operatorname{argmin}_{\boldsymbol{x} \in \mathcal{L}^n} \widehat{E}_{\emptyset}(\boldsymbol{x})$ be a minimizing assignment for $\widehat{E}_{\emptyset}$. From Lemma 4.4 and by construction of the upper bounds h in the definition (5.45), we know that $\widehat{E}_{\emptyset}$ is an upper bound to E. Furthermore, applying Lemma 5.7 and then using the optimality of $\boldsymbol{x}_0$, we conclude that

$$E(\hat{\boldsymbol{x}}) \leq E(\hat{\boldsymbol{x}}_0) \leq \widehat{E}_{\emptyset}(\hat{\boldsymbol{x}}_0) \leq (1+\gamma)\widehat{E}_{\emptyset}(\boldsymbol{x}_0) \leq (1+\gamma)\widehat{E}_{\emptyset}(\boldsymbol{x}^*). \tag{5.74}$$

To proceed, we abbreviate $\Gamma^*_{\mathfrak{a}} = \Gamma_{\mathfrak{a}}(\boldsymbol{x}^*)$ and use Proposition 5.1:

$$\widehat{E}_{\emptyset}(\boldsymbol{x}^*) = \sum_i \psi(x_i^*) + \sum_{\mathfrak{a} \in \mathcal{L}} h_{\mathfrak{a},\emptyset}(\Gamma^*_{\mathfrak{a}}) \tag{5.75}$$

$$= \sum_i \psi(x_i^*) + \frac{\sum_{\mathfrak{a} \in \mathcal{L}} h_{\mathfrak{a},\emptyset}(\Gamma^*_{\mathfrak{a}})}{\sum_{\mathfrak{a} \in \mathcal{L}} f(\Gamma^*_{\mathfrak{a}})} \sum_{\mathfrak{a} \in \mathcal{L}} f(\Gamma^*_{\mathfrak{a}}) \tag{5.76}$$

$$\leq \sum_i \psi(x_i^*) + \max_{\mathfrak{a} \in \mathcal{L}} \frac{h_{\mathfrak{a},\emptyset}(\Gamma^*_{\mathfrak{a}})}{f(\Gamma^*_{\mathfrak{a}})} \sum_{\mathfrak{a} \in \mathcal{L}} f(\Gamma^*_{\mathfrak{a}}) \tag{5.77}$$

Now, we use two observations to bound the fraction further. First, for any $\mathfrak{a} \in \mathcal{L}$ and $e_{\mathfrak{a}} \in \operatorname{argmax}_{e \in \Gamma^*_{\mathfrak{a}}} f_{\mathfrak{a}}(e)$, it holds that

$$h_{\mathfrak{a},\emptyset}(\Gamma^*_{\mathfrak{a}}) = \sum_{e \in \Gamma^*_{\mathfrak{a}}} f_{\mathfrak{a}}(e) \leq |\Gamma^*_{\mathfrak{a}}| f_{\mathfrak{a}}(e_{\mathfrak{a}}). \tag{5.78}$$

Second, by diminishing marginal costs, it holds that

$$f_{\mathfrak{a}}(\Gamma^*_{\mathfrak{a}}) \leq f(e_{\mathfrak{a}}) + \sum_{e \in \Gamma^*_{\mathfrak{a}} \setminus e} \rho^{\mathfrak{a}}(e | \Gamma^*_{\mathfrak{a}} \setminus e). \tag{5.79}$$

The bounds (5.78) and (5.79) serve to bound the maximum in (5.77) further:

$$\begin{aligned}
\widehat{E}_{\emptyset}(\boldsymbol{x}^*) &\leq \sum_i \psi(x_i^*) + \max_{\mathfrak{a} \in \mathcal{L}} \frac{h_{\mathfrak{a},\emptyset}(\Gamma^*_{\mathfrak{a}})}{f(\Gamma^*_{\mathfrak{a}})} \sum_{\mathfrak{a} \in \mathcal{L}} f(\Gamma^*_{\mathfrak{a}}) && (5.80)\\
&\leq \sum_i \psi(x_i^*) + \max_{\mathfrak{a} \in \mathcal{L}} \frac{|\Gamma^*_{\mathfrak{a}}| f_{\mathfrak{a}}(e_{\mathfrak{a}})}{\left(f_{\mathfrak{a}}(e_{\mathfrak{a}}) + \sum_{e \in \Gamma^*_{\mathfrak{a}} \setminus e_{\mathfrak{a}}} \rho^{\mathfrak{a}}(e | \Gamma^*_{\mathfrak{a}} \setminus e)\right)} \sum_{\mathfrak{a} \in \mathcal{L}} f_{\mathfrak{a}}(\Gamma^*_{\mathfrak{a}}) && (5.81)\\
&= \sum_i \psi(x_i^*) + \alpha \sum_{\mathfrak{a} \in \mathcal{L}} f(\Gamma^*_{\mathfrak{a}}) && (5.82)\\
&\leq \alpha \Big(\sum_i \psi(x_i^*) + \sum_{\mathfrak{a} \in \mathcal{L}} f(\Gamma^*_{\mathfrak{a}}) \Big) = \alpha E(\boldsymbol{x}^*). && (5.83)
\end{aligned}$$

The last inequality assumes that the unary potentials are nonnegative. The final result follows from the bounds (5.74) and (5.83):

$$E(\hat{\boldsymbol{x}}) \;\leq\; (1+\gamma) \widehat{E}_{\emptyset}(\boldsymbol{x}^*) \;\leq\; \alpha(1+\gamma) E(\boldsymbol{x}^*). \qquad \square$$

5.4.4. Cooperative expansion moves

The preceding section showed an approximate minimization of a cooperative cut multi-label energy via first approximating the energy by a pair-wise one and then performing expansion moves. For completeness, we here complement this approach by swapping the order. We show that it is possible to solve expansion moves for the cooperative cut energy directly as a cooperative cut. As above, for simplicity of exposition we assume that the terminal edges in the structure graph have modular edge weights, that is, that there are modular unary potentials.

Lemma 5.9. *The best expansion move with respect to an energy E_f (for M2), any fixed label $\mathfrak{a} \in \mathcal{L}$ and any given labeling $\boldsymbol{x}' \in \mathcal{L}^n$ can be computed as a minimum cooperative cut.*

To constructively prove Lemma 5.9, we construct a directed graph $\widetilde{\mathsf{G}} = (\widetilde{\mathsf{V}}, \widetilde{\mathsf{E}})$ and as above show Claims 5.1 and 5.2. The nodes $\tilde{\mathsf{v}}_i \in \widetilde{\mathsf{V}}$ in $\widetilde{\mathsf{G}}$ are in one-to-one correspondence with nodes $v_i \in \mathcal{V}$ in the given $\mathcal{G}$, and $\widetilde{\mathsf{G}}$ has two terminal nodes $\tilde{\mathsf{s}}, \tilde{\mathsf{t}}$ that are connected to each $\tilde{\mathsf{v}}_i$. Furthermore, Property 5.1 holds for $\widetilde{\mathsf{G}}$. If the marginal costs remain strictly positive, then every minimum cut is also minimal. Otherwise, we add tiny additional weights to all edges.

Proof. The last point of Property 5.1 justifies the following labeling corresponding to a given cut $\widetilde{C}$:

$$x_i = \begin{cases} \mathfrak{a} & \text{if } (\tilde{\mathsf{v}}_i, t) \in \widetilde{C} \\ x_i' & \text{if } (s, \tilde{\mathsf{v}}_i) \in \widetilde{C}. \end{cases} \tag{5.84}$$

First, we construct terminal edges in $\widetilde{\mathsf{G}}$ that express the unary potentials ψ_i. For all i, we assign weight $\psi_i(x_i')$ to edge $(\tilde{\mathsf{s}}, \tilde{\mathsf{v}}_i)$ and weight $\psi_i(x_i = \mathfrak{a})$ to edge $(\tilde{\mathsf{v}}_i, t)$. If $x_i' = \mathfrak{a}$, then edge $(\tilde{\mathsf{s}}, \tilde{\mathsf{v}}_i)$ has weight ∞, so that edge $(\tilde{\mathsf{v}}_i, \tilde{\mathsf{t}})$ is cut and $x_i = \mathfrak{a}$ is ensured. Then Labeling (5.84) is within one $\mathfrak{a}$-expansion of $\boldsymbol{x}'$; this proves Claim 5.1.

For Claim 5.2, we need to detail the structure of $\widetilde{\mathsf{G}}$. The construction of the remaining edges $\widetilde{\mathsf{E}}_c$ in $\widetilde{\mathsf{G}}$ is somewhat complicated because a directed cut only includes the edges between the $\tilde{\mathsf{s}}$-part (label $\mathfrak{a}$) and $\tilde{\mathsf{t}}$-part (label $\mathfrak{b} \neq \mathfrak{a}$), but we must keep track of the cost of all boundaries, not only $\Gamma_{\mathfrak{a}}(\boldsymbol{x})$. Thus, the cost of these other edges must be transferred appropriately. For additive edge weights, this can be achieved by adding the missing weight to some other edge. As the costs are nonlinear here, we must transfer arguments to a function, and sums become set unions. We define a mapping $\pi(\tilde{\mathsf{e}}) \in \mathcal{E}_n$ from edges in $\widetilde{\mathsf{E}}_c$ to edges in $\mathcal{E}_n$ (the set of non-terminal edges in $\mathcal{G}$). In addition, the edges $\widetilde{\mathsf{E}}_c$ make $\widetilde{\mathsf{G}}$ a multi-graph, and are partitioned into sets $S_{\mathfrak{b}} \subseteq \widetilde{\mathsf{E}}_c$. These sets are indexed by the labels $\mathfrak{b} \in \mathcal{L}$.

The cost of a set of edges $\widetilde{C} \subseteq \widetilde{\mathsf{E}}_c$ is defined by a submodular function $\tilde{f}$ that uses the original cost $f_{\mathfrak{b}}$ on the mappings $\pi(\widetilde{C})$ of edges in $\widetilde{C}$. The sets $S_{\mathfrak{b}}$ determine which $f_{\mathfrak{b}}$ is used:

$$\tilde{f}(\widetilde{C}) = \sum_{\mathfrak{b} \in \mathcal{L}} f_{\mathfrak{b}}(\pi(\widetilde{C} \cap S_{\mathfrak{b}})). \tag{5.85}$$

For any $e = (v_i, v_j) \in \mathcal{E}_n$ in $\mathcal{G}$, there are the following cases to construct edges in $\widetilde{\mathsf{G}}$. The resulting edges form the set $\widetilde{\mathsf{E}}_c$, and we indicate $\tilde{\mathsf{e}} \in S_{\mathfrak{b}}$ by a superscript $\mathfrak{b}$ on the edge $\pi(\tilde{\mathsf{e}})$.

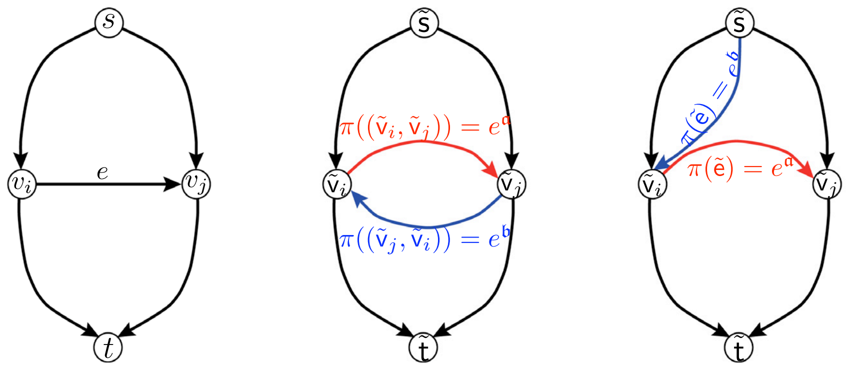
$x_i' = \mathfrak{a}, x_j' = \mathfrak{b}$: $e = (v_i, v_j)$ can at most be in $\Gamma_{\mathfrak{a}}(\boldsymbol{x})$; introduce an edge $\tilde{\mathsf{e}} = (\tilde{\mathsf{v}}_i, \tilde{\mathsf{v}}_j) \in S_{\mathfrak{a}}$, with $\pi(\tilde{\mathsf{e}}) = e^{\mathfrak{a}}$;

$x_i' = \mathfrak{b}, x_j' = \mathfrak{a}$: this direction is never cut in $\widetilde{\mathsf{G}}$, so introduce a reverse edge $\tilde{\mathsf{e}} = (\tilde{\mathsf{v}}_j, \tilde{\mathsf{v}}_i) \in S_{\mathfrak{b}}$, with $\pi(\tilde{\mathsf{e}}) = e^{\mathfrak{b}}$;

$x_i' = \mathfrak{b}, x_j' = \mathfrak{c}$: e is in $\Gamma_{\mathfrak{b}}(\boldsymbol{x})$ if $x_i = x_i' = \mathfrak{b}$, and otherwise possibly in $\Gamma_{\mathfrak{a}}(\boldsymbol{x})$; introduce $\tilde{\mathsf{e}} = (\tilde{\mathsf{v}}_i, \tilde{\mathsf{v}}_j) \in S_{\mathfrak{a}}$ with $\pi(\tilde{\mathsf{e}}) = e^{\mathfrak{a}}$, and an edge $\tilde{\mathsf{e}}' = (\tilde{\mathsf{s}}, \tilde{\mathsf{v}}_i) \in S_{\mathfrak{b}}$ with $\pi(\tilde{\mathsf{e}}') = e^{\mathfrak{b}}$;

$x_i' = \mathfrak{b}, x_j' = \mathfrak{b}$: e is currently in no boundary; introduce an edge $\tilde{\mathsf{e}} = (\tilde{\mathsf{v}}_i, \tilde{\mathsf{v}}_j) \in S_{\mathfrak{a}}$, with $\pi(\tilde{\mathsf{e}}) = e^{\mathfrak{a}}$, and an edge $\tilde{\mathsf{e}}' = (\tilde{\mathsf{v}}_j, \tilde{\mathsf{v}}_i) \in S_{\mathfrak{b}}$, with $\pi(\tilde{\mathsf{e}}') = e^{\mathfrak{b}}$;

$x_i' = \mathfrak{a}, x_j' = \mathfrak{a}$: labels are fixed, so we introduce no edges.


s
v_i
e
v_j
t
$\tilde{\mathsf{s}}$
$\pi((\tilde{\mathsf{v}}_i,\tilde{\mathsf{v}}_j)) = e^{\mathfrak{a}}$
$\tilde{\mathsf{v}}_i$
$\tilde{\mathsf{v}}_j$
$\pi((\tilde{\mathsf{v}}_j,\tilde{\mathsf{v}}_i)) = e^{\mathfrak{b}}$
$\tilde{\mathsf{t}}$
$\tilde{\mathsf{s}}$
$\pi(\tilde{\mathsf{e}}) = e^{\mathfrak{b}}$
$\pi(\tilde{\mathsf{e}}) = e^{\mathfrak{a}}$
$\tilde{\mathsf{v}}_i$
$\tilde{\mathsf{v}}_j$
$\tilde{\mathsf{t}}$

(a) $\mathcal{G}$ (b) $x'_i = x'_j = \mathfrak{b}$ (c) $x'_i = \mathfrak{b}$, $x'_j = \mathfrak{c}$

Figure 5.8. Expressing an edge e in the auxiliary graph by edges $\tilde{\mathsf{e}}$ when expanding label $\mathfrak{a}$. (a) Edge $e = (v_i, v_j)$ in $\mathcal{G}$. (b), (c) Implementations in $\tilde{\mathsf{G}}$ depending on the initial labels x'_i, x'_j. If the new labels $x_i = x_j$ are equal, then the edge e is not cut. If $x_i = \mathfrak{a}$ and $x_j \neq \mathfrak{a}$, then $e \in \Gamma_{\mathfrak{a}}(\boldsymbol{x})$, and e must contribute to that boundary. In that case, the cut $\tilde{\Gamma}(\boldsymbol{x})$ in $\tilde{\mathsf{G}}$ includes the red edge. If $x_i = x'_i = \mathfrak{b} \neq \mathfrak{a}$ and $x_j \neq x_i$, then $e \in \Gamma_{\mathfrak{b}}(\boldsymbol{x})$. There are two cases, (b) and (c). (b) If initially the variables had the same label, then $x_j \neq x_i$ can only happen if now $x_j = \mathfrak{a}$. Then the cut $\tilde{\Gamma}(\boldsymbol{x})$ must sever any edge $(\tilde{\mathsf{v}}_j, \tilde{\mathsf{v}}_i)$, that is, the blue edge. The mapping of that edge ensures that e is counted in $\Gamma_{\mathfrak{b}}$. (c) If the variables have different labels from the beginning, then $x_i \neq \mathfrak{a}$ implies that $e \in \Gamma_{\mathfrak{b}}(\boldsymbol{x})$. In this case, the terminal edge $\tilde{\mathsf{e}} = (\tilde{\mathsf{s}}, \tilde{\mathsf{v}}_i)$ in $\tilde{\mathsf{G}}$ is cut by $\tilde{\Gamma}(\boldsymbol{x})$, and $\pi(\tilde{\mathsf{e}})$ ensures that e is counted in $\Gamma_{\mathfrak{b}}$.

Figure 5.8 shows examples, and Table 5.2 lists all edges between two given nodes in the resulting $\tilde{\mathsf{G}}$. Given this $\tilde{\mathsf{G}}$, we now show that $E_f(\boldsymbol{x}) = \tilde{f}(\tilde{\Gamma}(\boldsymbol{x}) \cap \tilde{\mathsf{E}}_c) + \sum_i \psi_i(x_i)$. To do so, we prove that $\pi(\tilde{\Gamma}(\boldsymbol{x}) \cap S_{\mathfrak{b}}) = \Gamma_{\mathfrak{b}}(\boldsymbol{x})$ for all labels $\mathfrak{b} \in \mathcal{L}$.

Let $\tilde{\Gamma} \subseteq \tilde{\mathsf{E}}$ be a minimal (s,t)-cut in $\tilde{\mathsf{G}}$, and $\boldsymbol{x}$ the corresponding labeling (5.84), and $\Gamma_{\mathfrak{b}}(\boldsymbol{x})$ the boundary of label $\mathfrak{b}$ in $\mathcal{G}$. Then Table 5.2 implies for $\mathfrak{b} \neq \mathfrak{a}$ that

$$\pi(\tilde{\Gamma} \cap S_{\mathfrak{b}}) = \Big(\bigcup_{\substack{(\tilde{\mathsf{v}}_i,\tilde{\mathsf{v}}_j)\in S_{\mathfrak{b}} \\ x_i=\mathfrak{a}, x_j \neq \mathfrak{a}}} \pi(\tilde{\mathsf{v}}_i,\tilde{\mathsf{v}}_j) \Big) \cup \Big(\bigcup_{(\tilde{\mathsf{s}},\tilde{\mathsf{v}}_i)\in\tilde{\Gamma}\cap S_{\mathfrak{b}}} \pi(\tilde{\mathsf{s}},\tilde{\mathsf{v}}_i) \Big) \tag{5.86}$$

$$\begin{aligned} &= \Big(\bigcup_{\substack{(v_j,v_i)\in\mathcal{E}: \\ x'_i=\mathfrak{a}, \\ x_j = x'_j = \mathfrak{b}}} (v_j, v_i) \Big) \cup \Big(\bigcup_{\substack{(v_i,v_j)\in\mathcal{E}: \\ x_i=\mathfrak{a}, \\ x'_i = x_j = \mathfrak{b}}} (v_j, v_i) \Big) \\ &\quad \cup \Big(\bigcup_{x_i=\mathfrak{b}} \bigcup_{\substack{j:(v_i,v_j)\in\mathcal{E}: \\ x_j \neq \mathfrak{b},\mathfrak{a}}} (v_i, v_j) \Big) \end{aligned} \tag{5.87}$$

$$= \big(\Gamma_{\mathfrak{b}} \cap (X_{\mathfrak{b}} \times X_{\mathfrak{a}})\big) \cup \Big(\Gamma_{\mathfrak{b}} \cap \bigcup_{\mathfrak{c}\neq\mathfrak{b},\mathfrak{a}} (X_{\mathfrak{b}} \times X_{\mathfrak{c}})\Big) = \Gamma_{\mathfrak{b}}(\boldsymbol{x}). \tag{5.88}$$

x'_i	x'_j	$\pi(\tilde{\mathsf{e}})$ of $(\tilde{\mathsf{v}}_i, \tilde{\mathsf{v}}_j)$ in $\widetilde{\mathsf{G}}$
$\mathfrak{a}$	$\mathfrak{b}$	$(v_i, v_j)^{\mathfrak{a}}$ for $\tilde{\mathsf{e}} \in S_{\mathfrak{a}}$
$\mathfrak{b}$	$\mathfrak{c}$	$(v_i, v_j)^{\mathfrak{a}}$ for $\tilde{\mathsf{e}} \in S_{\mathfrak{a}}$
$\mathfrak{b}$	$\mathfrak{b}$	$(v_i, v_j)^{\mathfrak{a}}$ for $\tilde{\mathsf{e}} \in S_{\mathfrak{a}}$, $(v_j, v_i)^{\mathfrak{b}}$ for $\tilde{\mathsf{e}} \in S_{\mathfrak{b}}$

terminal edges in $\widetilde{\mathsf{G}}$		
edges	x'_i	$\pi(\tilde{\mathsf{e}})$
$(\tilde{\mathsf{s}}, \tilde{\mathsf{v}}_i)$	$\mathfrak{a}$	$\emptyset$
$(\tilde{\mathsf{s}}, \tilde{\mathsf{v}}_i)$	$\mathfrak{b}$	T_i
$(\tilde{\mathsf{v}}_i, \tilde{\mathsf{t}})$	$x'_i \in \mathcal{L}\emptyset$	

Table 5.2. Mappings $\pi(\tilde{\mathsf{e}})$ of edges $\tilde{\mathsf{e}}$ in $\widetilde{\mathsf{G}}$ (if $x'_j = \mathfrak{a}$, then there are no edges $(\tilde{\mathsf{v}}_i, \tilde{\mathsf{v}}_j)$). If (v_i, v_j) or $(v_j, v_i) \notin \mathcal{E}$, then replace it by $\emptyset$ in the table. Here, $\mathcal{N}_{i,\mathfrak{b}} = \{e = (v_i, v_j) \in \mathcal{E} \mid x'_i = \mathfrak{b}, x'_j \neq \mathfrak{b}, \mathfrak{a}\}$ and $T_i = \bigcup_{e \in \mathcal{N}_{i,\mathfrak{b}}} \{ e^{\mathfrak{b}} \}$.

Similarly, one can show that $\pi(\widetilde{\Gamma} \cap S_{\mathfrak{a}}) = \Gamma_{\mathfrak{a}}(\boldsymbol{x})$. In consequence, $\tilde{f}(\widetilde{\Gamma}(\boldsymbol{x})) = \sum_{\mathfrak{b}} f_{\mathfrak{b}}(\Gamma_{\mathfrak{b}}(\boldsymbol{x}))$. The expression of the unary terms is analogous in $\mathcal{G}$ and $\widetilde{\mathsf{G}}$, so indeed $E_f(\boldsymbol{x}) = \tilde{f}(\widetilde{\Gamma}(\boldsymbol{x}) \cap \widetilde{\mathsf{E}}_c) + \sum_i \psi_i(x_i)$. This proves Claim 2. $\square$

The cooperative cut that corresponds to the expansion move can be solved by any of the algorithms from Chapter 4. Using the iterative algorithm results in the same edge weights that result from the direct approximation described in Section 5.4.3.

5.5. Summary and discussion

Since cooperative cuts employ the same graph structure as standard graph cuts, they can enrich applications of graph cuts by higher-order coupling. In particular, in this chapter we introduce *cooperative cut energy functions* and show that cooperative cuts widen the range of functions representable by cuts to include higher-order and non-submodular functions. Indeed, energy functions represented by cooperative cuts need not have any of the simplifying properties that commonly facilitate inference in the corresponding probabilistic model. Nevertheless, the algorithms from Chapter 4 apply for performing approximate inference. Moreover, cooperative cut energies unify a number of recently defined models in computer vision.

The graph structure underlying the coupling in cooperative cuts has further benefits. We extend the binary domain of the initially discussed energy functions to wider discrete domains and design move-making algorithms for approximate minimization. Contrary to the move constructions by [Boykov et al., 2001], the graphs in our move-making algorithms do not require any auxiliary nodes.

In the following two chapters, we apply the representation of cooperative cut energies to image segmentation and approximate submodular minimization.

Chapter 6.

Applications in Computer Vision

Graph cuts have been successfully used in computer vision, in particular for image segmentation. However, graph cut models fail in a number of settings. In this chapter, we show how to significantly improve image segmentation results by coupling graph edges and thereby expressing a new global uniformity criterion. This criterion builds on the model discussed in Chapter 5.

Beyond image segmentation, cooperative cuts relate to structured regularization in machine learning, and to old ideas in image processing. The second part of this chapter establishes these ties.

6.1. Coupling edges for image segmentation

In this chapter, we address interactive figure-ground segmentation for color and grayscale images. In an interactive segmentation, a user labels a few pixels as 'object' or 'background', for example by brush strokes. The segmentation algorithm is to infer the binary labels of the remaining pixels.

Graph cuts have been popular for this task, but fail in particular difficult settings. Figure 6.1 shows an example that challenges several state-of-the-art segmentation algorithms.

6.1.1. Problems of cut-based algorithms

The image in Figure 6.1 is only one example. It is also known that the "Graph Cut" approach faces a *shrinking bias*: long, fine segments of objects tend to be cut off, as happens with the objects in Figure 6.7. To understand the reasons for those problems, we briefly review the standard model underlying Graph Cut for image segmentation.

Figure-ground segmentation is often formulated as maximum a posteriori (MAP) inference in a grid-structured Markov Random Field with unary terms and pairwise terms that span neighboring pixels. The energy associated with such a model is

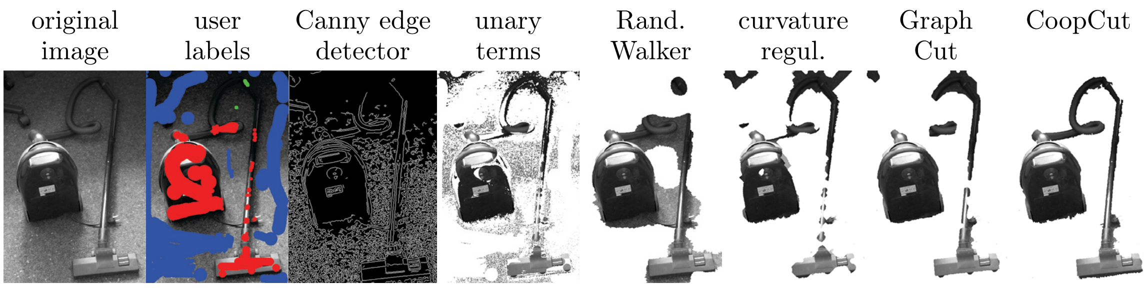


Figure 6.1. Segmentation results for an image with shading. The task is difficult despite many user labels. All algorithms used the same unary terms, except for the Random Walker, which got enhanced seeds (green). Column 4 is the segmentation obtained from unary terms alone. The algorithms are described in [Grady, 2006, El-Zehiry and Grady, 2010, Boykov and Jolly, 2001].

a function of the observed image pixels $\bar{\boldsymbol{z}}$ and variables $\boldsymbol{x}$, one x_i for each pixel i. The energy has the form

$$E(\boldsymbol{x}; \bar{\boldsymbol{z}}) = \sum_{i \in I} \psi_i(x_i) \quad + \quad \lambda \sum_{(i,j) \in \mathcal{E}_n} \psi_{ij}(x_i, x_j) \tag{6.1}$$

$$= w(\Gamma(\boldsymbol{x}) \cap \mathcal{E}_t) + \lambda w(\Gamma(\boldsymbol{x}) \cap \mathcal{E}_n) + \text{const.} \tag{6.2}$$

The unary terms ψ_i incorporate the user labels and judge the data fit for each pixel separately. The pairwise terms add a regularizing effect and enforce smooth boundaries and coherency.

The corresponding structure graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}_t \cup \mathcal{E}_n)$ is depicted in Figure 6.2. It has a node $v_i \in \mathcal{V}$ for each variable x_i, and two terminal nodes. Its *terminal edges* (s, v) and $(v, t) \in \mathcal{E}_t$ for all $v \in \mathcal{V}$ model the unary potentials, and the remaining grid edges – we call them $\mathcal{E}_n$ – represent the pairwise terms. The edge weights on $\mathcal{E}_n$ are a function of the intensity gradient, typically of the form

$$w(v_i, v_j) \propto \exp(-\|z_i - z_j\|^2), \tag{6.3}$$

and their sum may be seen as the weighted length of the object boundary. This penalty favors short boundaries, and thus results in the aforementioned shortcutting of long boundaries. Similarly, if all edge weights are large due to low contrast, then there is very little incentive towards the correct boundaries, and shortness becomes decisive. Lowering the coefficient λ is not a solution since it decreases the intended smoothing effect: boundaries become noisy and true background is included into the hypothesized foreground. Figure 6.7 from the experiments demonstrates this effect. This happens in particular if the unary potentials only provide very weak guidance, for example owing to the ambiguous color information in grayscale images (see, for example, the segmentation by unary potentials in Figure 6.1). In such images, it is very difficult to judge merely by *local* information where a boundary should be.

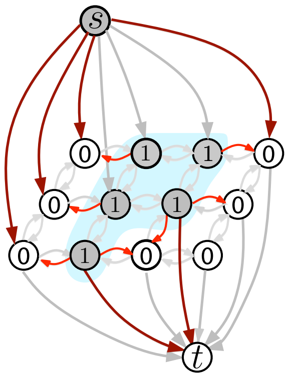


Figure 6.2 Commonly used graph cut representation of a local pairwise Markov random field. The energy of the labeling $\boldsymbol{x}$ indicated in the nodes of the graph is equivalent (up to constants) to the sum of weights of the edges in $\Gamma(\boldsymbol{x})$, i.e., the edges from label-one to label-zero nodes.

As lowering the weight of all edges is undesirable, we will *selectively* reward only boundaries with certain global properties. We use properties that are visible from the complete set of graph edges of which the boundary is composed. These properties go beyond the local criterion of mere contrast.

6.1.2. Structured cooperation for congruous boundaries

To remedy the problems outlined above, we aim to selectively reward discriminating features of true boundaries. We observe that, along true object boundaries, many images possess a certain *congruence*, and this may be true globally throughout the image. Boundary congruity materializes in a number of contexts. For example, of the many inter-pixel color gradients in Figure 6.7, only few occur along the true boundary in difficult regions (e.g., dark brown to sand-colored). Shortcutting introduces new, incongruous, boundary types (e.g., dark brown to slightly less dark brown). Moreover, the repetitiveness of patterned backgrounds retains congruity to a large extent. Similarly, if shade is neutralized, congruity extends across lit and shaded regions in Figure 6.1. In this latter case, we thus need a *shade-invariant* congruity criterion.

Let us briefly review the potential advantages of preferring congruous boundaries. First, this criterion can weaken the harmful smoothness penalty in areas that are likely to belong to a realistic boundary. It still retains the smoothing effect for non-congruous boundaries to remove parts of the background. By preferring certain boundaries over others, it complements the information given by the unary potentials. Second, the congruity criterion is very non-local. Therefore it has potential to transfer information from one region of the image to others. For example, if the object has some "easier" boundary parts, such as the body of the vacuum cleaner in Figure 6.1, then we can search for "similar" boundaries in more ambiguous parts of the image, such as shaded low-contrast regions. In that sense, the criterion contributes more than merely smoothing the segmentation provided by the unary potentials. This matters in cases where e.g. the color information is weak. As an example, by the unary potentials, large parts of the upper left part of the tube in Figure 6.1 would be excluded, and they are indeed excluded by the Graph Cut solution.

To implement the congruity criterion, we observe that the object boundary is merely the boundary $\Gamma(\boldsymbol{x}) \cap \mathcal{E}_n$ in the structure graph. Congruity then translates into having little diversity among the edges that make up the cut. It will turn out that such a criterion lends itself to cooperative cuts.

We define a cooperative model that retains the grid graph structure $\mathcal{G}$, and we replace only the over-smoothing inter-pixel cut (6.1) by a cooperative cut[1]:

$$\begin{aligned} E_f(\boldsymbol{x}) &= w(\Gamma(\boldsymbol{x}) \cap \mathcal{E}_t) + \lambda f(\Gamma(\boldsymbol{x}) \cap \mathcal{E}_n) && (6.4)\\ &= \sum\nolimits_{i \in I} \psi_i(\boldsymbol{x}_i) \quad + \quad \lambda f(\Gamma(\boldsymbol{x}) \cap \mathcal{E}_n). && (6.5)\end{aligned}$$

Since an object boundary consists of cut edges in $\mathcal{G}$, we desire a submodular edge cost f that captures the desirable boundary feature of congruity. Consequently, f should (i) decrease the penalty for globally congruous boundaries, (ii) retain the common smoothing effect of pairwise potentials for incongruous boundaries, and (iii) allow automatic and efficient adaptation of the congruence criterion to each image.

We define congruity in terms of classes or types $\mathcal{S}(\bar{\boldsymbol{z}}) = \{S_1, S_2, \ldots, S_\ell\}$ of similar edges, $S_i \subseteq \mathcal{E}_n$ and $\mathcal{E}_n = \bigcup_i S_i$. "Little diversity" then means that a congruous boundary uses only few edge types. To reward such boundaries, we make it costly to use an additional class, and relatively cheaper to stick to one group. Diminishing marginal costs can implement exactly this if they are employed appropriately. We make f submodular only *within* classes, and modular across classes:

$$f(\Gamma) = \sum\nolimits_{S \in \mathcal{S}(\bar{\boldsymbol{z}})} f_S(\Gamma \cap S). \tag{6.6}$$

As a result, (i) f is subadditive within a class, and *diminishing* costs mean that the discount on an additional edge increases with the number of edges included from that class. On the other hand, (ii) there is no discount for cuts that use edges from many classes, i.e., incongruous cuts. We define the class costs f_S to be thresholded discount functions,

$$f_S(\Gamma) = \begin{cases} w(\Gamma \cap S) & \text{if } w(\Gamma \cap S) \leq \theta_S \\ \theta_S + g(w(\Gamma \cap S) - \theta_S) & \text{if } w(\Gamma \cap S) > \theta_S \end{cases}, \tag{6.7}$$

for any nondecreasing, nonnegative concave function $g : \mathbb{R} \to \mathbb{R}$. The thresholds θ_S ensure that rewards are only granted once enough edges from a class are included. For our experiments, we chose $g(x) = \sqrt{x}$. Alternatives include $g(x) = \log(1+x)$ or roots $g(x) = x^{1/p}$. Figure 6.3 demonstrates their discounting effect. The modular case (6.1) corresponds to $g(x) = x$. The weights w here are the same as used for the standard modular Graph Cut model.

We call the cooperative cut energy E_f in (6.4) with submodular functions f_S as in (6.7) the "boundary congruity energy" (BC energy).

[1]Unary potentials can be coupled, too; this was used in Section 5.3.

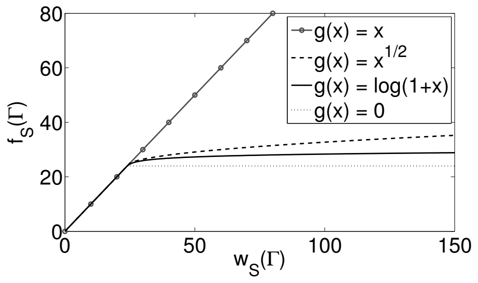


Figure 6.3 Effect of different concave functions g to introduce discounts in the cost function in Equation (6.7).

The effect of the cooperative cost function is that instead of "short" boundaries made up of few light-weight edges, we prefer boundaries that use few classes of edges. This idea is conceptually close to the formulations of "structured sparsity" for variable selection in machine learning, where instead of few variables, one aims to select variables from few groups. Moving from graph cuts to cooperative cuts then corresponds to moving from ℓ_1-regularization to (data-driven) group norms. We will further explore some of these relations in Section 6.2. We point out, however, that norms are commonly used for continuous problems and not for combinatorial discrete problems such as cuts.

Adaptation to a given image

The color statistics of different images can vary, and therefore we adapt the edge types to the image at hand, in an unsupervised manner. We infer edge classes by clustering the edges $\mathcal{E}_n$ for each image separately. Each cluster defines an edge class S_i.

Furthermore, the discount only sets in after a threshold θ_S is reached, and we adapt θ_S to the total weight of the class, i.e., $\theta_S = \vartheta w(S)$ for $\vartheta \in [0, 1]$, which improves scale-invariance. For large objects or images, more edges exist, and more similar edges must be used to deserve the label "congruous". In that case, more edges are in a class, and the adaptive threshold demands more edges before admitting a discount. The factor ϑ trades off between completely modular cuts ($\vartheta = 1$) and completely cooperative cuts ($\vartheta = 0$).

Similarity for inferring edge types

The quantitative gauge of "congruence" depends on the distance measure that we use to cluster the edges. For an edge $e = (v_i, v_j)$ with observed pixel values z_i, z_j, we define two possible feature vectors $\phi(e)$:

1. For uniformly lit images, we use linear color gradients, $\phi_l(e) = z_j - z_i$, and squared Euclidean distance for clustering.

2. To neutralize shading, we use logarithmic intensity ratios $\phi_r(e) = \log(z_j/z_i)$ which are roughly invariant to shading and the ℓ_1-distance for clustering.

For RGB images, ϕ_r is a vector where each entry is the ratio for a particular channel.

