Cut Pursuit: fast algorithms to learn piecewise constant functions on general weighted graphs

Loic Landrieu, Guillaume Obozinski

Cut Pursuit: fast algorithms to learn piecewise constant functions on general weighted graphs

Landrieu Loic[1,2] and Guillaume Obozinski[2]

[1]Université Paris-Est, IGN, MATIS, F-94160 Saint-Mandé, France
[2]Université Paris-Est, LIGM*, Ecole des Ponts, F-77454 Marne-la-Vallée, France
loic.landrieu@ign.fr, guillaume.obozinski@enpc.fr

April 25, 2016

Abstract

We propose working-set/greedy algorithms to efficiently solve problems penalized respectively by the total variation on a general weighted graph and its ℓ_0 counterpart the Mumford Shah total level-set boundary size when the piecewise constant solutions have a small number of distinct level-sets; this is typically the case when the total level-set boundary size is small, which is encouraged by these two forms of penalization. Our algorithms exploit this structure by recursively splitting the level-sets of a piecewise-constant candidate solution using graph cuts. We obtain significant speed-ups over state-of-the-art algorithms for images that are well approximated with few level-sets

1 Introduction

Estimation or approximation with piecewise constant functions has many applications in image and signal processing, machine learning and statistics. In particular, the assumption that natural images are well modeled by functions whose total variation is bounded motivates its use as a regularizer, which leads to piecewise constant images for discrete approximations. Moreover a number of models used in medical imaging (El-Zehiry and Elmaghraby, 2007) assume directly piecewise constant images. More generally piecewise constant models can be used for compression, for their interpretability and finally because they are typically adaptive to the local regularity of the function approximated (Wang et al., 2014). Piecewise constant functions display a form of structured sparsity since their gradient is sparse.

Both convex and non-convex formulations have been proposed to learn functions with sparse gradients. The most famous being the formulation of Rudin et al. (1992), hereafter referred to as ROF, who proposed to minimize the total variation subject to constraints of approximation of the noisy signal in the least squares sense, as well as the formulation of Mumford and Shah (Mumford and Shah, 1989) who proposed to penalize the total length of discontinuities of piecewise smooth functions. A fairly large literature is devoted to these formulations mainly in the image processing and optimization literature. Although the connection between the total variation, the Mumford-Shah energy and graph cuts is today well-established, algorithms that leverage this connection are relatively recent. In particular for ROF, Chambolle and Darbon (2009); Goldfarb and Yin (2009) use the fact that the problem can be formulated as a parametric max-flow. El-Zehiry and Grady (2011) use graph cuts to solve the formulation of Mumford and Shah for the case of two components.

The literature on sparsity in computational statistics and machine learning has shown how the sparsity of the solution sought can be exploited to design algorithms which use parsimonious computations to solve

*Laboratoire d'Informatique Gaspard Monge (UMR 8049), CNRS, ENPC, ESIEE Paris, UPEM.

the corresponding large-scale optimization problem with significant speed-ups (Bach et al., 2012). Our work is motivated by the fact that this has to the best of our knowledge not been fully leveraged to estimate and optimize with piecewise constant functions. In the convex case, the algorithm proposed to exploit sparsity are working set[1] algorithms and the related (fully corrective) Frank-Wolfe algorithm (Harchaoui et al., 2015). In the non-convex case, forward selection algorithms such as OMP, FoBa and others have been proposed (Mallat and Zhang, 1992; Needell and Tropp, 2009; Zhang, 2009)[2].

It is well understood that algorithms for the convex and non-convex case are in fact fairly related. In particular, for a given type of sparsity the forward step of working set methods, Frank-Wolfe and greedy algorithm is typically the same, and followed by the resolution of a reduced problem.

Given their similarity, we explore in this paper both greedy and working set strategies. The working set approach is used to solve optimization problem regularized by the total variation while the greedy strategy solves problems penalized by the Mumford-Shah penalty for piecewise constant functions (aka minimal partition problems). In the convex case, our algorithms do not apply only to the case where the data fitting term is the MSE or a separable smooth convex function, for which some efficient algorithms implicitly exploiting sparsity exist (Bach, 2013; Chambolle and Darbon, 2009; Kumar and Bach, 2015), but also to a general smooth convex term.

Our algorithms are very competitive for deblurring and are applicable to the estimation of piecewise constant functions on general weighted graphs.

1.1 Notations

Let $G = (V, E, w)$ be an unoriented weighted graph whose edge set is of cardinality m and $V = [1, \cdots, n]$. For convenience of notations and proofs, we encode the undirected graph G, as a directed graph with for each pair of connected nodes a directed edge in each direction. Thus E denotes a collection of couples (i, j) of nodes, with $(i, j) \in E$ if and only if $(j, i) \in E$. We also have $w \in \mathbb{R}^{2m}$ and $w_{ij} = w_{ji}$. For a set of nodes $A \subset V$ we denote $\mathbf{1}_A$ the vector of $\{0, 1\}^n$ such that $[\mathbf{1}_A]_i = 1$ if and only if $i \in A$. For $F \subset E$ a subset of edges we denote $w(F) = \sum_{(i,j)\in F} w_{ij}$. By extension, for two subsets A and B of V we denote $w(A, B) = w\big((A \times B) \cap E\big)$ the weight of the boundary between those two subsets. Finally we denote $\mathcal{C}$ the set of all partitions of V into connected components.

1.2 General problem considered

1.2.1 Problem formulation

In this work we consider the problem of minimizing functions Q of the form $f(x) + \lambda\Phi(x)$ with $f : \mathbb{R}^n \to \mathbb{R}$ differentiable and $\Phi : \mathbb{R}^n \to \mathbb{R}$ a penalty function that decomposes as $\Phi(x) = \sum_{(i,j)\in E} w_{ij}\, \phi(x_i - x_j)$ with $\phi : \mathbb{R} \to \mathbb{R}^+$ a sparsity inducing function such that $\phi(0) = 0$. The general problem writes $\min_{x\in\mathbb{R}^n} Q(x)$ with

$$Q(x) \doteq f(x) + \frac{\lambda}{2} \sum_{(i,j)\in E} w_{ij}\, \phi(x_i - x_j). \tag{1}$$

Energies of this form were first introduced by Geman and Reynolds (1992) for image regularization, and are widely used for their inducing spatial regularity as well as preserving discontinuities. The function ϕ is typically the absolute value, which corresponds to the total variation, or one minus the Kronecker delta at 0, which leads to the Mumford-Shah penalty for piecewise constant functions. More generally, for functions

[1] We distinguish *working set* algorithms (aka column generation algorithm) that maintain an expansion of the solution which may have zero coefficients from *active set* algorithms that maintain an expansion using only non-zero coefficients and discard all other directions (or variables). This distinction can also be understood in the dual, where working set algorithms (which are dually cutting plane algorithms) maintain a superset of the active constraints, while active set algorithms maintain the exact set of active constraints.

[2] Proximal methods that perform soft-thresholding or the non-convex IHT methods maintain sparse solutions, but typically need to update a full dimensional vector at each iteration, which is why we do not cite them here. They blend however very well with active set algorithms.