The features ϕ_l or ϕ_r are then used to cluster the edges. They are in fact *only* used for clustering; the cost functions f_S use the standard weights of the form (6.3).

6.1.3. Optimization

Chapter 4 presents a range of approximation algorithms. An efficient option for image segmentation is the iterative algorithm ITB described in Section 4.2.3. As a subroutine, it repeatedly uses a standard minimum cut in the grid-structured graph. Since this graph structure is widely used in computer vision, specialized efficient maximum flow algorithms have been devised for these graphs [Boykov and Kolmogorov, 2004]. Since we did not change the graph structure, we can profit from such algorithms as subroutines.

Moreover, the approximation bound in Lemma 4.6 relies on the minimum marginal cost any edge can take. We found that functions f_S yield better segmentations if their marginal costs do not completely decay to zero. Vanishing marginal costs lead to noisy segmentations, as has also been observed by Allène et al. [2009].Functions with non-vanishing marginal costs (curvature $\kappa_f < 1$) are not only empirically preferable but also result in better worst-case approximation bounds for ITB.

For the experiments in Section 6.1.4, the algorithm converged in five to ten iterations. We also found that only one initialization, $\mathcal{I} = \{\emptyset\}$, was sufficient. As the initialization only determines the approximation of costs for cooperating edges and not for those with modular costs, this initialization corresponds to starting with the graph cut solution of the model (6.1).

6.1.4. Experiments

For the task of interactive figure-ground segmentation, our experiments address three main questions: (i) What is the effect of coupling edges, and does this strengthen correct boundaries? We compare the cooperative BC energy E_f from Section 6.1.2 to the standard graph cut model (GC) [Boykov and Jolly, 2001] for pairwise potentials (referred to as "Graph Cut" in the sequel). (ii) What is the effect of the structure of coupling, i.e., the classes S_i? We compare to randomly assigning edges to classes. (iii) Does edge cooperation harm the segmentation of objects requiring standard smoothing? We address this using a standard benchmark data set.

When investigating the shrinking bias, we also compare to using "curvature regularity" that prefers boundaries with low curvature [Schoenemann et al., 2009] instead of considering length (pairwise terms) or congruity (cooperative model) of the boundary. Curvature is a higher-order feature, but, when implemented with graphs, it is still local and only takes into account few neighboring edges [El-Zehiry and Grady, 2010]. We re-implemented[2] the algorithm in [El-Zehiry and Grady, 2010]. All algorithms were implemented in C++, using the graph cut code by Boykov and Kolmogorov [2004], OpenCV for reading in and displaying images, and some preprocessing in Matlab. Code and data are available at `ssli.ee.washington.edu/~jegelka/cc`.

The congruity criterion requires defining classes of similar edges. We infer these classes via k-means clustering. If the image has shading, we use ℓ_1-distances between edge features $\phi_r(e) = \log(z_j/z_i)$, otherwise, we use squared Euclidean distances between linear features $\phi_l(e) = z_j - z_i$. All the cluster classes S_i, $1 \leq i \leq k$, of an image share the same threshold parameter ϑ in the cost (6.7) that determines the threshold $\theta_S = \vartheta w(S)$ when the discounts set in. In addition to the k cluster classes, one extra class S_{k+1} pools all edges whose incident pixels have identical color, i.e., $\phi_r(e) = 0$ or $\phi_l(e) = 0$. This extra class does not grant any discount and uses its own $\vartheta_{S_{k+1}} = 1$.

All of the algorithms involved have at least one parameter. In general, there are two ways to optimize the parameters of a segmentation model. One approach is such that the parameters for a model are tuned to each image to give the best results — this is similar to the small number of sliders in a photo editing program that lets the user adjust the parameters to optimize the segmentation boundary of a given image. An alternative strategy is more appropriate for off-line use, where there might be many images that need to be segmented but there is not the option to adjust the parameters individually to each image. In this case, we must choose one set of parameters for each technique (e.g., graph cut or cooperative cut), and those parameters must be used for a range of images. We compare the algorithms for both settings and will see that in both cases cooperative cuts outperform Graph Cuts with respect to the difficulties of low contrast and shrinking bias.

Data

We test the BC energy on three types of images: (i) shaded grayscale and color images; (ii) non-shaded color images with fine, delicate objects ("twigs and legs"); and (iii) the Grabcut image segmentation data [Rother et al., 2004, Blake et al., 2004]. For the Grabcut images, we use the given "Lasso" labeling. For (i) and (ii), we took high-resolution images using either a Canon 7D or a Canon 5DMkII DLSR camera. These images were hand-segmented and reduced in size. User labels were added by hand. The data for (i) consists of 8 grayscale and 7 color images, and the data for (ii) of 17 color images.

Grayscale images have the added difficulty that there is hardly any color information to guide the segmentation; the poor segmentation by mere unary terms in Figure 6.1 illustrates this.

[2]We thank Leo Grady and Noha El-Zehiry for fast responses and helpful hints about coding, as public code was not available.

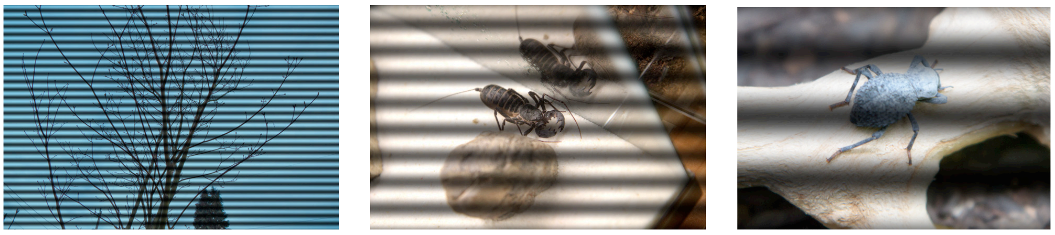


Figure 6.4. Examples for synthetically shaded images. Unary potentials were computed on the shaded images and hence less informative than those from the original images.

In addition to naturally shaded images, we synthetically added shading variations to equally lit color images. The artificial shading tests whether shade that varies locally with higher frequency affects the improvement achieved by cooperative cuts. Such variation can be harmful if shading is modelled explicitly on image patches. To generate examples for high-frequency shading, we selected images from the "twigs and legs" data. The pixel at location (x, y) in an image was multiplied by $0.4(1 + \sin(2\pi y/\gamma))$, and γ was varied across images. Figure 6.4 shows some examples.

Experimental Setup

To ensure equivalent conditions, all methods use the same weights on the terminal edges (i.e., the same unary potentials), the same 8-neighbor graph structure, and the same inter-pixel edge weights. The unary potentials either stem from color histograms [Boykov and Jolly, 2001], or from Gaussian mixture models (GMMs) with 5 components [Rother et al., 2004, Vicente et al., 2008]. The weight of an inter-pixel edge $e = (v_i, v_j) \in \mathcal{E}_n$ is

$$w(e) = 2.5 + 47.5 \exp(-0.5\|z_i - z_j\|^2/\sigma). \tag{6.8}$$

Recall that z_i is the observed color vector for pixel (node) v_i. These weights are equivalent to those used by Vicente et al. [2008], and the constant σ is the variance of inter-pixel color differences.

For all methods, λ refers to the "regularization coefficient", that is, the weight of the inter-pixel terms relative to the weight of the unary terms.

The total error is computed as the number of wrongly assigned pixels divided by the total number of unlabeled pixels ("unlabeled" means the pixels not labeled by the user). The total error, however, does not capture well the preservation of fine elongated structures, because those usually consist of only relatively few pixels. Therefore, we separately compute the *twig error*, where we only count the pixels in regions relevant to such fine difficult parts. Note that the twig error is in general much higher than the total error, because it refers to much fewer pixels, and a single mislabeled pixel contributes more to the error ratio than for the total error.

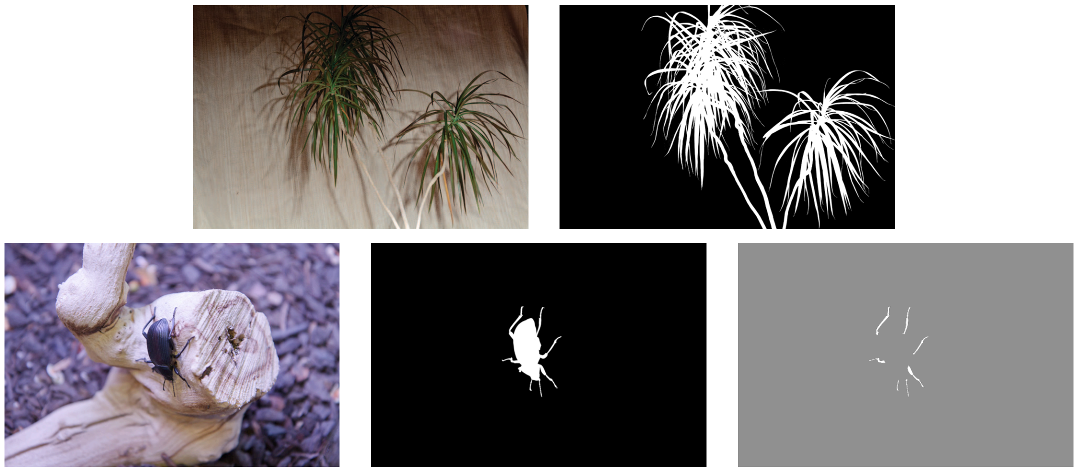


Figure 6.5. Examples for ground-truth labelings that were created by hand in Photoshop. The rightmost image of the beetle shows an example that was used to compute twig error — the gray region is ignored when computing the error.

Finally, we define a *joint error* as the weighted combination of total error and twig error,

$$\mathrm{err}_{\mathrm{joint}} = 2\mathrm{err}_{\mathrm{tot}} + \mathrm{err}_{\mathrm{twig}}. \tag{6.9}$$

Figure 6.5 shows some examples of ground truth labelings for total and twig error. The errors are with respect to a hand-segmented ground truth (obtained using Adobe Photoshop CS4 and CS5). The ground truth labelings contain a few gray pixels at the boundaries, wherever the exact boundary (at the pixel level) was not completely clear. These pixels were ignored when computing the error. This style of labeling is similar to that of the Grabcut data. For the Grabcut data, we used the given ground truth labelings.

6.1.5. Results: shrinking bias and the effect of the coefficient λ

All segmentation methods we test can be described as minimizing an objective

$$E(x) = \text{unary-terms}(x) + \lambda\, \text{smoothness-term}(x). \tag{6.10}$$

The smoothness term acts as a regularizer. We first investigate the regularization curve, i.e., the effect of varying λ.

Figure 6.6 demonstrates the behavior of Graph Cut and cooperative cut with respect to the regularization coefficient. For both methods, the total error shrinks as λ increases. As λ increases, Graph Cut however shortcuts many of the fine, elongated structures, resulting in a high twig error. This reflects the shrinking bias. As the fine structures only make up a small fraction of all pixels in the image, their disappearance does not affect the total error much. However, there is no value of λ that admits both a low total and a low twig error. In contrast, cooperative

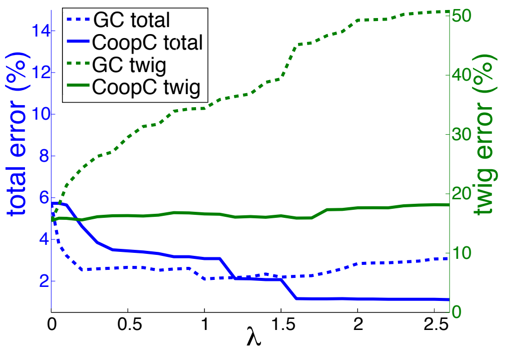


Figure 6.6. The effect of λ on the average total and twig error for Graph Cut (dashed line) and cooperative cut (solid line; $\vartheta = 10^{-4}$, 20 classes) on the "twigs and legs" data. The plot shows that a very low twig error, i.e., preservation of elongated structures, coincides with low λ and high total error for Graph Gut. For cooperative cut, the twig error increases much more slowly with λ, so that both low total and low twig error are possible simultaneously. The plot also demonstrates that cooperative cut is not too sensitive to the choice of parameters.

cut preserves fine structures even with higher regularization coefficient, and both total and twig error are minimized *simultaneously*. In addition, the plot shows that cooperative cut is not overly sensitive to parameter choice. The results in Sections 6.1.6 and 6.1.8, in particular Figures 6.7 and 6.12, illustrate the behavior indicated in Figure 6.6.

The errors in Figure 6.6 are averages over the "twigs and legs" data; we used $\vartheta = 10^{-4}$ and 20 classes for cooperative cut.

6.1.6. Results: fixed parameter for each data set

The results in Figure 6.6 use the same parameter values for all images in the data set. We now continue this viewpoint, and show qualitative and quantitative results for fixed parameters. However, we choose good parameters for each data set and method. The results visualize the tendencies indicated by Figure 6.6.

As a baseline, we show not only the results for standard Graph Cut, but also the results for cooperative cut (CoopCut) with only one class ($k = 1$, plus the class S_{k+1}), and for Graph Cut with logarithmic edge weights. Single-class CoopCut shows the effect of indiscriminate coupling, i.e., the discount is uniform on all edges (except S_{k+1}), and there is no group structure to it. The logarithmic weights are the non-cooperative equivalent of the edge features ϕ_r. This baseline has the ratio information contained in ϕ_r, but no coupling.

CoopCut always uses the edge weights in Equation (6.8), and the ratios ϕ_r only for finding edge types. For CoopCut, we show results for 1, 10, 15 and 20 edge

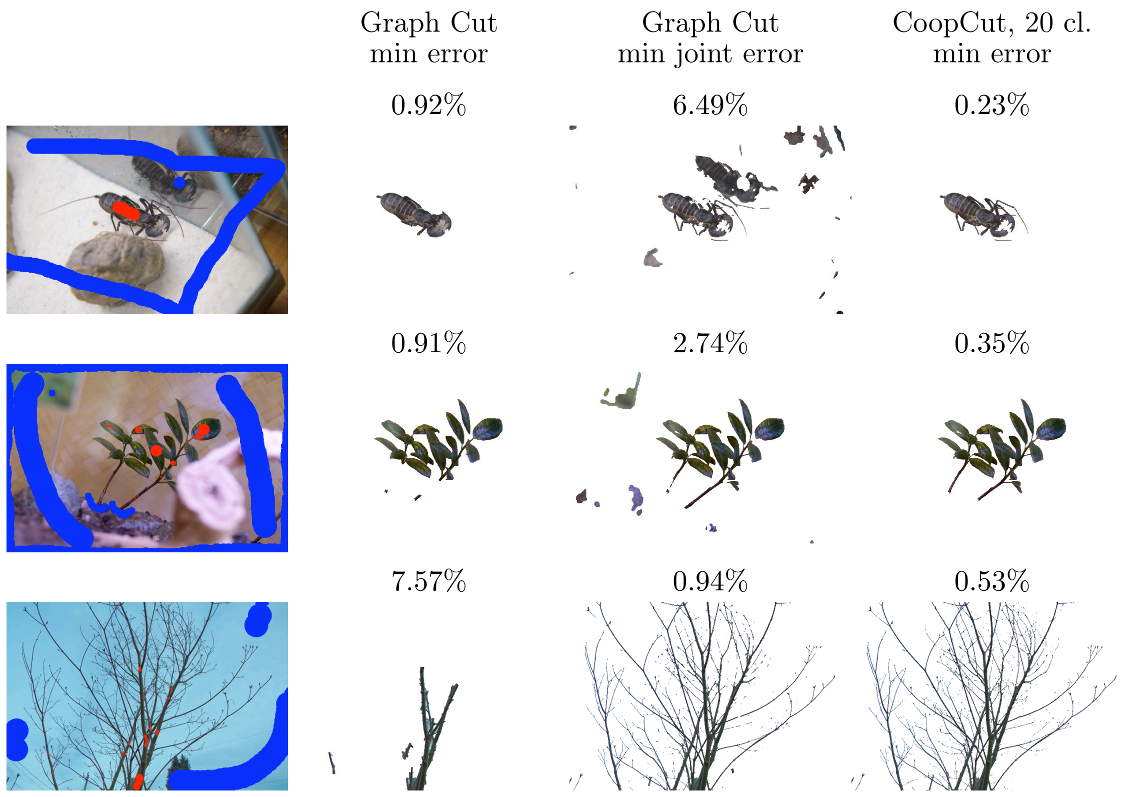


Figure 6.7. Effect of fixed parameters for Graph Cut and cooperative cut on the "twigs and legs" data. We show example results for parameters with minimum average error (Graph Cut, CoopCut) and minimum average joint error (Graph Cut). The first image contains a confounding mirrored insect that is, however, less sharp.

classes. The error denoted by "unary terms" is the error for Graph Cut with $\lambda = 0$, i.e., the segmentation resulting from unary terms alone.

Summarizing the results ahead, the quantitative results show that coupling edges (CoopCut) improves on the results by Graph Cut, and that structure in the coupling via edge groups ($k > 1$) improves on indiscriminate coupling. In fact, in many cases, a large part of the improvement is due to the grouping of edges. We do not know of any other method in the image segmentation literature that uses such groupings.

Shrinking bias and color images

On our benchmark images, Graph Cut appears to be very sensitive to parameter choice, and using the same single parameter (with the lowest error) for all images results in segmentations such as those shown in Figure 6.7: either fine structures are shortcut, or background pixels are included. The fixed parameters here are $\lambda = 1.5$ and $\lambda = 0.05$ for Graph Cut, and $(\lambda, 10^4\vartheta) = (1.5, 6)$ for CoopCut.

Twigs and legs: errors (in %)								
	GMM				histograms			
	min total error		min joint error		min total error		min joint error	
	total	twig	total	twig	total	twig	total	twig
unary terms	5.73	15.47	5.73	15.47	10.49	21.46	10.49	21.46
GC	2.10	34.40	3.78	18.08	3.00	50.58	6.74	28.95
CoopCut, 1	1.25	34.35	4.73	15.60	2.85	48.11	8.71	21.89
CoopCut, 10	1.01	18.27	1.17	16.43	**1.74**	**33.18**	2.17	29.41
CoopCut, 15	1.01	26.32	1.02	16.36	1.76	37.59	**2.82**	**27.40**
CoopCut, 20	**0.98**	**17.78**	**1.16**	**15.91**	1.75	35.05	2.66	28.51
curvature	3.82	56.09	5.73	16.00	5.00	64.44	10.31	21.85

Twigs and legs: parameters								
	GMM				histograms			
	λ	$10^4\vartheta$	λ	$10^4\vartheta$	λ	$10^4\vartheta$	λ	$10^4\vartheta$
GC	1.0	–	0.05	–	1.3	–	1.6	–
CoopCut, 1	1.5	4	10.0	0	1.6	4	12.0	0
CoopCut, 10	1.5	7	2.6	1	3.2	5	1.3	7
CoopCut, 15	22.0	0	1.7	2	3.5	6	2.0	1
CoopCut, 20	1.8	5	1.6	1	3.0	6	2.8	1
curvature	0.3	–	0.001	–	0.5	–	0.002	–

Table 6.1. Errors and parameters for the "twigs and legs" data set. Cooperative cut better preserves fine structures at low total error, i.e., without including too much background. "GMM" and "histogram" refer to the unary potentials.

Table 6.1 displays the corresponding quantitative results in detail. Shown are average percentages of mislabeled pixels (total and twig error) for two parameter settings: minimum total error and minimum joint error. Like the visual results, the quantitative results show that CoopCut can achieve low total error *and* low twig error, whereas the Graph Cut results minimize either one or the other.

Low contrast and shading in grayscale and color images

Figure 6.8 shows example segmentations of naturally shaded grayscale images obtained with fixed parameter settings, i.e., $\lambda = 0.1$ for Graph Cut, and $(\lambda, 10^4\vartheta) = (6, 5)$ for CoopCut; these are the parameters with the lowest average error. The error is indicated at the top of each image. Cooperative cut yields qualitatively and quantitatively better results.

The visual results for naturally shaded color images in Figure 6.9 are similar. The color however leads to more informative unary potentials, resulting in overall better segmentations. Therefore the differences between Graph Cut and cooperative cut lie mainly in the details of the segmentations, such as holes between the leaves of

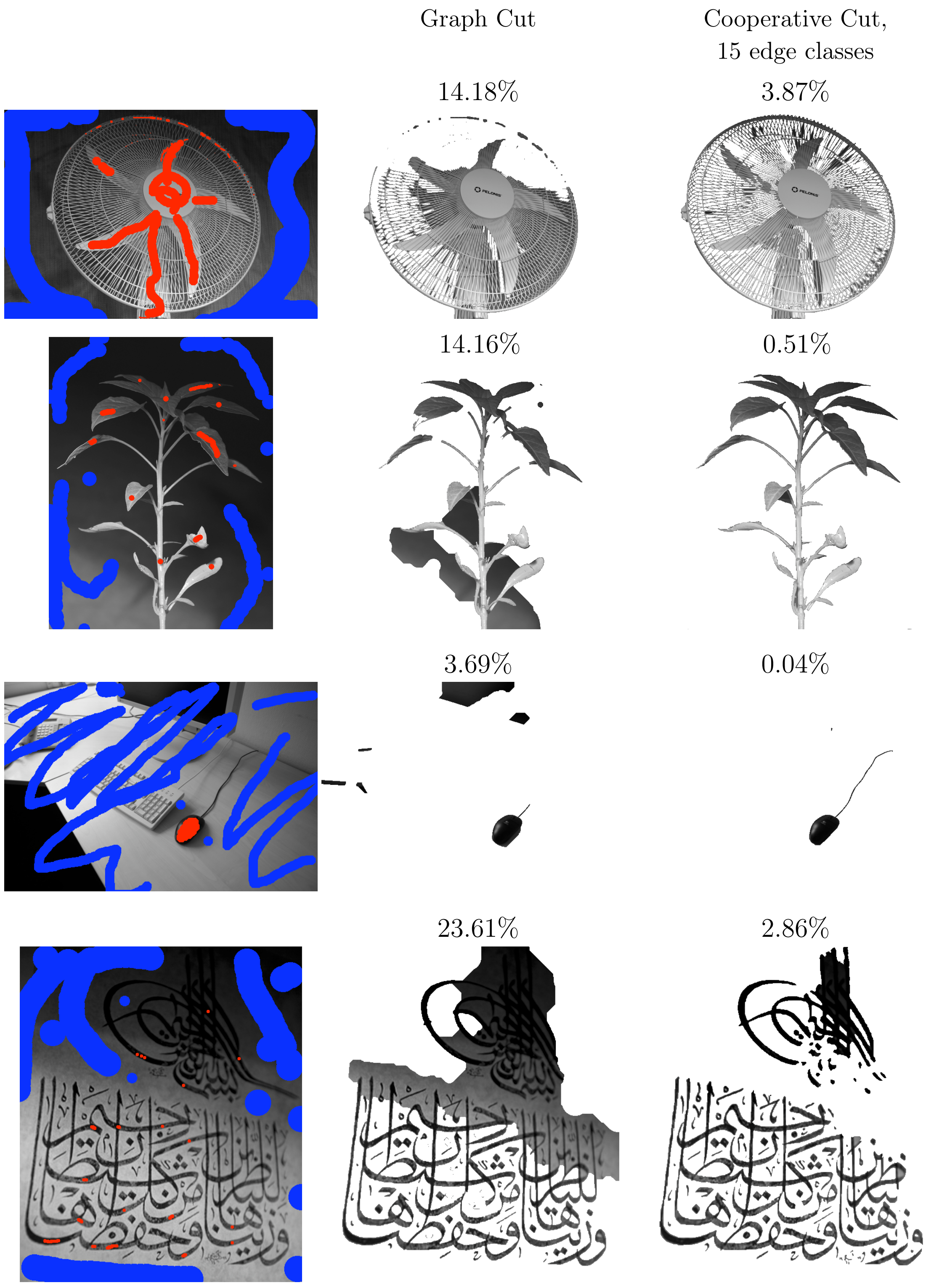


Figure 6.8. Example results with fixed parameters for shading in grayscale images.

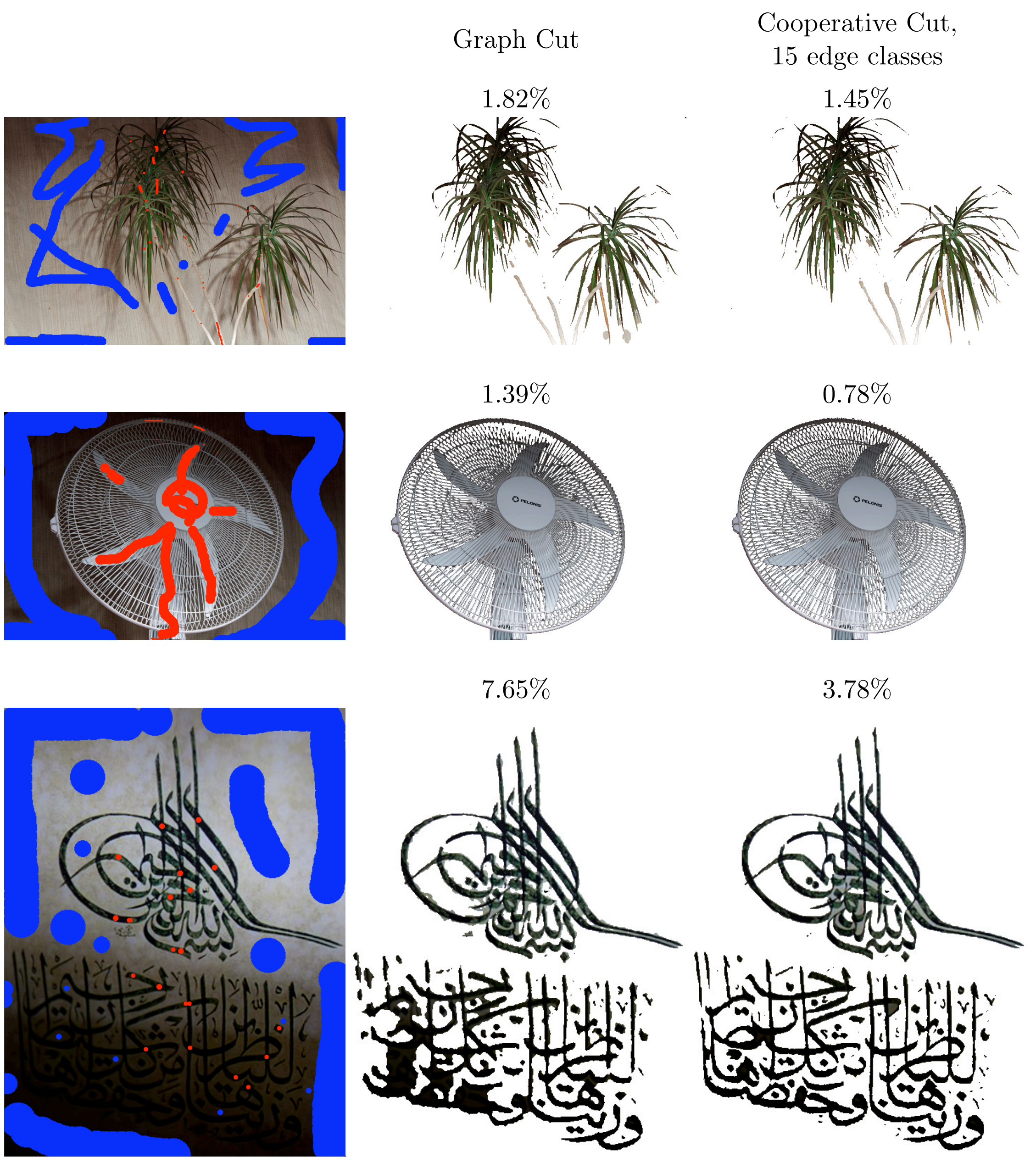


Figure 6.9. Example results with fixed parameters for shading in color images. As the color provides more guidance than grayscale values, the differences lie in the details.

Natural shading: errors (in %)				
	grayscale		color	
	GMM	hist	GMM	hist
unary terms	15.66	17.42	4.42	8.18
GC	14.03	14.71	3.41	6.49
GC,log wts	13.67	14.13	3.63	6.54
CoopCut, 1	11.58	10.61	2.95	5.31
CoopCut, 10	4.39	5.02	1.67	3.05
CoopCut, 15	**3.63**	**4.27**	1.69	**2.94**
CoopCut, 20	4.33	4.48	**1.62**	3.00
curvature reg.	17.40	19.48	3.93	7.37

Natural shading: parameters								
	grayscale				color			
	GMM		histogram		GMM		histogram	
	λ	$10^4\vartheta$	λ	$10^4\vartheta$	λ	$10^4\vartheta$	λ	$10^4\vartheta$
GC	0.10	–	0.6	–	0.05	–	0.10	–
GC, log wts	2.60	–	1.9	–	0.05	–	0.05	–
CoopCut, 1	70.0	1	70.0	6	100.0	0	22.0	3
CoopCut, 10	8.0	5	19.0	5	15.0	2	5.7	8
CoopCut, 15	6.0	5	20.0	4	19.0	2	3.5	10
CoopCut, 20	7.0	5	21.0	3	14.0	3	6.5	4
curvature reg.	0.01	–	0.01	–	0.025	–	0.04	–

Table 6.2. Errors in percent for the natural shading setting. Corresponding example visual results are shown in Figures 6.8 and 6.9. The lower table displays the parameters with which the errors in the upper table were achieved.

the plant, holes in the grid of the fan, or the smoothness of the boundary of the calligraphy.

Quantitative errors for both color and grayscale images with low-contrast regions are listed in Table 6.2. On grayscale images, CoopCut can reduce the total error by up to 74% compared to Graph Cut. Furthermore, the errors suggest that (i) the *coupling* is essential, since neither Graph Cut nor the logarithmic weights baseline or curvature regularization lead to segmentations as good as with cooperative cut; and (ii) the selective cooperation achieved via the edge classes S_i leads to a substantial improvement over indiscriminate uniform coupling (1 class). The actual number of classes, 10, 15 or 20, seems to be far less influential.

Synthetic high-frequency shading

As a proof of concept, we also show segmentation results for synthetic high-frequency shading. Such shades are less likely to occur naturally, but demonstrate that cooperative cuts do not rely on large patches having the same contrast: the results resemble those for the preceding two data sets.

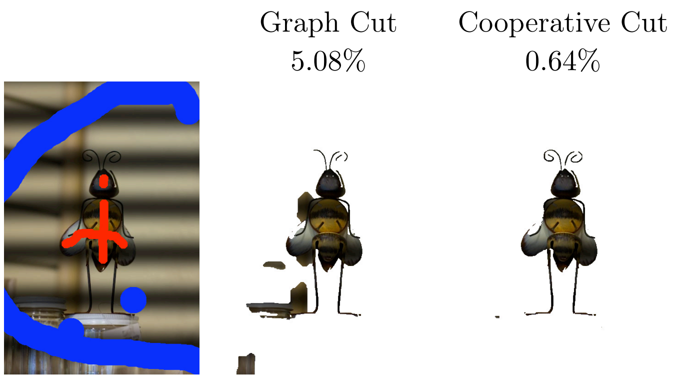


Figure 6.10. Example segmentations of a synthetically shaded image. For Graph Cut, $\lambda = 0.1$, and for CoopCut, $(\lambda, 10^4\vartheta) = (1.5, 50)$ (25 edge classes).

High-frequency shading: errors (in %)					parameters			
	min total error		min joint error		min tot. err.		min joint err.	
	total	twig	total	twig	λ	$10^4\vartheta$	λ	$10^4\vartheta$
unary terms	5.50	14.55	5.50	14.55	–	–	–	–
GC	2.56	20.96	3.43	13.54	0.2	–	0.05	–
GC,log wts	2.58	23.21	4.11	13.52	0.2	–	0.02	–
CoopCut, 1	1.49	33.03	3.10	12.53	0.9	6	0.05	40
CoopCut, 10	1.26	14.79	1.65	12.47	0.6	12	2.40	2
CoopCut, 15	1.27	14.69	1.73	12.39	0.7	9	0.80	6
CoopCut, 20	1.29	18.10	1.62	12.01	10.0	1	5.00	1
CoopCut, 25	**0.78**	**13.34**	**1.57**	**8.38**	0.9	15	2.2	1
curvature regul.	3.38	34.50	4.70	14.08	0.15	–	0.01	–

Table 6.3. Errors and parameters for synthetic high-frequency shading (GMM). Twig and total error together show that, while Graph Cut includes elongated structures at the cost of a higher total error, cooperative cut preserves them with much lower total error.

Unary terms are computed on the shaded image, so the color information is impaired compared to the original "twigs and legs" data from which the images were drawn. Table 6.3 shows both total and twig errors for the high-frequency data. The results are very similar to those for equally lit images: cooperative cut provides overall better segmentations and appears to be better suited to reduce the total error while preserving fine structures. Figure 6.10 illustrates an example segmentation.

6.1.7. Results: influence of the edge groups

The preceding sections suggest that cooperation between edges can be beneficial for segmentation. Moreover, the result tables indicate that restricting cooperation

method	total error	twig error	λ	$10^4\vartheta$
Shaded grayscale images				
CoopCut, 1 class	11.58		70.0	1
CoopCut, 15 random classes	10.97	–	4.0	7
CoopCut, 15 clustered classes	3.63	–	6.0	5
Shaded color images				
CoopCut, 1 class	2.95		100.0	0
CoopCut, 15 random classes	2.95	–	20.0	1
CoopCut, 15 clustered classes	1.69	–	19.0	2
Synthetic shading, min total error parameters				
CoopCut, 1 class	1.49	33.03	0.9	6
CoopCut, 15 random classes	0.98	23.24	0.6	7
CoopCut, 15 clustered classes	1.27	14.69	0.7	9
Synthetic shading, min joint error parameters				
CoopCut, 1 class	3.10	12.53	0.05	40
CoopCut, 15 random classes	2.64	8.96	0.1	25
CoopCut, 15 clustered classes	1.73	12.39	0.8	6

Table 6.4. Influence of clusters S_i in the cost function. For the naturally shaded images, much better segmentations result from restricting cooperation to *similar* edges. The results for random edge classes are similar to those for one class only. This tendency is not as clear for synthetic shading, but the tendency of a higher twig error with low total error seems to be stronger for random groups than for clusters. The errors here are averages, obtained with unary terms from GMMs.

to groups of edges strongly improves the results compared to an indiscriminate coupling resulting from a single edge class. To test how much the relation of the clusters to the actual image matters, we compare the results in Tables 6.2 and 6.3 for clusters of *similar* edges to results where edge classes S_i are assigned arbitrarily at random. That means we replace the k-means clustering by a random partition of edges into 15 classes, retaining the special class S_{k+1}. Table 6.4 shows that the errors for random groups are comparable to those for one uniform group and suggests that indiscriminate coupling is as good as global uniform coupling. These results indicate that meaningful edge types are indeed beneficial, and that the edge types do influence the results.

6.1.8. Results: best parameters per image

One may argue that the results in Section 6.1.6 do not show the full picture, and that some methods may yield better results if the parameters are optimized for each image individually, as could be done in a photo processing program. Therefore, we next show qualitative results where we choose parameters that lead to the lowest error on the given image.

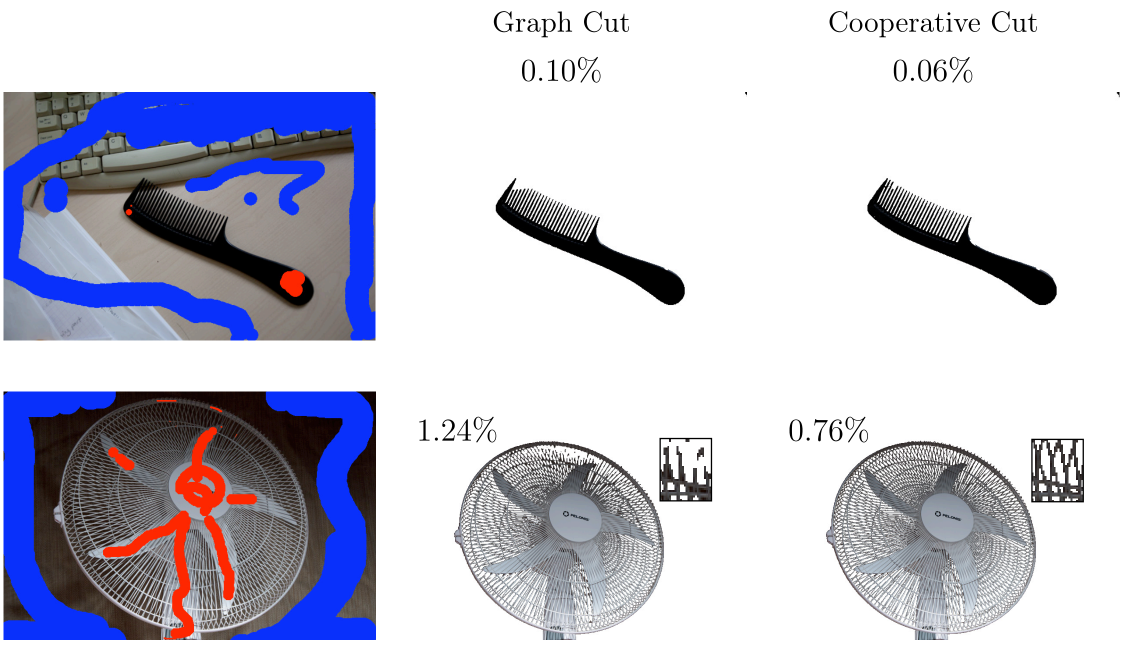


Figure 6.11. Example segmentations when choosing the optimal parameters (by total error) for each image separately. The comb illustrates a different type of shading, its segmentation is obstructed by its own shadow. (Parameters: GC $\lambda = 0.2, 0.05$, CoopCut $(\lambda, 10^4\vartheta) = (5.5, 10), (7.0, 3)$, 15 and 20 edge classes.)

Figures 6.12 and 6.13 suggest that even when choosing very good parameters for an image, shrinking bias is still a problem for Graph Cut, and the bias towards congruous boundaries improves the segmentations. The results with curvature regularization are worse than with Graph Cuts, possibly because the unary terms here give a preference to background parts that do not actually belong to the object.

Figures 6.11 and 6.14 show example segmentations for objects with shaded parts. The results are similar to those in the previous sections. The unary terms in Figure 6.14 demonstrate how limited the information in the grayscale color models is. In particular with such little guidance by color, including additional information about the boundary helps achieve better segmentations.

6.1.9. Results: Grabcut benchmark

As a “sanity check”, we address the effect of cooperation with objects that do not have elongated fine structures and that are not shaded either. Thanks to those properties, we do not expect a great gain from preferring congruous boundaries. Figure 6.15 shows two example segmentations for objects that are known to be challenging. Regarding average errors, Table 6.5 shows that the BC energy leads to slightly better segmentations than Graph Cut, but the gain is not as big as for

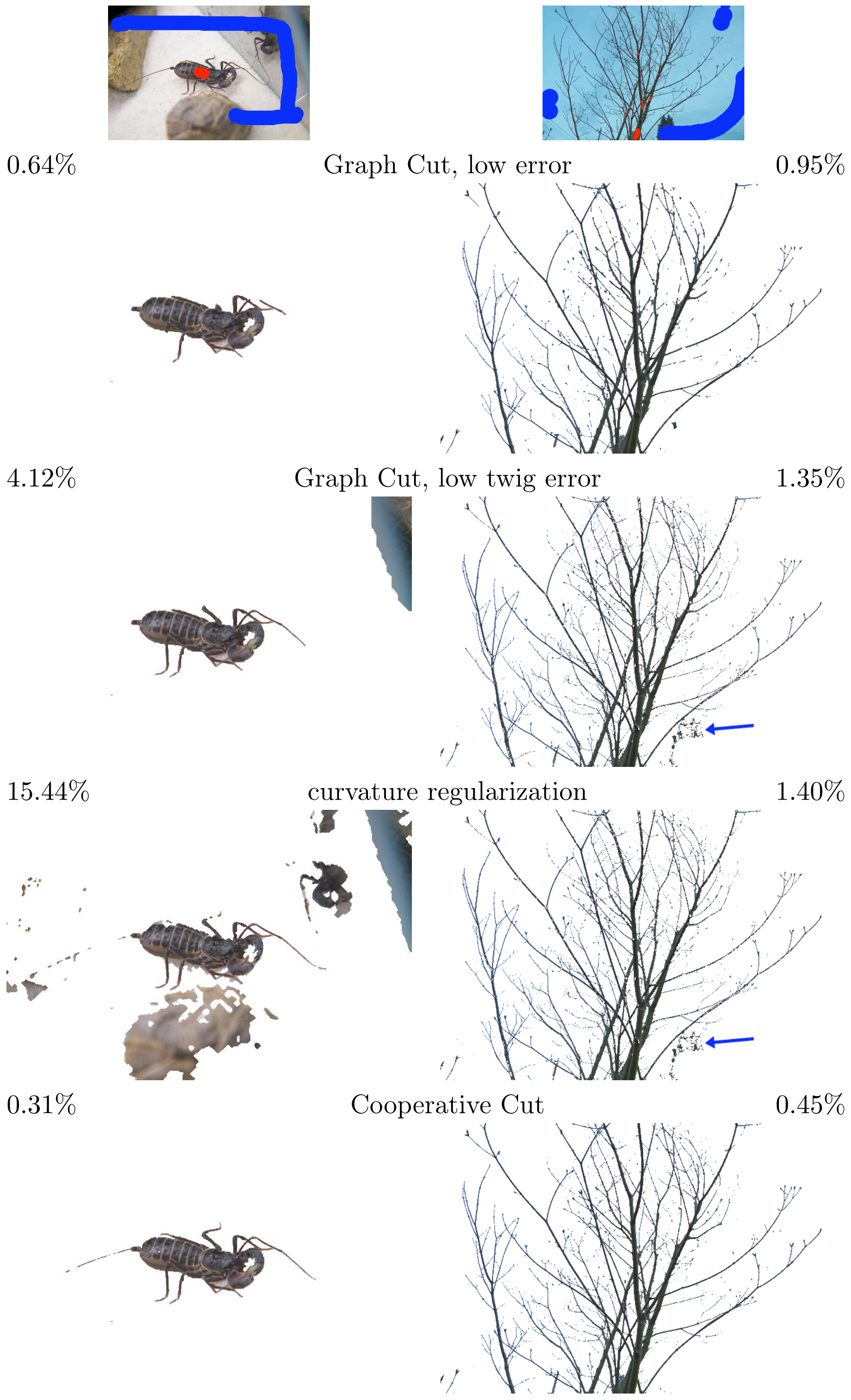


Figure 6.12. Example results for segmentation with the parameters that lead to the lowest error. When preserving fine structures, Graph Cut and curvature regularization still include parts of the background, as those methods are more strongly guided by the unary potentials. (Parameters: GC low error $\lambda = 1.5, 0.05$, GC low err_{twig} $\lambda = 1.0, 0.001$; curvature $\lambda = 0.03, 0.002$, CoopCut $(\lambda, 10^4\vartheta) = (1.5, 9), (1.8, 10)$).

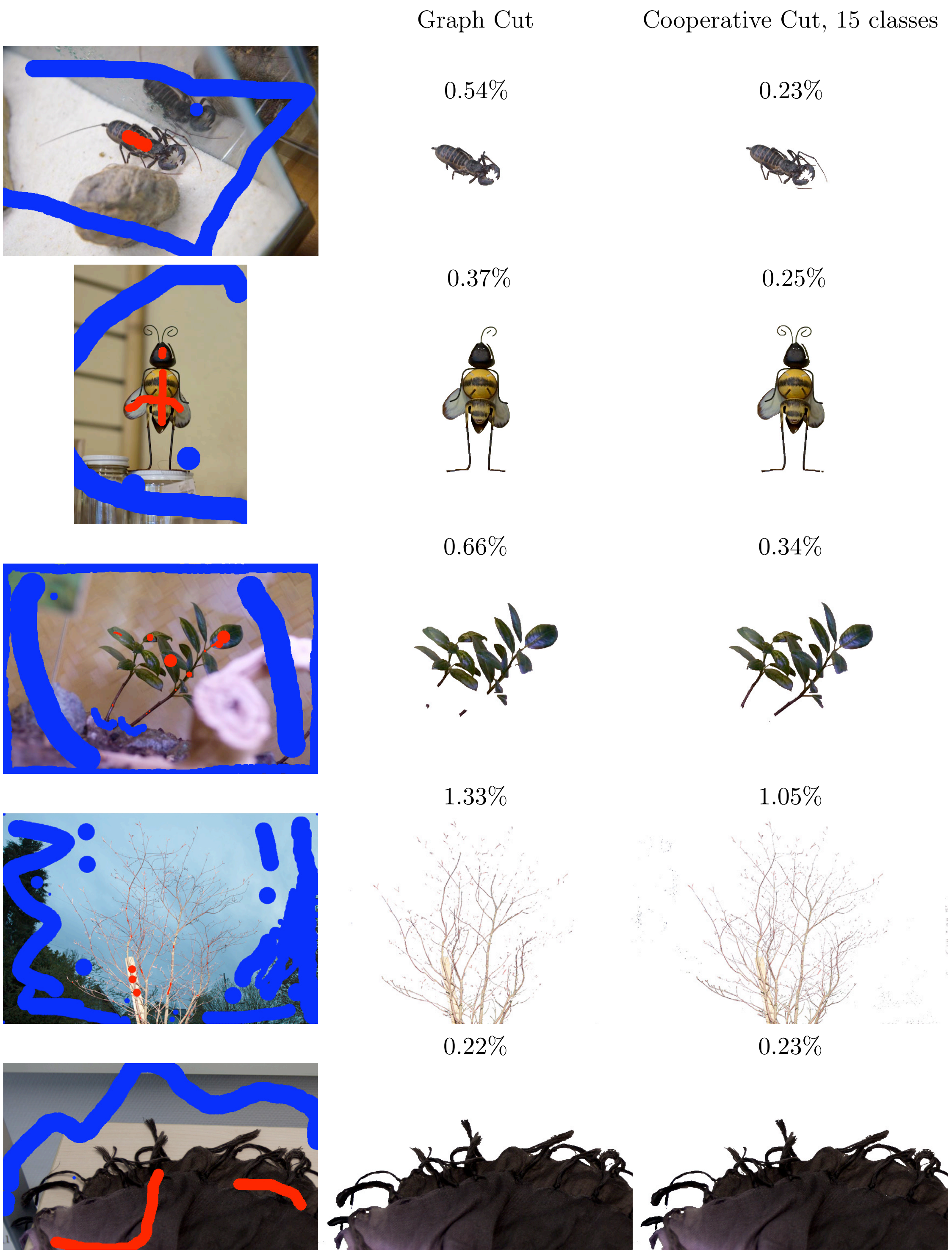


Figure 6.13. Example segmentations when choosing the optimal parameters (by total error) for each image separately.

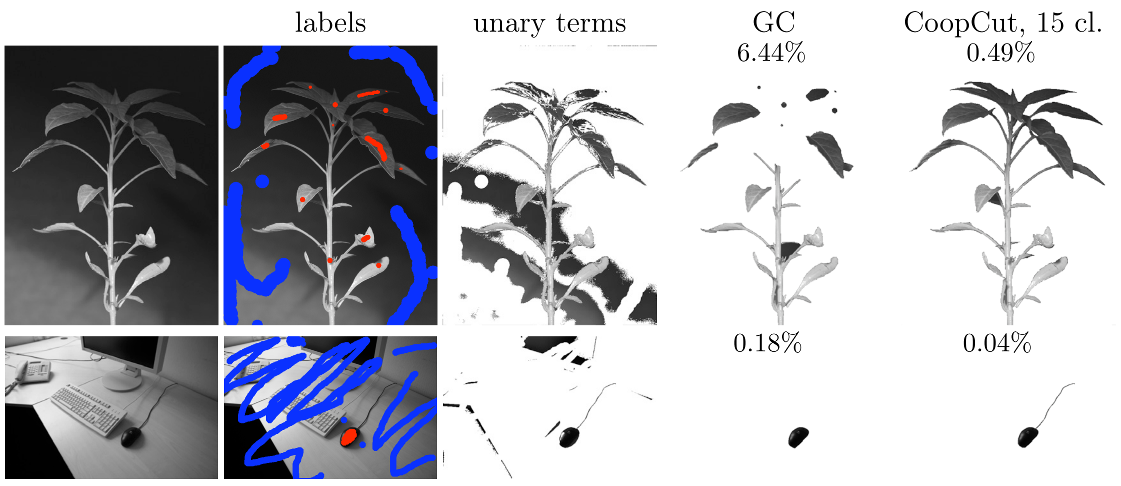


Figure 6.14. Example segmentations for parameters that lead to the lowest total error, for each image separately. The third column shows the segmentation by unary terms alone, illustrating the limited guidance owing to the lack of color information.

the other data sets. Nevertheless, these results suggest that cooperation does not hurt either.

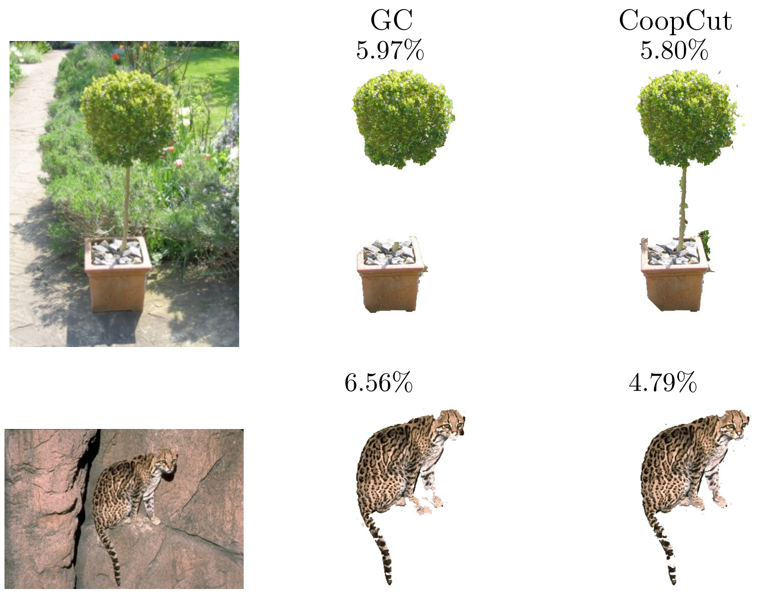


Figure 6.15. Two examples from the Grabcut data that are known to be difficult. Graph Cut either cuts off the tail or trunk, or includes parts of the background (compare the segmentations in [Blake et al., 2004, Lempitsky et al., 2009]). The cat is challenging for cooperative cuts as well, because, thanks to camouflage, boundaries congruous to the true outer boundary may be found even on the animal. (Parameters: GC $\lambda = 1.3, 0.05$, CoopCut $(\lambda, 10^4\vartheta) = (12, 3), (0.4, 7)$, 15 and 10 edge classes).

Grabcut data: errors and parameters						
	error (%)		par. GMM		par. hist.	
	GMM	histograms	λ	$10^4\vartheta$	λ	$10^4\vartheta$
GC	5.33 ± 3.7	6.88 ± 5.0	0.8	–	0.8	–
CoopCut, ϕ_l, 10 cl.	5.19 ± 3.5	6.51 ± 4.7	0.8	10.0	30.0	2.0
CoopCut, ϕ_l, 20 cl.	4.95 ± 3.2	6.27 ± 4.2	8.0	2.0	10.0	2.0
CoopCut, ϕ_r, 10 cl.	5.28 ± 3.7	6.50 ± 4.3	0.5	30.0	10.0	2.0
CoopCut, ϕ_r, 20 cl.	4.79 ± 3.1	6.12 ± 4.0	10.0	1.0	10.0	2.0

Table 6.5. Results on the Grabcut data with edge features ϕ_l and ϕ_r.

6.1.10. Summary and outlook

Many models for image segmentation essentially include two terms: the first one, here the unary potentials, biases the assignment of each pixel to foreground or background, judging, for instance, the fit of model and data by color or texture. The second part smoothens out the segmentation induced by the first part alone, for example by penalizing the curvature of the boundary, or the length of the boundary as with Graph Cut. If the guidance by the unary terms is weak and therefore overridden by the smoothness penalty, then the resulting segmentations will degrade. The shrinking bias, where few pixels inside a branch are outweighed by the edges around a long boundary, and low-contrast regions, where the color fails to provide discriminative guidance, are just two examples of this phenomenon.

The cooperative criterion for boundary congruity does a bit more than mere smoothing, as the global coupling of edges can provide additional guidance. This might be one reason why the segmentations obtained with cooperative cut in the experiments are much better than those obtained with Graph Cut.

The class-based boundary congruity energy changes the preference from using few edges, that means, short boundaries, to globally using few types of edges, that means, congruous boundaries. The experiments illustrate that coupling edges indeed improves segmentation results — Figure 6.6 indicates the suitability as a smoothness criterion — but that the type of coupling strongly influences the results too: indiscriminate uniform coupling via one class, or via random classes, is not as good as meaningful edge classes derived from clustering. Finally, compared to Graph Cut, edge cooperation does not corrupt segmentations if the object neither has elongated fine structures nor has low-contrast areas.

As an additional concrete example application, we mention the segmentation of biological images here. Genetic plant experiments, usually require observing the growth of a range of modified plants over time (Figure 6.16)[3]. This surveillance includes the tedious work of measuring the area and shape of plant parts, in particular of leaves. The automatic fine segmentation of leaf structures is one step towards

[3]Photos courtesy of George Wang (MPI for Developmental Biology).

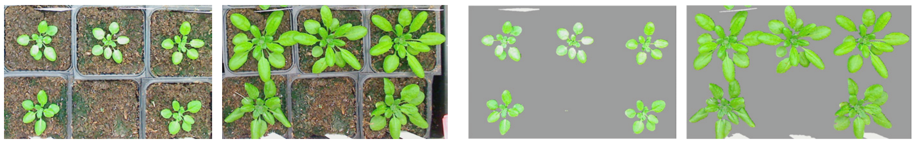


Figure 6.16. Segmentation of *Arabidopsis thaliana* at two stages (day 18 and 25). The color model was learned for a series of images together, and leaves in other images had strong white content. The segmentations are preliminary results.

an entirely automatic data collection that circumvents tedious measurements by hand.

6.2. Relaxations, regularization and further models in computer vision

The congruity bias for image segmentation was an application of the discrete MinCoopCut problem, and the above sections demonstrate its suitability as a smoothness or regularization criterion. Next we address the relaxation of MinCoopCut and its relations to regularization methods. These methods are mostly in computer vision, but the introduced non-uniform *edge-norms* may be of interest for a range of fields. We also derive an algorithm for solving a problem with a regularizing term derived from a cooperative cut. For readability, we here use non-bold letters for vectors x, y. To distinguish matrices from sets, we denote them by bold letters.

Many problems in machine learning and computer vision are stated in terms of a convex loss function $L(x)$ and a regularizing[4] term $\Omega(x)$, and the key problem becomes solving the optimization problem

$$\min_{x \in \mathbb{R}^{\mathcal{V}}} \quad L(x) + \lambda \Omega(x). \tag{6.11}$$

In the previous section, we already indicated that cooperative cut energies closely relate to such composite functions.

6.2.1. Stating MinCoopCut as regularized minimization

The next Lemma relates composite functions like (6.11) to cooperative cut energies, in particular to the BC energy (6.4). We denote by $(y)_+$ the element-wise maximum of 0 and y, and we employ the *node-edge incidence matrix* $\boldsymbol{A} \in \{0, 1, -1\}^{\mathcal{E} \times \mathcal{V}}$ of a

[4]We use the common term "regularizer" here for introducing a bias towards smoothness. (Total variation, for example, has frequently been used in this respect.) We leave statistical aspects as an interesting direction for future research.

graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ that is defined via its entries a_{e,v_i} for $v_i \in \mathcal{V}$ and $e = (v_j, v_k) \in \mathcal{E}$ as

$$a_{e,v_i} = \begin{cases} 0 & \text{if } i \neq j, k \\ 1 & \text{if } i = k \\ -1 & \text{if } i = j. \end{cases} \tag{6.12}$$

Lemma 6.1. *Minimizing a cooperative cut energy E_f (which is equivalent to solving an instance of* MinCoopCut*) for a structure graph with node-edge incidence matrix $\boldsymbol{A}$ corresponds to minimizing a problem of the form*

$$\min_{x \in \{0,1\}^n} \quad L(x) + \Omega(x),$$

with $\Omega(x) = \tilde{f}(|\boldsymbol{A}x|)$ for an undirected graph, and $\Omega(x) = \tilde{f}((\boldsymbol{A}x)_+)$ for a directed graph. In particular, the energy (6.4) corresponds to

$$\min_{x \in \{0,1\}^n} \quad \sum_{i=1}^{n} \psi_i(x) + \tilde{f}((\boldsymbol{A}x)_+).$$

The above cost functions are equally defined for continuous variables $x \geq 0$.

Proof. In a slightly modified fashion compared to the statement in Chapter 4, a relaxation of a cooperative cut problem in $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is

$$\begin{aligned} \min \quad & \tilde{f}(y) \\ \text{s.t.} \quad & y_e \geq x_j - x_i \quad \text{for all } e = (v_i, v_j) \in \mathcal{E} \\ & x \geq 0, \quad y \geq 0 \\ & x_s = 0 \\ & x_t = 1 \\ & y \in [0,1]^{\mathcal{E}}, \; x \in [0,1]^{\mathcal{V}}. \end{aligned} \tag{6.13}$$

The first constraint is defined by the matrix $\boldsymbol{A}$: in vector form it reads as $y \geq \boldsymbol{A}x$. Together with the nonnegativity constraint on y, this becomes $y \geq (\boldsymbol{A}x)_+$. Furthermore, the cost function $\tilde{f}(y)$ on y is nondecreasing. This means that for a given feasible x, an optimal $y(x)$ must be $y(x) = (\boldsymbol{A}x)_+$. If $\mathcal{G}$ is undirected, then the cut can be solved as a directed cut problem on a directed equivalent of $\mathcal{G}$. The directed graph has edges (v_i, v_j) and (v_j, v_i) for each undirected edge $\{v_i, v_j\}$ in $\mathcal{G}$. This implies that $y(x)_e = \max\{(x_j - x_i)_+, (x_i - x_j)_+\} = |x_i - x_y|$ and therefore an optimal solution is $y(x) = |\boldsymbol{A}x|$.

Moreover, we can include the constraints $x_s = 0, x_t = 1$ into the cost function, for example as $L(x) = \gamma((x_s - 0)^2 + (x_t - 1)^2)$ for a large enough $\gamma > 0$. With these observations, solving the Program (6.13) is equivalent to solving

$$\min_{x \in [0,1]^n} \quad L(x) + \Omega(x) \tag{6.14}$$

with $\Omega(x) = \tilde{f}((\boldsymbol{A}x)_+)$ for a directed graph, and $\Omega(x) = \tilde{f}(|\boldsymbol{A}x|)$ for an undirected graph. For a discrete solution, the minimization ranges over $x \in \{0,1\}^n$.

For the BC energy (6.4), the terminal edges in $\mathcal{G}$ are charged by an additive function, that is, $f(S) = \sum_{e \in S} w(e)$ for all S that only consist of edges incident to s or t. To obtain the second formulation in Lemma 6.1, we enforce the constraints on x_s and x_t. Fixing $x_s = 0$ sets $y_e = x_i$ for any edge $e = (s, v_i)$, and fixing $x_t = 1$ sets $y_e = 1 - x_i$ for any edge $e = (v_i, t)$. If $\boldsymbol{A}$ is the incidence matrix for the sub-graph $\mathcal{G}_{\setminus s,t} = (\mathcal{V} \setminus \{s,t\}, \mathcal{E}_n)$, then the energy for fixed x_s, x_t is minimized by solving

$$\min_{x \in \{0,1\}^n} \quad \sum\nolimits_{i=1}^{n} w(s,v_i)x_i + w(v_i,t)(1-x_i) + \tilde{f}((\boldsymbol{A}x)_+) \tag{6.15}$$

$$\equiv \min_{x \in \{0,1\}^n} \quad \sum_i \psi_i(x_i) + \tilde{f}((\boldsymbol{A}x)_+) + const. \tag{6.16}$$

The derivation for undirected graphs is analogous. □

The proof shows that $\Omega(x)$ here penalizes the differences of x_i along edges. Depending on the graph structure, these differences can be viewed as a discrete gradient. We call it the *edge gradient*.

Properties of the function $\Omega(x) = \tilde{f}((\boldsymbol{A}x)_+)$

To understand the formulations in Lemma 6.1 better, we investigate some properties of the function $\Omega(x)$. We argue that it is convex and a norm on the edge gradient.

Proposition 6.1. *The functions $\Omega(x) = \tilde{f}(|\boldsymbol{A}x|)$ and $\Omega(x) = \tilde{f}((\boldsymbol{A}x)_+)$ are convex.*

To prove the proposition, we will use another observation.

Proposition 6.2. *Let $f : 2^{\mathcal{E}} \to \mathbb{R}_+$ be a nondecreasing submodular function. Then its Lóvasz extension is monotone: for two vectors $x, y \in \mathbb{R}_+$, where y dominates x ($y_i \geq x_i$ for all entries i), it holds that $\tilde{f}(y) \geq \tilde{f}(x)$.*

Proof (Proposition 6.2). Without loss of generality, assume $f(\emptyset) = 0$. The Lovász extension can also be defined in terms of level sets $X_\theta = \{i \mid x_i \geq \theta\}$ as

$$\tilde{f}(x) = \int_0^\infty f(X_\theta) d\theta. \tag{6.17}$$

Then, using $X_\theta \subseteq Y_\theta$ for all θ and the monotonicity of f, it follows that

$$\tilde{f}(x) = \int_0^\infty f(X_\theta) d\theta \leq \int_0^\infty f(Y_\theta) d\theta = \tilde{f}(y). \qquad \square$$

Proof (Proposition 6.1). The proposition follows in a straightforward way from the convexity of $\tilde{f}(|\cdot|)$ [Bach, 2010]. For the second function, we observe that $x_+ + y_+ \geq (x+y)_+$ for any $x, y \in \mathbb{R}$. To derive convexity, we use Proposition 6.2 and the convexity of the Lovász extension $\tilde{f}$.

$$\begin{aligned}\tilde{f}((\boldsymbol{A}(\gamma x_1 + (1-\gamma)x_2)_+) &\leq \tilde{f}(\gamma(\boldsymbol{A}x_1)_+ + (1-\gamma)(\boldsymbol{A}x_2)_+) \qquad (6.18)\\ &\leq \gamma\tilde{f}((\boldsymbol{A}x_1)_+) + (1-\gamma)\tilde{f}((\boldsymbol{A}x_2)_+). \qquad \square\end{aligned}$$

Bach [2010, Prop.1] shows that the function $\tilde{f}(|y|)$ is a norm if f is nondecreasing and normalized ($f(\emptyset) = 0$). This means that $\Omega(x)$ is a norm on the vector y_+ or y of differences $y = \boldsymbol{A}x$. The function Ω, however, is not a norm on x, because the null space of $\boldsymbol{A}$ is nontrivial and includes the all-ones vector. Nevertheless, being a norm on y means that the relaxation of MinCoopCut belongs to a larger family of problems that is described next.

6.2.2. Edge-norms

We observed that $\Omega(x)$ can be written in terms of a particular norm defined by a Lovász extension, $\|\boldsymbol{A}x\|$ for undirected and $\|(\boldsymbol{A}x)_+\|$ for directed graphs. We will call such norms of cut vectors *edge-norms.* For continuous variables, the cut vector is a vector of differences.

Definition 6.1 (Edge-norm)**.** Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, let $\boldsymbol{A} \in \{-1, 0, 1\}^{\mathcal{E}\times\mathcal{V}}$ denote its node-edge incidence matrix. For a discrete $x \in \{0,1\}^{\mathcal{V}}$ or a continuous $x \in \mathbb{R}_+^{\mathcal{V}}$, we define its corresponding *edge-norm* with respect to a norm $\|\cdot\|$ as

$$\begin{aligned}\Omega(x) &\triangleq \|\boldsymbol{A}x\| && \text{if } \mathcal{G} \text{ is undirected, and}\\ \Omega(x) &\triangleq \|(\boldsymbol{A}x)_+\| && \text{if } \mathcal{G} \text{ is directed.}\end{aligned}$$

If $\|\cdot\|$ is a norm, then $\|(\cdot)_+\|$ is a semi-norm. Depending on the graph structure and the norm, Ω can express a variety of established and novel terms. Examples will be shown in Section 6.2.4. By the reasoning in Section 6.2.1, cooperative cuts and their relaxations correspond to edge-norms. Conversely, many edge-norms define cooperative cuts:

Lemma 6.2. *An edge-norm over discrete variables for any ℓ_p-norm with $p \geq 1$ corresponds to a cooperative cut. Similarly, a discrete edge-norm for any $\ell_{1,p}$ group norm with $p \geq 1$ corresponds to a cooperative cut.*

Proof. Let $y = \boldsymbol{A}x$ be the vector of differences between the node labels x_i. By definition, y is the characteristic vector of a cut $y = \delta(X_0)$ for $X_0 = \{v_i \in \mathcal{V} \mid x_i = 0\}$. Therefore, it holds that

$$\|y\|_p = \Big(\sum_{e\in\mathcal{E}} y_e^p\Big)^{1/p} = \Big(\sum_{e\in\delta(X_0)} 1\Big)^{1/p} = |\delta(X_0)|^{1/p}. \qquad (6.19)$$

This is a concave function of the cardinality of a set of edges, and therefore a submodular function on sets of edges. An equivalent statement holds for the directed case. The group norm is a sum of ℓ_p-norms, and the result follows analogously. $\square$

The reader should however be warned that the Lovász extension $\tilde{f}(|x|)$ for the function $f(S) = |S|^{1/p}$ is different from $\|x\|_p$ for $p > 1$.

The proof of Lemma 6.2 shows that the common ℓ_p-norms correspond to a uniform, indiscriminate coupling of all edges in the graph. The function $f(S) = |S|^{1/p}$ only depends on the cardinality of the argument (it can be extended to a weighted sum via a Mahalanobis norm for a diagonal matrix) and is symmetric in this sense. It does not involve a specific coupling of certain groups of edges like the BC energy we used for image segmentation. In fact, independently to the work at the beginning of this chapter, Bach [2010] (extended in [Bach, 2011]) introduced the Lovász extension norm $\Omega(x) = \tilde{f}(|x|)$ on the variables directly as a sparsity-inducing, regularizing term for variable selection. There, the underlying submodular function is defined on subsets of nonzero variables, and, as for BC energies, induces preferences for certain sparsity structures, that is, patterns of nonzero variables.

Section 6.2.4 will show that instances of edge-norms have been used as regularizing terms. Therefore, we derive an algorithm to solve a composite problem like (6.11) consisting of a convex loss function and a cooperative edge-norm.

6.2.3. A proximal splitting algorithm

Motivated by a common use of edge-norms as regularizing terms, we derive an algorithm to minimize the sum of a strictly convex loss function and a cooperative edge-norm,

$$\min_{x \in \mathbb{R}^{\mathcal{V}}} \quad L(x) + \lambda \tilde{f}(|\boldsymbol{A}x|). \tag{6.20}$$

We assume that the loss $L(x)$ prevents trivial or unbounded solutions. Minimization problems composed of a strictly convex, differentiable function and a nonsmooth convex function can be solved by proximal methods [Combettes and Pesquet, 2011]. At their core, proximal methods iterate two steps. First, in the forward step, they take a gradient descent step with respect to the loss L, which leads to a point z. Then, in the backward step, the proximity operator finds a close feasible point with relatively small Ω. The backward step is the generalization of a projection. The proximity operator is defined as

$$\text{prox}_\Omega(z) = \text{argmin}_x \tfrac{1}{2}\|x - z\|^2 + \lambda\Omega(x). \tag{6.21}$$

Together, the forward and backward steps then define the iteration

$$x_{t+1} = \text{prox}_\Omega(x_t - \gamma_t \nabla L(x_t)). \tag{6.22}$$

The gradient step can also be accelerated [Beck and Teboulle, 2009]. The key issue here is computing the proximity operator. In the rare case that the cooperative cut energy for f and $\mathcal{G}$ is submodular, the operator (6.21) can be computed by a parametric maxflow [Chambolle and Darbon, 2009]. However, as implied by Lemma 5.1, cooperative edge-norms $\Omega(x)$ are in general not submodular, and the flow method does not apply. Therefore, we derive a more general algorithm that applies to any $\Omega(x) = \tilde{f}(|\boldsymbol{A}x|)$ derived from a nondecreasing, normalized submodular function f. If L is the squared loss, then the entire problem (6.20) corresponds to the proximal problem, and we can solve it directly.

Let $h_z(x) = \frac{1}{2}\|x - z\|^2$. We first derive the dual of the objective (6.21). To do so, we recall the relaxation (6.13) that has an edge indicator variable $y = \boldsymbol{A}x$. With this auxiliary variable, the proximal problem becomes

$$\min_{x,y} \quad h_z(x) + \lambda \tilde{f}(|y|) \quad \text{s.t. } \boldsymbol{A}x = y. \tag{6.23}$$

Let μ denote the dual variables for the constraint. Then the Lagrangian is

$$\mathscr{L}(x, y, \mu) = h_z(x) + \lambda \tilde{f}(|y|) + \mu^\top(\boldsymbol{A}x - y). \tag{6.24}$$

Using the shorthand $g(y) = \tilde{f}(|y|)$, the Lagrangian leads to the dual function $\mathscr{H}(\mu) \triangleq \inf_{x,y} \mathscr{L}(x, y, \mu)$, which can be computed as

$$\begin{aligned}
\mathscr{H}(\mu) &= \inf_x (h_z(x) + \mu^\top \boldsymbol{A}x) + \inf_y (\lambda g(y) - \mu^\top y) && (6.25)\\
&= -\sup_x(-\mu^\top \boldsymbol{A}x - h_z(x)) - (\sup_y \mu^\top y - \lambda g(y)) && (6.26)\\
&= -h_z^*(-\boldsymbol{A}^\top \mu) - \lambda g^*(\lambda^{-1}\mu), && (6.27)\\
&= \begin{cases} -h_z^*(-\boldsymbol{A}^\top \mu) & \text{if } |\lambda^{-1}\mu| \in P_f \\ -\infty & \text{otherwise.} \end{cases} && (6.28)
\end{aligned}$$

Here, h^* and g^* are the Fenchel duals of h and g, respectively. We derive g^* via the definition of the Fenchel dual and the Lovász extension:

$$\begin{aligned}
g^*(w) &= \sup_y \quad w \cdot y - \tilde{f}(|y|) && (6.29)\\
&= \sup_y \quad w \cdot y - \max_{s \in P_f} s \cdot |y| && (6.30)\\
&= \sup_y \min_{s \in P_f} \quad w \cdot y - s \cdot |y| && (6.31)\\
&= \sup_y \min_{|s| \in P_f} \quad w \cdot y - s \cdot y && (6.32)\\
&= \sup_y \min_{|s| \in P_f} \quad (w - s) \cdot y && (6.33)\\
&= \begin{cases} 0 & \text{if } |w| \in P_f \\ \infty & \text{otherwise.} \end{cases} && (6.34)
\end{aligned}$$

To get from (6.31) to (6.32), we use the following proposition. Since f is nonnegative and nondecreasing, P_f must contain a nonnegative vector.

Proposition 6.3. *If f is nondecreasing, then any $s^* \in \operatorname{argmax}_{|s| \in P_f} s \cdot y$ satisfies $|s^*| \in \operatorname{argmax}_{s \in P_f} s \cdot |y|$. Vice versa, for each $t^+ \in \operatorname{argmax}_{s \in P_f} s \cdot |y|$, there exists $t^* \in \operatorname{argmax}_{|s| \in P_f} s \cdot y$ with $|t^*| = t^+$ and $t^+ \cdot |y| = t^* \cdot y$.*

Proof. We first observe that $s^* \cdot y = |s^*| \cdot |y|$. This equality follows because if there exists an index j with $\operatorname{sign}(s^*_j) \neq \operatorname{sign}(y_j)$, then either $y_j = 0$, in which case $s^*_j y_j = |s^*_j||y_j|$, or we can construct a vector s' with $s'_i = s^*_i$ for $i \neq j$, and $s'_j = -s^*_j$. This new vector s' contradicts the optimality of s^*, because $|s'| = |s^*| \in P_f$ and $s' \cdot y = s^* \cdot y - 2s^*_j y_j > s^* \cdot y$.

We prove the Proposition by contradiction. Assume $|s^*| \notin \operatorname{argmax}_{s \in P_f} s \cdot |y|$. Then there exists a nonnegative vector $t \in P_f$ with $t \cdot |y| > |s^*| \cdot |y|$. We can construct t by sorting the elements in $|y|$ in descending order, and by setting $t_j = f(A_j) - f(A_{j-1}) \geq 0$ for the chain $A_0 = \emptyset$ and $A_j = \{e_{\pi(i)} | i \leq j\}$. The nonnegativity of t results from f being nondecreasing, and it implies that $t = |t|$, and thus $|t| \in P_f$ too.

From this vector t, we construct a vector t' such that $t' \cdot y = t \cdot |y|$: let $t'_i = \operatorname{sign}(y_i) t_i$. Then it holds that

$$t' \cdot y = t \cdot |y| > |s^*| \cdot |y| \geq s^* \cdot y. \tag{6.35}$$

As $|t'| = t \in P$, the vector s^* cannot be in $\operatorname{argmax}_{|s| \in P_f} s \cdot y$.

The reverse is analogous. We construct t^* from t^+ like t' from t. If t^* was not a maximizer of $s \cdot y$, then $|t^*| = t^+$ would not be a maximizer of $s \cdot |y|$, a contradiction. □

The indicator function (6.34) follows because if $|w| \in P_f$, then setting $s = w$ is feasible and results in $(w - s) \cdot y = 0$. Thus, in that case, $\min_{|s| \in P_f} (w - s) \cdot y \leq 0$ for any vector y, and $y = 0$ is a maximizer. If $|w| \notin P_f$, then there exists a set $A \subseteq \mathcal{E}$ with $|w|(A) > f(A) \geq |s|(A)$ for all feasible s. Consider a vector y' with $y'_i = 0$ for all $e_i \notin A$, and $y'_i = c \operatorname{sign}(w_i)$ for some constant c otherwise. For this y', it holds that

$$(w - s) \cdot y' \geq w \cdot y' - |s| \cdot |y'| = (|w| - | s|) \cdot |y'| = (|w|(A) - |s|(A))c. \tag{6.36}$$

As c tends to infinity, this product becomes unbounded, because $(|w|(A) - |s|(A)) > 0$ for any $|s| \in P_f$.