ϕ that have a non-differentiability at 0, the solution $x^\star$ of (1) has a sparse gradient $\{x_i^\star - x_j^\star \mid (i,j) \in E\}$. As a consequence, these solutions are constant on the elements of a certain partition of V that is typically coarse, i.e. such that has much fewer elements than $|V|$. We therefore reformulate the problem for candidate solutions that have that property. We define the *support* of a vector $x \in \mathbb{R}^n$ as the set $S(x)$ of edges supporting its gradients

$$S(x) \doteq \{(i,j) \in E \mid x_i \neq x_j\}, \tag{2}$$

and we will use $S^c(x) \doteq E \backslash S(x)$ for the set on which the gradients are zero.

In the general case the approach presented in Section 2 can be easily adapated to functions ϕ that are differentiable in $\mathbb{R} \setminus \{0\}$, are decreasing on $\mathbb{R}^-$, non-decreasing on $\mathbb{R}^+$ and such that $\lim_{h \to 0, h>0} \phi'(h) > 0$ and $\lim_{h \to 0, h<0} \phi'(h) < 0$. We will limit our scope however to the absolute value.

1.2.2 Decomposition on a partition

Any $x \in \mathbb{R}^n$ can be written as $x = \sum_{i=1}^k c_i \mathbf{1}_{A_i}$ with $\Pi = \{A_1, \cdots, A_k\} \in \mathcal{C}$ a partition of V into k connected components and $c \in \mathbb{R}^k$. Conversely we say that x can be expressed by partition $\Pi = (A_1, \cdots, A_k)$ if it is in the set $\operatorname{span}(\Pi) = \operatorname{span}(\mathbf{1}_{A_1}, \cdots, \mathbf{1}_{A_k}) = \{\sum_{i=1}^k c_i \mathbf{1}_{A_i} \mid c \in \mathbb{R}^k\}$. We denote

$$x_\Pi \doteq \operatorname*{arg\,min}_{z \in \operatorname{span}(\Pi)} Q(z), \tag{3}$$

the solution of (1) when x is constrained to be in $\operatorname{span}(\Pi)$. With this notation, we can rewrite problem (1) as the problem of finding an optimal partition $\Pi^\star$:

$$\Pi^\star \doteq \operatorname*{arg\,min}_{\Pi \in \mathcal{C}} Q(x_\Pi). \tag{4}$$

An additional motivation to consider a sequence of partitions and solve sequentially problems with x constrained to $\operatorname{span}(\Pi)$ is that the vectors of the form $w(B, B^c)^{-1}\mathbf{1}_B$ are extreme points of the set $\{x | \mathrm{TV}(x) \leq 1\}$. In fact, the total variation is an *atomic gauge* in the sense of Chandrasekaran et al. (2012) and the vectors of the form $w(B, B^c)^{-1}\mathbf{1}_B$ are among the *atoms* of the gauge. We do not develop this more abstract point of view in the paper, but provide a discussion in appendix A.

Before presenting our approach we review some of the main relevant ideas in the related literature.

1.3 Related work

Mumford and Shah (1989) describe an image as *simple* if it can be expressed as a piecewise-smooth function with few and small discontinuities, i.e. if the space can partitioned in regions with short contours and such that the image varies little in each of these regions. Given an observed noisy image viewed as a function $J : \mathbb{R}^2 \mapsto \mathbb{R}$, Mumford and Shah therefore propose to recover the original image $I : \mathbb{R}^2 \mapsto \mathbb{R}$ via the minimization of an energy composed of three terms: a fidelity term quantifying the distortion between I and J, a part evaluating the smoothness of I outside of the one dimensional boundary Γ, and finally the length of this boundary, as follows:

$$\min_{I,\Gamma} \int_\Omega \big(I(x) - J(x)\big)^2 dx + \mu \int_{\Omega \setminus \Gamma} \|\nabla I(x)\|^2 dx + \lambda \int_\Gamma dl. \tag{MS}$$

μ and λ are two nonnegative regularization coefficients. When $\mu \to \infty$, the smoothness term forces the function to be infinitely smooth outside of the boundary, i.e. constant on each set R_i of a collection $\Pi = \{R_i\}_{i=1}^k$ of disjoint connected regions. This special instance of the Mumford-Shah problem is also called the *minimal partition problem* (Santner et al., 2011) and can be reformulated as

$$\min_{\Pi, I, k} \sum_{i=1}^k \int_{R_i} \big(I_i - J(x)\big)^2 dx + \lambda \operatorname{length}(\Pi), \tag{MPP}$$

with I_i the constant value of I on R_i and length(Π) the total length of the boundaries between pairs of sets in Π. The setting in which the number of regions k is fixed a priori, typically with $k = 2$, is known as the Chan-Vese problem and was first solved using active contour methods (Kass et al., 1988). Chan and Vese (2001) propose a level-set based method for the binary case, which has the advantage of foregoing edges and gradient completely, as they are typically very sensitive to noise. This method has since been extended to the so called *multiphase* setting where the number of *phases*, that is of level-sets of the function, is a power of two (Vese and Chan, 2002). The resolution of those problems is substantially sped up by the introduction of graph-cut methods, for binary phase (El-Zehiry and Elmaghraby, 2007) and in the multiphase setting (El-Zehiry and Grady, 2011).

Independently of the work of Mumford and Shah, Rudin, Osher and Fatemi proposed in Rudin et al. (1992) the idea that the class of functions with bounded variation is a good model for images, and relied on this idea to motivate the minimization of the total variation under MSE approximation constraint as an approach for image denoising. The introduction of the total variation had a lasting impact in imaging sciences and was used for various tasks including denoising, deblurring and segmentation (Chambolle et al., 2010). The ROF problem can today be viewed as a convex relaxation of the MPP problem. When the total variation is used as a regularizer[3], the ROF problem can be formulated as

$$\min_{I \in \mathrm{BV}} \int_\Omega \big(I(x) - J(x)\big)^2 dx + \lambda \mathrm{TV}(I), \tag{ROF}$$

where BV is the space of functions with bounded total variation.

In this paper we consider discretized versions of these formulations, in which the function takes its value on the node set of a weighted graph $G = (V, E, w)$. Such discretizations are for example naturally obtained if an a priori fine grained partition of the space in a collection of elementary regions[4] $\mathcal{R}_0$ is chosen and the image or function I is constrained to be constant on each of these regions. The edge set E captures adjacencies between the elements, and the weights w the size of the boundary between each pair of regions.

A first approach to minimizing functions regularized by the total variation is to consider explicitely the set of edges presenting discontinuities and iteratively update this set using calculus of variations based on the Euler-Lagrange equations (Aubert et al., 2003). This class of methods is known as *active contours*.

The level-sets approach (Osher and Sethian, 1988; Tsai and Osher, 2005) takes an opposite point of view and defines the discontinuity set as the zero set of an auxiliary function. This allows to indirectly handle continuously the evolution of the curve, thereby avoiding complications associated to making discrete changes in the structure of the contours.

In the recent literature, problem regularized with the total variation are typically solved using proximal splitting algorithms (Chambolle and Pock, 2011; Raguet et al., 2013).