The conjugate (6.34), finally, implies the dual function (6.28), which leads to the optimization problem

$$\max_{\mu} \quad -h^*_z(-\boldsymbol{A}^\top \mu) \quad \text{s.t. } |\lambda^{-1}\mu| \in P_f \tag{6.37}$$

$$\equiv \max_{\mu} \quad -h^*_z(-\boldsymbol{A}^\top \mu) \quad \text{s.t. } |\mu| \in P_{\lambda f} \tag{6.38}$$

Substituting $h^*_z(x) = z \cdot x + \frac{1}{2}\|x\|^2$, we obtain $h^*_z(-\boldsymbol{A}^\top \mu) = \frac{1}{2}\|\boldsymbol{A}^\top \mu\|^2 - z^\top \boldsymbol{A}^\top \mu + \frac{1}{2}\|z\|^2 - \frac{1}{2}\|z\|^2$ and the dual problem

$$\min_{\mu} \quad \tfrac{1}{2}\|\boldsymbol{A}^\top \mu - z\|^2 \quad \text{s.t. } |\mu| \in P_{\lambda f}. \tag{6.39}$$

Two questions remain to be solved: how to optimize the above dual problem and how to recover the optimum x^* of the primal problem from it.

To solve the second problem, we use the Karush-Kuhn-Tucker optimality conditions that the derivative of the Lagrangian with respect to x disappears at x^*:

$$\frac{\partial \mathscr{L}(x, y, \mu)}{\partial x} = (x - z) + \boldsymbol{A}^\top \mu. \tag{6.40}$$

Setting this derivative to zero yields $x^* = z - \boldsymbol{A}^\top \mu^*$. Next we address how to solve the dual problem (6.39).

Minimizing a quadratic function over P_f

The dual problem (6.39) is a convex minimization over a subset of the submodular polyhedron. An algorithm by von Hohenbalken [1975] maximizes pseudoconcave function h over a submodular polyhedron, if the condition $\partial h(\boldsymbol{D}y + w)/\partial y = 0$ can be solved efficiently (e.g., in closed form) for transformation matrices $\boldsymbol{D}$. This algorithm applies to minimize $h(\mu) = \frac{1}{2}\|\boldsymbol{A}^\top \mu - z\|^2$, because the function h is convex and the derivative condition holds. The derivative is

$$\begin{aligned} \partial h(Dy + w)/\partial y &= \partial/\partial y(\tfrac{1}{2}\|\boldsymbol{A}^\top \boldsymbol{D}y + \boldsymbol{A}^\top w - z\|^2) && (6.41)\\ &= \boldsymbol{D}^\top \boldsymbol{A}(\boldsymbol{A}^\top \boldsymbol{D}y + \boldsymbol{A}^\top w - z). && (6.42)\end{aligned}$$

Setting this derivative to zero yields

$$\begin{aligned} \boldsymbol{D}^\top \boldsymbol{A}\boldsymbol{A}^\top \boldsymbol{D}y &= -\boldsymbol{D}^\top \boldsymbol{A}\boldsymbol{A}^\top w + \boldsymbol{D}^\top \boldsymbol{A}z && (6.43)\\ \Rightarrow \quad y &= -(\boldsymbol{D}^\top \boldsymbol{A}\boldsymbol{A}^\top \boldsymbol{D})^\dagger(\boldsymbol{D}^\top \boldsymbol{A}\boldsymbol{A}^\top w - \boldsymbol{D}^\top \boldsymbol{A}z). && (6.44)\end{aligned}$$

The constraints of Problem (6.39), however, do not directly refer to P_f, but require the *absolute value* of μ to be in P_f. This corresponds to μ being in a polytope that is a subset of P_f. Von Hohenbalken's algorithm invokes the polytope only for a linear maximization over it. Proposition 6.3 shows that we can find $\mu^* \in \operatorname{argmax}_{|\mu| \in P_f} c \cdot \mu$ by finding $\mu^+ \in \operatorname{argmax}_{\mu \in P_f} |c| \cdot \mu$. For the latter problem, the greedy algorithm applies.

6.2.4. Special cases of edge-norms in the literature

Finally, we point out special cases of edge-norms in the literature. As opposed to the BC energies in Section 6.1.2, most of the examples in the sequel use uniform and not group-inducing norms. Lemma 6.2 implies that the discrete versions of several of the examples below are cooperative cuts. Furthermore, one of the most prominent examples for edge-norms is the standard minimum cut, an edge-norm that uses the ℓ_1-norm.

Cuts with uniform norms

If the norm in Ω is an ℓ_p-norm, we will refer to the problem as a cut with a *uniform norm*. Sinop and Grady [2007] define a cut cost

$$\Omega(x) = \|\boldsymbol{W}\boldsymbol{A}x\|_p, \tag{6.45}$$

where $\boldsymbol{W} \in \mathbb{R}^{\mathcal{E}\times\mathcal{E}}$ is a diagonal matrix of edge weights and $p \geq 1$. They minimize this function subject to some hard boundary conditions, i.e., the value of some x_i is fixed. If the ℓ_p-norm is the only term in the cost function, then one can instead minimize Ω^p, a separable function. In contrast, submodular norms do not in general offer such separability. Sinop and Grady [2007] point out that the uniform edge-norm with $p = 2$ corresponds to the random walk criterion used for image segmentation in [Grady, 2006]. The authors also address the ℓ_∞-norm, which penalizes the maximum $w(v_i, v_j)|x_i - x_j|$ for any edge $e = (v_i, v_j)$, and they show a solution using shortest paths. Relations between norms of gradients and flows or paths were also studied by Strang [1982]. Jimenez and Sra [2010] address efficient algorithms for minimizing terms $\|\boldsymbol{A}x\|_p$ for $p = 1$ and $p = 2$.

Watersheds

An alternative image segmentation approach has been motivated by watersheds [Maxwell, 1870, Jordan, 1872]. For *watershed cuts*, an edge weight is viewed as the height of that edge, and a watershed (cut) would be the "rim" in a corresponding topographical surface, that is, the lines from where water can flow down in at least two different directions. Cousty et al. [2009] show that watershed cuts can be obtained from a minimum spanning forest that leaves initially given seed points disconnected. The forest defines the partition via its connected components. The maximum spanning forest is equivalent to a minimum spanning forest for transformed edge weights w, which in turn corresponds to the solution of a minimum cut with edge weights w^p raised to a sufficiently high power p [Allène et al., 2009]. For discrete $x \in \{0,1\}^\mathcal{V}$, such a cut defines an edge-norm (taken to the power p). The cut cost is then

$$f(S) = \sum_{e\in S} w^\alpha(e). \tag{6.46}$$

As $\alpha \to \infty$, the problem becomes a minimum cut for the ℓ_∞ norm.

The discrete edge-ℓ_∞-norm is equivalent to a cooperative cut with the submodular cost function $f_{\max}(S) = \max_{e\in S} w(e)$. The cut obtained from a maximum spanning forest is indeed also a minimum cooperative cut for $f_{\max}$. The function $f_{\max}$ penalizes only the maximum edge in the cut, and therefore commonly admits many optimal cuts, several of which can be long. In other words, it strongly relaxes the smoothness induced by a sum of edge weights. Allène et al. [2009] too observe that high powers p may encourage longer cuts that remedy the shrinking bias, but that these cuts can be noisy. Lower powers in contrast bias towards smoothing. In Section 6.1.2 the complete loss of the smoothing effect is counteracted by restricting cooperation to edge groups and by using non-vanishing marginal costs, which correspond to lower powers.

Power watersheds [Couprie et al., 2010] combine the idea in [Sinop and Grady, 2007] with the work on watersheds in [Allène et al., 2009, Cousty et al., 2009], by allowing different powers on the weights and on $\boldsymbol{A}x$. If we view the weights raised to a (finite) power as fixed input, or if we add an integrality constraint on x, then power watersheds too relate to the norm framework.

Powers of norms

Several further examples exist in the literature where the norm $\|\cdot\|_p$ is replaced by its pth power. Strictly speaking, the resulting term is not a edge-norm, but sufficiently related. Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a common form is

$$\Omega^p(x) = \sum_{(v_i,v_j)\in\mathcal{E}} (w_{ij}|x_i - x_j|)^p. \tag{6.47}$$

A well-known example from machine learning is Laplacian regularization, where $p = 2$. The weights w_{ij} in (6.47) are then the square roots of the edge weights in the graph. Commonly, in semi-supervised learning, some labels x_i are given, and the others must be inferred. The regularizer leads to a labeling that varies smoothly along edges. Its name stems from the graph Laplacian $\boldsymbol{L}$, for which $\Omega^2(x) = x^\top \boldsymbol{L}x$. This form is also used for spectral clustering [von Luxburg, 2007]. Bühler and Hein [2009] motivate spectral clustering with the graph p-Laplacian, which corresponds to function (6.47) for arbitrary $p > 0$ (see also [Amghibech, 2003]). In particular, they address the case $1 < p < 2$. When expressing the p-Laplacian, the weights in (6.47) are $w_{ij} = \tilde{w}_{ij}^{1/p}$, where $\tilde{w}_{ij}$ are the edge weights in the graph.

Related is also the work by Cho et al. [2010], who use different powers of the norm of local gradients, and these powers vary across the image, adapting to the textures' gradient statistics. This is not the same as a non-uniform edge-norm, but it shares the idea that a non-uniform, adaptive "prior" can be beneficial.

Discrete total variation and related terms

For image denoising in 2D, Rudin et al. [1992] introduced a discretization of the continuous total variation. This discretization, with respect to the ℓ_2-norm, is defined in terms of labels x_{ij} of vertices v_{ij} in a grid with coordinates i, j:

$$\begin{aligned}
\mathrm{TV}_1(x) &= \sum\nolimits_{i,j} \|[(x_{i+1,j} - x_{i,j}),\ (x_{i,j+1} - x_{i,j})]^\top\|_2 &\quad (6.48)\\
&= \sum\nolimits_{i,j} \sqrt{(x_{i+1,j} - x_{ij})^2 + (x_{i,j+1} - x_{ij})^2} &\quad (6.49)\\
&= \sum\nolimits_{i,j} \|\boldsymbol{A}_{ij}x\|_2. &\quad (6.50)
\end{aligned}$$

The term (6.49) is a group norm, i.e., an ℓ_1-norm of the vector of ℓ_2-norms of the differences at a pixel. These differences can be written in terms of edges $(v_{i,j}, v_{i+1,j})$ and $(v_{i,j}, v_{i,j+1})$ in a grid-structured graph. The norm uses a group of edges $\mathcal{E}_{ij} = \{(v_{i,j}, v_{i+1,j}), (v_{i,j}, v_{i,j+1})\}$ for each node $v_{i,j}$. We write the norm in terms of submatrices $\boldsymbol{A}_{ij}$ of the node-edge incidence matrix $\boldsymbol{A}$ that refer to the groups $\mathcal{E}_{ij}$.

The discretization TV_1 easily generalizes to arbitrary graphs; for instance, Couprie et al. [2011] define the *combinatorial total variation*

$$\mathrm{TV}_2(x) = \sum\nolimits_i \nu_i \sqrt{\sum_{(v_i,v_j)\in\mathcal{E}} (x_i - x_j)^2} \tag{6.51}$$

$$= \sum\nolimits_i \sqrt{\sum_{(v_i,v_j)\in\mathcal{E}} \nu_i^2 (x_i - x_j)^2}. \tag{6.52}$$

The coefficients ν_i are inversely proportional to the gradient of the image at the particular point i. Relations of total variation and related terms have also been studied in [Strang, 1983, 2008].

Since group norms are also norms, all the discrete formulas above fall into the framework of edge-norms. The last term (6.52) has a Mahalanobis norm as the inner norm.

Chambolle and Darbon [2009] define as a discrete total variation $\Omega : \mathbb{R}^n \to \mathbb{R}_+$ a function that satisfies a discrete co-area formula, i.e., $\Omega(x) = \int_{-\infty}^{\infty} \Omega(\chi_\theta) d\theta$, where χ_θ is the characteristic vector of set of points with $x_i \geq z$. They mention the following examples that we re-write as cuts with group norm total variation. Here, x has two indices and is seen as a 2D image. The first example is an ℓ_1-edge-norm,

$$\mathrm{TV}_3(x) = \sum_{1\leq i,j\leq M} |x_{i+1,j} - x_{i,j}| + \sum_{1\leq i,j\leq M} |x_{i,j+1} - x_{i,j}| \tag{6.53}$$

$$= \|\boldsymbol{A}x\|_1. \tag{6.54}$$

Their second example is an "oscillation". Define the neighborhood of x_{ij} as $\mathcal{N}_{i,j} = \{x_{i,j}, x_{i+1,j}, x_{i,j+1}, x_{i+1,j+1}\}$. We denote by the sub-matrix $A_{\mathcal{N}_{i,j}}$ the edge-node incidence matrix for the complete subgraph over $\mathcal{N}_{i,j}$. The oscillation is an edge-norm for an $\ell_{1,\infty}$-norm:

$$\mathrm{TV}_4(x)$$

$$= \sum_{1\leq i,j\leq M} \max\{x_{i,j}, x_{i+1,j}, x_{i,j+1}, x_{i+1,j+1}\} - \min\{x_{i,j}, x_{i+1,j}, x_{i,j+1}, x_{i+1,j+1}\}$$

$$= \sum\nolimits_{i,j} \max_{y,z\in\mathcal{N}_{i,j}} |y - z| \tag{6.55}$$

$$= \sum\nolimits_{i,j} \|A_{\mathcal{N}_{i,j}} x\|_\infty. \tag{6.56}$$

For discrete variables x, this cost corresponds to a cooperative cut. The resulting TV_4, however, is still submodular. The reason is that the cooperating edges form a clique in which the edges are treated uniformly. In that respect, TV_4 is a continuous case of the submodular P^n functions by Kohli et al. [2009a] that we discuss in Section 5.3.2.

The final example is a pairwise circulant oscillation and involves cliques $\mathcal{N}_{i,j} = \{x_{i,j}, x_{i+1,j}, x_{i,j+1}\}$ over triples of nodes. It couples the edges of such cliques, and its discrete variant is again a cooperative cut:

$$\begin{aligned}\mathrm{TV}_5(x) &= \sum_{1 \leq i,j \leq M} \max\{|x_{i,j} - x_{i+1,j}|, |x_{i,j} - x_{i,j+1}|, |x_{i+1,j} - x_{i.j+1}|\} \\ &= \sum\nolimits_{i,j} \|A_{\mathcal{N}_{i,j}} x\|_\infty. \end{aligned} \tag{6.57}$$

6.3. Summary and discussion

This chapter explored cooperative cuts for implementing prior information or regularizing criteria. As a concrete example and application of cooperative cut energies, we defined a new criterion for image segmentation that reduces the segmentation error compared to the popular Graph Cut approach by up to 70%. The criterion implements a bias towards congruous object boundaries by using structured cooperations of edges. The resulting non-submodular energy function couples variables anywhere in the image. Still, the iterative algorithm yields visually improved solutions efficiently.

Section 6.2 further develops the viewpoint of regularization and advances the notion of *edge-norms*. Edge-norms unify a number of models from the literature, including the oft-encountered total variation, and connect them to the energy models discussed in Section 5.3. Combined with ideas from Section 6.1, they lead to new non-uniform continuous smoothness criteria. In addition, we proposed an optimization algorithm for the newly introduced cooperative edge-norms. Exploring the potential of cooperative edge-norms in further applications is a topic for future research.

Chapter 7.

Representation and Efficient Approximate Minimization of Submodular Functions

Polynomial-time algorithms for minimizing submodular functions have been the subject of research for many years. But to date, the known polynomial-time algorithms are computationally too expensive for large data sets. Therefore, research efforts comprise algorithms that efficiently minimize particular sub-families of submodular functions. However, not every submodular function falls into such a "nice" sub-family. Thus, this chapter addresses an efficient minimization algorithm that applies to any submodular function at the cost of a (controlled) approximation.

7.1. Submodular function minimization

For a long time, it remained an open question whether every submodular function could be minimized in polynomial time. This question was resolved by Grötschel et al. [1981] as an implication of the ellipsoid method and the solvability of the separation problem for a polymatroid polytope [Edmonds, 1970]. The resulting algorithm runs in $\widetilde{O}(n^5\tau+n^7)$ time ([McCormick, 2006], citing an email from L. Lovász). Here and in the sequel of this chapter, τ denotes the time to evaluate an oracle of f. A further milestone were the first combinatorial polynomial-time algorithms. Using results of Edmonds [1970], Cunningham [1985a] set up a network flow-like framework that led to a pseudo-polynomial algorithm. This framework set the foundation for subsequent polynomial-time algorithms. The first strongly polynomial algorithms were then devised by Schrijver [2000] and Iwata et al. [2001]. These combinatorial algorithms and their successors (surveyed in [McCormick, 2006]) run in polynomial time, but with high complexity: the currently fastest algorithm by Orlin [2009] still takes $O(n^6 + n^5\tau)$ time.

Those combinatorial algorithms commonly build on a linear program and results from Edmonds [1970] for finding a vector $\boldsymbol{x}$ in P_f that maximizes $\mathbf{1}_n \cdot \boldsymbol{x} = \boldsymbol{x}(\mathcal{V})$ subject to given upper bounds $x_i \leq b_i$. In the end, these algorithms either minimize a function $f'(S) + \boldsymbol{y}(\mathcal{V} - S)$ with respect to a point $\boldsymbol{y}$, derived via a reduction by Cunningham; or they find a point $\boldsymbol{x}$ that maximizes $\boldsymbol{x}^-(\mathcal{V}) = \sum_{i\in\mathcal{V}} \min\{x_i, 0\}$, building on the equality

$$\min_{S\subseteq\mathcal{V}} f(S) \;=\; \max\{\mathbf{1}\cdot\boldsymbol{x} \mid \boldsymbol{x}\in P_f, \boldsymbol{x}\geq 0\} = \max\{\boldsymbol{x}^-(\mathcal{V}) \mid \boldsymbol{x}\in B_f\}. \qquad (7.1)$$

If $\boldsymbol{x}^*$ is the solution of the last optimization, then the minimal optimal set is the set $S^* \subseteq \mathcal{V}$ of indices i with negative x_i^*, and the maximal optimal set is the set of indices i with nonnegative x_i^*.

Owing to the still substantial running times of the combinatorial algorithms, McCormick [2006] recommends first investigating whether alternative methods may be feasible. In particular in machine learning or computer vision, where data sets are often large, a limited complexity of optimization algorithms can be extremely beneficial.

7.1.1. The minimum norm point algorithm

One alternative to the combinatorial algorithms that is considered faster in practice is the minimum norm point or Fujishige-Wolfe algorithm [Fujishige et al., 2006, Fujishige and Isotani, 2011]. This algorithm finds the vector $\boldsymbol{x}^*$ with minimum ℓ_2-norm in the base polytope B_f. The set $\{i \mid x_i^* < 0\}$ indexed by negative entries in $\boldsymbol{x}^*$ is the minimal optimal solution, and the set $\{i \mid x_i^* \leq 0\}$ indexed by all nonnegative entries is the maximal optimal solution. To find $\boldsymbol{x}^*$, Fujishige et al. [2006] apply Wolfe's algorithm [Wolfe, 1976]. This algorithm merely requires that a linear optimization over the polytope is tractable, and such a problem is solved by the "greedy algorithm" [Edmonds, 1970].

The empirical results in [Fujishige and Isotani, 2011] show a running time between $O(n^3)$ and $O(n^{3.5})$, mostly on minimum cut problems (numbers from [McCormick, 2006]). However, it is not known whether this algorithm always runs in polynomial time. Indeed, Hui Lin observed that a simple modification to a graph cut function can increase the running time of the minimum norm algorithm to become impractical for large data sets [Jegelka et al., 2011].

The minimum norm point algorithm can become impractical

The difficult example originated from a data subset selection problem in machine learning and is a submodular function that is induced by a bipartite graph as in Figure 2.1. Recall that such functions arise from a bipartite graph $\mathcal{H} = (\mathcal{V}, \mathcal{U}, \mathcal{E})$ with left and right nodes $\mathcal{V}$ and $\mathcal{U}$, respectively, and a nondecreasing submodular weight function $g : 2^{\mathcal{U}} \to \mathbb{R}_+$. We define the *neighborhood* of a set $S \subseteq \mathcal{V}$ be $\mathcal{N}(S) = \{u \in \mathcal{U} : \exists \text{ edge } (v, u) \in \mathcal{E} \text{ with } v \in S\}$. Then $f : 2^{\mathcal{V}} \to \mathbb{R}_+$, defined as $f(S) = g(\mathcal{N}(S))$, is nondecreasing submodular (Proposition 2.1). In our example, the full function is the sum of a nonnegative modular function m and such induced submodular functions defined on subsets $\mathcal{U}_i \subseteq \mathcal{U}$,

$$f(S) = \lambda \sum\nolimits_i g_i(\mathcal{N}(S) \cap \mathcal{U}_i) + m(\mathcal{V} \setminus S). \tag{7.2}$$

The difficult family uses functions g that are a square root of a nonnegative modular function w. Figure 1.3 in the introduction shows that on average for (not the worst instance of) such a problem, the algorithm scales as n^4. Such complexities rule out large data sets that are involved in the subset selection application from which these functions arose.

For the experiment in the figure, 20 sub-graphs were sampled from a large graph for each n. The shown times are averages over those 20 repetitions for the cost function $f(S) = -m(S) + \lambda\sqrt{w(\mathcal{N}(S))}$. The large graph stems from a corpus subset extraction problem that is described next and has $|\mathcal{V}| = 54915$ and $|\mathcal{U}| = 6871$ nodes.

A motivating application

The proliferation of large real-world machine-learning data sets is both a blessing and a curse for machine learning algorithms. On the one hand, a large data set can accurately represent a complex task, thereby giving an algorithm the opportunity to live up to its potential. On the other hand, the size of such data sets presents its own problem: many algorithms have complexities that scale at a level that renders them impractical for all but small problem sizes.

Thus, Lin and Bilmes [2010, 2011] address the question of how to empirically evaluate new or expensive algorithms on large data sets without spending an inordinate amount of time doing so. We may wish to test various algorithms quickly, identifying the one that performs best. If a new idea ends up performing poorly, knowing this sooner rather than later will avoid futile work. Often the complexity of a training iteration is linear in the number of samples n but polynomial in the number c of classes or types. For example, for object recognition, it typically takes $O(c^k)$ time to segment an image into regions that each correspond to one of c objects, using an MRF with non-submodular k-interaction potential functions. In speech recognition, moreover, a k-gram language model with size-c vocabulary has a complexity of $O(c^k)$, where c is in the hundreds of thousands and k can be as large as six.

To reduce complexity one can reduce k, but this can be unsatisfactory since the nature and novelty of the algorithm might entail this very cost. An alternative is to extract and use a subset of the training data, one with a small number c of classes.

We would want any such subset to possess as much as possible of the richness and intricacy of the original data set while simultaneously ensuring that c is bounded. We also may wish to impose a bias in favor of certain classes, say, on those between which it is most difficult to discriminate. For example, in computer vision, we may wish to choose a large subset of a face recognition database that limits the number of different people but that includes people looking as similar to each other as possible. In speech recognition, we might desire a large number of utterances that, collectively, have a small confusable vocabulary [Lin and Bilmes, 2011].

One approach might be to choose a subset of classes of the appropriate size, and then choose all training samples that contain any of those classes. The resulting set of training samples will contain more than this set of classes, however. This would be problematic since, say in computer vision, much of the image would contain real objects that cannot be recognized, and, say, in speech recognition, it would mean that the resulting corpus has a large out-of-vocabulary set. Alternatively, one could choose only those training samples that have no more than the selected set of classes, but this then might result in an extremely small set of training samples. We thus desire an approach that allows us to, on the one hand, choose a bounded set of classes, but on the other hand, chooses a subset of training samples that is very large.

This problem can be solved via submodular function minimization (SFM) using the family of induced functions described above. Let $\mathcal{U}$ be the set of types contained in a set of training samples $\mathcal{V}$. Moreover, let the modular function w measure the cost of a type $u \in \mathcal{U}$ (this corresponds e.g. to the "undesirability" of type u). Define also a modular function $m : 2^{\mathcal{V}} \to \mathbb{R}_+$, $m(S) = \sum_{v \in S} m(v)$ as the benefit of training samples (e.g., in vision, $m(v)$ is the number of different objects in an image $v \in \mathcal{V}$, and in speech, this is the length of utterance v). Then the above optimization problem can be solved by finding $\operatorname{argmin}_{S \subseteq \mathcal{V}} w(\mathcal{N}(S)) - \lambda m(S) = \operatorname{argmin}_{S \subseteq \mathcal{V}} w(\mathcal{N}(S)) + \lambda m(\mathcal{V} \setminus S)$ where λ is a tradeoff coefficient. As shown below, this can be easily represented and solved efficiently via graph cuts. In some cases, however, we prefer to pick certain subclasses of $\mathcal{U}$ *together*. We partition $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$ into blocks, and make it beneficial to pick items from the same block. Such cooperation within blocks can arise from non-negative non-decreasing submodular functions $g : 2^{\mathcal{U}} \to \mathbb{R}_+$ restricted to blocks. The resulting optimization problem is $\min_{S \subseteq \mathcal{V}} \sum_i g(\mathcal{U}_i \cap \mathcal{N}(S)) + \lambda m(\mathcal{V} \setminus S)$; the sum over i expresses the obvious generalization to a partition into more than just two blocks. Unfortunately, it is not obvious how this class of functions can be efficiently represented by graph cuts, and we must resort to general SFM.

7.2. Representations for efficient minimization

To circumvent general SFM algorithms and even the minimum norm point algorithm, (provably) more efficient algorithms exist for subclasses of submodular functions. Two popular classes are functions that are minimizable as graph cuts, and functions that are scalar concave functions concatenated with modular functions. We survey both of these classes.

Other specific families that admit more efficient algorithms include symmetric submodular functions [Queyranne, 1998] and graphic matroids [Preissmann and Sebő, 2009].

7.2.1. Graph cuts

The minimum (s,t)-cut problem is equivalent to the minimization of a (submodular) quadratic pseudo-boolean polynomial that has one variable for each node and constants 1 for terminal node s and 0 for terminal t. The minimum (s,t)-cut induces a partition of the nodes $\mathcal{V}$ into the set X_s reachable from s after cutting and into its complement X_t. Analogously, in the optimal solution of the equivalent pseudo-boolean polynomial, the variables $x_i = 1$ indicate the nodes $X_s = X(\boldsymbol{x})$, and the variables $x_i = 0$ indicate the variables in X_t. This equivalence between the (pseudo-boolean or set) function representation and the graph cut representation has been exploited especially in computer vision [Boykov and Jolly, 2001, Greig et al., 1989], because more efficient algorithms exist for minimum cut than for general SFM.

Furthermore, this apparently useful equivalence can be extended by introducing auxiliary variables (and thus graph nodes) $\mathcal{U}$. Formally, to represent the submodular function $f : 2^{\mathcal{V}} \to \mathbb{R}$, one constructs a graph $\mathcal{G} = (\mathcal{V} \cup \mathcal{U} \cup \{ s,t\}, \mathcal{E})$ with nodes $\mathcal{V} \cup \mathcal{U}$ and edge weights $w : \mathcal{E} \to \mathbb{R}_+$ such that the "restricted" minimum cuts in $\mathcal{G}$ are equivalent to f. That means, to evaluate a set S, we enforce the case that we assign S to X_s and $\mathcal{V} \setminus S$ to X_t, and freely distribute the nodes in $\mathcal{U}$ over X_s and X_t such that the cut weight $w(\delta(s \cup X_s)) = \sum_{e \in \delta(s \cup X_s)} w(e)$ of the edges between X_s and X_t is minimized. This cut value is then equivalent, up to a constant, to $f(S)$. Formally, we define the class of *graph-representable* functions as follows:

Definition 7.1 ($\mathcal{F}_{gc}$)**.** A submodular function $f : 2^{\mathcal{V}} \to \mathbb{R}$ is *graph-representable* if there exists a graph $\mathcal{G} = (\mathcal{V} \cup \mathcal{U} \cup \{ s,t\}, \mathcal{E}, w)$ with nonnegative (modular) edge weights w and a constant γ such that for any $S \subseteq \mathcal{V}$, it holds that

$$f(S) + \gamma = \min_{U \subseteq \mathcal{U}} w(\delta_s(S \cup U)) = \min_{U \subseteq \mathcal{U}} \sum_{e \in \delta_s(S \cup U)} w(e). \tag{7.3}$$

We denote the family of all graph-representable functions by $\mathcal{F}_{gc}$.

Here and in the sequel, we use the shorthand $\delta_s(S) = \delta(s \cup S)$. With such a graph cut representation, any minimizer S^* of f corresponds to a minimum cut in $\mathcal{G}$. In the sequel, we will assume without loss of generality that f is normalized, and that it is shifted (if necessary) to correspond to the cut directly.

Proposition 7.1. *Let S^* be a minimizer of f, and let $U^* \in$ $\operatorname{argmin}_{U \subseteq \mathcal{U}} w(\delta_s(S^* \cup U))$. Then the boundary $\delta_s(S^* \cup U^*) \subseteq \mathcal{E}$ is a minimum cut in $\mathcal{G}$.*

Proof. Assume that the Proposition is not true and that there is a cut $\delta_s(T)$ around $T \subseteq \mathcal{V} \cup \mathcal{U}$ with $w(\delta_s(T)) < w(\delta_s(S^* \cup U^*))$. Let $T_{\mathcal{V}} = T \cap \mathcal{V}$. Then it holds that

$$f(T_{\mathcal{V}}) = \min_{U \subseteq \mathcal{U}} w(\delta_s(T_{\mathcal{V}} \cup U)) \leq w(\delta_s(T)) < w(\delta_s(S^* \cup U^*)) = f(S^*), \tag{7.4}$$

contradicting the optimality of S^*. □

Conversely, the proof also illustrates that any minimum cut $\delta_s(T^*)$ in $\mathcal{G}$ leads to a minimizer $S^* = T^* \cap \mathcal{V}$ of f.

The efficiency of minimum cut algorithms compared to general SFM raised the question which submodular functions can be represented in the form (7.3). This question has been studied not only in computer vision, but also in discrete optimization and with respect to constraint satisfaction (we detail references below). Several of these studies allow the set $\mathcal{U}$ to be exponentially larger than $\mathcal{V}$ if need be, and rather address the general representability question. In practice, it is desirable that $|\mathcal{U}|$ be at most linear or quadratic in $|\mathcal{V}|$, and similarly $|\mathcal{E}|$. An upper bound on the number of auxiliary variables needed for representing a representable function is the Dedekind number of the order of the function[1] [Ramalingam et al., 2011]. (This upper bound is still rather large, e.g., the Dedekind number of 5 is $M(5) = 7581$.)

Recall that set functions can equivalently be written as pseudo-boolean functions in terms of characteristic vectors. It is well-known that a pseudo-boolean polynomial $f(\boldsymbol{x}) = \sum_i a_i x_i + \sum_{i,j} a_{ij} x_i x_j$ is submodular if all coefficients a_{ij} are non-positive [Boros and Hammer, 2002]. This non-positivity is a necessary and sufficient criterion for quadratic functions to be graph-representable [Hansen and Simeone, 1979, Picard and Ratliff, 1975, Ivănescu (Hammer), 1965]. In fact, these quadratic polynomials define exactly the functions that can be represented without any auxiliary variables apart from s and t. Cunningham [1985b] proves this for an equivalent criterion: for any three disjoint sets $A, B, C \subseteq \mathcal{V}$, it must hold that

$$\begin{aligned} &f(A \cup B \cup C) \\ &\quad = f(A \cup B) + f(A \cup C) + f(B \cup C) - f(A) - f(B) - f(C) + f(\emptyset). \end{aligned} \tag{7.5}$$

Higher-order functions can be reduced to quadratic polynomials (and thus potentially also to graph cuts) by adding additional variables. A sufficient criterion for graph-representability is that in the representation as a polynomial the coefficients of all higher-order terms are non-positive [Freedman and Drineas, 2005]. In a similar fashion, Fujishige and Patkar [2001] study the representability of submodular functions as hypergraph cuts without additional nodes. In essence, both papers (and also [Kolmogorov and Zabih, 2004]) use the Möbius inversion formula for set functions:

$$f(S) = \sum_{T \subseteq S} f^{(|T|)}(T) \tag{7.6}$$

$$f^{(|T|)}(T) = \sum_{Z \subseteq T} (-1)^{|S-T|} f(Z). \tag{7.7}$$

A nonnegative function can be represented by a hypergraph cut if $f^{(i)} \leq 0$ for all $i \geq 2$ [Fujishige and Patkar, 2001] (the condition that $f(\emptyset) = f(\mathcal{V}) = 0$ can be overridden by introducing terminal nodes and asking for an (s,t)-cut). Equation (7.6)

[1]The Dedekind number $M(k)$ is the number of monotone boolean functions of k variables.

shows that the Möbius inversion formula yields the coefficients of a corresponding pseudo-boolean function. In the polynomial, all terms will be active whose variables denote elements of S, and each term will contribute its coefficient. Those coefficients are exactly the $f^{(|T|)}(T)$ counted in the sum. The viewpoint of polynomials was taken by Freedman and Drineas [2005].

However, the non-positivity of the $f^{(i)}$ (or higher order coefficients) is not a necessary condition, and subsequent work addressed further representations [Živný and Jeavons, 2010, Zalesky, 2003]. It is also known that cubic submodular polynomials [Billionet and Minoux, 1985], and binary valued functions [Cohen et al., 2005] are representable. Moreover, other types of representability have been studied [Ramalingam et al., 2008, Charpiat, 2011] that are not addressed here. Figure 7.2 shows examples of submodular functions represented by graph cuts.

Finally, Živný et al. [2009] show that there exist submodular functions that are not graph cut representable in the sense of Definition 7.1. For functions f over a ground set of size four, they state necessary and sufficient conditions to be graph-representable. Being graph-representable for f is equivalent to a particular function being a "multi-morphism" of f, or to f being in the cone generated by the family of functions called "fans". Not all submodular functions over a ground set of size 4 satisfy these conditions.

7.2.2. Concave functions

Another family of submodular functions that lends itself to more efficient algorithms is the one of concave functions applied after modular functions:

Definition 7.2 ($\mathcal{F}_{conc}$)**.** We denote by $\mathcal{F}_{conc}$ the class of all submodular functions $f : 2^{\mathcal{V}} \to \mathbb{R}$ that can be written as

$$f(S) = \sum_{j=1}^{k} \psi_j(w_j(S)) \tag{7.8}$$

for nonnegative modular functions w_j and concave scalar functions ψ_j.

This class is what Stobbe and Krause [2010] term *decomposable submodular functions.* A sub-class of $\mathcal{F}_{conc}$ are discounted price functions for which $f(S) = \psi(w(S))$ and ψ is concave and nondecreasing.

In addition to the work summarized below, an exact algorithm for integer-valued functions was given in [Kolmogorov, 2012]. The constructions in [Kohli et al., 2009b] apply to piecewise linear concave functions, but can require many additional edges and nodes.

Optimization by parametric maximum flow

If the function f consists of only one concave function and a modular function m, $f(S) = \psi(w(S)) + m(S)$, then, under certain conditions, f can be efficiently

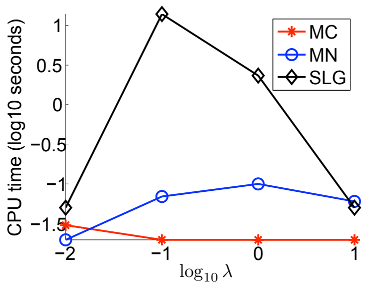


Figure 7.1 Running time[5] of MN, MC and SLG with varying λ on a graph with $|\mathcal{V}| = 300$, $|\mathcal{U}| = 636$, and with cut function $f(S) = -m(S) + \lambda w(\mathcal{N}(S))$.

minimized over a distributive lattice via a parametric maximum flow [Fujishige and Iwata, 1999]. The conditions are that either ψ is piecewise linear with a polynomial number of breakpoints, or that w is such that $w(S)$ only takes a polynomial number of values on the lattice, e.g., $w(i) = 1$ for all i. In the latter case, one can replace ψ by a piecewise linear function ϕ with $\phi(z) = \psi(z)$ for all values of $z = w(S)$, by linearly interpolating between the values z. This function ϕ has only a polynomial number of breakpoints.

The function ϕ is the minimum of a set of linear functions ϕ_k that each lead to a modular set function $\phi_k(w(\cdot))$. When replacing ϕ by any ϕ_k, the resulting function can be minimized as a graph cut. Fujishige and Iwata [1999] find the minimizer S_k for each of those functions, and from those select the overall minimizer, knowing that $S^* = \min_k \phi_k(S_k)$. The minimization for all the linear functions ϕ_k simultaneously can be computed as a parametric maximum flow.

Optimization by accelerated subgradient descent

Instead of representing f as the minimum of concave functions, Stobbe and Krause [2010] decompose concave functions as a sum of truncations of the form $g(S) = \min\{\sum_{v \in S} w(v), \gamma\}$. They show how such functions can be minimized by an accelerated gradient descent [Nesterov, 2004] minimizing the smoothed Lovász extension. This algorithm (termed SLG by its authors) has the benefit that it applies to *any* function in $\mathcal{F}_{conc}$.

However, experiments show that this algorithm too may not always be fast. Figure 7.1 compares the running time of SLG[2] to those of the minimum norm point algorithm (MN) and minimum cut[3] on a function induced by a bipartite graph. The reason for the extended running times is the computation of the gradient that requires finding gradients of $|\mathcal{U}|$ truncation functions.

[2]implementation in C++ by Hui Lin
[3]code by V. Kolmogorov [Boykov and Kolmogorov, 2004].
[5]This experiment was performed by Hui Lin.

Are all submodular functions in $\mathcal{F}_{conc}$?

As it is known that not all submodular functions can be represented as graph cuts, and as there appears to be a decomposition into efficiently optimizable base functions, the question arises whether all submodular functions can be represented by functions in $\mathcal{F}_{conc}$. The answer to this question is unfortunately negative:

Theorem 7.1. *The class $\mathcal{F}_{conc}$ of concave concatenated with modular functions is a strict subset of the class of all submodular functions.*

To show this result, we need another lemma.