Some of the connections between graph-cuts and the total variation were already known in Picard and Ratliff (1975) but some of these connections have been only fully exploited recently, when Chambolle and Darbon (2009) and Goldfarb and Yin (2009) among others, exploited the fact that the ROF model can be reformulated as a parametric maximum flow problem, which they moreover show can be solved by a divide-and-conquer strategy: This algorithm requires to solve a sequence of max-flow problems on the same graph, and the algorithm makes it possible to efficiently reuse partial computations performed in each max-flow problem with a push-relabel algorithm. These results on the total variation are actually an instance of results that apply more generally to submodular functions (Bach, 2013). Indeed, the intimate relation existing between the total variation and graph-cuts is due fundamentally to the fact that the former is the Lovász extension of the value of the cut, which is a submodular function. Beyond the case of the total variation, Bach (2011) considers regularizers that are obtained as Lovász extensions of symmetric submodular functions and recent progress made on the efficient optimization of submodular functions produces simultaneously new fast algorithms to compute proximal operators of the Lovász extension of submodular function (Jegelka et al., 2013; Kumar and Bach, 2015).

[3] In Rudin et al. (1992) the TV(I) is minimized under a constraint on the L_2 distance between I and J.

[4] In the context of images these could be though of as super-pixels, for example.

Problems regularized by the total variation or the Mumford-Shah energy are also related to the Potts model. Indeed, if the values of the level-set are quantized, the corresponding energy to minimize is that of a discrete valued conditional random field (CRF), with as many values as there are quantization levels (Ishikawa, 2003; Tsai and Osher, 2005). A number of optimization techniques exist for CRFs (Szeliski et al., 2006). One of the fastest is the α-expansion algorithm of Boykov et al. (2001b), which is relying on graph-cut algorithms (Boykov and Kolmogorov, 2004).

In the literature on sparsity, a number of algorithms have been proposed to take advantage computationally of the sparsity of the solution. In the convex setting, these algorithms count homotopy algorithms such as the LARS (Efron et al., 2004) or working set algorithms (Friedman et al., 2010; Obozinski et al., 2006; Roth and Fischer, 2008). It should be noted that the Frank-Wolfe algorithm (Jaggi, 2013), which has been revived and regained popularity in recent years, is closely related to working set methods and also provides a rationale to algorithmically exploit the sparsity of solution of optimization problems. Although originally designed to solve constrained optimization problems, Harchaoui et al. (2015) have shown how a variant can be naturally constructed for the regularized setting, and apply it to the case of total variation regularization. The counterparts of these algorithms in the ℓ_0 setting are (a) greedy forward selection approaches that compute a sequence of candidate solutions by iteratively decreasing the sparsity of the candidate solutions, such as orthogonal matching pursuit (Mallat and Zhang, 1992), orthogonal least squares (Chen et al., 1991) and related algorithms (Needell and Tropp, 2009), (b) forward-backward selection approaches such as the Single Best Replacement (SBR) algorithm (Soussen et al., 2011), based on an ℓ_0 penalization or the FoBa algorithm (Zhang, 2009), which add backwards steps to remove previously introduced variables that are no longer relevant. See (Bach et al., 2012) for a review. Bach (2013) proposes a number of algorithms to minimize submodular functions, compute the associated proximal operators of the corresponding Lovász extensions. In particular, generic primal and dual active set algorithms are proposed to solve a linear regression problem regularized with the Lovász extension of a submodular function (Bach, 2013, Chap. 7.12).

2 A working set algorithm for total variation regularization

In this section, we consider the problem of solving the minimization of a differentiable function f regularized by a weighted total variation of the form $\mathrm{TV}(x) = \frac{1}{2}\sum_{(i,j)\in E} w_{ij}\,|x_i - x_j|$ with w_{ij} some nonnegative weights. Based on the considerations of Section 1.2.2, we propose a working set algorithm which alternates between solving a reduced problem of the form $\min_{x\in\mathrm{span}(\Pi)} Q(x)$ for $Q(x) = f(x) + \lambda \mathrm{TV}(x)$, and refining the partition Π. We will discuss in Section 2.3 how to solve the reduced problem efficiently, but first present a criterion to refine the partition Π.

2.1 Steepest binary cut

Given a current partition Π and the solution of the associated reduced problem $x_\Pi = \arg\min_{x\in\mathrm{span}(\Pi)} Q(x)$, our goal is to compute a finer partition Π_{new} leading to the largest possible decrease of Q. To this end we consider updates of x of the form $x_\Pi + h\,u_B$ with $u_B = \gamma_B \mathbf{1}_B - \gamma_{B^c}\mathbf{1}_{B^c}$ for some set $B \subset V$ and some scalars h, γ_B and γ_{B^c} such that $\|u_B\|_2 = 1$. We postpone to Section 2.2 the precise discussion of how the choice of B leads to a new partition and focus first on a rationale to choose B, but essentially, introducing u_B in the expansion of x will lead to a new partition in which the elements of Π are split along the boundary between B and B^c. A natural criterion is to choose the set B such that u_B is a descent direction which is as steep as possible, in the sense that Q decreases the most, at first order. We denote $Q'(x, v) = \lim_{h\to 0} h^{-1}(Q(x + hv) - Q(x))$ so that, when $d \in \mathbb{R}^n$ is a unit vector, $Q'(x, d)$ denotes the directional derivative of Q at $x \in \mathbb{R}^n$ in the direction d. Consequently, choosing B for which the direction u_B is steepest requires solving $\min_{B\subset V} Q'(x_\Pi, u_B)$.

To further characterize Q' we decompose the objective function: Since the absolute value is differentiable on $\mathbb{R}_*$, setting $S \doteq S(x_\Pi)$ allows us to split Q into two parts Q_S and $\mathrm{TV}|_{S^c}$ which are respectively

differentiable and non-differentiable at x_Π:

$$\begin{cases} Q_S(x) & \doteq f(x) + \frac{\lambda}{2} \sum_{(i,j)\in S} w_{ij} \, |x_i - x_j|, \\ \mathrm{TV}|_{S^c}(x) & \doteq \frac{\lambda}{2} \sum_{(i,j)\in S^c} w_{ij} \, |x_i - x_j|. \end{cases}$$

$\mathrm{TV}|_{S^c}$ is a weighted total variation on the graph G but with weights w_{S^c} such that $[w_{S^c}]_{i,j} \doteq w_{ij}$ for $(i,j) \in S^c$ and 0 for $(i,j) \in S$. We extend the previous notations and define $w_{S^c}(A,B) \doteq w_{S^c}(A \times B) = w((A \times B) \cap S^c)$.

Proposition 1. *For $x \in \mathbb{R}^n$, if we set $S = S(x)$ then the directional derivative in the direction of $\mathbf{1}_B$ is*

$$Q'(x, \mathbf{1}_B) = \langle \nabla Q_S(x), \mathbf{1}_B \rangle + \lambda w_{S^c}(B, B^c).$$

Moreover if $\langle \nabla f(x), \mathbf{1}_B \rangle = 0$ then

$$Q'(x, u_B) = (\gamma_B + \gamma_{B^c}) \, Q'(x, \mathbf{1}_B).$$

Proof. See appendix B. □

Considering the case $x = x_\Pi$, then for $S = S(x_\Pi)$, $\nabla Q_S(x_\Pi)$ is clearly orthogonal to $\mathrm{span}(\Pi)$ and thus to $\mathbf{1}$. Therefore, by the previous proposition, finding the steepest descent direction of the form u_B requires solving

$$\min_{B \subset V} (\gamma_B + \gamma_{B^c}) \, Q'(x_\Pi, \mathbf{1}_B)$$

To keep a formulation which remains amenable to efficient computations, we will assume that $\gamma_B + \gamma_{B^c}$ is constant or ignore this factor[5]. This leads us to define a *steepest binary cut* as any cut (B_Π, B_Π^c) such that

$$B_\Pi \in \arg\min_{B \subset V} \langle \nabla Q_S(x_\Pi), \mathbf{1}_B \rangle + \lambda w_{S^c}(B, B^c). \tag{5}$$

Note that since $Q'(x, \mathbf{1}_\varnothing) = 0$, we have $\min_{B \subset V} Q'(x, \mathbf{1}_B) \leq 0$. If $\varnothing$ is a solution to (5), we set $B_\Pi = \varnothing$. As formulated, it well known at least since Picard and Ratliff (1975) that problem (5) can be interpreted as a minimum cut problem in a suitably defined flow graph. Indeed consider the graph $G_{flow} = (V \cup \{s,t\}, E_{flow})$ illustrated in Figure 1, where s and t are respectively a source and sink nodes, and where the edge set E_{flow} and the associated nonzero (undirected) capacities $c \in \mathbb{R}^{|S^c|+n}$ are defined as follows

$$E_{flow} = \begin{cases} (s,i), \forall i \in \nabla_+, & \text{with} \quad c_{si} = \nabla_i Q_S(x) \,, \\ (i,t), \forall i \in \nabla_-, & \text{with} \quad c_{it} = -\nabla_i Q_S(x) \,, \\ (i,j), \forall (i,j) \in S^c, & \text{with} \quad c_{ij} = \lambda w_{ij}, \end{cases} \tag{6}$$

where $\nabla_+ \doteq \{ i \in V \mid \nabla_i Q_S(x) > 0 \}$ and $\nabla_- \doteq V \backslash \nabla_+$. The vector $\nabla Q_S(x)$ is directly computed as $\nabla Q_S(x) = \nabla f(x) + \frac{1}{2} \lambda D^\top y$, with $D \in \mathbb{R}^{2m \times n}$ the weighted edge incidence matrix whose entries are equal to $D_{(i,j),k} \doteq w_{ij}(1_{\{i=k\}} - 1_{\{j=k\}})$ and $y \in \mathbb{R}^{2m}$ is the vector whose entries are indexed by the elements of E and such that $y_{(i,j)} \doteq \mathrm{sign}(x_i - x_j)$ with the convention that $\mathrm{sign}(0) = 0$.

As stated in the next proposition, finding a minimal cut in this graph provides us with the desired steepest binary cut.

Proposition 2. *Let $S = S(x)$ then $(C, V_{flow} \backslash C)$ is a minimal cut in G_{flow} if and only if $C \backslash \{s\}$, and its complement in V are minimizers of $B \mapsto Q'(x, \mathbf{1}_B)$.*

[5] γ_B and γ_{B^c} could otherwise be determined by requiring that $\langle \mathbf{1}, u_B \rangle = 0$. More rigorously, descent directions considered could be required to be orthogonal to $\mathrm{span}(\Pi)$, but this leads to even less tractable formulations, that we therefore do not consider here.

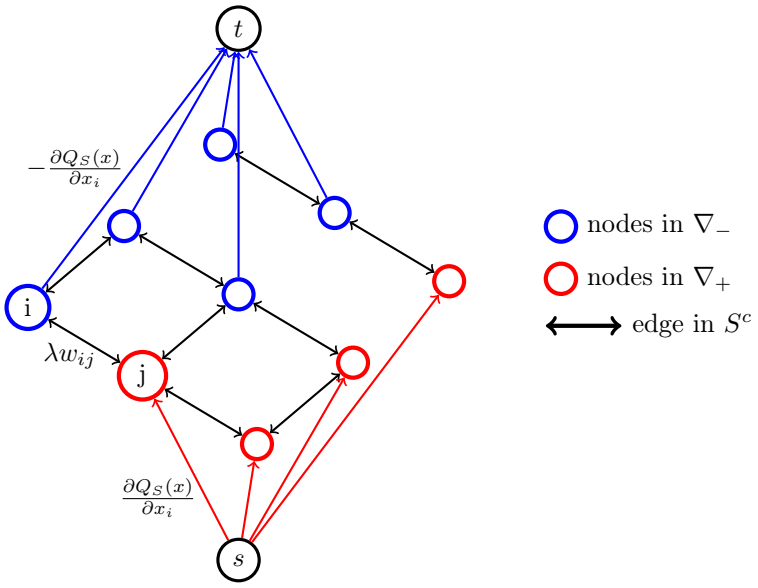


Figure 1: Directed graph for which finding a maximal flow is equivalent to solving (5). Neighboring nodes with different values of x in the original graph are linked by an undirected edge with capacity λw_{ij}, nodes with non-negative gradient are linked to the source, and nodes with negative gradient to the sink with capacity $|\nabla Q_S(x)|$.

This result is a well-know result that was first discussed in Picard and Ratliff (1975). We refer the reader to Kolmogorov and Zabih (2004) for a proof.

Note that the min-cut/max-flow problem of Figure 1 decouples on each of the connected components of the graph $G|_{S^c} \doteq (V, S^c)$ and that as a result solving (5) is equivalent to solving separately

$$\min_{C \subset A} \langle \nabla Q_S(x_\Pi), 1_C \rangle + \lambda w(C, A \backslash C)$$

for each set A that is a connected components of $G|_{S^c}$. The binary steepest cut thus actually reduces to computing a steep cut in each connected component of the graph, and they can all be computed in parallel. Let us insist that the connected components of $G|_{S^c}$ are often but not always the elements of Π since they can be unions of adjacent elements of Π when they share the same value.

We can now characterize the optimality of x_Π or of the corresponding partition Π, based on the value of the steepest binary partition:

Proposition 3. *We have* $x = \arg\min_{z \in \mathbb{R}^n} Q(z)$ *if and only if* $\min_{B \subset V} Q'(x, \mathbf{1}_B) = 0$ *and* $Q'(x, \mathbf{1}_V) = 0$.