Lemma 7.1. *Every submodular function that is a concatenation $f(S) = \psi(w(S))$ of a nondecreasing nonnegative modular function and a concave scalar function ψ can be represented as a graph cut (up to a constant).*

Before proving the lemma, we observe that it is sufficient to consider nondecreasing concave functions:

Proposition 7.2. *Every continuous concave function $\psi : [0, \theta] \to \mathbb{R}$ can be decomposed into a non-increasing linear function c and a nondecreasing concave function ϕ.*

Proposition 7.2 a continuous analogue to Theorem 18 in [Cunningham, 1983].

Proof (Proposition 7.2). If ψ is nondecreasing, we set $c \equiv 0$ and are done. Otherwise, we define a linear function $c(x) = \gamma x$ with $\gamma <\psi\ (x) - \psi(y) < 0$ for all $x \leq y \in [0, \theta]$. (If ψ is differentiable at θ, we can use $\gamma = \psi'(\theta) < 0$). Then

$$\psi(x) = c(x) + (\psi - c)(x); \tag{7.9}$$

this implies that $\phi(x) = (\psi - c)(x)$. Owing to the definition of c, the function ϕ is nondecreasing: it holds that $\phi(x + h) - \phi(x) = \psi(x + h) - \psi(x) - c(h) \geq 0$ for any $x \in [0, \theta]$, and $0 \leq h \leq \theta - x$. As ϕ is the difference between a concave and a linear function, it is also concave. □

Proof (Lemma 7.1). To prove Lemma 7.1, we construct a graph with terminal nodes s, t, and one node v_i for each element i in the ground set. We first use Proposition 7.2 to decompose f into a negative modular part, $m(S) = c(w(S))$, and a nondecreasing part $\phi(w(S))$. Without loss of generality, we can assume the submodular function f to be normalized. We represent the modular part (up to a constant, as will be described in Section 7.3.1) by introducing edges (s, v_i) with weight $-c(w(i))$.

We observe that the only critical points of ϕ are those where it is actually evaluated; those are all the values that $w(S)$ can take on $2^{\mathcal{V}}$, at most 2^n many. Therefore we replace ϕ with a piecewise linear function φ that has a break-point at each value that $w(S)$ takes; and $\varphi(z) = \phi(z)$ at each such break-point z.

(This function is merely the linear interpolation between the points $(z,\phi(z))$.) Note that $\phi(w(S)) = \varphi(w(S))$ for all $S \subseteq \mathcal{V}$. We number the breakpoints $0 = z_0 < z_1 < \ldots < z_\ell$, and define the slopes $m_j = \frac{\phi(z_j)-\phi(z_{j-1})}{z_j - z_{j-1}}$. Then φ can equivalently be written as

$$\phi(x) = \sum_{j, z_j < x} m_j(z_j - z_{j-1}) + m_{j(x)}(x - z_{j(x)-1}), \tag{7.10}$$

where $j(x)$ is such that $x \in (z_{j(x)-1}, z_{j(x)}]$. Using Equation (7.10), we re-write φ as a sum of truncations. Beginning with $c_\ell = m_\ell$, we define constants

$$c_k = m_k - \sum_{j=k+1}^{\ell} c_j \geq 0. \tag{7.11}$$

Then we can write

$$\begin{aligned}
\sum_{k=1}^{\ell} c_k \min\{x, z_k\} &= \sum_{j'<j(x)} c_{j'} z_{j'} + \sum_{j' \geq j(x)} c_{j'} x & (7.12)\\
&= \sum_{j'<j(x)} (z_{j'} - z_{j'-1}) \sum_{i=j'}^{\ell} c_i + \sum_{j' \geq j(x)} c_{j'}(x - z_{j(x)-1}) & (7.13)\\
&= \sum_{j'<j(x)} (z_{j'} - z_{j'-1}) m_{j'} + m_{j(x)}(x - z_{j(x)-1}) & (7.14)\\
&= \varphi(x). & (7.15)
\end{aligned}$$

We represent each function $c_k \min\{w(S), z_k\}$ by introducing an auxiliary node t_k and an edge (t_k, t) with weight $c_k z_k$ (as in Figure 7.2(c)). Moreover, we add edges (v_i, t_k) with weight $w(i)c_k$. Selecting S means to disconnect the corresponding nodes from t, that means, on each truncation, we either cut the edges to t_k with weight $c_k w(S)$, or the edge (t_k, t) with weight $c_k z_k$. □

With this lemma and known results on representation capacities of graph cuts, the theorem follows straightforwardly.

Proof (Theorem 7.1). By Lemma 7.1, all of the functions in $\mathcal{F}_{conc}$ can be represented as (potentially exponentially large) graph cuts. If all submodular functions were in $\mathcal{F}_{conc}$, then any submodular function would be graph-representable, and this contradicts the results by Živný et al. [2009] that some submodular functions cannot be represented by graphs as defined in Equation (7.3). □

Remark: Jan Vondrák has proved the same result as Theorem 7.1 by directly constructing a non-representable counterexample [Vondrák, 2010]. After proving Theorem 7.1, we found that Shapley [1971] also discusses this question for nondecreasing supermodular functions.

7.3. Representing submodular functions by cooperative cuts

The preceding sections indicate a gap in the optimization landscape for submodular functions. There exist sub-classes that can be minimized more efficiently, but these sub-classes are restricted. Thus, for certain functions, one still needs to revert to the general, computationally expensive algorithms for SFM. Furthermore, even the supposedly fast minimum norm point algorithm is not always practically applicable.

Motivated by this gap and the need for scalable algorithms, we extend the graph representation (7.3) to comprise all submodular functions. This extension is achieved by allowing submodular weights on some edges. The resulting cooperative cut problem is of course not NP-hard, but the new representation suits other algorithms, in particular the algorithms for MinCoopCut. These algorithms solve the problem *approximately* but potentially faster than generic exact algorithms for SFM. For the approximations, we additionally try to ensure that the number of auxiliary nodes does not grow too large.

The minimum norm point algorithm too allows to trade accuracy for speed. It can be accelerated by loosening its accuracy parameter. Unfortunately, this can sometimes result in very poor solutions, as shown by the experiments in Section 7.5. Hence, in the sequel we investigate whether using an approximation right from the beginning can offer a better tradeoff between accuracy and speed.

7.3.1. Basic construction

Unless the submodular function f is already a graph cut function and directly representable, we first decompose f into a modular function and a nondecreasing submodular function, and then build up the graph part by part. A sum of graph-representable functions can be expressed by joining the respective representing sub-graphs into one large graph.

We first introduce a relevant decomposition result by Cunningham [1983]. A polymatroid rank function is *totally normalized* if $f(\mathcal{V} \setminus i) = f(\mathcal{V})$ for all $i \in \mathcal{V}$.

Theorem 7.2 ([Cunningham, 1983, Thm. 18]). *Any submodular function f can be decomposed as $f(S) = m(S) + g(S)$ into a modular function m and a totally normalized polymatroid rank function g. The components are defined as $m(S) = \sum_{v\in S} \rho_f(v|\mathcal{V} \setminus v)$ and $g(S) = f(S) - m(S)$ for all $S \subseteq \mathcal{V}$.*

For unconstrained minimization, we may assume that $m(v) < 0$ for all $v \in \mathcal{V}$. For if $m(v) \geq 0$ for any $v \in \mathcal{V}$, then the property of diminishing marginal costs implies that we can discard element v immediately. By diminishing marginal costs, it holds that $f(S) - f(S \setminus v) \geq f(\mathcal{V}) - f(\mathcal{V} \setminus v) = m(v)$ for all S containing v. If $m(v) > 0$, then $f(S) \geq m(v) + f(S \setminus v)$, and excluding v from S never increases the value of an SFM solution.

To express the negative costs m in a graph cut, we point out an equivalent formulation with positive weights: since $m(\mathcal{V})$ is constant, minimizing $m(S) = \sum_{v \in S} m(v)$ is equivalent to minimizing the shifted function $m(S) - m(\mathcal{V}) = -m(\mathcal{V} \setminus S)$. Thus, we instead minimize the sum of positive weights on the complement of the solution. We implement this shifted function in the graph by adding an edge (s, v) with *nonnegative* weight $-m(v)$ for each $v \in \mathcal{V}$. Every element $v_j \in X_t$ (i.e., $v_j \notin S$) that is not selected must be separated from s, and the edge (s, v_j) contributes $-m(v_j)$ to the total cut cost. If f is modular, then $f = m$ and the optimal solution is the empty cut separating t from $\mathcal{V}$, i.e., the set of all elements with negative weights as shown in Figure 7.2(a).

Example constructions

Having constructed the modular part of the function f by edges (s, v) for all $v \in \mathcal{V}$, we address its submodular part g. If g is a sum of functions, we can add a sub-graph for each function. We begin with some example functions that are explicitly graph-representable with polynomially many auxiliary nodes $\mathcal{U}$. Figure 7.2 illustrates the construction of some such example graphs; it includes the modular part m as well. Graph-representable functions include the following examples.

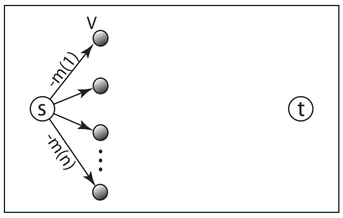
Maximum. The maximum function $g(S) = \max_{v \in S} w(v)$ charges the maximum weight of any element in S, for nonnegative weights w. To represent it, we introduce $n - 1$ auxiliary nodes u_j and connect them to form an imbalanced tree with leaves $\mathcal{V}$, as illustrated in Figure 7.2(b). The minimum-cost way to disconnect a set S from t is to cut the single edge (u_{j-1}, u_j) with weight $w(v_j)$ of the largest element $v_j = \operatorname{argmax}_{v \in S} w(v)$.

Truncations. Truncated functions $f(S) = \min\{w(S), \gamma\}$ for w,$\gamma \geq 0$ can be modeled by one extra variable, as shown in Figure 7.2(c). If $w(S) > \gamma$, then the minimization in the equivalence (7.3) puts u in X_s and cuts the γ-edge. This construction has been used successfully in computer vision [Kohli et al., 2009b]. Truncations can model piecewise linear concave functions of $w(S)$ [Kohli et al., 2009b, Stobbe and Krause, 2010], and also represent negative terms in a pseudo-boolean polynomial[6]. Furthermore, these functions include rank functions $g(S) = \min\{|S|, k\}$ of uniform matroids, and rank functions of partition matroids (Fig. 7.2(d)).

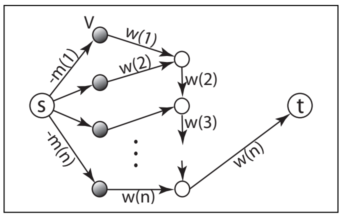
Bipartite neighborhoods. We already encountered bipartite submodular functions $f(S) = \sum_{u \in \mathcal{N}(S)} w(u)$ in Section 7.1.1. The bipartite graph that defines $\mathcal{N}(S)$ is part of the representation shown in Figure 7.2(i), and its edges get infinite weight. As a result, if $S \in X_s$, then all neighbors $\mathcal{N}(S)$ of S must also be in X_s, and the edges (u, t) for all $u \in \mathcal{N}(S)$ are cut. Each $u \in \mathcal{U}$ has such an edge (u, t), and the weight of that edge is the weight $w(u)$ of u.

Dual rank functions. A matroid with rank function $r : 2^{\mathcal{V}} \to \mathbb{R}_+$ has a dual matroid with rank function $r^*(S) = |S| + r(\mathcal{V} \setminus S) - r(\mathcal{V})$ [Schrijver, 2004]. If we can

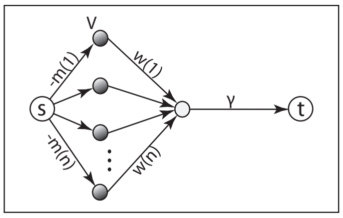
[6]To represent the term $-c \prod_{v_i \in Q} x_i$ for $Q \subseteq \mathcal{V}$, introduce edges (s, v_i) with weight $c > 0$ for all $v_i \in Q$, and an auxiliary node t' with edges (v_i, t') weighted c, and (t', t) weighted $\gamma = |Q|c - c$.



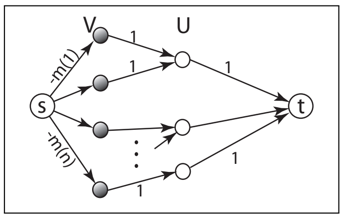
(a) $f = m$



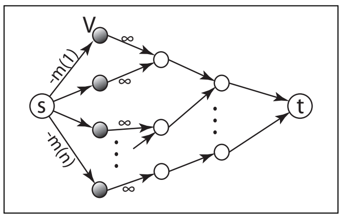
(b) maximum



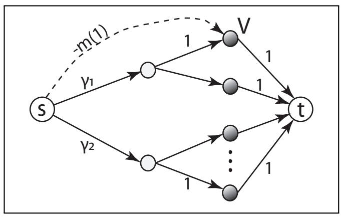
(c) truncation



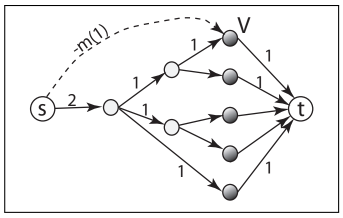
(d) partition matroid



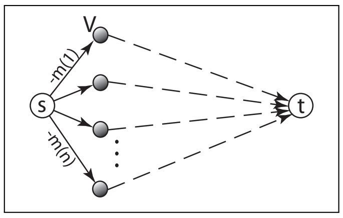
(e) bipartite & truncation



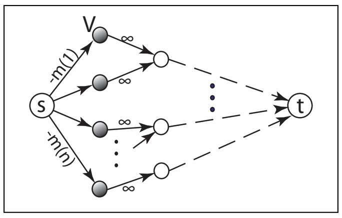
(f) dual partition matroid



(g) graphic matroid via its dual



(h) basic submodular construction



(i) bipartite

Figure 7.2. Example graph constructions. Dashed edges ending in t can have submodular weights; auxiliary nodes are white. The bipartite graph can have arbitrary representations between $\mathcal{U}$ and t, 7.2(e) is one example. The duals 7.2(f), 7.2(g) can also have edges (s, v) with weight $-m(v)$ that are not all shown.

represent r, then we can represent r^* too. Figure 7.2(f) shows the construction for the dual of a partition matroid, and Proposition 7.3 formalizes the construction.

Proposition 7.3. *Let $\mathcal{G}$ be a graph that represents the rank r of a matroid. Construct a graph $\mathcal{G}^*$ that has the same nodes as $\mathcal{G}$ and that, for each edge (v_i, v_j) in $\mathcal{G}$ with $v_i, v_j \in (\mathcal{V} \cup \mathcal{U}\ ) \setminus \{s, t\}$, has an inverted edge (v_j, v_i). Furthermore, for each edge (s, v) in $\mathcal{G}$, there is an edge (v, t) in $\mathcal{G}^*$, and for each edge (v, t) in $\mathcal{G}$, there is an edge (s, v) in $\mathcal{G}^*$. Then we add an edge (v_j, t) to $\mathcal{G}^*$ with $w(v_j, t) = 1$ for each $v_j \in \mathcal{V}$. The resulting graph $\mathcal{G}^*$ represents the rank of the dual matroid: $\min_{U \subseteq \mathcal{U}} w(\delta(s \cup S \cup U)) = r^*(S) + const$ for all $S \subseteq \mathcal{V}$.*

Proof. First, one can check that $\mathcal{G}^*$ without the added weights (v_j, t) represents the function $r'(S) = r(\mathcal{V} \setminus S)$. With those added edges, a cut contains an edge (v_j, t) for each element $j \in S$; these edges contribute $|S|$ to the cost. In sum, it then holds that $\min_{U \subseteq \mathcal{U}} w(\delta(s \cup S \cup U)) = |S| + r(\mathcal{V} \setminus S) = r^*(S) + r(\mathcal{V})$, and $r(\mathcal{V})$ is constant. □

Sometimes the representation via the dual matroid can be simpler than a direct representation. Figure 7.2(g) shows the representation of a graphic matroid as the dual of its dual. Here, the rank $r(S)$ is the size of the largest subset of S that does not contain a cycle in the defining graph; the elements are edges in that defining graph. For the dual, $r^*(S) = |S|$ if $|S| \leq 2$ and $S \neq \{1,4\}, \{2,3\}$. For any other set, $r^*(S) = 2$. We represent r^* via two truncations and place it between s and $\mathcal{V}$. It can be checked that the resulting graph represents r. However, not all graphic matroids might be representable in this fashion.

All the above constructions can also be applied to subsets $Q \subset \mathcal{V}$ of nodes. In fact, the decomposition and constructions above permit us to address arbitrary sums and restrictions of the represented functions. These example families of functions shown here already cover a wide variety of functions occurring in applications.

7.3.2. Submodular edge weights

Next we address the generic case of a submodular function that is not (efficiently) graph-representable or whose functional form is unknown. We can still decompose this function into a modular part m and a polymatroid g. Then we construct a simple graph as shown in Figure 7.2(h). The representation of m via edges (s, v) is the same as above, but the polymatroid rank function g will be represented by edges with non-additive weights. Each edge (v, t) is associated with exactly one ground set element $v \in \mathcal{V}$, and selecting v (i.e., assigning v to X_s) is equivalent to cutting the edge (v, t). Thus, the cost of edge (v, t) will model the cost $g(v)$ of its element $v \in \mathcal{V}$. Let $\mathcal{E}_t$ be the set of edges (v, t) (i.e., edges adjacent to t), and denote, for any subset $C \subseteq \mathcal{E}_t$ the set of ground set elements adjacent to C by $V(C) = \{v \in \mathcal{V} \,|\, (v, t) \in C\}$. Equivalently, C is the boundary of $V(C)$ in $\mathcal{E}_t$: $\delta_s(V(C)) \cap \mathcal{E}_t = C$. We define the cost of C to be the cost of its adjacent ground set elements, $h_g(C) \triangleq g(V(C))$; this implies $h_g(\delta_s(S) \cap \mathcal{E}_t) = g(S)$. The equivalent of Equation (7.3) becomes

$$f(S) = \min_{U \subseteq \mathcal{U}} w(\delta_s(S \cup U) \setminus \mathcal{E}_t) + h_g(\delta_s(S \cup U) \cap \mathcal{E}_t) = -m(\mathcal{V} \setminus S) + g(S), \quad (7.16)$$

with $\mathcal{U} = \emptyset$ in Figure 7.2(h). This generalization from the standard sum of edge weights to a nondecreasing submodular function permits us to express many more functions, in fact *any* submodular function.

The simple construction in Figure 7.2(h) itself corresponds to a general submodular function minimization. It becomes powerful when combined with parts of f that are explicitly representable. If g decomposes into a sum of graph-representable functions and a (nondecreasing submodular) remainder g_r, then we construct a sub-graph for each graph-representable function, and combine these sub-graphs with the submodular-edge construction for g_r. All the sub-graphs share the same ground set nodes $\mathcal{V}$. In addition, we are in no way restricted to separating graph-representable and general submodular functions. The cost function in our application is a submodular function induced by a bipartite graph $\mathcal{H} = (\mathcal{V}, \mathcal{U}, \mathcal{E})$. Let, as before, $\mathcal{N}(S)$ be the neighborhood of $S \subseteq \mathcal{V}$ in $\mathcal{U}$. Given a nondecreasing submodular function $g_{\mathcal{U}} : 2^{\mathcal{U}} \to \mathbb{R}_+$ on $\mathcal{U}$, the graph $\mathcal{H}$ defines a function $g(S) = g_{\mathcal{U}}(\mathcal{N}(S))$.

For any such function, we represent $\mathcal{H}$ explicitly in $\mathcal{G}$, and then add submodular-cost edges from $\mathcal{U}$ to t with $h_g(\delta_s(\mathcal{N}(S))) = g_{\mathcal{U}}(\mathcal{N}(S))$, as shown in Figure 7.2(i). If $g_{\mathcal{U}}$ is itself exactly representable, then we add the appropriate sub-graph instead (Figure 7.2(e)).

7.4. Approximate optimization

The graph representation helps minimize a given submodular function f as a cooperative cut. Any of the approximation algorithms from Chapter 4 applies; we henceforth build on the efficient iterative algorithm ITB. This algorithm is exact if the edge costs are modular (instead of submodular), that is, if the function f is efficiently graph-representable.

The algorithm we will describe approximates f in each iteration by a submodular function $\hat{f}$ that is efficiently graph-representable, and minimizes $\hat{f}$ instead. In this section, we switch from costs $f, \hat{f}$ of node sets S, T to costs w, h of edge sets A, B, C and back.

The approximation $\hat{f}$ arises from the cut representation constructed in Section 7.3.2. Recall that the representing graph $\mathcal{G}$ has two types of edges: those whose weights w are counted as the usual sum, and those charged via a submodular function h_g derived from g. We denote the latter set by $\mathcal{E}_t$, and the former by $\mathcal{E}_m$. Following the idea of the iterative, adjusting upper bounds, we approximate the edge weights in $\mathcal{G}$ by modular weights ν. For any $e \in \mathcal{E}_m$, we use the exact cost $\nu(e) = w(e)$. The submodular cost h_g of the remaining edges is upper bounded by referring to a fixed set $B \subseteq \mathcal{E}$ that we specify later. For any $A \subseteq \mathcal{E}_t$, we define

$$\hat{h}_B(A) \triangleq h_g(B) + \sum_{e \in A \setminus B} \rho_h(e|B \cap \mathcal{E}_t) - \sum_{e \in B \setminus A} \rho_h(e|\mathcal{E}_t \setminus e) \geq h_g(A). \tag{7.17}$$

This inequality holds thanks to diminishing marginal costs, and the approximation is tight at B, i.e., $\hat{h}_B(B) = h_g(B)$. The proof of Lemma 4.5 shows that, up to a constant shift, this function is equivalent to the edge weights

$$\nu_B(e) = \rho_h(e|B \cap \mathcal{E}_t) \quad \text{if } e \in \mathcal{E}_t \setminus B; \qquad \text{and} \qquad \nu_B(e) = \rho_h(e|\mathcal{E}_t \setminus e) \quad \text{if } e \in B \cap \mathcal{E}_t. \tag{7.18}$$

Replacing the edge weights w and h_g by ν_B in the representation equivalence (7.16) yields an approximation $\hat{f}$ of f. In Algorithm 6, B is always the boundary $B = \delta_s(T)$ of a set of nodes $T \subseteq (\mathcal{V} \cup \mathcal{U})$. Then $\mathcal{G}$ with weights ν_B represents the function

$$\hat{f}(S) = \min_{U \subseteq \mathcal{U}} \; \nu_B(\delta_s(S \cup U) \cap \mathcal{E}_m) + \nu_B(\delta_s(S \cup U) \cap \mathcal{E}_t) \tag{7.19}$$

$$= \min_{U \subseteq \mathcal{U}} \; w(\delta_s(S \cup U) \cap \mathcal{E}_m) + \sum_{(u,t) \in \delta_s(S \cup U) \cap B} \rho_g(u|\mathcal{V} \cup \mathcal{U} \setminus u) \tag{7.20}$$

Algorithm 6 Minimizing graph-based approximations.

```
construct the representation graph $\mathcal{G} = (\mathcal{V} \cup \mathcal{U} \cup \{ s, t\}, \mathcal{E})$
set $S_0 = T_0 = \emptyset$
for $i = 1, 2, \dots$ do
    compute edge weights $\nu_{i-1} = \nu_{\delta_s(T_{i-1})}$ (Equation 7.18)
    find the (maximal) minimum $(s,t)$-cut $T_i = \text{argmin}_{T \subseteq (\mathcal{V} \cup \mathcal{U})} \nu_{i-1}(\delta_s T)$
    if $f(T_i) = f(T_{i-1})$ then
        return $S_i = T_i \cap \mathcal{V}$
    end if
end for
```

$$+ \sum_{(u,t) \in \delta_s(S \cup U) \setminus B} \rho_g(u|T). \tag{7.21}$$

Here, we used the definition $h_g(C) \triangleq g(V(C))$. Importantly, the edge weights ν_B are always nonnegative, because, by Theorem 7.2, g is guaranteed to be nondecreasing. Hence, we can efficiently minimize $\hat{f}$ as a standard minimum cut. Lemma 7.2 states properties of the T_i.

Lemma 7.2. *Assume $\mathcal{G}$ is any of the graphs in Figure 7.2, and let $T^* \subseteq \mathcal{V} \cup \mathcal{U}$ be the maximal set defining a minimum-cost cut $\delta_s(T^*)$ in $\mathcal{G}$, so that $S^* = T^* \cap \mathcal{V}$ is a minimizer of the function represented by $\mathcal{G}$. Then, in any iteration i of Algorithm 6, it holds that $T_{i-1} \subseteq T_i \subseteq T^*$. In particular, $S \subseteq S^*$ for the returned solution S.*

Lemma 7.2 has three important implications. First, the algorithm never picks any element outside the maximal optimal solution. Second, because the T_i are growing, there are at most $|T^*| \leq |\mathcal{V} \cup \mathcal{U}|$ iterations, and the algorithm is strongly polynomial. Finally, the chain property $T_{i-1} \subseteq T_i$ permits more efficient implementations. The proof of Lemma 7.2 relies on the definition of ν and on submodularity.

Proof. We prove the result for the case of a bipartite graph in Figure 7.2(i) with node sets $\mathcal{V}, \mathcal{U}$, and cost $f(A) = m(A) + g(\mathcal{N}(A))$ for $A \subseteq \mathcal{V}$. The proof carries over to the other submodular-weight graphs by identifying $\mathcal{V}$ with the right hand side of the bipartite graph. For a graph with completely modular edge weights, $\hat{f} = f$, and T_1 is an optimal solution.

Let $T_i \subseteq \mathcal{V} \cup \mathcal{U}$ be the selected nodes at iteration i, i.e., the nodes reachable from s in the min-cut. We first observe that the T_i form a chain, i.e., $T_i \subseteq T_{i+1}$ for all i. The reason lies in the adaptive edge weights. Let ν_{i-1} be the weights based on T_{i-1} by which T_i was chosen. Assume there was a nonempty set $Q = T_i \setminus T_{i+1}$, with $Q_{\mathcal{V}} = Q \cap \mathcal{V}$, $Q_{\mathcal{U}} = Q \cap \mathcal{U}$. Let $\delta^+ Q$ be the outgoing edges from Q to t that are cut if Q is selected, and let $\delta^- Q$ be the incoming edges from s to Q that are cut if Q is not selected. Since Q is a subset of the optimal T_i, it must hold that

$$\begin{aligned} 0 &\geq \nu_{i-1}(\delta_s T_i) - \nu_{i-1}(\delta_s(T_i \setminus Q)) && (7.22)\\ &= \nu_{i-1}(\delta^+ T_i) + \nu_{i-1}(\delta^-(\mathcal{V} \setminus T_i)) - (\nu_{i-1}(\delta^+(T_i \setminus Q_{\mathcal{U}}))) \\ &\quad - \nu_{i-1}(\delta^-(\mathcal{V} \setminus (T_i \setminus Q_{\mathcal{V}}))) \\ &= \nu_{i-1}(\delta^+ Q_{\mathcal{U}}) - \nu_{i-1}(\delta^- Q_{\mathcal{V}}). && (7.23) \end{aligned}$$

In the next iteration $i+1$, the weight for any $(u,t) \in \delta^+ Q_{\mathcal{U}}$ cannot increase, because $\nu_i(u,t) = g(\mathcal{U}) - g(\mathcal{U} \setminus u) \leq \nu_{i-1}(u,t)$ by the property of diminishing marginal costs. Together with Inequality (7.23), this implies that $\nu_i(\delta^+ Q_{\mathcal{U}}) \leq \nu_{i-1}(\delta^+ Q_{\mathcal{U}}) \leq \nu_{i-1}(\delta^- Q_{\mathcal{V}}) = -m(Q_{\mathcal{V}}) = \nu_i(\delta^- Q_{\mathcal{V}})$. Thus, cutting $\delta^+ Q_{\mathcal{U}}$ instead of $\delta^- Q_{\mathcal{V}}$ and including Q in T_{i+1} can never increase the cost: as in Equations (7.22) to (7.23), it holds that

$$\nu_{i-1}(\delta_s(T_{i+1} \cup Q)) - \nu_{i-1}(\delta_s T_{i+1}) = \nu_i(\delta^+ Q_{\mathcal{U}}) - \nu_i(\delta^- Q_{\mathcal{V}}) \leq 0. \qquad (7.24)$$

This contradicts the maximality and optimality of T_{i+1}, thus, Q must be empty and therefore $T_i \subseteq T_{i+1}$.

Next, we will see that all the T_i are subsets of T^*. Initially, the solution is $T_0 = \emptyset$, so clearly $T_0 \subseteq T^*$. Assume that i is the first iteration where $T_i \setminus T^* \neq \emptyset$, and let $Q = T_i \setminus T^*$. As before, we define $Q_{\mathcal{U}} = Q \cap \mathcal{U}$ and $Q_{\mathcal{V}} = Q \cap \mathcal{V}$. We will show that $g(Q_{\mathcal{U}} \cup T^*_{\mathcal{U}}) - m(\mathcal{V} \setminus (Q_{\mathcal{V}} \cup T^*_{\mathcal{V}})) \leq g(T^*_{\mathcal{U}}) - m(\mathcal{V} \setminus T^*_{\mathcal{V}})$, and then T^* cannot be the maximal optimal solution, a contradiction. By diminishing marginal costs, it holds that

$$\begin{aligned} f(S^* \cup Q_{\mathcal{V}}) - f(S^*) &= g(Q_{\mathcal{U}} \cup T^*_{\mathcal{U}}) - m(\mathcal{V} \setminus (Q_{\mathcal{V}} \cup T^*_{\mathcal{V}})) - g(T^*_{\mathcal{U}}) + m(\mathcal{V} \setminus T^*_{\mathcal{V}}) \\ &= \rho_g(Q_{\mathcal{U}} | T^*_{\mathcal{U}}) + m(Q) && (7.25)\\ &\leq \rho_g(Q_{\mathcal{U}} | T_{i-1} \cap \mathcal{U}) + m(Q), && (7.26) \end{aligned}$$

since $T_{i-1} \subseteq T^*$ by assumption, and thus $Q \cap T_{i-1} = \emptyset$. Note that $\rho_g(Q_{\mathcal{U}} | T_{i-1} \cap \mathcal{U}) \leq \sum_{u \in Q_{\mathcal{U}}} \rho_g(u | T_{i-1} \cap \mathcal{U})$: numbering the elements in $u_k \in Q_{\mathcal{U}}$ arbitrarily shows that

$$\rho_g(Q_{\mathcal{U}} | T_{i-1} \cap \mathcal{U}) = \sum_k \rho_g(u_k | (T_{i-1} \cap \mathcal{U}) \cup u_1 \cup \ldots \cup u_{k-1}) \leq \sum_k \rho_g(u_k | T_{i-1} \cap \mathcal{U})$$

by diminishing marginal costs. Thus, by the definition of ν_{i-1},

$$\begin{aligned} g(Q_{\mathcal{U}} \cup T^*_{\mathcal{U}}) - m(\mathcal{V} \setminus (Q_{\mathcal{V}} \cup T^*_{\mathcal{V}}) - g(T^*_{\mathcal{U}}) + m(\mathcal{V} \setminus (T^*_{\mathcal{V}})) && (7.27)\\ \leq \sum_{u \in Q_{\mathcal{U}}} \rho_g(u | T_{i-1} \cap \mathcal{U}) + m(Q) && (7.28)\\ = \sum_{e \in \delta^+ Q_{\mathcal{U}} \setminus \delta^+ T_{i-1}} \rho_h(e | \delta^+ T_{i-1}) + m(Q_{\mathcal{V}}) && (7.29)\\ = \nu_{i-1}(\delta^+ Q_{\mathcal{U}}) - \nu_{i-1}(\delta^- Q_{\mathcal{V}}) && (7.30)\\ = \nu_{i-1}(\delta_s T_i) - \nu_{i-1}(\delta_s(T_i \setminus Q)) && (7.31)\\ \leq 0. && (7.32) \end{aligned}$$

The last part follows like Equations (7.22) to (7.23) and the optimality of T_i. Hence, including Q in T^* would be optimal, and contradicts the maximality of T^*. Thus, Q must be empty. Finally, if $T_i \subseteq T^*$, then $S_i = (T_i \cap \mathcal{V}) \subseteq (T^* \cap \mathcal{V}) = S^*$. □

Moreover, Lemma 4.6 holds here as well and implies an approximation bound[7]:

Corollary 7.1. *Let $S^* \subseteq \mathcal{V}$, and $T^* \subseteq \mathcal{V} \cup \mathcal{U}$ be defined as in Lemma 7.2. For the solution S returned by Algorithm 6, it holds that*

$$f(S) \leq \frac{|\delta_s T^*|}{1 + (|\delta_s T^*| - 1)\beta(T^*)} f(S^*) \quad \leq \quad |\delta_s T^*| f(S^*), \tag{7.33}$$

where $\beta(T^*) = \min_{u \in T^*} \rho_g(u|\mathcal{U} \setminus u) / \max_{u \in T^*} g(u)$.

7.4.1. Improvement via summarizations

The approximation $\hat{f}$ is loosest if the sum of edge weights $\nu_i(A)$ significantly overestimates the true joint cost $h_g(A)$ of sets of edges $A \subseteq \delta_s T^* \setminus \delta_s T_i$ that are still to be cut. This happens if the joint marginal cost $\rho_h(A|\delta_s T_i)$ is much smaller than the estimated sum of weights $\nu_i(A) = \sum_{e \in A} \rho_h(e|\delta_s T_i)$. Luckily, many of the functions we are interested in that show this behavior strongly resemble truncations. Thus, to tighten the approximation, we summarize the joint cost of groups of edges by a construction similar to Figure 7.2(c). Then the algorithm can take larger steps and pick groups of elements.

We partition the set $\mathcal{E}_t$ of submodular-weight edges into disjoint groups G_k of edges (u, t). For each group, we introduce an auxiliary node t_k and re-connect all edges $(u, t) \in G_k$ to end in t_k instead of t. Their cost remains the same. An extra edge e_k connects t_k to t, and carries the joint weight $\nu_i(e_k)$ of all edges in G_k. This is a tighter approximation than using marginal costs of single edges. Like the previously described weights $\nu_i(e)$, the weight $\nu_i(e_k)$ is also adapted in each iteration. Initially, we set $\nu_0(e_k) = h_g(G_k) = g(V(G_k))$. Subsequent approximations ν_i refer to cuts $\delta_s T_i$, and such a cut can contain either single edges from G_k or the group edge e_k. We set the next reference set B_i to be a copy of $\delta_s T_i$ in which each group edge e_k was replaced by all its group members G_k. The joint group weight $\nu_i(e_k)$ for any k is then $\nu_i(e_k) = \rho_h(G_k \setminus B_i|B_i) + \sum_{e \in G_k \cap B_i} \rho_h(e|\mathcal{E}_t \setminus e) \leq \sum_{e \in G_k} \nu_i(e)$. Formally, these weights represent the tighter upper bound

$$\begin{aligned} &h_g(A) \leq \\ &h_g(B) + \sum_{G_k \subseteq A} \rho_h(G_k \setminus B|B) + \sum_{\substack{e \in (G_k \cap A) \setminus B, \\ G_k \not\subseteq A}} \rho_h(e|B) - \sum_{e \in B \setminus A} \rho_h(e|\mathcal{E}_t \setminus e) \leq \hat{h}_B(A), \end{aligned}$$

[7]The approximation bound holds as stated in Corollary 7.1 if f is equivalent to the graph cuts without any constant shift.

where we replace G_k by e_k whenever $G_k \subseteq A$. In the experiments in Section 7.5, this summarization helps improve the results while simultaneously reducing the running time.

7.4.2. Parametric constructions for special cases

For certain functions of the form $f(S) = m(S) + g(\mathcal{N}(S))$, the graph representation in Figure 7.2(i) admits a specific exact algorithm. This algorithm uses approximations that are exact on limited ranges, and eventually picks the best range. A sufficient condition for the method to apply is that g has the form $g(U) = \psi(\sum_{u \in U} \tilde{w}(u))$ for weights $\tilde{w} \geq 0$ and one piecewise linear, concave function ψ with a small (polynomial) number ℓ of breakpoints. Alternatively, ψ can be *any* concave function if the weights $\tilde{w}$ are such that $\tilde{w}(U) = \sum_{u \in U} \tilde{w}(u)$ can take at most polynomially many distinct values x_k. For example, if $\tilde{w}(u) = 1$ for all u, then ψ gets evaluated at only $|\mathcal{U}| + 1$ points, and we can construct an equivalent piecewise linear concave function with $\ell = |\mathcal{U}| + 1$ by using the x_k as breakpoints and interpolating. In all these cases, ψ is equivalent to the minimum of at most ℓ linear (modular) functions.

We build on the approach in [Fujishige and Iwata, 1999], but, whereas their functions are defined on the ground set $\mathcal{V}$, the function g here is defined on the right hand side $\mathcal{U}$ of a bipartite graph. Contrary to their functions and owing to our decomposition, the function ψ here is nondecreasing. We define ℓ linear functions, one for each break-point x_k (and use $x_0 = 0$):

$$\psi_k(t) = (\psi(x_k) - \psi(x_{k-1}))(t - x_k) + \psi(x_k) = \alpha_k t + \beta_k. \tag{7.34}$$

The ψ_k are defined such that $\psi(t) = \min_k \psi_k(t)$. Therefore, we approximate f by a series $\hat{f}_k(S) = -m(\mathcal{V} \setminus S) + \psi_k(\tilde{w}(\mathcal{N}(S)))$, and find the exact minimizer S_k for each k. To compute S_k via a minimum cut in $\mathcal{G}$ (Fig. 7.2(i)), we define edge weights $\nu_k(e) = w(e)$ for edges $e \notin \mathcal{E}_t$ as in Section 7.4, and $\nu_k(u,t) = \alpha_k \tilde{w}(u)$ for $e \in \mathcal{E}_t$. Then $T_k = S_k \cup \mathcal{N}(S_k)$ defines a minimum cut $\delta_s T_k$ in $\mathcal{G}$ with respect to weights ν_k. We compute the full cost $\hat{f}_k(S_k) = \nu_k(\delta_s T_k) + \beta_k + m(\mathcal{V})$ for each S_k; the optimal solution is the S_k with minimum cost $\hat{f}_k(S_k)$. This method is exact. To solve for all k within one maximum flow, we use a parametric max-flow method [Gallo et al., 1989, Hochbaum, 2008]. Parametric max-flow usually works with edges both from s and to t. Here, the changing weights ν_k are nonnegative because ψ is nondecreasing, and therefore we only need t-edges which already exist in the bipartite graph $\mathcal{G}$.

This method is limited to few breakpoints. For more general concave ψ and arbitrary $\tilde{w} \geq 0$, we can *approximate* ψ by a piecewise linear function. Still, the parametric approach does not directly generalize to more than one nonlinearity, e.g., to $g(U) = \sum_i g_i(U \cap W_i)$ for sets $W_i \subseteq \mathcal{U}$. In contrast, Algorithm 6 (with the summarization) can approximate all of these cases.

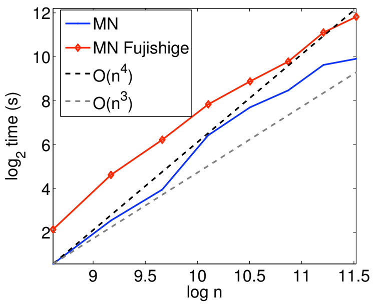


Figure 7.3 Comparison of our implementation for the minimum norm point algorithm with that by S. Fujishige on graph cut functions (experiment run and implemented by Hui Lin). The graphs were generated by GENRMF, and problem size refers to the number of nodes. The tolerance was set to 10^{-10}.

We point out that without indirection via the bipartite graph, that means if $f(S) = m(S) + \psi(w(S))$ for a function ψ with few breakpoints, we can minimize f very simply: The solution for ψ_k includes all $j \in \mathcal{V}$ with $\alpha_k \leq -m(j)/w(j)$. The advantage of the graph cut is that it easily combines with other objectives.

7.5. Experiments

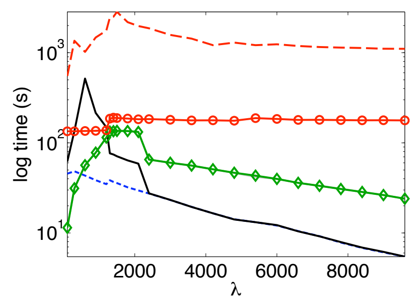
Having derived algorithms for approximately minimizing submodular functions as cooperative cuts, we empirically compare those algorithms to generic exact methods such as the minimum norm point algorithm. We compare the following methods:
MN: a re-implementation of the minimum norm point algorithm in C++ that is about four times faster than the C code used in [Fujishige and Isotani, 2011], Figure 7.3 compares both implementations and shows that our results are not caused by a slow implementation;
MC: a minimum cut with static edge weights $\nu(e) = h_g(e)$;
GI: the graph-based iterative Algorithm 6, implemented in C++ with the max-flow code by Boykov and Kolmogorov [2004], (i) by itself; (ii) with summarization via $\sqrt{|\mathcal{E}_t|}$ random groups (GIr); (iii) with summarization via groups generated by sorting the edges in $\mathcal{E}_t$ by their weights $h_g(e)$, and then forming groups G_k of edges adjacent in the order such that for each $e \in G_k$, $h_g(e) \leq 1.1 h_g(G_k)$ (GIs);
GP: the parametric method from Section 7.4.2, using $|\mathcal{E}_t|$ equispaced breakpoints; based on C code from RIOT[8].

As the SLG method was expensive on the type of functions used here (Figure 7.1), it is excluded from the experiments in the sequel.

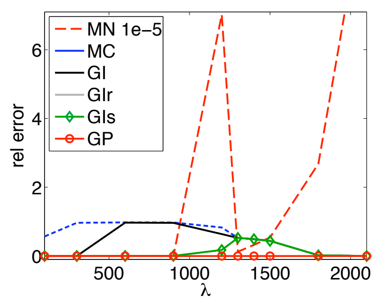
Solution quality with solution size

The running time and accuracy depend on the size of the optimal solution S^*. Therefore we test problem instances with varying solution sizes. We use an example from the the corpus subset extraction problem described in Section 7.1.1 with a speech data set [Godfrey et al., 1992]. The cost function is based on a bipartite

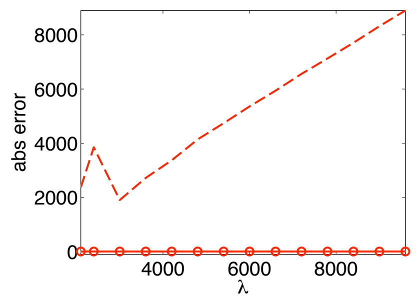
[8]`http://riot.ieor.berkeley.edu/riot/Applications/Pseudoflow/parametric.html`



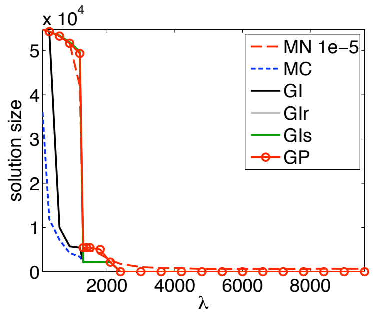
(a)



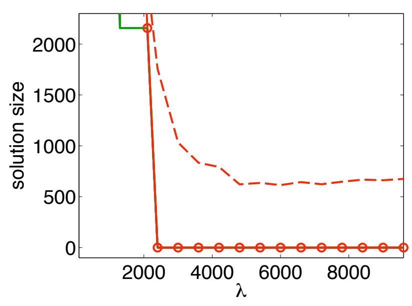
(b)



(c)



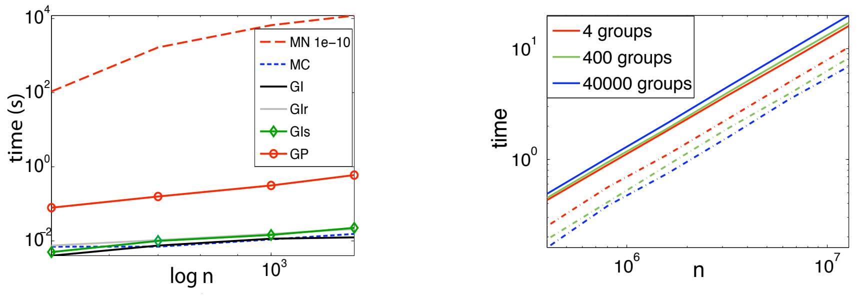
(d)



(e)

Figure 7.4. (a) Running time, (b) relative and (c) absolute error and (d) solution sizes with varying λ for a data set as described in Section 7.1.1, $|\mathcal{V}| = 54915$, $|\mathcal{U}| = 6871$, and $f(S) = -m(S) + \lambda\sqrt{|\mathcal{N}(S)|}$. Where $f(S^*) = 0$, we show absolute errors. (e) is a zoom into (d) that shows that the minimum norm point algorithm returns solutions of size 500 and more where all other methods find the optimal (empty) set.

graph and has the form $f(S) = -m(S) + \lambda\sqrt{w(\mathcal{N}(S))}$. The bipartite graph has $|\mathcal{V}| = 54915$ and $|\mathcal{U}| = 6871$ nodes, and uniform weights $w(u) = 1$ for all $u \in U$. The results look similar with non-uniform weights, but for uniform weights the parametric method from Section 7.4.2 always finds the optimal solution and thus enables us to report errors. The size of the optimal solution depends on the tradeoff parameter λ. We vary λ from 50 ($S^* \approx \mathcal{V}$) to 9600 ($S^* = \emptyset$). Figure 7.4 shows



(a) Bipartite graph (b) Iwata's test function

Figure 7.5. Running times with respect to $|\mathcal{V}|$. (a) Running times for $f(S) = -m(S) + \lambda\sqrt{w(\mathcal{N}(S))}$. Apart from MC, all solution qualities are similar. (b) Groups in Satoru Iwata's test function. The solid lines are the running times of the minimum norm point algorithm, the dashed lines of Algorithm 6 (GI). The steepness of the graphs across these two figures is not comparable, as the scale of the x axis is different.

the running times and the relative error $\mathrm{err}(S) = |f(S) - f(S^*)|/|f(S^*)|$ (note that $f(S^*) \leq 0$). If $f(S^*) = 0$, we report absolute errors. The running times were recorded on a machine with CPU 3.8GHz. Because of the large graph, we used the minimum-norm algorithm with accuracy 10^{-5}. Still, it takes up to 100 times longer than the other methods. It works well if S^* is large, but as λ grows, its accuracy becomes poor. In particular when $f(S^*) = f(\emptyset) = 0$, it returns large sets with large positive cost. In contrast, the deviation of the approximate edge weights ν_i from the true cost is bounded. All algorithms except MN return an optimal solution for $\lambda \geq 2000$. Updating the weights ν clearly improves the performance of Algorithm 6, as does the summarization (GIr/GIs perform identically here). With the latter, the solutions are very often optimal, and almost always very good.

Scaling

To test how the methods scale with the size $|\mathcal{V}|$ of the ground set, we sample small graphs from the big speech corpus graph, and report average running times across 20 graphs for each size. As the graphs have non-uniform weights, we use GP as an approximation method and estimate the nonlinearity $\sqrt{w(U)}$ by a piecewise linear function with $|\mathcal{U}|$ breakpoints. All algorithms find the same (optimal) solution. Figure 7.5 shows that the minimum-norm algorithm with high accuracy is much slower than the other methods. Empirically, MN scales as up to $O(n^4)$ or $O(n^5)$, the parametric version approximately as $O(n^2)$, and the variants of GI as up to $O(n^{1.5})$. Figure 1.3 in the introduction shows results on a larger range for a very similar data set.

Groups of Iwata's test function

We also test running time and performance for a function that is supposed to be "ideal" for the minimum-norm algorithm [McCormick, 2006]. For this function, proposed by Satoru Iwata, the elements are numbered $j = 1, \dots, n \in \mathcal{V}$. Then $f_I(S) = |S||\mathcal{V} \setminus S| - \sum_{j \in S}(5j - 2n)$ [Fujishige and Isotani, 2011]. We modify this function as follows: randomly assign the elements in $\mathcal{V}$ to m groups Q_i. The members of each Q_i are now numbered 1 to $|Q_i|$, and we apply f_I group-wise: $f(S) = \sum_{i=1}^{m} f_I(S \cap Q_i)$. Here, we implement the simplest version (Figure 7.2(h)) of Algorithm 6, so that we actually do not need the graph explicitly, and for each element only compare whether $\nu(s,j) = -m(j) > \nu(j,t)$. Since the solutions form a chain, we never need to test any selected element again. Figure 7.5(b) shows that this implementation is even slightly faster than the minimum-norm algorithm, and scales similarly. In particular, the minimum-norm algorithm becomes slower as there are more groups, whereas Algorithm 6 (GI) becomes faster. For Algorithm 6, smaller groups imply fewer iterations. Moreover, it always finds the optimal solution.

7.6. Summary and discussion

Submodular functions can be minimized in polynomial time, but currently known algorithms may become practically infeasible for large data. Therefore in this chapter we advance the idea of designing approximation algorithms even for polynomial-time solvable problems. We designed an algorithm that builds on the function representation discussed in Chapter 5. Contrary to standard graph cuts, cooperative cuts extend this formulation to cover all submodular functions. We therefore minimized submodular functions as cooperative cuts.

We proved that any element selected by the algorithm is a member of the maximal optimal solution. The basic algorithm works well if the interactions determining the minimum are detectable by looking at small groups of elements. In particular, it finds an optimal solution S^* if the elements in S^* can be ordered such that $\rho_g(e_i|\{e_1, \dots, e_{i-1}\}) < -m(e_i)$ for all $e_i \in S^*$. To handle the case that discounts can only be detected at larger scale, we introduce another layer of groupings. The experiments show that this modification improves the results.

We tested the algorithm especially on a combination of two types of cost functions that occur frequently: neighborhoods in bipartite graphs and functions from $\mathcal{F}_{conc}$. The minimum norm point algorithm, a common method of choice, converges slowly on the problem, and the SLG method requires too many terms to model the bipartite graph. On the speech corpus data, the proposed algorithm runs up to 1.5 orders of magnitude faster than the minimum norm point algorithm. While the minimum norm algorithm can output very inaccurate solutions, the empirical approximation factors for the new algorithm were never larger than two.

Specifically for the corpus selection problem, we additionally proposed an algorithm that uses parametric maximum flow. This algorithm returns the optimal solution for a sub-class of problems and works well in practice, but it does not apply to direct sums of cost functions on $\mathcal{U}$. The iterative algorithm does not have any such restrictions.

Chapter 8.

Structured Online Decision Problems with Submodular Losses

There exist settings where we do not only wish to solve a problem once, but instead to solve the same problem repeatedly, while the unknown cost function changes over time. Sequential Decision Problems cover this setting and implement the assumption that the cost function is only revealed after we have chosen a solution. This online setting becomes more challenging if in each round, a combinatorial *structure* must be chosen as the solution, e.g., a path or spanning tree [Kalai and Vempala, 2005, Kakade et al., 2009, Koolen et al., 2010]. Most algorithms for sequential decision problems involving combinatorial decision spaces address only linear (modular) cost functions, or otherwise only very simple constraints. This chapter poses the question whether one can build on the algorithms developed in Chapter 4 to derive Hannan-consistent algorithms for combinatorial sequential problems with submodular cost functions.

8.1. Introduction

The typical setting of sequential decision problems proceeds in rounds $t = 1, \dots, T$. In each round, the decision maker must decide upon a solution from a decision space $\mathcal{S}$. Here, we address discrete spaces, and the solution S_t must be a subset of a given fixed ground set $\mathcal{E}$. In the continuous setting, one picks a vector x_t from a feasible set. We consider the *full information* setting, where at iteration t, we have observed the cost functions up to step $t-1$ (i.e., we can query previous costs), but not the current cost f_t. After the player has picked a solution S_t, the cost function f_t is revealed, and he incurs the loss $f_t(S_t)$.

The problem of selecting S_t becomes harder when the decision space is combinatorial, that means in each round we must pick a set of elements having particular structure, such as a spanning tree or a cut. This combinatorial problem has also been termed "learning structured concept classes" [Koolen et al., 2010]. In most other work, the cost function is assumed to be modular and separates over the single elements. This separability helps cope with the combinatorial explosion of the space $\mathcal{S}$. It yields, for instance, a compressed representation for algorithms that maintain weights for each possible structure in $\mathcal{S}$.

In the sequel, we address the case where $\mathcal{S}$ is combinatorial, and, in addition, the cost functions f_t can be nondecreasing submodular functions that are not separable. The algorithms and bounds below extend known results for linear (modular) costs to a wider range of problems. Section 2.5 lists examples where the restriction to linearity fails to capture the cost in real-world situations. The offline correspondent of our problem is

$$\min f_t(S) \quad \text{subject to } S \in \mathcal{S}; \tag{8.1}$$

this is what we would solve if we knew f_t beforehand. The results in Chapter 3 show the hardness even of the offline problem where the cost function is known, and thereby imply that apart from the combinatorial decision space and the non-linear cost function, a third complication of our setting is that we can only use approximations.

Regret

The crucial question to be solved is how to choose the solution S_t as well as possible. We wish to pick solutions that are competitive to the best solution in hindsight. This competitiveness is measured by the commonly used (external) *regret*,

$$R(T) = \frac{1}{T}\left(\sum_{t=1}^{T} f_t(S_t) - \min_{S\in\mathcal{S}} \sum_{t=1}^{T} f_t(S)\right). \tag{8.2}$$

An algorithm is *Hannan-consistent* if its regret vanishes, $R(T) \to 0$, as $T \to \infty$. Regret is commonly used for problems where the minimization $\min_{S\in\mathcal{S}} f(S)$ for a known cost f can be solved exactly.

However, we simultaneously aim at an algorithm that does not take exponential time to select the next solution S_t. If the minimization problem $\min_{S\in\mathcal{S}} f(S)$ is NP-hard even when knowing f, then we cannot expect the online algorithm to choose an arbitrarily good S_t in polynomial time. Therefore, in this case one commonly aims for *online approximation algorithms* and instead compares to the best result that is achievable in polynomial time. This is measured by the α-regret that includes the approximation factor α achievable for the corresponding offline problem:

$$R_\alpha(T) = \frac{1}{T}\left(\sum_{t=1}^{T} f_t(S_t) - \alpha \min_{S\in\mathcal{S}} \sum_{t=1}^{T} f_t(S)\right). \tag{8.3}$$

In the sequel we address sequential decision problems that involve combinatorial problems with submodular costs. Many of them are NP-hard, as we saw in Chapter 3, and therefore we devise online approximation algorithms that minimize the respective α-regret.

8.1.1. Related work

Most existing online and bandit[1] algorithms for combinatorial problems expect a modular (linear) cost function [Awerbuch and Kleinberg, 2004, Kalai and Vempala, 2005, Balcan and Blum, 2007, Dani et al., 2008, Abernethy et al., 2008, Koolen et al., 2010, Cesa-Bianchi and Lugosi, 2009, Helmbold and Warmuth, 2009, Audibert et al., 2011]. Many exploit the *separability* of this function to handle the exponential number of choices, e.g., when maintaining weights. Submodular functions are not separable in this way. One example for non-separable costs with multi-task constraints is the work by Lugosi et al. [2009]. Their dynamic programming approach, however, works only for limited constraint sets that keep the state graph small.

When allowing arbitrary sets and removing the combinatorial constraints of choosing a structure, that is, $\mathcal{S} = 2^{\mathcal{E}}$, then we end up with an unconstrained submodular minimization problem, which is not NP-hard. Hazan and Kale [2009] derive online algorithms for unconstrained online submodular minimization and show regret bounds of $O(m/\sqrt{T})$ for m elements. We partially build on their techniques and, as a corollary, tighten their bound to $O(\sqrt{m/T})$.

Contrary to the problems in most of the work above, most instances of submodular minimization over combinatorial structures are NP-hard. Integrating approximations into existing online algorithms can be challenging, and there is no generic solution [Kakade et al., 2009, Kalai and Vempala, 2005]. Kalai and Vempala [2005] extend the regret bound for the *Follow-the-perturbed leader* (FPL) algorithm to NP-hard combinatorial problems with a *modular* cost function if there is an algorithm that provides a coordinate-wise approximation to the optimal solution. This approximation, however, does not apply here. Moreover, Kakade et al. [2009] show an example where FPL fails when directly used with the greedy set cover algorithm, and ask how to use FPL in general with approximations. In Section 8.3, we integrate a class of approximation algorithms for Problem (8.1) into the FPL framework.

Kakade et al. [2009] show how to derive online approximation algorithms from offline approximation algorithms for linear-cost problems, generalizing online gradient descent [Zinkevich, 2003] by approximate projections. Their approach is very general and considers any offline approximation algorithm as a black box. It takes a step in a descent direction and then uses the offline algorithm to project back onto the (scaled) feasible set; these are the approximate projections. The authors too consider only a certain family of cost functions and pose the case of nonlinear costs as an open problem. Their cost function is of the form $c : 2^{\mathcal{E}} \times \mathbb{R}^d \to \mathbb{R}$, $c(S, w) = \langle \phi(S), w \rangle$ and must be linear in w. That means it is the dot product between some feature vector of S and a weight vector. To use this framework, we must express any nondecreasing submodular function f via a cost vector w^f as

[1]In the *bandit setting* we never observe the true cost function, only its evaluation $f_t(S_t)$ after choosing solution S_t.

$c(S, w^f) = f(S)$. One possible encoding is via exponential-size feature vectors that have one entry for each possible set. Simple linear algebra shows that a full basis is needed to represent all such f meaning that w has an exponential dimension d. But then a straightforward use of the regret bound by Kakade et al. [2009] leads to a bound that is exponential in $|\mathcal{E}|$, since their bound is linear in $\|w\|$, i.e., growing with the dimension as $\sqrt{d}$. The exponential bound is of course due to the representation, but we do not know of any compact non-exponential representations of this form that are powerful enough to represent all submodular functions. The non-constant lower bound for learning submodular functions [Goemans et al., 2009] rather suggests that there is probably no simple polynomial-size representation of all submodular functions in terms of linear combinations of features. Furthermore, the algorithm also assumes that, given any $w \in \mathbb{R}^d$, we can project it in polynomial time onto the set of those w for which $c(\cdot, w)$ is a nondecreasing submodular function. Given the results by Seshadri and Vondrák [2010] and the hardness of recognizing even a quartic submodular posiform [Gallo and Simeone, 1988], this too seems to be non-trivial, and thus we use a different approach.

For an online version of approximate submodular maximization, greedy methods exist [Streeter and Golovin, 2008] that however satisfy constraints in expectation only. Shalev-Shwartz and Singh [2011] use the multilinear extension [Calinescu et al., 2011] for an algorithm close to online gradient descent [Zinkevich, 2003] and then apply pipage rounding. The solutions S_t satisfy the given matroid constraints. Other rounding techniques for other constraints [Chekuri et al., 2010] probably apply equivalently. Adaptive or interactive submodularity [Golovin and Krause, 2010, Guillory and Bilmes, 2011] also implies greedy algorithms, but considers a setting different from ours.

8.1.2. Three types of algorithms

We build on the offline approximation algorithms from Chapter 4 to tackle submodular cost functions in combinatorial online problems.

First, we show two generic Hannan-consistent algorithms for two main approximation strategies, one based on subgradient descent (Section 8.2), and one based on a Follow-the-leader scheme (Section 8.3). Table 8.1 shows regret bounds with details plugged in for various problems. As a corollary, our Theorem 8.1 tightens Theorem 1 in [Hazan and Kale, 2009] for unconstrained online submodular minimization. While the first two parts address general submodular functions, the third part focuses on a special class, namely label costs (LC). This class admits better approximation factors if class-specific algorithms are used. We reformulate LC problems as label selection problems with cover-type constraints, and derive an online algorithm that can use *any* offline algorithm for the LC problem at hand. Beyond standard LC, our formulation extends to multiple labels and simple discounts.

	subgradient descent (Sec. 8.2)	FPL (Sec. 8.3)	label costs (Sec. 8.4)
set cover	$O(\underline{k}\sqrt{m/T})$	–	$O(\underline{\ln\lvert\mathcal{U}\rvert}\sqrt{\lvert\mathcal{L}\rvert/T})$
vertex cover	$O(\underline{2}\sqrt{m/T})$	–	$O(\underline{\ln\lvert\mathcal{E}\rvert}\sqrt{\lvert\mathcal{L}\rvert/T})$
(s,t)-cut	$O(\underline{n}\sqrt{m/T})$	$O(\underline{n}m/\sqrt{T})$	$O(\sqrt{\underline{m}\lvert\mathcal{L}\rvert/T})$
spanning tree	–	$O(\underline{n}m/\sqrt{T})$	$O(\underline{\ln n}\sqrt{\lvert\mathcal{L}\rvert/T})$
perfect matching	–	$O(\underline{n}m/\sqrt{T})$	$O(\underline{\lvert\mathcal{L}\rvert}\sqrt{\lvert\mathcal{L}\rvert/T})^*$
monotone MSCA	$O(\underline{\log m}\sqrt{m/T})$	–	–
submodular MP	$O(\underline{2}\sqrt{m/T})$	–	–

Table 8.1. Overview of the regret bounds derived in this chapter, when applied to a range of problems. The approximation factor α is underlined, k is the maximum frequency, $\mathcal{U}$ the universe to cover. In a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $n = |V|$ is the number of nodes and $m = |\mathcal{E}|$ the number of edges; for set cover, m is the number of sets. MSCA is the minimum submodular-cost allocation problem [Chekuri and Ene, 2011b], which subsumes e.g. submodular-cost facility location. MP stands for "multiway partition". The last two lines refer to randomized rounding methods and the bounds are in expectation. *The label cost result hold for perfect matching in complete bipartite graphs.

Many approximation algorithms relate an inherently difficult problem to an easier one, and for the first two algorithms we build on exactly this relation. We categorize the approaches as in Section 4.1: they simplify either (i) the constraints or (ii) the cost function. When simplifying the constraints, we consider in particular relaxations. The respective algorithms treat f as a pseudo-boolean function on indicator vectors, relax the feasible set $\mathcal{S}$ to its convex hull, and finally round the solution of the relaxed problem. The relaxation is a convex non-smooth minimization problem with linear constraints. Such a problem is naturally amenable to an online, possibly exponentiated, subgradient descent. Algorithms motivated by (ii) replace f by a tractable approximation $\hat{f}$, and minimize $\hat{f}$ over $\mathcal{S}$. We use this $\hat{f}$ in a specific way in the Follow-the-leader framework, and show example functions that fit our framework. The generic $\hat{f}$ by Goemans et al. [2009] does not fit Algorithm 8, but we present a modification that does.

Most of the approximation algorithms surveyed in Chapter 4 fall into one of the two categories. For any of those, we can use the generic algorithms in Sections 8.2 and 8.3, respectively.

For ease of reading, we denote vectors $x \in \mathbb{R}^m$ by non-bold lowercase characters. As in previous chapters, m is the size of the ground set.

Algorithm 7 Rounded subgradient descent

```
**Input:** $\eta > 0$, initial $x_1 \in \mathcal{K}$
**for** $t = 1$ **to** $T$ **do**
    get $S_t$ from $x_t$ by rounding with factor $\alpha$
    obtain $f_t$
    compute $g_t = \mathrm{argmax}_{g \in P_{f_t}} g \cdot x_t$ and
    $x_{t+1} = \Pi_{\mathcal{K}}(x_t - \eta g_t)$
**end for**
```

8.2. Relaxations

We begin with an algorithm that operates on a relaxation and then rounds the continuous solution in each iteration. The rounding procedure determines the approximation factor. Suitable rounding procedures exist for covering constraints [Iwata and Nagano, 2009], cuts (Section 4.3.2), the monotone MINIMUM SUBMODULAR-COST ALLOCATION problem [Chekuri and Ene, 2011b] (with the rounding method by Kleinberg and Tardos [1999]), and SUBMODULAR MULTIWAY PARTITION with Half-rounding [Chekuri and Ene, 2011a]. The resulting approximation factors form the underlined parts of the factors in Table 8.1.

Let $\mathcal{K} \subseteq [0,1]^{\mathcal{E}}$ be the convex hull of the decision space $\mathcal{S} \subseteq \{0,1\}^{\mathcal{E}}$; both are described by the same linear inequalities. The cost function on $\mathcal{K}$ corresponding to f_t on $\mathcal{S}$ is the convex Lovász extension $\tilde{f}_t$. Algorithm 7 maintains two variables: it performs a subgradient descent based on [Zinkevich, 2003] in continuous space that yields x_t, and then rounds x_t to S_t. In each round t, it takes a step into the direction of the negative subgradient $-g_t$ of $\tilde{f}_t$ and projects back onto $\mathcal{K}$. The strongly convex projection $\Pi_{\mathcal{K}}(y) = \mathrm{argmin}_{x \in \mathcal{K}} \|x - y\|^2$ is in general easier to solve than a minimization of the full non-smooth relaxation. As x_t is rounded to S_t before the actual cost function f_t is revealed, the rounding procedure must not explicitly depend on f_t. That is, given a continuous vector x, the procedure must return a solution S with $f(S) \leq \alpha \tilde{f}(x)$ for any submodular function f that can occur. All of the above-mentioned rounding procedures satisfy this constraint.

Theorem 8.1. *When using a rounding scheme with approximation factor α, constants $\eta = \sqrt{m}(M\sqrt{T})^{-1}$ and $M = \max_{t, A \subseteq \mathcal{E}} \beta_t |f_t(A)|$ with $\beta_t = 1$ if f_t is nondecreasing and $\beta_t = 3$ otherwise, then the α-regret of Algorithm 7 is bounded as $R_\alpha(T) \leq \alpha M \sqrt{m/T} = O\big(\alpha\sqrt{m/T}\big)$.*

As a corollary, Theorem 8.1 tightens a bound by Hazan and Kale [2009, Thm. 7] for unconstrained online submodular minimization (using $\alpha = 1$, $\mathcal{S} = \{0,1\}^{\mathcal{E}}$ and their thresholded rounding).

Corollary 8.1. *The regret for online submodular minimization with Algorithm 7 is bounded by $O(\sqrt{m/T})$.*

Crucial for the improvement is a bound on the ℓ_2-norm of the subgradient. We assume everywhere that all functions f_t are normalized.

Lemma 8.1. *Let g_t be a subgradient of f_t (obtained by the greedy algorithm). Then $\|g_t\| \leq \beta \max_{A \subseteq \mathcal{E}} |f_t(A)|$, where $\beta = 1$ if f_t is nondecreasing, and $\beta = 3$ otherwise.*

Proof (Lemma 8.1). Since t is fixed, we drop the subscript in this proof. Essential for the proof is that $g \in P_f$, in fact, it lies in the base polytope (Section 2.3.1, [Fujishige, 2005, Lemma 6.19]). This means that

$$g \cdot \chi_A \leq f(A) \tag{8.4}$$

for all $A \subseteq \mathcal{E}$. Assume first that f is nonnegative and nondecreasing. Then Equation (8.4) immediately leads to a bound on $\|g\|$, by bounding the ℓ_2 norm by the ℓ_1 norm:

$$\|g\|_2 \leq \|g\|_1 = g \cdot \chi_{\mathcal{E}} \leq f(\mathcal{E}). \tag{8.5}$$

This proves the lemma for nondecreasing functions.

For arbitrary submodular functions, we use the construction of g in slightly more detail, but the basic arguments are the same. For ease of notation, let $\gamma = \max_{A \subseteq \mathcal{E}} |f(A)|$. We first recall how g was constructed, given $x \geq 0$. We denote the components of x by x_i, $1 \leq i \leq m$. We find a permutation π such that $x_{\pi(1)} \geq x_{\pi(2)} \geq \ldots \geq x_{\pi(m)}$. This ordering induces a maximal chain of sets, $\emptyset = A_0 \subset A_1 \subset \ldots \subset A_m$ with $A_0 = \emptyset$ and $A_i = A_{i-1} \cup \{e_{\pi(i)}\}$. Setting

$$g_{\pi(i)} = f(A_i) - f(A_{i-1}) \tag{8.6}$$

yields the subgradient g, with $g \cdot \chi_{A_i} = f(A_i)$. Let $g^+ = \max\{g, 0\}$ denote the element-wise maximum of g and 0.

Claim 8.1. $\|g^+\|_1 = \sum_{i=1}^M g_i^+ \leq \gamma$.

Consider the subset $\mathcal{E}^+$ of elements e_k with $g_k \geq 0$, and let B_j be the set of the j first such elements, where we use the ordering π induced by x, restricted to $\mathcal{E}^+$. We call this restriction π^+: $\{1, \ldots |\mathcal{E}^+|\} \to \{ 1, \ldots, |\mathcal{E}|\}$. The jth element in $\mathcal{E}^+$, $e_{\pi^+(j)}$, also occurs at some point $\pi(i(j))$ in the full sequence, so that $e_{\pi^+(j)} = e_{\pi(i(j))}$. Since the nonnegative elements are a subsequence, we know that $i(j) \geq j$ and thus $B_j \subseteq A_{i(j)}$. By the definition of g_j and diminishing marginal costs (submodularity), it holds for all $e_{\pi^+(j)} \in \mathcal{E}^+$ that

$$g_{\pi^+(j)} = f(A_{i(j)-1} \cup \{e_{\pi^+(j)}\}) - f(A_{i(j)-1}) \tag{8.7}$$
$$\leq f(B_{j-1} \cup \{e_{\pi^+(j)}\}) - f(B_j) =: g'_j. \tag{8.8}$$

The definition of the g_j^+ implies that $\sum_{j=1}^k g_j^+ = f(B_k) \leq \gamma$. In consequence,

$$\|g^+\|_1 = \sum_{j=1}^{|\mathcal{E}^+|} g_{\pi^+(j)} \leq \sum_{j=1}^{|\mathcal{E}^+|} g'_j \leq \gamma. \tag{8.9}$$

This proves the claim.

We next use this result to bound the sum of the absolute values of the negative entries. By the definition of g, we know that $\sum_{i=1}^k g_i = f(A_k)$, in particular also for $k = m$. Since $-\gamma \leq f(A_k) \leq \gamma$ for any k ($1 \leq k \leq m$), it follows that

$$-\gamma \leq \sum_{i:g_i<0} g_i + \sum_{i:g_i\geq 0} g_i \leq \gamma. \tag{8.10}$$

Claim 8.1 now implies that

$$\sum_{i:g_i<0} g_i \geq -\gamma - \sum_{i:g_i\geq 0} g_i \geq -2\gamma, \tag{8.11}$$

and thus $\sum_{i:g_i<0} |g_i| \leq 2\gamma$. In total, this shows that

$$\|g\|_1 = \sum_{i:g_i<0} |g_i| + \sum_{i:g_i\geq 0} |g_i| \leq 3\gamma. \tag{8.12}$$

With $\|g\|_2 \leq \|g\|_1$, Lemma 8.1 follows. □

Proof (Theorem 8.1). The proof consists of two steps. First, we bound the α-regret with $\alpha = 1$ for the sequence $\{x_t\}$ analogous to [Zinkevich, 2003], and then use this result to bound the α-regret for the sequence S_t.

Let $S^* \in \operatorname{argmin}_{S\in\mathcal{S}} \sum_{t=1}^T f_t(S)$. The definition $\tilde{f}_t(x) = \max_{g\in P_{f_t}} g \cdot x$ implies $f_t(S^*) = \tilde{f}_t(\chi_{S^*})$ and

$$\sum_{t=1}^T \tilde{f}_t(x_t) - \sum_{t=1}^T f_t(S^*) \ \leq\ \sum_{t=1}^T g_t \cdot x_t - \sum_{t=1}^T g_t \cdot \chi_{S^*}. \tag{8.13}$$

A proof similar to that in [Zinkevich, 2003] leads to a bound on the right hand side of Inequality (8.13) that we bound further:

$$2\sum_{t=1}^T g_t \cdot (x_t - \chi_{S^*}) \leq \max_{x,y\in\mathcal{K}} \|x-y\|^2/\eta + \eta T \max_t \|g_t\|^2 \tag{8.14}$$

$$\leq m/\eta + M^2 T\eta. \tag{8.15}$$

For the second inequality, we used $\|x-y\|^2 \leq m$ for all $x, y \in \mathcal{K}$ because $\mathcal{K} \subseteq [0,1]^{\mathcal{E}}$. Furthermore, we bounded the ℓ_2 norm of g_t by Lemma 8.1:

$$\|g_t\| \leq g_t \cdot \chi_{\mathcal{E}} \leq \max\nolimits_{t,A\subseteq\mathcal{E}} \beta_t |f_t(\mathcal{E})| \leq M. \tag{8.16}$$

Finally, the approximation factor for the rounding procedure implies that $f_t(S_t) = \tilde{f}_t(\chi_{S_t}) \leq \alpha \tilde{f}_t(x_t)$, so

$$\sum_{t=1}^T f_t(S_t) - \alpha\sum_{t=1}^T f_t(S^*) \leq \alpha\sum_{t=1}^T \tilde{f}_t(x_t) - \alpha\sum_{t=1}^T f_t(S^*) \tag{8.17}$$

$$\leq 0.5\alpha(\, m/\eta + M^2 T\eta\,). \tag{8.18}$$

The regret bound follows with $\eta = \sqrt{m}(M\sqrt{T})^{-1}$. □

A similar strategy works with exponentiated gradients – the regret bounds are as for linear cost problems, scaled by a factor α. The proof is analogous.

Suitable rounding techniques are not available for all submodular-cost problems. Instead, several algorithms approximate the cost function. Such an approach fits the Follow-the-leader framework that we describe next.

8.3. Approximations of the cost function

We now address algorithms of type (ii) that replace the cost function f by an approximation $\hat{f}$ and solve the resulting tractable problem instead. Several approximations $\hat{f}$ integrate with the Follow-the-leader principle [Hannan, 1957].

The Follow-the-leader principle suggests playing the best feasible set S given the costs observed so far. To prevent an adversary to exploit the determinism in this scheme, we add a regularizing perturbation $r : 2^{\mathcal{E}} \to \mathbb{R}$. In round t, pick [Kalai and Vempala, 2005]

$$S_t \in \operatorname*{argmin}_{S \in \mathcal{S}} \sum_{\tau=1}^{t-1} f_\tau(S) + r(S). \tag{8.19}$$

In the simplest case, the last term is a modular perturbation $r(S) = r \cdot \chi_S$ by a random vector r (Follow-the-perturbed-leader, FPL). For appropriately chosen r, the expected regret is then of order[2] $O(m/\sqrt{T})$, and $O(\sqrt{m/T})$ for modular functions f_τ.