Proof. See appendix B □

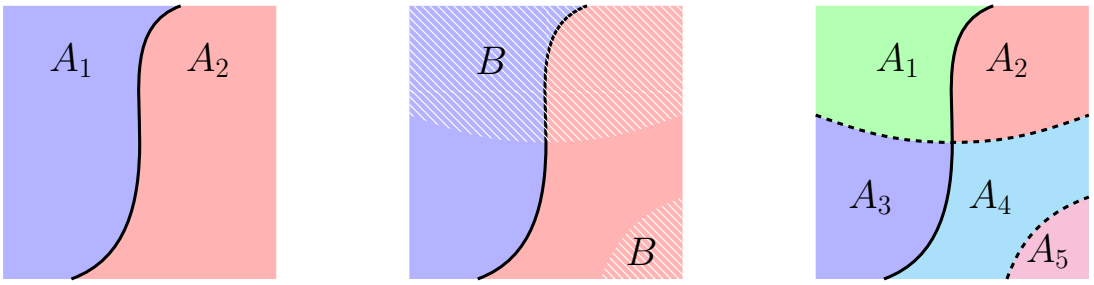
Note that the rationale we propose to choose the new direction $\mathbf{1}_B$ is different than the one typically used for working set algorithms in the sparsity literature and variants of Frank-Wolfe. When considering the minimization of an objective of the form $f(x) + \lambda\Omega(x)$, where $f : \mathbb{R}^n \to \mathbb{R}$ is a differentiable function and Ω is a norm, the optimality condition in terms of subgradient is $-\frac{1}{\lambda}\nabla f(x) \in \partial\Omega(x)$, where $\partial\Omega(x)$ is the subgradient of the norm Ω at x. A classical result from convex analysis is that $\partial\Omega(x) = \{s \in \mathbb{R}^n \mid \langle s, x \rangle = \Omega(x) \text{ and } \Omega^\circ(s) \leq 1\}$ where Ω° denotes the dual norm (Rockafellar, 1970, Thm. 23.5). In particular, the subgradient condition is not satisfied if $\Omega^\circ(-\nabla f(x)) \geq \lambda$ and since $\Omega^\circ(s) = \max_{\Omega(\xi) \leq 1} \langle s, \xi \rangle$ then $\operatorname{argmax}_{\Omega(\xi) \leq 1} \langle -\nabla f(x), \xi \rangle$ provides a direction in which the inequality constraint is most violated. This

direction is the same as the Frank-Wolfe direction for the optimization problem $\min_{x:\Omega(x)\leq\kappa} f(x)$, also the same as the direction proposed in a variant of the Frank-Wolfe algorithm proposed by Harchaoui et al. (2015) for the regularized problem, and again the same as the direction that would be used in the primal active set algorithm of Bach (2013, Chap. 7.12) for generic Lovász extensions of submodular function, which is essentially a fully corrective and active-set version of the algorithm of Harchaoui et al. (2015). This rationale extends to the case where Ω is more generally a gauge and is most relevant when it is an atomic norm or gauge (Chandrasekaran et al., 2012), which we discuss in appendix A. For decomposable atomic norms (Negahban et al., 2009) that have atoms of equal Euclidean norm, one can check that the steepest descent direction that we propose and the Franck-Wolfe direction are actually the same. However, for the the total variation the two differ. The Frank-Wolfe direction leads to the choice $B^\star = \arg\max_{B\subset V} -w(B, B^c)^{-1}\langle \nabla f(x_\Pi), \mathbf{1}_B\rangle$. We show in Section 4.1 and via results presented in Figure 6 that using the steepest cut direction outperforms the Frank-Wolfe direction.

2.2 Induced new partition in connected sets and new reduced problem

For $\Pi = (A_1, \cdots, A_k)$, B_Π is chosen so that the addition of a term of the form $h\, u_B = h\gamma_B \mathbf{1}_B - h\gamma_{B^c}\mathbf{1}_{B^c}$ to $x = \sum_{i=1}^k c_i \mathbf{1}_{A_i}$ decreases the objective function Q the most. At the next iteration, we could thus consider solving a reduced problem that consists of minimizing Q under the constraint that $x \in \operatorname{span}(\mathbf{1}_{A_1}, \ldots, \mathbf{1}_{A_k}, \mathbf{1}_B)$ with $B = B_\Pi$. But there is in fact a simpler and more relevant choice. Indeed, on the set $\operatorname{span}(\mathbf{1}_{A_1}, \ldots, \mathbf{1}_{A_k}, \mathbf{1}_B)$, the values $x_{i_1}, x_{i_2}, x_{i_3}$ and x_{i_4} with $i_1 \in A_j \cap B$, $i_2 \in A_j \cap B^c$, $i_3 \in A_{j'} \cap B$ and $i_4 \in A_{j'} \cap B^c$ are a priori coupled; also, if $A_j \cap B$ has several connected components $i \mapsto x_i$ must take the same value on these components. These constraints seem unnecessarily restrictive.

Consider $S_\Pi \doteq \bigcup_{(A,A')\in\Pi^2} \partial(A, A')$ with $\partial(A, A') \doteq (A \times A') \cap E$. With the notion of support $S(x)$ that we defined in (2) we actually have $\operatorname{span}(\Pi) = \{x \in \mathbb{R}^n \mid S(x) \subset S_\Pi\}$. Now, if $x \in \operatorname{span}(\mathbf{1}_{A_1}, \ldots, \mathbf{1}_{A_k}, \mathbf{1}_B)$, we have in general $S(x) \subset S_{\text{new}} \doteq S_\Pi \cup \partial(B, B^c)$, which corresponds to allowing a larger support. But then it makes sense to allow x to remain in the largest set with this maximal support S_{new}, that is equivalently to stay in the vector space $\mathcal{X}_{S_{\text{new}}} \doteq \{x' \mid S(x') \subset S_{\text{new}}\}$. But, if we now define Π_{new} as the partition of V defined as the collection of all connected components in G of all sets $A_j \cap B_\Pi$ and $A_j \cap B_\Pi^c$ for $A_j \in \Pi$, then it is relatively immediate that $\operatorname{span}(\Pi_{\text{new}}) = \mathcal{X}_{S_{\text{new}}}$. The construction of Π_{new} from Π is illustrated in Figure 2.

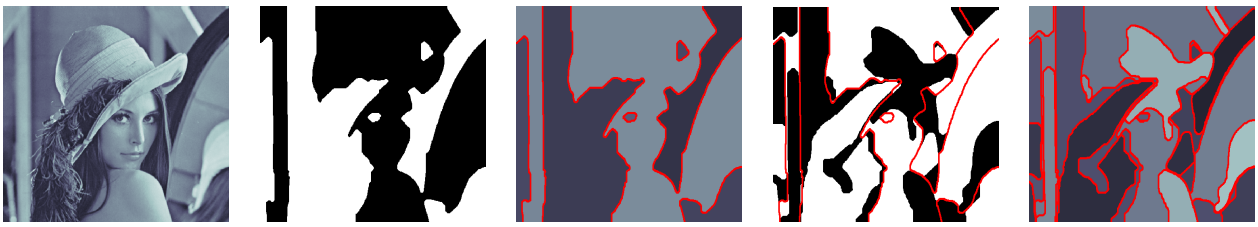


(a) Initial partition $\Pi = \{A_1, A_2\}$ (b) Steepest Binary Cut B (c) $\Pi_{new} = \{A_1, A_2, A_3, A_4, A_5\}$

Figure 2: Illustration of the induced new partition. From an initial partition Π, the steepest binary cut B induced a new partition Π_{new}. The solid line ——— represent the initial contours S, and the dashed line - - - - - the new contours $S_{new} \setminus S$ introduced by B. Note that the binary partition induced by B can more than double the number of resulting components.

We therefore set Π_{new} to be the new partition and solve the reduced problem constrained to $\operatorname{span}(\Pi_{\text{new}})$. Note that in general we do not have $S(x_\Pi) = S_\Pi$, because the total variation regularization can induce that the value of x_Π on several adjacent elements of Π are the same.