In our setting, however, finding a minimizer S_t is NP-hard. Kakade et al. [2009] show that in general, an approximate minimizer S_t from an approximation algorithm does not suffice. Thus, instead of approximately solving the minimization (8.19), Algorithm 8 uses the *exact* expected minimizer of the *approximate* costs. The advantage of function-wise approximations is that the deviation of *each* $\hat{f}_t$ from f_t is bounded, whereas a general approximation factor for Problem (8.19) only refers to the quality with respect to the *sum* of costs. We define two general conditions for $\hat{f}$ and its associated factor α:

C1 The approximation $\hat{f}$ of f satisfies $f(A) \leq \hat{f}(A) \leq \alpha f(A)$ for all $A \in \mathcal{S}$.

C2 The following problem can be solved exactly in polynomial time:

$$\operatorname{argmin}_{S \in \mathcal{S}} \sum\nolimits_t \hat{f}_t(S) + \alpha r(S). \tag{8.20}$$

These constraints apply in a variety of settings, as we discuss later. Algorithm 8 integrates such an $\hat{f}$ into the FPL framework and adapts the perturbation r accordingly. Condition (C2) ensures that we can find S_t in polynomial time. If

[2]The first bound follows from the proof of Theorem 8.2 for $\alpha = 1$, the second is in [Kalai and Vempala, 2005]. The discrepancy by a factor $\sqrt{m}$ results from a different analysis that is needed for non-additive functions.

Algorithm 8 Follow the approximate perturbed leader

```
Input: $\eta > 0$
pick $r \in [0, M/\eta]^{\mathcal{E}}$ uniformly at random
for $t = 1$ to $T$ do
    set $S_t = \operatorname{argmin}_{S \in \mathcal{S}} \sum_{\tau=1}^{t-1} \hat{f}_\tau(S) + \alpha r(S)$
    obtain $f_t$
    approximate $f_t$ by $\hat{f}_t$
end for
```

(C1) and (C2) hold, then we can bound the expected regret for submodular costs in an approximation setting even with FPL.

Theorem 8.2. *For an approximation $\hat{f}$ that satisfies (C1) and (C2), and for $M = \max_t f_t(\mathcal{E})$ and $\eta = 1/\sqrt{2T}$, the α-regret of Algorithm 8 is bounded in expectation as $\mathbb{E}[R_\alpha(T)] \leq 2\sqrt{2}\alpha m M/\sqrt{T} = O(\alpha m/\sqrt{T})$.*

Proof. We define the minimizers

$$S_t \in \operatorname*{argmin}_{S \in \mathcal{S}} \sum_{\tau=1}^{t-1} \hat{f}_\tau(S) + \alpha r(S); \quad \widehat{S}_t \in \operatorname*{argmin}_{S \in \mathcal{S}} \sum_{\tau=1}^{t-1} \hat{f}_\tau(S); \quad S_t^* \in \operatorname*{argmin}_{S \in \mathcal{S}} \sum_{\tau=1}^{t} f_\tau(S). \tag{8.21}$$

First, we show a relation for $\sum_{t=1}^T \hat{f}_t(S_{t+1})$ and later relate it to the actual cost $\sum_{t=1}^T \hat{f}_t(S_t)$. The first inequality is[3]

$$\sum\nolimits_{t=1}^T \hat{f}_t(\widehat{S}_{t+1}) \;\leq\; \sum\nolimits_{t=1}^T \hat{f}_t(\widehat{S}_{T+1}). \tag{8.22}$$

It holds trivially for $T = 1$. The case $T+1$ follows by induction and the optimality of $\widehat{S}_{T+1}$:

$$\begin{aligned}
\sum\nolimits_{t=1}^{T+1} \hat{f}_t(\widehat{S}_{t+1}) &\leq \sum\nolimits_{t=1}^T \hat{f}_t(\widehat{S}_{T+1}) + \hat{f}_{T+1}(\widehat{S}_{T+2}) && (8.23)\\
&\leq \sum\nolimits_{t=1}^T \hat{f}_t(\widehat{S}_{T+2}) + \hat{f}_{T+1}(\widehat{S}_{T+2}) && (8.24)\\
&= \sum\nolimits_{t=1}^{T+1} \hat{f}_t(\widehat{S}_{T+2}). && (8.25)
\end{aligned}$$

We now replace $\hat{f}_1$ in Equation (8.22) by $\hat{f}_1 + \alpha r$ to get a bound similar to (8.22) but in terms of $\hat{f}_t + \alpha r$ rather than just in terms of $\hat{f}_t$. We also note that $S_1 \in \operatorname{argmin}_S r(S)$. This yields

$$\begin{aligned}
\sum\nolimits_{t=1}^T \hat{f}_t(S_{t+1}) + \alpha r(S_1) &\leq \sum\nolimits_{t=1}^T \hat{f}_t(S_{T+1}) + \alpha r(S_{T+1}) && (8.26)\\
&\leq \sum\nolimits_{t=1}^T \hat{f}_t(\widehat{S}_{T+1}) + \alpha r(\widehat{S}_{T+1}). && (8.27)
\end{aligned}$$

[3]Strictly speaking, we here imagine the algorithm to run for $T+2$ steps.

Rearranging the terms yields

$$\sum\nolimits_{t=1}^{T} \hat{f}_t(S_{t+1}) \leq \sum\nolimits_{t=1}^{T} \hat{f}_t(\widehat{S}_{T+1}) + \alpha(r(\widehat{S}_{T+1}) - r(S_1)). \tag{8.28}$$

To transfer this result to the series of S_t, we use that $\hat{f}_t(S_t) = \hat{f}_t(S_{t+1}) + (\hat{f}_t(S_t) - \hat{f}_t(S_{t+1}))$:

$$\sum_{t=1}^{T} \hat{f}_t(S_t) \leq \sum_{t=1}^{T} \hat{f}_t(\widehat{S}_{T+1}) + \sum_{t=1}^{T}(\hat{f}_t(S_t) - \hat{f}_t(S_{t+1})) + \alpha(r(\widehat{S}_{T+1}) - r(S_1)). \tag{8.29}$$

Condition (C1) implies that

$$\sum\nolimits_{t=1}^{T} \hat{f}_t(\widehat{S}_{T+1}) \leq \sum\nolimits_{t=1}^{T} \hat{f}_t(S_T^*) \leq \alpha \sum\nolimits_{t=1}^{T} f_t(S_T^*), \tag{8.30}$$

and that $\sum_{t=1}^{T} f_t(S_t) \leq \sum_{t=1}^{T} \hat{f}_t(S_t)$. Together with Equation (8.29), this yields

$$\begin{aligned}&\sum\nolimits_{t=1}^{T} f_t(S_t) - \alpha \sum\nolimits_{t=1}^{T} f_t(S_T^*) \\ &\quad \leq \sum\nolimits_{t=1}^{T}(\hat{f}_t(S_t) - \hat{f}_t(S_{t+1})) + \alpha(r(\widehat{S}_{T+1}) - r(S_1)).\end{aligned} \tag{8.31}$$

It remains to bound the two terms on the right hand side, and these bounds depend on $r \in [0, M/\eta]^{\mathcal{E}}$. We first address the random perturbation r in $[0, M/\eta]^{\mathcal{E}}$. The expectation of the second term can be bounded as

$$\alpha\mathbb{E}[r(\widehat{S}_{T+1}) - r(S_1)] \leq \alpha m M/\eta. \tag{8.32}$$

To bound the expected sum of differences of the function values, we use a technique by Hazan and Kale [2009]. For the analysis, one can assume that r is resampled in each round. We first bound $P(S_t \neq S_{t+1})$. A simple union bound holds:

$$P(S_t \neq S_{t+1}) \leq \sum_{i=1}^{m} P(e_i \in S_t \text{ and } e_i \notin S_{t+1}) + \sum_{i=1}^{m} P(e_i \notin S_t \text{ and } e_i \in S_{t+1}). \tag{8.33}$$

To bound the right hand side, we fix i and look at $P(e_i \in S_t \text{ and } e_i \notin S_{t+1})$. Denote the components of r by r_j and define $r' : 2^{\mathcal{E}} \to \mathbb{R}$ as $r'(S) = \sum_{e_j \in S, j \neq i} r_j$, so $r'(e_j) = r(e_j) = r_j$ for all $j \neq i$, but $r'(e_i) = 0$; and define $\Phi'_t : 2^{\mathcal{E}} \to \mathbb{R}$ as $\Phi'_t(S) = \sum_{\tau=1}^{t-1} \hat{f}_\tau(S) + \alpha r'(S)$. Now let

$$S^1 = \operatorname*{argmin}_{S \in \mathcal{S}, e_i \in S} \Phi'_t(S); \qquad S^2 = \operatorname*{argmin}_{S \in \mathcal{S}, e_i \notin S} \Phi'_t(S). \tag{8.34}$$

The event $e_i \in S_t$ only happens if $\Phi'_t(S^1) + \alpha r_i \leq \Phi'_t(S^2)$ and in such case $S_t = S^1$. On the other hand, to have $e_i \notin S_{t+1}$, it must be that $\Phi'_t(S^1) + \alpha r_i + \alpha M \geq \Phi'_t(S^2)$, since otherwise

$$\begin{aligned}\sum\nolimits_{\tau=1}^{t+1} \hat{f}_t(S^1) + \alpha r(S^1) &= \Phi'_t(S^1) + \alpha r_i + \hat{f}_t(S^1) && (8.35)\\ &< \Phi'_t(S^2) && (8.36)\\ &< \Phi'_t(B) + \hat{f}_t(B) && (8.37)\end{aligned}$$

for all $B \in \mathcal{S}$ with $e_i \notin B$. Here, we used that $\hat{f}_t(S) \leq \alpha f_t(S) \leq \alpha M$ for all $S \subseteq E$. Let $v = \alpha^{-1}(\Phi'_t(S^2) - \Phi'_t(S^1))$, then $e_i \in S_t$ and $e_i \notin S_{t+1}$ only if $r_i \in [v - M, v]$. The number r_i is in this range with probability at most η since it is chosen uniformly at random from $[0, M/\eta]$, so $P(e_i \in S_t \text{ and } e_i \notin S_{t+1}) \leq \eta$. The bound on $P(e_i \notin S_t \text{ and } e_i \in S_{t+1})$ follows by an analogous argumentation. Together, those results bound the right hand side of (8.33):

$$\begin{aligned}P(S_t \neq S_{t+1}) &\leq \sum_{i=1}^{m} P(e_i \in S_t \text{ and } e_i \notin S_{t+1}) + \sum_{i=1}^{m} P(e_i \notin S_t \text{ and } e_i \in S_{t+1})\\ &\leq 2m\eta. && (8.38)\end{aligned}$$

Equation (8.38) helps to bound the sum of function values, using $\hat{f}(C) \leq \alpha M$ for all C:

$$\begin{aligned}\sum\nolimits_{t=1}^{T} \mathbb{E}\big[\hat{f}_t(S_t) - \hat{f}_t(S_{t+1})\big] &\leq \sum\nolimits_{t=1}^{T} P(S_t \neq S_{t+1}) \max_{B \in \mathcal{S}} \hat{f}(B)\\ &\leq 2\alpha m M T \eta. && (8.39)\end{aligned}$$

Combining Inequalities (8.31), (8.32) and (8.39) results in

$$\mathbb{E}\big[\sum\nolimits_{t=1}^{T} f_t(S_t)\big] - \alpha \sum\nolimits_{t=1}^{T} f_t(S_T^*) \leq \alpha M m/\eta + 2\alpha m M T \eta.$$

The final regret bound follows for $\eta = 1/\sqrt{2T}$. □

8.3.1. Approximations fitting Algorithm 8

Conditions (C1) and (C2) show that a suitable approximation $\hat{f}$ is decisive. We list and derive examples for $\hat{f}$ that can be plugged into Algorithm 8.

Spanning tree and matching

The best approximation bound for Minimum spanning tree (MST) and Perfect matching with general submodular costs is $O(|\mathcal{V}|)$, and is achieved with the simple approximation $\hat{f}_{add}(S) = \sum_{e \in S} f(e)$ [Goel et al., 2009]. Since $\hat{f}_{add}$ is additive, any standard algorithm for MST or matching applies for satisfying (C2). (C1) holds by subadditivity of f.

The simple $\hat{f}_{add}$, however, often leads to rather loose approximation factors. Based on the approximation with polymatroidal flows in Section 4.2.2, we derive a better, nontrivial approximation for the problem structure of cuts.

Approximation for (s,t)-cuts

The approximation for sequential MinCoopCut builds on the approximation in Section 4.2.2, but requires some modifications. In particular, if we replace each f_t in the problem (8.20) by its approximation $\hat{f}_{pf}$, then almost always each such approximation will use a different optimal partition of the set of edges. However, a *sum* of such approximations, each using a different partition, is not a straightforward equivalent of a set of polymatroid network flow constraints that are easy to handle. Therefore, we instead use *the same* partition Π for all approximations $\hat{f}_t$. Selecting the fixed partition uniformly at random ensures in expectation the same approximation bound as the one in Theorem 4.1.

Let $\{E_v^-\}_{v\in V}$ be the partition of $\mathcal{E}$ where E_v^- contains all edges $e=(u,v)$ with head v, and $\{E_u^+\}_{u\in V}$ be the analogous partition that assigns each edge $e=(u,v)$ to its tail node u. For either partition, we define an approximate cost function similar to $\hat{f}_{pf}$:

$$\hat{f}^-(S)=\sum_{v\in V} f(S\cap E_v^-); \qquad \hat{f}^+(S)=\sum_{v\in V} f(S\cap E_v^+).$$

By subadditivity, both functions are upper bounds on f. At the beginning of Algorithm 8, we decide uniformly at random whether to use $\hat{f}\triangleq\hat{f}^-$ or $\hat{f}\triangleq\hat{f}^+$ and retain this choice throughout, so that $\sum_{\tau=1}^{t-1}\hat{f}_\tau=\hat{\Phi}_t$ for $\Phi_t\triangleq\sum_{\tau=1}^{t-1} f_t$ throughout.

With this strategy, the factor α in (C1) improves from m (for $\hat{f}_{add}$) to $|\mathcal{V}|/2$:

Lemma 8.2. *Let $\hat{f}$ be chosen uniformly at random as either $\hat{f}^-$ or $\hat{f}^+$. Then $f(S)\leq\mathbb{E}[\hat{f}(S)]\leq(|\mathcal{V}|/2)f(S)$ for all minimal (s,t)-cuts S.*

The lemma follows analogously to Theorem 4.1. If we use the same partition for all $\hat{f}_t$, then we can solve the minimization (8.20) as a polymatroidal network flow problem, similar to the approach for $\hat{f}_{pf}$ in Section 4.2.2. Recall that the dual to a polymatroidal flow is a minimum cut problem with cost function [Lovász, 1983]

$$c(S)=\min_{T\subseteq S}\sum_v \mathrm{cap}_v^{\mathrm{in}}(T\cap E_v^-)+\mathrm{cap}_v^{\mathrm{out}}((S\setminus T)\cap E_v^+).$$

As for $\hat{f}_{pf}$, we set the capacity functions of the the flow to represent the cost in (C2). If $\hat{f}=\hat{f}^-$, then we set $\mathrm{cap}_v^{\mathrm{out}}$ to some large value so that $c(S)$ only uses $\mathrm{cap}_v^{\mathrm{in}}$, and set $\mathrm{cap}_v^{\mathrm{in}}(A)=\sum_t f_t(A\cap E_v^-)+\alpha r(A\cap E_v^-)$. Then

$$c(S)=\sum\nolimits_v\sum\nolimits_t f_t(S\cap E_v^-)+\alpha r(S\cap E_v^-)=\sum\nolimits_t \hat{f}_t(S)+\alpha r(S).$$

The procedure for $\hat{f}^+$ is analogous.

A generic approximation

A generic nontrivial approximation $\hat{f}_{ea}$ for submodular functions was proposed by Goemans et al. [2009] (outlined in Section 4.1.1), but it does not satisfy (C2) in a straightforward way. Its functional form is $\hat{f}(S) = \sqrt{\sum_{e\in S} w(e)}$, and a sum of square roots is hard to optimize.

Nevertheless, it is possible to use the square $\hat{f}^2_{ea}$. The square satisfies $\hat{f}^2_{ea}(A) \leq f^2(A) \leq \alpha_g^2 \hat{f}^2_{ea}(A)$ for all $A \subseteq \mathcal{E}$, with $\alpha_g = O(\sqrt{m}\log m)$. Even better, $\hat{f}^2$ is a modular function, and modular functions have been studied much more than nonlinear functions in the online combinatorial setting. We can use any algorithm for modular costs, and, when observing f_t, we pretend to have seen $\hat{f}^2_t$.

To state the regret bound, let $\nu = \min_{t,S\in\mathcal{S}} f_t(S)$. We make the reasonable assumption that each element in $\mathcal{E}$ has nonzero cost, and then $\nu > 0$.

Lemma 8.3. *Let $\widehat{R}_{\mathcal{A}}$ be the regret of an online algorithm $\mathcal{A}$ when used with linear cost functions $\hat{f}^2_t$. Using $\mathcal{A}$ with $\hat{f}^2_t$ when observing f_t leads to an α_g-regret of $R_{\alpha_g}(T) \leq \alpha_g \widehat{R}_{\mathcal{A}}/\nu$.*

Proof. Since we use $\hat{f}^2_t$ in $\mathcal{A}$, the regret $\widehat{R}_{\mathcal{A}}$ bounds the difference $\frac{1}{T}\sum_t(\hat{f}^2_t(S_t) - \hat{f}^2_t(\widehat{S}^*))$ to $\hat{S}^* = \operatorname{argmin}_{S\in\mathcal{S}} \sum_t \hat{f}^2_t(S)$. Therefore, we relate the actual regret, $\frac{1}{T}\sum_t(f_t(S_t) - \alpha_g f_t(S^*))$, to the regret of $\mathcal{A}$. We use that $\hat{f}^2(S) \leq f^2(S) \leq \alpha_g^2 \hat{f}^2(S)$ and that $\hat{S}^*$ is optimal for $\hat{f}^2$. It holds that

$$\begin{aligned}
\sum\nolimits_t (f_t(S_t) - \alpha_g f_t(S^*)) &= \sum\nolimits_t \frac{(f_t^2(S_t) - \alpha_g^2 f_t^2(S^*))}{(f_t(S_t) + \alpha_g f_t(S^*))} \\
&\leq \sum\nolimits_t (f_t^2(S_t) - \alpha_g^2 f_t^2(S^*))/(\alpha_g \nu) \\
&\leq \sum\nolimits_t \alpha_g^2(\hat{f}_t^2(S_t) - \hat{f}_t^2(S^*))/(\alpha_g \nu) \\
&\leq \sum\nolimits_t \alpha_g(\hat{f}_t^2(S_t) - \hat{f}_t^2(\widehat{S}^*))/(\nu) \\
&= \alpha_g \widehat{R}_{\mathcal{A}}/\nu.
\end{aligned}$$

□

8.4. Label costs and related functions

The two preceding sections proposed online algorithms for general submodular costs. The associated approximation factors α usually match their lower bounds and in the general case cannot be improved further. However, certain sub-classes of submodular functions admit better approximation factors. For example, the approximation factor for Minimum Spanning Tree drops from linear to logarithmic in $|\mathcal{V}|$ if f is a label cost function. Therefore, we address a specific algorithm for the label costs introduced in Section 2.5. We begin by a unifying viewpoint on algorithms for the offline problem in Section 8.4.1. This perspective leads to generalizations (matroids and Section 8.4.3) and an online algorithm described in Section 8.4.2.

8.4.1. Label costs and approximations: a cover viewpoint

A label cost function assigns a set of labels $\pi(e)$ from a label space $\mathcal{L}$ to each element e in the ground set. In the simplest, most often addressed case, $\pi(e)$ contains only one label. Each label $\ell \in \mathcal{L}$ has a cost $c(\ell)$, and the cost of a set $S \subseteq \mathcal{E}$ of items is the additive cost of its labels $\pi(S) = \{\ell \mid \ell \in \bigcup_{e \in S} \pi(e)\}$:

$$f(S) = c\Big(\bigcup_{e \in S} \pi(e)\Big) = \sum_{\ell \in \pi(S)} c(\ell). \tag{8.40}$$

When designing algorithms for label costs, a key observation is that this cost is additive in terms of labels. We thus rephrase Problem 8.1 as a modular-cost label selection problem. To relate labels back to items, we invert the function π and define $\mathcal{E}(L) \triangleq \{e \mid \pi(e) \in L\}$. In other words, selecting a label ℓ means selecting all items that carry this particular label. The problem now reads as follows: select the cheapest set $L \subseteq \mathcal{L}$ of labels such that $\mathcal{E}(L)$ satisfies a relaxed constraint. The final solution will be a subset of $\mathcal{E}(L)$.

The initial constraints require selecting a structure $S \in \mathcal{S}$. In terms of labels, we demand that the selected $\mathcal{E}(L)$ must contain a subset $T \subseteq \mathcal{E}(L)$ that is in $\mathcal{S}$. Finding *some* feasible set T within $\mathcal{E}(L)$ is in general much easier than finding a minimum-cost structure, and this pruning step finishes the algorithm. As the cost f is nondecreasing, the set T is not more costly than $\mathcal{E}(L)$, that is, $f(T) \leq f(\mathcal{E}(L))$. Mathematically, we have relaxed the family of feasible sets $\mathcal{S}$ to be up-monotone: any superset of a feasible solution is also feasible. In this light, the new label-focused problem has the structure of a covering problem: we must include at least one $S \in \mathcal{S}$ completely, and may include arbitrary other elements.

Even though not explicitly stated in the respective works, several existing algorithms for label cost problems [Krumke and Wirth, 1998, Monnot, 2005, Hassin et al., 2007, Zhang et al., 2011] can be viewed as solving this type of covering problem. Those algorithms include methods for (budgeted) covering to select groups $\mathcal{E}(\ell)$ via labels.

The difficulty of the transformed problem depends on how easily the structural constraint $S \in \mathcal{S}$ propagates via the labels. Covering constraints and matroid constraints are particularly suitable for the reformulation. The former remain covering constraints, and matroid constraints result in a submodular cover problem, as we demonstrate below. The transformation below applies to any set $\mathcal{S}$ consisting of the independent sets of a matroid; spanning trees are a prominent but not the only example.

Minimum label cost problems with matroid constraints

Before addressing the online algorithm, we demonstrate the above covering scheme for sets $\mathcal{S}$ consisting of the maximal independent sets of a matroid. The known greedy algorithm for spanning trees follows as a special case. Let $r : 2^{\mathcal{E}} \to \mathbb{N}$ be the rank function of the matroid. For spanning trees, this is the rank of a graphic matroid. The constraint $S \in \mathcal{S}$ is equivalent to demanding that

$$r(S) \geq r(\mathcal{E}), \tag{8.41}$$

and that S is minimal in this respect (no subset of S satisfies (8.41)). First, we relax the minimality constraint and merely enforce the rank constraint (8.41). Second, to formulate the constraint in terms of labels, we define the function $g : 2^{\mathcal{L}} \to \mathbb{N}$,

$$g(L) \triangleq r(\mathcal{E}(L)). \tag{8.42}$$

The function g is an integer-valued polymatroid rank function: it is nondecreasing submodular, for example by Proposition 2.1. Furthermore, $g(L) = g(\mathcal{L}) = r(\mathcal{E})$ if and only if $\mathcal{E}(L)$ contains a feasible set $T \in \mathcal{S}$.

Given the cost vector c of the labels, the label selection problem is now a submodular cover problem:

$$\min\ c(L) \quad \text{s.t. } g(L) \geq g(\mathcal{L}). \tag{8.43}$$

Problem (8.43) is solved by a greedy algorithm for submodular cover problems, with approximation factor $\alpha = H(\max_\ell g(\ell)) = O(\log |\mathcal{V}|)$ [Wolsey, 1982]. Here, $H(n) = \sum_{k=1}^{n} \frac{1}{k}$ is the nth harmonic number.

8.4.2. Online algorithm

Next we design an algorithm for sequentially selecting a structure $S_t \in \mathcal{S}$ with low label cost f_t. We treat the problem as a label selection problem with covering constraints. The batch approximation algorithm will help find the desired labels. We assume in the sequel that the labels of the elements are fixed, but that the cost of the labels changes over time.

The online algorithm is outlined as Algorithm 9. Instead of directly choosing a structure, Algorithm 9 picks in each step a set of *labels* $L_t \subseteq \mathcal{L}$ with minimum cost. This L_t must be such that the set of elements $\mathcal{E}(L_t)$ contains the desired structure $S_t \in \mathcal{S}$, e.g., a tree.

It remains to determine (i) by which criterion to choose L_t, and (ii) how to find an L_t that contains the desired structure. As to (i), we use an approximate gradient descent that applies thanks to the structure of f_t. Algorithm 9 maintains a continuous correspondent y_t of L_t and moves from y_t into the direction of the negative gradient $-c_t$. The resulting point is projected onto the feasible set of label sets via the approximate projections by Kakade et al. [2009], denoted by ApproxProj. This method applies because the cost of our re-formulation is additive in the labels.

Finally, the approximate projections rely on finding a set L_t with minimum cost c' such that $\mathcal{E}(L_t)$ contains a set $T \in \mathcal{S}$. If $\mathcal{S}$ is defined as the bases of a matroid,

Algorithm 9 Online label cost minimization

```
pick any $S_1 \in \mathcal{S}$; set $L_1 = \bigcup_{e \in S_1} \pi(e),\ y_1 = \chi_{L_1}$
for $t = 2$ to $T$ do
    $(y_t, L_t) = \mathrm{ApproxProj}(y_{t-1} - \eta c_{t-1}, L_{t-1}, y_{t-1})$
    find $S_t \subseteq \mathcal{E}(L_t)$, $S_t \in \mathcal{S}$
    obtain $f_t$ and extract $c_t$
end for
```

then we use the algorithm from Section 8.4.1. Some structures, such as paths, are not easily represented by matroids. Given such an $\mathcal{S}$, we solve a minimum label cost problem with cost c'. Let S^* be the resulting solution. Then the desired label set is the set $\pi(S^*)$ of all labels used by S^*. Approximation algorithms for label costs exist for paths [Hassin et al., 2007], matchings [Monnot, 2005] and cuts [Zhang et al., 2011].

The regret bound for Algorithm 9 involves the total number of labels, $|\mathcal{L}| \leq m$, instead of the size m of the ground set. The factor α is the approximation factor of the respective offline algorithm.

Theorem 8.3. *The regret of Algorithm 9 is bounded as $R_\alpha(T) = O(\alpha\sqrt{M|\mathcal{L}|/T})$, where $M = \max_t f_t(E)$.*

The bound follows from Theorem 3.2 in [Kakade et al., 2009] for $\eta = (\alpha + 1)\sqrt{|\mathcal{L}|/(MT)}$, and from the equivalence of picking labels and structures, outlined above. For spanning trees, Theorem 8.3 immediately implies a regret bound of $O(\sqrt{|\mathcal{L}|M}\log|\mathcal{V}|/\sqrt{T})$.

8.4.3. Multiple labels and truncated costs

If $\mathcal{S}$ is the set of all spanning trees of a graph, then the transformation into a label selection problem extends to multiple labels per edge and to simple thresholded costs. When facing multi-label costs, let k be the maximum number of labels any edge can have. We assign k "slots" to each edge, as shown in Figure 8.1(a). Each label $\ell \in \pi(e)$ occupies $1 \leq \gamma_e(\ell) \leq k$ slots in edge e, such that $\sum_{\ell \in \pi(e)} \gamma_e(\ell) = k$. Define k copies $\mathcal{G}_i = (\mathcal{V}, \mathcal{E}_i)$ of the original graph $\mathcal{G}$. Edge e is contained in $\mathcal{E}_i(L)$ if at least i of its slots are filled by labels in L. Then we use

$$g(L) = \sum\nolimits_{i=1}^{k} r(\mathcal{E}_i(L)). \tag{8.44}$$

This sum is still submodular, and maximum only if $\mathcal{E}(L)$ contains a tree of full edges, meaning that for at least one tree, all necessary labels are chosen. The approximation factor increases moderately to $O(\log(nk))$.

In a similar spirit, truncated costs of the form $c(L) = \min\{w \cdot \chi_L, \gamma\}$ can be simulated by parallel edges, so that the algorithm can pick a full group or single edges via labels. Figure 8.1(e) illustrates an example.

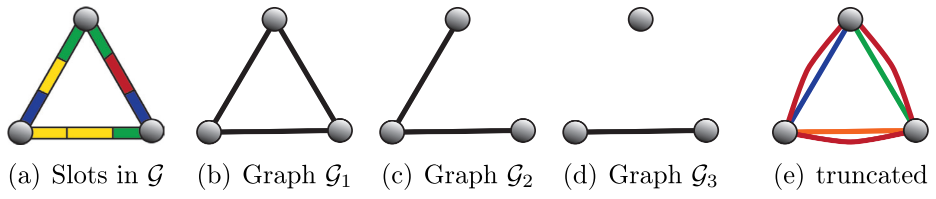

(a) Slots in $\mathcal{G}$
(b) Graph $\mathcal{G}_1$
(c) Graph $\mathcal{G}_2$
(d) Graph $\mathcal{G}_3$
(e) truncated

Figure 8.1. Construction of g for multiple labels. Colors indicate labels, and graphs $\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3$ refer to a selection of $L = \{\text{green, yellow}\}$ with $g(L) = 5$. In (e), the cost of the red label is $\gamma < w(\{\text{blue, green, orange}\})$.

8.5. Summary and discussion

Previous work on online learning of combinatorial problems has focused on linear cost functions or simple constraints. Online submodular-cost combinatorial problems combine three challenges: an exponential decision space, non-separable cost functions and only approximate batch optimization methods.

We showed two generic approaches to online algorithms for submodular costs and combinatorial structures: a projected subgradient algorithm that works with relaxations and a Follow-the-leader strategy for approximations of the cost function. The algorithms rely on the classification of approximation techniques that we discuss in Chapter 4. Despite being not completely ignorant of the underlying approximation strategy, they are still generic enough to cover almost all existing offline approximation methods for structured concepts with submodular costs. Algorithm 9 is even more generic: it fits *any* algorithm for a label cost problem, and yields vanishing α-regret even for the better approximation factors α attainable with label costs. The cover viewpoint extends the algorithm to more general multi-label and truncated label costs.

Our work contributes to extending results for the online combinatorial setting from linear costs to nonlinear costs. In particular, it yields the first regret bounds for structured concepts with submodular costs. This complements the work in the unconstrained setting [Hazan and Kale, 2009], and the work on online submodular maximization [Streeter and Golovin, 2008, Streeter et al., 2009]. "Combinatorial bandits" have been explored for linear cost functions [Cesa-Bianchi and Lugosi, 2009] – an open question remains what is achievable in the combinatorial bandit setting for nonlinear, nonseparable costs.

Chapter 9. Conclusion and Outlook

In Chapter 1 we argued that machine learning and computer vision drive a quest for modeling discrete problems with interactions between elements. Large data sets add the demand for algorithms whose running time scales moderately with the size of the input data. We posed the questions: *Which models might well trade off the representation of rich interactions with algorithmic efficiency? Can we design models and algorithms that effectively find practically applicable solutions considering higher-order interactions?*

This thesis proposes answers to the above questions by introducing cooperative cuts and connecting them to applications. In particular, the results in the preceding chapters illustrate three aspects: (1) submodular functions in combinatorial problems can model relevant interactions; (2) approximation algorithms return solutions that obey theoretical bounds and that improve results in applications; (3) the algorithms are applicable to real-world data. The results also raise open questions that we outline after a discussion of these three points.

Submodular interactions are relevant

The examples in Section 2.5 illustrate that submodular cost functions in combinatorial problems express various types of variable interactions that are relevant in real-world problems. Cooperative cuts, in particular, represent a family of functions on discrete domains that admit higher-order interactions, and these functions unify and generalize several existing models in computer vision. The improved model for image segmentation in Chapter 6 and the approximation algorithm for submodular minimization in Chapter 7 provide further evidence that coupling edges in graphs introduces not only rich, but also useful interactions. In addition, the relaxed problem relates to established models in computer vision and machine learning.

Approximate solutions

MinCoopCut is a very hard optimization problem. Important is thus not only a rich model but also the approximation algorithm: how good are the solutions it finds, in terms of the mathematical optimization problem and of the application? Even though we cannot efficiently obtain exact solutions, the results in Chapters 6 and 7 suggest that the proposed approximation algorithms can provide visually and numerically appealing empirical results. As to optimization, all algorithms discussed in Chapter 4 have bounded approximation factors, and we demonstrated by an example that the proven factors can be decisive in preventing arbitrarily poor solutions. Furthermore, the theoretical results in Chapters 3 and 4 complement the recent theoretical analyses of other combinatorial problems with submodular cost functions. Chapter 8 extends some of the discussed approximation techniques to an online framework while ensuring Hannan-consistency. Besides theoretical analyses, the empirical results in Chapter 4 indicate that the approximation factors for average-case inputs are lower than the theoretical worst-case bounds. We however also saw that in the worst case, some of the proven bounds are tight.

Applicability

Lastly, some of the proposed algorithms are efficient enough to be used with realistic data: images (Chapter 6) and a speech corpus (Chapter 7). In particular the iterative and randomized greedy algorithms are efficient in practice.

The three points suggest that cooperative cuts offer both richness and usable approximate solutions. Still, the presented results also raise a host of open research questions.

A selection of open problems

1. Cooperative cuts provide models with exploitable structure that yield appealing empirical results. Nevertheless, the structure is not enough to admit exact solutions or constant approximation factors in polynomial time. Is there a combinatorial structure (e.g., certain graphs or matroids) that admits better approximations while retaining sufficient interaction between elements? In general, continuing the idea of using combinatorial structures for inference appears to be an interesting direction to pursue.
2. The approximation factors and the lower bound provided in this thesis address general nondecreasing submodular functions. Are there (expressive) sub-classes of the discussed problems that admit better approximations? The answer to this question depends on the combination of graph structure and cost function. While for general graphs, the family of label cost functions is still hard [Zhang et al., 2011], sums of weights and bottleneck costs $f(S) = \max_{e \in S} w(e)$ are not. The approximation results for specific cases in [Nikolova, 2010, Goyal and Ravi, 2008, Mittal and Schulz, 2012] might add to the discussion, as well as the level at which interaction can be detected, as mentioned in Section 2.3.3.
3. This thesis provides a theoretical foundation for defining energy functions via cooperative cuts, including models for binary and multi-label discrete variables, and also continuous regularizing terms. While a selection of applications is shown, there is much room for further applications of the offered

coupling, using general cooperative cut energies or the specific boundary congruity energies. In addition, the combinatorial model itself is applicable, as indicated e.g. in Section 2.5.

4. We designed online algorithms for submodular-cost combinatorial problems in the full information setting. An interesting open question is what is achievable for submodular costs and combinatorial constraints with bandit feedback, where instead of the function oracle, only the evaluation at chosen solutions is accessible.

In summary, cooperative cuts are one possibility to incorporate higher-order interactions into combinatorial problems. Despite the hardness of the resulting optimization problem, the generality and the empirical results suggest that introducing and approximating structured variable interactions is a valuable route to take.

Appendix A.

Notation

$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	(structure) graph
$\mathcal{E}$	ground set of elements, usually edges in a graph
$2^{\mathcal{E}}$	power set of $\mathcal{E}$
n	number of nodes
m	number of edges and size of the ground set
A, B, C, S, T	sets of elements
χ_A	characteristic vector of set A
e	an edge
$w(S)$	modular function defined as $w(S) = \sum_{e \in S} w(e)$
$A + e$	$A \cup \{e\}$
$A - e$	$A \setminus \{e\}$
f	cost function (a set function)
$\tilde{f}$	Lovász extension of f
$\hat{f}$	function that approximates f
$\rho_f(A\|B)$	marginal cost of A with respect to B
α	approximation factor
$\mathsf{G} = (\mathsf{V}, \mathsf{E})$	graphical model or auxiliary graph
$\delta(X)$	boundary of $X \subseteq \mathcal{V}$: $\{(u,v) \in \mathcal{E} \mid u \in X, v \notin X\}$
$X(\boldsymbol{x})$	set of nodes selected by characteristic vector $\boldsymbol{x}$
$\Gamma(\boldsymbol{x})$	boundary induced by binary vector $\boldsymbol{x}$; $\Gamma(\boldsymbol{x}) = \delta(X(\boldsymbol{x}))$
$E(\boldsymbol{x})$	energy function

Appendix B.

Further Details

B.1. Derivation of $D'(n)$ (Chapter 3)

In this section, we derive the number $D'(n)$ of modified derangements that was used to prove MinCoopCut to be NP-hard. A derangement is a permutation, i.e., a mapping $\sigma : \{1, \dots, n\} \to \{1, \dots, n\}$, where no element can be mapped to itself: $\sigma(i) \neq i$ for all $1 \leq i \leq n$. We define a relaxed version of a derangement, where one pre-specified element i' can be mapped to itself, but no other element can: $\sigma(i') \in \{1, \dots, n\}$, but $\sigma(i) \neq i$ for all $i \neq i'$. The number $D'(n)$ is the number of such relaxed derangements given a specific i'.

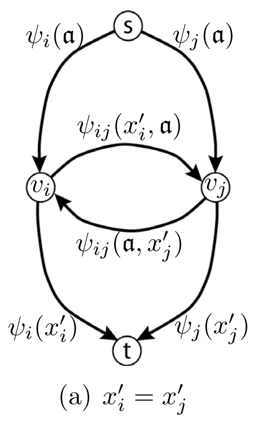
We derive $D'(n)$ by the method of the forbidden board [Stanley, 1997, pp. 71-73]. Let, without loss of generality, $i' = n$. Then the forbidden board is $B = \{(1,1), (2,2), \dots, (n-1, n-1)\}$. Let N_j be the number of permutations σ for which $\left|\{(i,\sigma(i)\}_{i=1}^{n} \cap B\right| = j$; the graph of these permutations coincides with B in j positions. Furthermore, let r_k be the number of k-subsets of B such that no two elements have a coordinate in common. The polynomial

$$N_n(x) = \sum_j N_j x^j = \sum_{k=0}^{n} r_k (n-k)!(x-1)^k \tag{B.1}$$

gives the desired solution $D'(n) = N_0 = N_n(0)$. For the board B above, $r_k = \binom{n-1}{k}$. Thus,

$$\begin{aligned} N_n(x) &= \sum_{k=0}^{n} r_k (n-k)!(x-1)^k && \text{(B.2)} \\ &= \sum_{k=0}^{n} \binom{n-1}{k} (n-k)!(x-1)^k && \text{(B.3)} \\ &= \sum_{k=0}^{n} \frac{(n-1)!}{k!(n-1-k)!} (n-k)!(x-1)^k && \text{(B.4)} \\ &= \sum_{k=0}^{n} \frac{(n-1)!}{k!} (n-k)(x-1)^k. && \text{(B.5)} \end{aligned}$$

Then $D'(n) = N_n(0) = \sum_{k=0}^{n} \frac{(n-1)!}{k!}(n-k)!(-1)^k$ and $D'(n-1) = N_{n-1}(0) = \sum_{k=0}^{n-1} \frac{(n-2)!}{k!}(n-1-k)!(-1)^k$.



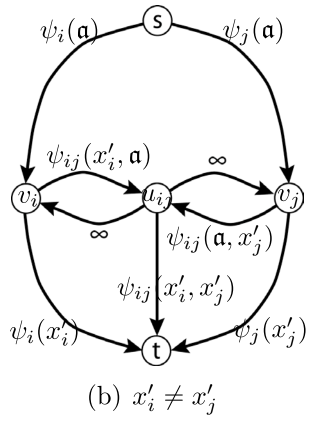


Figure B.1. Graph construction for expanding label $\mathfrak{a}$ for the approximate energy $\widehat{E}$. If $x'_i = \mathfrak{a}$, we set the weight of edge (v_i, t) to ∞ rather than to $\psi_i(\mathfrak{a})$, and equally for $x'_j = \mathfrak{a}$.

B.2. Expansion moves for non-symmetric potentials (Chapter 5)

In this section, we show an alternative construction for expansion moves that apply to the approximation $\widehat{E}$ of a multi-label cooperative cut energy without assuming the composite form of Lemma 5.6. This construction is closer to that by [Boykov et al., 2001][1], with the difference that it does not require symmetry.

The construction in the sequel applies in fact to any type of symmetric or non-symmetric potential. The only requirement is a triangle inequality for the (uni-directional) potentials $\psi_{ij,\Gamma'}$, which is satisfied by M1 and M2. We show it for M2:

$$\begin{aligned}\psi_{ij,\Gamma'}(\mathfrak{a}, \mathfrak{b}) &= w_{\mathfrak{a}}((v_i, v_j)) + w_{\mathfrak{b}}((v_j, v_i)) && \text{(B.6)}\\ &\leq w_{\mathfrak{a}}((v_i, v_j)) + w_{\mathfrak{c}}((v_j, v_i)) + w_{\mathfrak{c}}((v_i, v_j)) + w_{\mathfrak{b}}((v_j, v_i)) && \text{(B.7)}\\ &= \psi_{ij,\Gamma'}(\mathfrak{a}, \mathfrak{c}) + \psi_{ij,\Gamma'}(\mathfrak{c}, \mathfrak{b}). && \text{(B.8)}\end{aligned}$$

An expansion move with respect to a label $\mathfrak{a}$ and a given current assignment $\boldsymbol{x}'$ finds the best relabeling

$$\boldsymbol{x}^{\mathfrak{a}} \in \operatorname{argmin} \widehat{E}(\boldsymbol{x}) \quad \text{s.t. } \boldsymbol{x} \in \mathcal{X}(\boldsymbol{x}', \mathfrak{a}).$$

Lemma 5.5 states that given a current assignment $x' \in \mathcal{L}^n$ and any label $\mathfrak{a}$, a minimizer $\boldsymbol{x}^{\mathfrak{a}} \in \mathcal{X}(\boldsymbol{x}', \mathfrak{a})$ of $\widehat{E}$ can be found via a minimum cut in a graph. We prove this lemma here without assuming composite potentials.

[1]We have recently been told that an asymmetric construction was also known to the authors of that paper but was never published. The construction here is ours.

Proof (Lemma 5.5). Given a structure graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we construct an auxiliary graph $\mathsf{G} = (\mathsf{V}, \mathsf{E})$. Its nodes consist of the nodes in $\mathcal{V}$, terminal nodes s,t, and additional auxiliary nodes $\mathsf{V}_\mathfrak{a}$.

For each $v_i \in \mathcal{V}$, we introduce edges (s, v_i) with weight $\psi_i(\mathfrak{a})$ and (v_i, t) with weight $\psi_i(x'_i)$. If a variable already has label $\mathfrak{a}$, $x'_i = \mathfrak{a}$, then we set the weight of edge (v_i, t) to infinity, so that node v_i will always be in the partition belonging to t. We will infer a labeling from an (s,t)-cut as follows: if edge (s, v_i) is cut, then we set $x_i = \mathfrak{a}$, otherwise edge (v_i, t) is cut, and we retain the label $x_i = x'_i$. If the weight $w(v_i, t) = \infty$, then any minimum (s,t)-cut will assign v_i to t and yield $x_i = \mathfrak{a}$. This proves that any labeling resulting from a minimum cut must be in $\mathcal{X}(\boldsymbol{x}', \mathfrak{a})$.

Now we set the remaining edges such that there is a direct correspondence between labelings and cuts. In particular, for any assignment $\boldsymbol{x} \in \mathcal{X}(\boldsymbol{x}', \mathfrak{a})$ the resulting graph will satisfy

$$\widehat{E}(\boldsymbol{x}) = \min_{Y \subseteq \mathsf{V}_\mathfrak{a}} w(\delta(X(\boldsymbol{x}) \cup Y)). \tag{B.9}$$

That means, if we assign each $v_i \in \mathcal{V}$ to s or t according to x_i, then we assign the remaining nodes in G to minimize the corresponding cut.

For each pair v_i, v_j in $\mathcal{V}$ that is connected in $\mathcal{G}$, we introduce connecting edges. If the current labeling is $x'_i = x'_j$, then we introduce edges (v_i, v_j) with weight $\psi_{ij}(x'_i, \mathfrak{a})$ and (v_j, v_i) with weight $\psi_{ij}(\mathfrak{a}, x'_j)$, as shown in Figure B.1(a). This is the same construction as in the proof of Lemma 5.6.

If the labels differ in $\boldsymbol{x}'$, i.e., $x'_i \neq x'_j$, then we introduce an auxiliary node $u_{ij} \in \mathsf{V}_\mathfrak{a}$, and construct edges as shown in Figure B.1(b). Enumerating all cuts reveals that this part of the graph satisfies Equation (B.9) for the pair x_i, x_j. Table B.1 lists all those cuts and their costs. For each assignment x_i, x_j, there are two possibilities of assigning u_{ij}, either to s or to t. One of those cuts is worse, either because its weight is infinite or thanks to the triangle inequality (B.8). The cost of the better cut is always equal to the energy $\widehat{E}(x_i, x_j)$. This shows that Equation (B.9) holds.

In summary, Equation (B.9) holds for all pairs x_i, x_j, and thus for $\boldsymbol{x}$. As a result, a minimum cut in G yields a minimum energy labeling, the sought minimizer $\boldsymbol{x}^\mathfrak{a}$. $\square$

	to s (old label)	to t (new label $\mathfrak{a}$)	cut cost
	v_i, u_{ij}	v_j	$\psi_i(x'_i) + \psi_j(\mathfrak{a}) + \psi_{ij}(x'_i, x'_j) + \infty$
$\star$	v_i	v_j, u_{ij}	$\psi_i(x'_i) + \psi_j(\mathfrak{a}) + \psi_{ij}(x'_i, \mathfrak{a})$
	v_j, u_{ij}	v_i	$\psi_i(\mathfrak{a}) + \psi_j(x'_j) + \psi_{ij}(x'_i, x'_j) + \infty$
$\star$	v_j	v_i, u_{ij}	$\psi_i(\mathfrak{a}) + \psi_j(x'_j) + \psi_{ij}(\mathfrak{a}, x'_j)$
$\star$	v_i, v_j, u_{ij}	—	$\psi_i(x'_i) + \psi_j(x'_j) + \psi_{ij}(x'_i, x'_j)$
	v_i, v_j	u_{ij}	$\psi_i(x'_i) + \psi_j(x'_j) + \psi_{ij}(x'_i, \mathfrak{a}) + \psi_{ij}(\mathfrak{a}, x'_j)$
$\star$	—	v_i, v_j, u_{ij}	$\psi_i(\mathfrak{a}) + \psi_j(\mathfrak{a})$
	u_{ij}	v_i, v_j	$\psi_i(\mathfrak{a}) + \psi_j(\mathfrak{a}) + 2\infty$

Table B.1. Enumeration of all possible (s, t)-cuts, i.e., assignments of nodes v_i, v_j, u_{ij} in Figure B.1(b). The preferred cut (for fixed assignments of v_i, v_j, varying only that of u_{ij}) is marked by a star.

Bibliography

A.M. Abdelbar and S.M. Hedetniemi. Approximating MAPs on belief networks is NP-hard and other theorems. *Artificial Intelligence*, 102, 1998.

J. Abernethy, E. Hazan, and A. Rakhlin. Competing in the dark: An efficient algorithm for bandit optimization. In *Conference on Learning Theory (COLT)*, 2008.

V. Aggarwal, V. G. Tikekar, and L.-F. Hsu. Bottleneck assignment problem under categorization. *Computers & Operations Research*, 13:11–26, 1986.

R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows.* Prentice Hall, 1993.

C. Allène, J.-Y. Audibert, M. Couprie, and R. Keriven. Some links between extremum spanning forests, watersheds, and min-cuts. *Image and Vision Computing*, 2009.

S. Amghibech. Eigenvalues of the discrete p-Laplacian for graphs. *Ars Combin.*, 67, 2003.

E. Angel, E. Bampis, and L. Gourvès. On the minimum hitting set of bundles problem. *Theoretical Computer Science*, 410(45):4534–4542, 2009.

A. Atamtürk and V. Narayanan. Polymatroids and mean-risk minimization in discrete optimization. *Operations Research Letters*, 36(5):618–622, 2008.

J.-Y. Audibert, S. Bubeck, and G. Lugosi. Minimax policies for combinatorial prediction games. In *Conference on Learning Theory (COLT)*, 2011.

I. Averbakh and O. Berman. Categorized bottleneck-minisum path problems on networks. *Operations Research Letters*, 16:291–297, 1994.

B. Awerbuch and R. D. Kleinberg. Adaptive routing with end-to-end feedback: distributed learning and geometric approaches. In *Symposium on Theory of Computing (STOC)*, 2004.

F. Bach. Structured sparsity-inducing norms through submodular functions. In *Advances in Neural Information Processing Systems (NIPS)*, 2010.

F. Bach. Shaping level sets with submodular functions. In *Advances in Neural Information Processing Systems (NIPS)*, 2011.

S. Bagon. Matlab wrapper for graph cut, December 2006. `http://www.wisdom.weizmann.ac.il/~bagon`.

M. F. Balcan and A. Blum. Approximation algorithms and online mechanisms for item pricing. *Theory Comput.*, 3(9):179–195, 2007.

A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imaging Sciences*, 2(1), 2009.

A. Billionet and M. Minoux. Maximizing a supermodular pseudo-boolean function: a polynomial algorithm for cubic functions. *Discrete Applied Mathematics*, 12 (1):1–11, 1985.

J. Bilmes. Dynamic graphical models – an overview. *IEEE Signal Processing Magazine*, 27(6):29–42, 2010.

J. Bilmes and C. Bartels. On triangulating dynamic graphical models. In *Uncertainty in Artificial Intelligence (UAI)*, pages 47–56, Acapulco, Mexico, 2003. Morgan Kaufmann Publishers.

A. Blake, C. Rother, M. Brown, P. Perez, and P. Torr. Interactive image segmentation using an adaptive GMMRF model. In *Europ. Conf. on Computer Vision (ECCV)*, 2004.

E. Boros and P. Hammer. Pseudo-boolean optimization. *Discrete Applied Mathematics*, 123(1-3):155–225, 2002.

Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. In *Int. Conf. on Computer Vision (ICCV)*, 2001.

Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, 2004.

Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 23, 2001.

T. Bühler and M. Hein. Spectral clustering based on the graph p-Laplacian. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2009.

T.N. Bui and C. Jones. Finding good approximate edge and vertex partitions is np-hard. *Information Processing Letters*, 42(3):153–159, 1992.

F. Bunke, H. W. Hamacher, F. Maffioli, and A. Schwahn. Minimum cut bases in undirected networks. *Discrete Applied Mathematics*, 158(4), 2010.

G. Calinescu, S. Kapoor, A. Olshevsky, and A. Zelikovsky. Network lifetime and power assignment in ad hoc wireless networks. In *Proc. Europ. Symp. on Algorithms (ESA)*, 2003.

G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák. Maximizing a submodular set function subject to a matroid constraint. *SIAM J. Computing, Special Issue for STOC 2008*, 40(6), 2011.

N. Cesa-Bianchi and G. Lugosi. Combinatorial bandits. In *Conference on Learning Theory (COLT)*, 2009.

A. Chambolle and J. Darbon. On total variation minimization and surface evolution using parametric maximum flows. *Int. Journal of Computer Vision*, 84(3), 2009.

V. Chandrasekaran, N. Srebro, and P. Harsha. Complexity of inference in graphical models. In *Uncertainty in Artificial Intelligence (UAI)*, 2008.

G. Charpiat. Exhaustive family of energies minimizable exactly by a graph cut. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2011.

C. Chekuri and A. Ene. Approximation algorithms for submodular multiway partition. In *Proc. IEEE Symp. on Foundations of Computer Science (FOCS)*, 2011a.

C. Chekuri and A. Ene. Submodular cost allocation problems and applications. In *Int. Colloquium on Automata, Languages and Programming (ICALP)*, 2011b.

C. Chekuri, J. Vondrák, and R. Zenclusen. Dependent randomized rounding via exchange properties of combinatorial structures. In *Proc. IEEE Symp. on Foundations of Computer Science (FOCS)*, 2010.

T. S. Cho, N. Joshi, C. L. Zitnick, S. B. Kang, R. Szeliski, and W. Freeman. A content-aware image prior. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2010.

L Choquet. Theory of capacities. *Ann. Inst. Fournier Grenoble*, 5, 1955.

F. A. Chudak and K. Nagano. Efficient solutions to relaxations of combinatorial problems with submodular penalties via the Lovász extension and non-smooth convex optimization. In *Proc. SIAM-ACM Symp. on Discrete Algorithms (SODA)*, 2007.

J. Chuzhoy and J.S. Naor. The hardness of metric labeling. In *Proc. IEEE Symp. on Foundations of Computer Science (FOCS)*, 2004.

D. Cohen, M. Cooper, P. Jeavons, and A. Krokhin. Supermodular functions and the complexity of max-csp. *Discrete Applied Mathematics*, 2005.

P. L. Combettes and J.-C. Pesquet. *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, chapter Proximal splitting methods in signal processing, pages 185–212. Springer, 2011.

M. Conforti and G. Cornuéjols. Submodular set functions, matroids and the greedy algorithm: tight worst-case bounds and some generalizations of the Rado-Edmonds theorem. *Discrete Applied Mathematics*, 7, 1984.

C. Couprie, L. Grady, L. Najman, and H. Talbot. Power watershed: A unifying graph-based optimization framework. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2010.

C. Couprie, L. Grady, H. Talbot, and L. Najman. Combinatorial continuous maximum flow. *SIAM J Imaging*, pages 905–930, 2011.

J. Cousty, G. Bertrand, L. Najman, and M. Couprie. Watershed cuts: Minimum spanning forests and the drop of water principle. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2009.

W. H. Cunningham. Decomposition of submodular functions. *Combinatorica*, 3 (1):53–68, 1983.

W. H. Cunningham. Testing membership in matroid polyhedra. *J. Combinatorial Theory B*, 36:161–188, 1984.

W. H. Cunningham. On submodular function minimization. *Combinatorica*, 3: 185–192, 1985a.

W. H. Cunningham. Minimum cuts, modular functions, and matroid polyhedra. *Networks*, 1985b.

V. Dani, T. P. Hayes, and S. M. Kakade. The price of bandit information for online optimization. In *Advances in Neural Information Processing Systems (NIPS)*, 2008.

G.B. Dantzig and D.R. Fulkerson. On the max flow min cut theorem of networks. Technical Report P-826, The RAND Corporation, 1955.

G.B. Dantzig and D.R. Fulkerson. *Linear Inequalities and related systems*, chapter On the max-flow min-cut theorem of networks. Princeton, 1956.

A. Das and D. Kempe. Submodular meets spectral: Greedy algorithms for subset selection, sparse approximation and dictionary selection. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2011.

A. Delong, A. Osokin, H. N. Isack, and Y. Boykov. Fast approximate energy minimization with label costs. *Int. Journal of Computer Vision*, 2011.

J. Edmonds. Minimum partition of a matroid into independent subsets. *Journal of Research of the National Bureau of Standards—B. Mathematics and Physics*, 69B(1–2):67–72, 1965.

J. Edmonds. *Combinatorial Structures and their Applications*, chapter Submodular functions, matroids and certain polyhedra, pages 69–87. Gordon and Breach, 1970.

N. El-Zehiry and L. Grady. Fast global optimization of curvature. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2010.

U. Feige, R. Krauthgamer, and K. Nissim. On cutting a few vertices from a graph. *Discrete Applied Mathematics*, 127:643–649, 2003.

U. Feige, V.S. Mirrokni, and J. Vondrák. Maximizing non-monotone submodular functions. *SIAM Journal on Computing*, 40(4), 2011.

L. Fleischer. Recent progress in submodular function minimization. *Mathematical Programming Society Newsletter*, (64):1–11, 2000.

L.R. Ford and D.R. Fulkerson. Maximal flow through a network. Technical Report P-605, The RAND Corporation, 1954.

L.R. Ford and D.R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956.

A. Frank. *Szubmoduláris függvények a kombinatorikus optimalizálásban (Submodular functions in combinatorial optimization)*. PhD thesis, University of Szeged, 1989.

A. Frank. *Surveys in Combinatorics*, volume 187 of *London Mathematical Society Lecture Note Series*, chapter Applications of submodular functions, pages 85–136. London Mathematical Society, 1993.

D. Freedman and P. Drineas. Energy minimization via graph cuts: Settling what is possible. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2005.

S. Fujishige. Foreword. *Discrete Applied Mathematics*, 131(2):253–254, 2003.

S. Fujishige. *Submodular Functions and Optimization*. Number 58 in Annals of Discrete Mathematics. Elsevier Science, 2nd edition, 2005.

S. Fujishige and S. Isotani. A submodular function minimization algorithm based on the minimum-norm base. *Pacific Journal of Optimization*, 7:3–17, 2011.

S. Fujishige and S. Iwata. Minimizing a submodular function arising from a concave function. *Discrete Applied Mathematics*, 92, 1999.

S. Fujishige and S. B. Patkar. Realization of set functions as cut functions of graphs and hypergraphs. *Discrete Mathematics*, 226:199–210, 2001.

S. Fujishige, T. Hayashi, and S. Isotani. The minimum norm-point algorithm applied to submodular function minimization and linear programming. Technical Report Preprint RIMS-1571, Research Institute for the Mathematical Sciences, Kyoto University, 2006.

G. Gallo and B. Simeone. On the supermodular knapsack problem. *Mathematical Programming*, 45:295–309, 1988.

G. Gallo, M.D. Grigoriadis, and R.E. Tarjan. A fast parametric maximum flow algorithm and applications. *SIAM J Computing*, 18(1), 1989.

M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1(3):237–267, 1976.

I. Giotis and V. Guruswami. Correlation clustering with a fixed number of clusters. *Theory of Computing*, 2:249–266, 2006.

J.J. Godfrey, E.C. Holliman, and J. McDaniel. Switchboard: Telephone speech corpus for research and development. In *Proc. ICASSP*, volume 1, pages 517–520, 1992.

G. Goel, C. Karande, P. Tripati, and L. Wang. Approximability of combinatorial problems with multi-agent submodular cost functions. In *Proc. IEEE Symp. on Foundations of Computer Science (FOCS)*, 2009.

G. Goel, P. Tripathi, and L. Wang. Combinatorial problems with discounted price functions in multi-agent systems. In *Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, 2010.

M. X. Goemans and J. A. Soto. Symmetric submodular function minimization under hereditary family constraints. *arXiv:1007.2140v1*, 2010.

M. X. Goemans, N. J. A. Harvey, R. Kleinberg, and V. S. Mirrokni. On learning submodular functions – a preliminary draft. Unpublished Manuscript, 2008.

M.X. Goemans, N. J. A. Harvey, S. Iwata, and V. S. Mirrokni. Approximating submodular functions everywhere. In *Proc. SIAM-ACM Symp. on Discrete Algorithms (SODA)*, 2009.

D. Golovin and A. Krause. Adaptive submodularity: A new approach to active learning and stochastic optimization. In *Conference on Learning Theory (COLT)*, 2010.

D. Golovin and A. Krause. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *Journal of Artificial Intelligence Research (JAIR)*, 42:427–486, 2011.

V. Goyal and R. Ravi. An FPTAS for minimizing a class of low-rank quasi-concave functions over a convex domain. Technical Report 366, Tepper School of Business, Carnegie Mellon University, 2008.

L. Grady. Random walks for image segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 28(11), 2006.

D. M. Greig, B. T. Porteous, and A. H. Seheult. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society*, 51(2), 1989.

M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid algorithm and its consequences in combinatorial optimization. *Combinatorica*, 1:499–513, 1981.

M. Grötschel, L. Lovász, and A. Schrijver. Corrigendum to the paper "the ellipsoid algorithm and its consequences in combinatorial optimization". *Combinatorica*, 4:291–295, 1984.

M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer, 1988.

A. Guillory and J. A. Bilmes. Simultaneous learning and covering with adversarial noise. In *Proc. Int. Conf. on Machine Learning (ICML)*, Bellevue, Washington, 2011.

J. Hannan. Approximation to Bayes risk in repeated play. In *Contributions to the Theory of Games*, volume III. 1957.

P. Hansen and B. Simeone. A class of quadratic pseudoboolean functions whose maximization is reducible to a network flow problem. Technical Report CORR 79-39, University of Waterloo, 1979.

N. J. A. Harvey. *Matchings, matroids and submodular functions*. PhD thesis, Massachusetts Institute of Technology, 2008.

R. Hassin. Minimum cost flow with set constraints. *Networks*, 12:1–21, 1982.

R. Hassin, J. Monnot, and D. Segev. Approximation algorithms and hardness results for labeled connectivity problems. *Journal of Combinatorial Optimization*, 14(4):437–453, 2007.

E. Hazan and S. Kale. Online submodular minimization. In *Advances in Neural Information Processing Systems (NIPS)*, 2009.

D. P. Helmbold and M. K. Warmuth. Learning permutations with exponential weights. *JMLR*, 10:1705–1736, 2009.

D. Hochbaum. The pseudoflow algorithm: a new algorithm for the maximum flow problem. *Operations Research*, 58(4), 2008.

D. Hochbaum. Submodular problems – approximations and algorithms. *arXiv:1010.1945.v1*, 2010.

D. S. Hochbaum and A. Pathria. The bottleneck graph partitioning problem. *Networks*, 28(4):221–225, 1996.

D. S. Hochbaum and D. B. Shmoys. A unified aproach to approximation algorithms for bottleneck problems. *Journal of the ACM*, 33(3):533–550, 1986.

M. Iri and S. Fujishige. Use of matroid theory in operations research, circuits and systems theory. *Int. Journal of Systems Sciences*, 12(1):27–54, 1981.

H. Ishikawa. Higher order clique reduction in binary cut. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2009.

P. Ivănescu (Hammer). Some network flow problems solved with pseudo-boolean programming. *Operations Research*, 13:388–399, 1965.

S. Iwata and K. Nagano. Submodular function minimization under covering constraints. In *Proc. IEEE Symp. on Foundations of Computer Science (FOCS)*, 2009.

S. Iwata, L. Fleischer, and S. Fujishige. A combinatorial strongly polynomial algorithm for minimizing submodular functions. *Journal of the ACM*, 48:761–777, 2001.

S. Jegelka, H. Lin, and J. Bilmes. On fast approximate submodular minimization. In *Advances in Neural Information Processing Systems (NIPS)*, 2011.

S. Jha, O. Sheyner, and J.M. Wing. Two formal analyses of attack graphs. In *Proc. of the 15th Computer Security Foundations Workshop*, pages 49–63, 2002.

A. Barbero Jimenez and S. Sra. Fast algorithms for total-variation based optimization. Technical Report 194, Max Planck Institute for Biological Cybernetics, 2010.

C. Jordan. Nouvelles observations sur les lignes de faites de et de thalweg. *Comptes Rendus des Séances de l'Académie des Sciences*, 75:1023–1025, 1872.

S. Kakade, A. T. Kalai, and K. Ligett. Playing games with approximation algorithms. *SIAM J. Comput.*, 39(3):1088–1106, 2009.

A.T. Kalai and S. Vempala. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71:26–40, 2005.

D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *Proc. 9th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, 2003.

S. Khot. Ruling out PTAS for graph min-bisection, densest subgraph and bipartite clique. In *Proc. IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 136–145, 2004.

J. Kleinberg and É. Tardos. Approximation algorithms for classification problems with pairwise relationships: Metric labeling and markov random fields. In *Proc. IEEE Symp. on Foundations of Computer Science (FOCS)*, 1999.

T. Kloks. *Treewidth: Computations and Approximations*. Springer, 1994.

P. Kohli and M.P. Kumar. Energy minimization for linear envelope MRFs. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2010.

P. Kohli, M. P. Kumar, and P. Torr. P3 & beyond: solving energies with higher-order cliques. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2007.

P. Kohli, M.P. Kumar, and P.H.S. Torr. P^3 & beyond: Move making algorithms for solving higher order functions. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, pages 1645–1656, 2009a.

P. Kohli, L. Ladický, and P.H.S. Torr. Robust higher order potentials for enforcing label consistency. *International Journal of Computer Vision*, 82(3):302–324, 2009b.

V. Kolmogorov. Minimizing a sum of submodular functions. *Discrete Applied Mathematics*, 2012. accepted.

V. Kolmogorov and C. Rother. Minimizing nonsubmodular functions with graph cuts–a review. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 29 (7):1274–1279, 2007.

V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 26(2):147–159, 2004.

N. Komodakis and N. Paragios. Beyond pairwise energies: efficient optimization of higher-order energies. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2009.

W.M. Koolen, M. K. Warmuth, and J. Kivinen. Hedging structured concepts. In *Conference on Learning Theory (COLT)*, 2010.

C. Koufogiannakis and N. E. Young. Greedy Δ-approximation algorithm for covering with arbitrary constraints and submodular costs. In *Int. Colloquium on Automata, Languages and Programming (ICALP)*, 2009.

A. Krause. Matlab toolbox for submodular function optimization, 2009. `http://www.cs.caltech.edu/~krausea/sfo/`.

A. Krause, B. McMahan, C. Guestrin, and A. Gupta. Robust submodular observation selection. *Journal of Machine Learning Research*, 9:2761–2801, 2008.

A. Krokhin and B. Larose. Maximizing supermodular functions on product lattices, with applications to maximum constraint satisfaction. *SIAM Journal on Discrete Math.*, 22(1):312–328, 2008.

S. O. Krumke and H.-C. Wirth. On the minimum label spanning tree problem. *Information Processing Letters*, 66(2):81–85, 1998.

L. Ladický, C. Russell, P. Kohli, and P.H.S. Torr. Graph cut based inference with co-occurrence statistics. In *Europ. Conf. on Computer Vision (ECCV)*, 2010.

E. L. Lawler and C. U. Martel. Computing maximal "Polymatroidal" network flows. *Mathematics of Operations Research*, 7(3):334–347, 1982.

B. Lehmann, D. J. Lehmann, and N. Nisan. Combinatorial auctions with decreasing marginal cost. *Games and Economic Behavior*, 55:270–296, 2006.

V. Lempitsky, P. Kohli, C. Rother, and T. Sharp. Image segmentation with a bounding box prior. In *Int. Conf. on Computer Vision (ICCV)*, 2009.

V. Lempitsky, C. Rother, S. Roth, and A. Blake. Fusion moves for Markov Random Field optimization. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 32(8):1392–1405, 2010.

H. Lin and J. Bilmes. An application of the submodular principal partition to training data subset selection. In *NIPS workshop on Discrete Optimization in Machine Learning*, 2010.

H. Lin and J. Bilmes. Optimal selection of limited vocabulary speech corpora. In *Proc. Interspeech*, 2011.

L. Lovász. *Mathematical programming – The State of the Art*, chapter Submodular Functions and Convexity, pages 235–257. Springer, 1983.

G. Lugosi, O. Papaspiliopoulos, and G. Stoltz. Online multi-task learning with hard constraints. In *Conference on Learning Theory (COLT)*, 2009.

J. Maxwell. On hills and dales. *Philosophical Magazine*, 4/40:421–427, 1870.

S. T. McCormick. *Handbook on Discrete Optimization*, chapter Submodular Function Minimization, pages 321–391. Elsevier, 2006. updated version 3a (2008).

I. Milis. Task assignment in distributed systems using network flow methods. *Proc. Conf. on Combinatorics and Computer Science (CCS 95)*, pages 396–405, 1996.

S. Mittal and A. Schulz. An FPTAS for optimizing a class of low-rank functions over a polytope. *Mathematical Programming*, 2012.

J. Monnot. The labeled perfect matching in bipartite graphs. *Information Processing Letters*, 96:81–88, 2005.

H. Nagamochi and T. Ibaraki. A note on minimizing submodular functions. *Information Processing Letters*, 67:239–244, 1998.

K. Nagano, Y. Kawahara, and K. Aihara. Size-constrained submodular minimization through minimum norm base. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2011.

M. Narasimhan and J. Bilmes. A submodular-supermodular procedure with applications to discriminative structure learning. In *Uncertainty in Artificial Intelligence (UAI)*, 2005.

M. Narasimhan, N. Jojic, and J. Bilmes. Q-clustering. In *Advances in Neural Information Processing Systems (NIPS)*, Vancouver, Canada, December 2006.

H. Narayanan. *Submodular Functions and Electrical Networks.* Elsevier Science, 1997.

G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular functions - i. *Mathematical Programming*, 14:265–294, 1978.

Y. Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103(1):127–152, 2004.

E. Nikolova. Approximation algorithms for reliable stochastic combinatorial optimization. In *APPROX*, 2010.

J. B. Orlin. A faster strongly polynomial time algorithm for submodular function minimization. *Mathematical Programming*, 118(2):237–251, 2009.

J. G. Oxley. *Matroid Theory.* Oxford University Press, 1992.

C. Papadimitriou and K. Steiglitz. *Combinatorial Optimization.* Dover Publications, 1998.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, 2nd printing edition, 1988.

J.C. Picard and H.D. Ratliff. Minimum cuts and related problems. *Networks*, 5 (4):357–370, 1975.

M. Preissmann and A. Sebő. *Research Trends in Combinatorial Optimization*, chapter Graphic Submodular Function Minimization: A Graphic Approach and Applications, pages 365–385. Springer, 2009.

A. P. Punnen. Traveling salesman problem under categorization. *Operations Research Letters*, 12:89–95, 1992.

A. P. Punnen. On bottleneck assignment problems under categorization. *Computers & Operations Research*, 31:151–154, 2004.

M. Queyranne. Minimizing symmetric submodular functions. *Mathematical Programming*, 82:3–12, 1998.

S. Ramalingam, P. Kohli, K. Alahari, and P. Torr. Exact inference in multi-label crfs with higher order cliques. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2008.

S. Ramalingam, C. Russell, L. Ladicky, and P. H. S. Torr. Efficient minimization of higher order submodular functions using monotonic boolean functions. *arXiv:1109.2304*, 2011.

M. B. Richey and A. P. Punnen. Minimum perfect bipartite matchings and spanning trees under categorization. *Discrete Applied Mathematics*, 39:147–153, 1992.

C. Rother, V. Kolmogorov, and A. Blake. Grabcut – interactive foreground extraction using iterated graph cuts. In *SIGGRAPH*, 2004.

L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D*, (60), 1992.

T. Schoenemann, F. Kahl, and D. Cremers. Curvature regularity for region-based image segmentation and inpainting: A linear programming relaxation. In *Int. Conf. on Computer Vision (ICCV)*, 2009.

A. Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *J. Combin. Theory Ser. B*, 80:346–355, 2000.

A. Schrijver. *Combinatorial Optimization.* Springer, 2004.

C. Seshadri and J. Vondrák. Is submodularity testable? *arXiv:1008.0831v1*, 2010.

C.R. Seshan. Some generalizations of the time minimizing assignment problem. *Journal of the Operational Research Society*, 32:489–494, 1981.

S. Shalev-Shwartz and V. Singh. Online submodular maximization. manuscript, Hebrew University, 2011.

R. Shamir, R. Sharan, and D. Tsur. Cluster graph modification problems. In *Proc. 28th Workshop on Graph Theory (WG '02)*, pages 379–390, 2002.

L. S. Shapley. Cores of convex games. *International Journal of Game Theory*, 1 (1):11–26, 1971.

A.K. Sinop and L. Grady. A seeded image segmentation framework unifying graph cuts and random walker which yields a new algorithm. In *Int. Conf. on Computer Vision (ICCV)*, 2007.

R. P. Stanley. *Enumerative Combinatorics*, volume I of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, 1997.

P. Stobbe and A. Krause. Efficient minimization of decomposable submodular functions. In *Advances in Neural Information Processing Systems (NIPS)*, 2010.

G. Strang. l^1 and l^∞ approximation of vector fields in the plane. In *Nonlinear Partial Differential Equations in Applied Science; Proceedings of the U.S. - Japan Seminar*, volume 5 of *Lecture Notes in Num. Appl. Anal.*, pages 273–288, 1982.

G. Strang. Maximum flows through a domain. *Mathematical Programming*, pages 123–143, 1983.

G. Strang. Maximum flows and minimum cuts in the plane. *Journal of global optimization*, 2008.

M. Streeter and D. Golovin. An online algorithm for maximizing submodular functions. In *Advances in Neural Information Processing Systems (NIPS)*, 2008.

M. Streeter, D. Golovin, and A. Krause. Online learning of assignments. In *Advances in Neural Information Processing Systems (NIPS)*, 2009.

Z. Svitkina and L. Fleischer. Submodular approximation: Sampling-based algorithms and lower bounds. In *Proc. IEEE Symp. on Foundations of Computer Science (FOCS)*, 2008.

E. Tardos, C. A. Tovey, and M. A. Trick. Layered augmenting path algorithms. *Mathematics of Operations Research*, 11(2), 1986.

S. Vicente, V. Kolmogorov, and C. Rother. Graph cut based image segmentation with connectivity priors. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2008.

B. von Hohenbalken. A finite algorithm to maximize certain pseudoconcave functions on polytopes. *Mathematical Programming*, 8, 1975.

U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17, 2007.

J. Vondrák. *Submodularity in Combinatorial Optimization.* PhD thesis, Charles University, 2007.

J. Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. In *Symposium on Theory of Computing (STOC)*, 2008a.

J. Vondrák. Submodularity and curvature: the optimal algorithm. *RIMS Kôkyûroku Bessatsu*, 2008b.

J. Vondrák. email communication, 2010.

J. Vondrák. Symmetry and approximability of submodular maximization problems. *arXiv:1110.4860v1*, 2011.

D. Wagner and F. Wagner. Between min cut and graph bisection. In *Proc. IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 744–750, 1993.

P.-J. Wan, G. Călinescu, X.-Y. Li, and O. Frieder. Minimum-energy broadcasting in static ad hoc wireless networks. *Wireless networks*, 2002.

P.-J. Wan, D.-Z. Du, P. Pardalos, and W. Wu. Greedy approximations for minimum submodular cover with submodular cost. *Comput Optim Appl*, 45:463–474, 2010.

D.J.A. Welsh. *Matroid Theory.* Academic Press, 1976.

H. Whitney. On the abstract properties of linear dependence. *American Journal of Mathematics*, 57(3):509–533, 1935.

J.E. Wieselthier, G.D. Nguyen, and A. Ephremides. On the construction of energy-efficient broadcast and multicast trees in wireless network. In *IEEE Infocom*, 2000.

P. Wolfe. Finding the nearest point in a polytope. *Mathematical Programming*, 11: 128–149, 1976.

L. Wolsey. An analysis of the greedy set cover algorithm for the submodular set covering problem. *Combinatorica*, 2(4), 1982.

S. Yuan, S. Varma, and J. P. Jue. Minimum-color path problems for reliability in mesh networks. In *IEEE Infocom*, 2005.

B. Zalesky. Efficient determination of Gibbs estimators with submodular energy functions. *arXiv:math/0304041*, 2003.

P. Zhang, Cai J.-Y, L.-Q. Tang, and W.-B. Zhao. Approximation and hardness results for label cut and related problems. *Journal of Combinatorial Optimization*, 2011.

M. Zinkevich. Online convex programming and infinitesimal gradient ascent. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2003.

S. Živný and P.G. Jeavons. Classes of submodular constraints expressible by graph cuts. *Constraints*, 15:430–452, 2010. ISSN 1383-7133.

S. Živný, D. A. Cohen, and P. G. Jeavons. The expressive power of binary submodular functions. *Discrete Applied Mathematics*, 157(15):3347–3358, 2009.