The following result shows that if a non-trivial cut (B_Π, B_Π^c) was obtained as a solution to (5) then



(a) (b) (c) (d) (e)

Figure 3: Two first iterations of cut pursuit for the ROF problem on the picture in (a). Images (b) and (d) represent the new cut at iterations 1 and 2 with B_Π and B_Π^c respectively in black and white, and (c) and (e) represent the partial solution in levels of gray, with the current set of contours S in red. The contours induced by the cut in (b) (resp. (d)) are superimposed on (c) (resp. (e)).

the new reduced problem has a solution $x_{\Pi_{\text{new}}} = \arg\min_{x \in \text{span}(\Pi_{\text{new}})} Q(x)$ which is strictly better than the previous one.

Proposition 4. *If* $B_\Pi \neq \varnothing$, $Q(x_{\Pi_{\text{new}}}) < Q(x_\Pi)$.

Proof. We clearly have

$$\text{span}(\Pi) \subset \text{span}(\mathbf{1}_{A_1}, \dots, \mathbf{1}_{A_k}, \mathbf{1}_{B_\Pi}) \subset \text{span}(\Pi_{\text{new}}),$$

so that

$$Q(x_{\Pi_{\text{new}}}) = \min_{x \in \text{span}(\Pi_{\text{new}})} Q(x) \leq \min_{x \in \text{span}(\Pi)} Q(x) = Q(x_\Pi).$$

Moreover, if $B_\Pi \neq 0$, then $Q'(x_\Pi, \mathbf{1}_B) < 0$, which entails that there exists $\varepsilon > 0$ such that $Q(x_{\Pi_{\text{new}}}) \leq Q(x_\Pi + \varepsilon \mathbf{1}_B) < Q(x_\Pi)$. This completes the proof. □

Algorithm 1: Cut Pursuit

Initialize $\Pi \leftarrow \{V\}$, $x_\Pi \in \arg\min_{z = c\mathbf{1}_V, c \in \mathbb{R}} Q(z)$, $S \leftarrow \varnothing$
while $\min_{B \subset V} \langle \nabla Q_S(x_\Pi), \mathbf{1}_B \rangle + \lambda w_{S^c}(B, B^c) < 0$ **do**
  Pick $B_\Pi \in \arg\min_{B \subset V} \langle \nabla Q_S(x_\Pi), \mathbf{1}_B \rangle + \lambda w_{S^c}(B, B^c)$
  $\Pi \leftarrow \{B_\Pi \cap A\}_{A \in \Pi} \cup \{B_\Pi^c \cap A\}_{A \in \Pi}$
  $\Pi \leftarrow$ connected components of elements of Π
  Pick $x_\Pi \in \arg\min_{z \in \text{span}(\Pi)} Q(z)$
  $S \leftarrow S(x_\Pi)$
return (Π, x_Π)

We summarize the obtained working set scheme as Algorithm 1, and illustrate its two first steps on a ROF problem in Figure 3. Propositions 4 and 2 together show that this algorithm guarantees a monotonic decrease of the objective function Q and convergence to the optimum $\Pi^\star$. In terms of complexity, at each iteration Π_{new} has at least one more component than Π, so that the algorithm converges in at most n steps, in the worst case. We now discuss how to exploit the sparse structure of x_Π to solve the reduced problem efficiently.

2.3 A reduced graph for the reduced problem

Let Π be a coarse partition of V into connected components. We argue that $\min_{z \in \text{span}(\Pi)} Q(z)$ can be solved efficiently on a smaller weighted graph whose nodes are associated with the elements of partition Π, and whose edges corresponds to pairs of adjacent elements in the original graph. Indeed, consider the graph

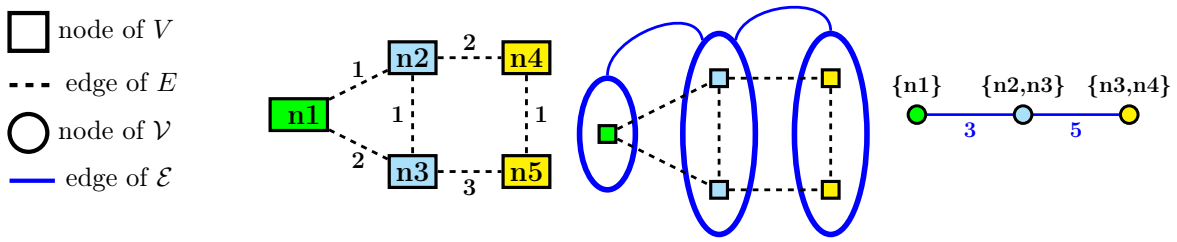


Figure 4: Example of reduced graph. Left: graph G with weights $(w_{ij})_{(i,j)\in E}$ on the edges, middle: partition Π of G into connected components, right: reduced graph $\mathcal{G}$ with weights $(w_{AB})_{(A,B)\in\mathcal{E}}$ on the edges.

$\mathcal{G}=(\mathcal{V},\mathcal{E})$ with $\mathcal{V}=\Pi$ and $\mathcal{E}=\{(A,B)\in\mathcal{V}^2 \mid \exists (i,j)\in(A\times B)\cap E\}$. Figure 4 shows an example of graph reduction on a small graph. For $x\in\operatorname{span}(\Pi)$ we can indeed express $\mathrm{TV}(x)$ simply:

Proposition 5. *For $x=\sum_{A\in\Pi} c_A \mathbf{1}_A$ we have $\mathrm{TV}(x)=\mathrm{TV}_{\mathcal{G}}(c)$ with $\mathrm{TV}_{\mathcal{G}}(c)\doteq\frac{1}{2}\sum_{(A,B)\in\mathcal{E}} w(A,B)\,|c_A-c_B|$.*

Proof.

$$2\mathrm{TV}(x)=\sum_{(i,j)\in E} w_{ij}\,|x_i-x_j|=\sum_{(i,j)\in E} w_{ij}\sum_{(A,B)\in\Pi^2}\mathbf{1}_{\{i\in A,j\in B\}}\,|c_A-c_B|=\sum_{(A,B)\in\Pi^2}|c_A-c_B|\sum_{(i,j)\in E\cap(A\times B)} w_{ij},$$

hence the result using the definition of $w(A,B)$. □

Note that if TV is the total variation associated with the weighted graph G with weights $(w_{ij})_{(i,j)\in E}$ then $\mathrm{TV}_{\mathcal{G}}$ is the total variation associated with the weighted graph $\mathcal{G}$ and the weights $\big(w(A,B)\big)_{(A,B)\in\mathcal{E}}$. Denoting $\tilde{f}: c\mapsto f(\sum_{A\in\Pi} c_A\mathbf{1}_A)$, the reduced problem is equivalent to solving $\min_{c\in\mathbb{R}^k}\tilde{f}(c)+\lambda\mathrm{TV}_{\mathcal{G}}(c)$ on $\mathcal{G}$. If Π is a coarse partition, we have $|\mathcal{E}|\ll 2m$ and computations involving $\mathrm{TV}_{\mathcal{G}}$ are much cheaper than those involving TV. As illustrated in Section 2.4, the structure of $\tilde{f}$ can often be exploited as well to reduce the computational cost on the reduced problem. The construction of the reduced graph itself $\mathcal{G}$ is cheap compared to the speed-ups allowed, as it is obtained by computing the connected components of the graph $(V,E\backslash S(x))$, which can be done in linear time by depth-first search. Note that once the reduced problem is solved if $c_\Pi\in\arg\min_c \tilde{f}(c)+\lambda\mathrm{TV}_{\mathcal{G}}(c)$ then $S(x_\Pi)$ is directly computed as $S(x_\Pi)=\bigcup\{\partial(A,A') \mid (A,A')\in\mathcal{E}, c_A\neq c_{A'}\}$.

2.4 Solving linear inverse problems with TV

A number of classical problems in image processing such as deblurring, blind deconvolution, and inpainting are formulated as ill-posed linear inverse problems (Chan et al., 2005), where a low TV prior on the image provides appropriate regularization. Typically if x_0 is the original image, H a blurring linear operator typically computed as a convolution on the image, ϵ additive noise, and $y=Hx_0+\epsilon$ the degraded image, this leads to problems of the form

$$x^\star=\operatorname*{arg\,min}_{x\in\mathbb{R}^n}\ \frac{1}{2}\|Hx-y\|^2+\lambda\mathrm{TV}(x) \tag{7}$$

Since n is large, manipulating the matrices H or $H^\intercal$ directly should be avoided, but if the forward operator $x\mapsto Hx$ is a convolution, it can be computed quickly using for example the fast Fourier transform. First order optimization algorithms, such as proximal methods, only require the computation of the gradient $H^\intercal Hx-H^\intercal y$ of f and can be used to solve (7) efficiently. In the case of a blurring operator with adequate symmetry, $H^\intercal Hx$ is also a blurring operator. These fast computations of $x\mapsto Hx$ can be exploited in our algorithm to compute the Hessian and loading vector of the reduced problem very efficiently. Indeed, for a k-partition Π of V we denote $K\in\{0,1\}^{n\times k}$ the matrix whose columns are the vectors $\mathbf{1}_A$ for $A\in\Pi$. Any

$x \in \text{span}(\Pi)$ can be rewritten as Kc with $c \in \mathbb{R}^k$. The gradient of the discrepancy function with respect to c writes: $\nabla_c 1/2 \left\| HKc - y \right\|^2 = K^\intercal H^\intercal HKc - K^\intercal Hy$. As a result, the reduced problem can be solved by a similar forward backward scheme of much smaller size, with parameters $K^\intercal H^\intercal HK$ and $K^\intercal Hy$, which are of size $k \times k$ and k respectively, and which can be computed in $\mathcal{O}(k^2\, n \log n)$ time. Indeed, each column of $H^\intercal K$ and $HH^\intercal K$ can be computed in $\mathcal{O}(n \log n)$ time using the FFT, which saves an order of magnitude.

2.5 Complexity analysis

The computational bottlenecks of the algorithm could a priori be (a) the computation of the steepest binary cut which requires to solve a min cut/max flow problem, (b) the cost of solving the reduced problem, (c) the computation the reduced graph itself, (d) the number of global iterations needed.

(a) The steepest binary cut is obtained as the solution of a max-flow/min-cut optimization problem. It is well-known that there is a large discrepancy between the theoretical upper bound on the complexity of many graph-cut algorithms and the running times observed empirically, the former being too pessimistic. In particular, the algorithm of Boykov et al. (2001a) has a theoretical exponential worst case complexity, but scales essentially linearly with respect to the graph size in practice. In fact, it is known to scale better than some algorithms with polynomial complexity, which is why we chose it.

(b) Solving the reduced problem can be done with efficient proximal splitting algorithms such as Raguet and Landrieu (2015), which is proved to reach a primal suboptimality gap of ε in $\mathcal{O}(1/\varepsilon)$ iterations; in practice, the observed convergence rate is almost linear. Preconditioning greatly speeds up convergence in practice. Moreover, the problems induced on the reduced graph can typically be solved at a significantly reduced cost: in particular, as discussed in section 2.4, for a quadratic data fitting term, the gradient in the subgraph can be computed in $\mathcal{O}(k^2)$ time, based on a single efficient FFT-based computation of the Hessian per global iteration which itself takes $\mathcal{O}(k^2 n \log n)$ time. For problems with coarse solutions, this algorithm is only called for small graphs so that this step only contributes to a small fraction of the the running time.

(c) Computing the reduced graph, requires to compute the connected components of the graph obtained when removing the edges in S, and the weights $w(A, B)$ between all paris of components (A, B). These can be efficiently performed in $\mathcal{O}(m + n)$ through a depth-first exploration of the nodes of the original graph.

(d) The main factor determining the computation time is the number of global iterations needed. In the worst case, this is $\mathcal{O}(n)$. In practice, the number of global iterations seems to grow logarithmically with the number of constant regions at the optimum. If for simple images or strongly regularized natural images 4 or 5 cuts seems to suffice, a very complex image with very weak regularization might need many more. In the end, our algorithm is only efficient on problems whose solutions do not have too many components. E.g. in the deblurring task, it is competitive for solutions with up to $10,000$ components for a 512×512 image.

2.6 Regularization path of the total variation

Since the regularization coefficient λ is difficult to choose a priori, it is typically useful to compute an approximate regularization path, that is the collection of solutions to (1) for a set of values $\lambda_0 > \cdots > \lambda_j > 0$. For ℓ_1 sparsity, Efron et al. (2004) showed how a fraction of the exact regularization path can be computed in a time of the same order of magnitude as the time need to compute of the last point. In general, when the path is not piecewise linear, the exact path cannot be computed, but similar results have been shown for group sparsity (Obozinski et al., 2006; Roth and Fischer, 2008). The case of total variation has been studied as well for 1-dimensional signals in Bleakley and Vert (2011). We propose a warm start approach to compute an approximate[6] solution path for the total variation.

[6] In fact for a quadratic data fitting term regularized by the total variation, the regularization path is piecewise linear and could thus in theory computed exactly, with a scheme similar to the LARS algorithm (Efron et al., 2004). It should however

The rationale behind our approach is that, if λ_i and λ_{i+1} are close, the associated solutions $x_i^\star$ and $x_{i+1}^\star$ should also be similar, as well as their associated optimal partition, which we will refer to as $\Pi_i^\star$ and $\Pi_{i+1}^\star$. Consequently, it is reasonable to use a warm-start technique which consists in initializing Algorithm 1 with $\Pi_i^\star$ to solve the problem associated with λ_{i+1} and to expect that it will converge in a small number of binary cuts. It is important to note, that while our algorithm lends itself naturally to warm starts, to the best of our knowledge similar warm-start techniques do not exist for proximal splitting approaches such as Raguet et al. (2013) or Chambolle and Pock (2011). Indeed solutions whose primal solutions are close can have vastly different auxiliary/dual solutions, and in our experiments no initialization heuristics consistently outperformed a naive initialization.

3 Minimal partition problems

We consider now a generalization of the minimal partition problem of the form $\min_{x\in\mathbb{R}^n} Q(x)$ with $Q(x) = f(x) + \lambda\Gamma(x)$ where $\Gamma(x) \doteq \frac{1}{2}\sum_{(i,j)\in S(x)} w_{ij}$ the Mumford-Shah penalty. This non-convex non-differentiable problem being significantly harder than the previous one, we restrict the functions f we consider to be separable functions of the form $f(x) = \sum_{i\in V} f_i(x_i)$ with $f_i : \mathbb{R} \mapsto \mathbb{R}$ continuously differentiable and convex. We call the corresponding problem *generalized minimal partition problem.*

Inspired by greedy feature selection algorithms in the sparsity literature and by the working set algorithm we presented for TV regularization, we propose to exploit that $|\Pi^*|$ is not too large to construct an algorithm that greedily optimizes the objective by adding and removing cuts in the graph.

Indeed, the problem that we consider has a fixed regularization coefficient λ, and so its natural counterpart for classical sparsity is the problem of minimizing an objective of the form $f(x) + \lambda\|x\|_0$ which subsumes AIC, BIC and other information criteria. The algorithmic approach we consider is thus the counterpart of a very natural greedy algorithm to minimize the former objective, which surprisingly is almost absent from the literature, perhaps for the following reasons: On the one hand, work on *stagewise regression* and forward-backward greedy algorithms, which both add and remove variables, goes back to the 60ies (Efroymson, 1960), but the algorithms then considered were based on sequences of tests as opposed to a greedy minimization of a penalized criterion. On the other hand, the literature on greedy algorithm for sparse models has almost exclusively focused on solving the constrained problem $\min_x f(x)$ s.t. $\|x\|_0 \leq k$, with algorithms such as OMP, Orthogonal least squares (OLS), FoBa, and CoSamp, which can alternatively be viewed as algorithms that are greedily approximating the corresponding Pareto frontier. A notable exception is IHT.

A very natural variant of OLS solving $\min_x f(x)+\lambda\|x\|_0$ can however be obtained by adding the ℓ_0 penalty to the objective. This algorithm was formally considered in Soussen et al. (2011) under the name Single Best Replacement (SBR), in reference to the similar Single Maximum Likelihood Replacement (SMLR) of Kormylo and Mendel (1982). At each iteration, the algorithm considers adding or removing a single variable, whichever reduces most the value of the objective. It should be noted that while the similar OLS and OMP are forward algorithms, SBR is a forward-backward algorithm, which can remove a variable provided doing so only increases f by less than λ.

We argue in the following sections that a similar natural algorithm can be designed for the generalized minimal partition problem, where forward steps split existing components and backward steps merge two components (with the further possibility of combined merge-resplit moves). We call this algorithm ℓ_0-Cut Pursuit, since it is also naturally very similar to Cut Pursuit.

3.1 A greedy algorithm for regularized minimal partition

As for the working set algorithm, we propose to build an expansion of x of the form $x = \sum_{i=1}^k c_i \mathbf{1}_{A_i}$, for $\Pi = (A_1, \cdots, A_k)$ a partition of V, by recursively splitting some of the existing sets $A \in \Pi$. Assume that we split the set of existing regions $(A_j)_{1\leq j\leq k}$ by introducing a global cut (B, B^c) for some set $B \subset V$. This cut induces a cut on each element A_j of the form $(A_j \cap B, A_j \cap B^c)$. Two simple properties should be noted: (a)

be expected that this path has many point of discontinuity of the gradient, which entails that the cost of computation of the whole path is likely to be prohibitively high. We therefore do not consider further this possibility.

the additional boundary length incurred with the cut is simply the sum of the lengths of the cuts induced within each element A_j and is precisely of the form $\sum_{j=1}^{k} w(A_j \cap B, A_j \cap B^c)$ — the boundary of previously accepted component is thus "free" (cf Figure 2), (b) if the value of x is re-optimized under the constraint that it should be constant on each of the elements $A_j \cap B$ and $A_j \cap B^c$ of the new partition, then the separability of f entails that the optimization is independent on each set A_j. As a consequence of (a) and (b) the choice of an optimal cut reduces to independent choices of optimal cut on each set A_j as defined by the objective

$$\min_{B \subset V} \min_{(h_j, h'_j)} \sum_{i \in A_j \cap B} f_i(h_j) + \sum_{i \in A_j \cap B^c} f_i(h'_j) + \lambda w(A_j \cap B, A_j \cap B^c).$$

We should therefore design an algorithm that cuts a single set A at a time. To simplify notations we consider hereafter the case $\Pi = \{V\}$, which corresponds to the very first cut of the algorithm.

3.1.1 Optimal binary cut with alternating minimization

In the same way that we defined the steepest binary cut in the working set algorithm, we define the optimal binary partition (B, B^c) of V such that Q optimized over $\mathrm{span}(\mathbf{1}_B, \mathbf{1}_{B^c})$ is as small as possible. Ideally, we should impose that B and B^c have a single connected component each, because as argued in section 2.3, it does not make sense to impose that x_i should have the same values in different connected components. However, since this constraint is too difficult to enforce, we first ignore it and address it later with post-processing. Note however that the penalization of the length of the boundary between B and B^c should strongly discourage the choice of sets B with many connected components.

Since $\Gamma(h\mathbf{1}_B + h'\mathbf{1}_{B^c}) = \Gamma(\mathbf{1}_B) = w(B, B^c)$, and ignoring the connectedness constraint, the corresponding optimization problem is of the form

$$\min_{B \subset V} \min_{h, h' \in \mathbb{R}} \sum_{i \in B} f_i(h) + \sum_{i \in B^c} f_i(h') + \lambda w(B, B^c). \tag{8}$$

This problem is a priori hard to solve in general, because $B \mapsto \min_{h,h' \in \mathbb{R}} f(h\mathbf{1}_B + h'\mathbf{1}_{B^c})$ is not a submodular function. However, when h, h' are fixed, the assumption that f is separable entails that $B \mapsto f(h\mathbf{1}_B + h'\mathbf{1}_{B^c})$ is a modular function, so that the objective can be optimized with respect to B by solving a max-flow problem. Similarly as for the flow problem (6) we define the flow graph $G_{flow} = (V \cup \{s,t\}, E_{flow})$ whose edge set and capacities are defined by:

$$E_{flow} = \begin{cases} (s,i), \forall i \in \nabla_+, & \text{with} \quad c_{si} = f_i(h) - f_i(h'), \\ (i,t), \forall i \in \nabla_-, & \text{with} \quad c_{it} = f_i(h') - f_i(h), \\ (i,j), \forall (i,j) \in E, & \text{with} \quad c_{ij} = \lambda w_{ij}, \end{cases} \tag{9}$$

where $\nabla_+ \doteq \{i \in V \mid f_i(h) > f_i(h')\}$ and $\nabla_- \doteq V \backslash \nabla_+$.

The smoothness and convexity of f with respect to h and h' guarantee that the objective can be minimized efficiently with respect to these variables. A local minimum of the objective can thus be obtained efficiently by alternatively minimizing with respect to B and (h, h') as suggested by Bresson et al. (2007) or El-Zehiry et al. (2011).

3.1.2 From binary cut to partition in connected components

Like the working set algorithm proposed for the total variation, ℓ_0-Cut Pursuit recursively splits the components of the current partition Π. The sets B and B^c obtained as a solution of (8) are not necessarily connected sets, but splitting B and B^c into their connected components and assigning each connected component its own value obviously does not change the contour length Γ and can only decrease f. Given the collection of connected components $A_1, \ldots, A_k$ of B and B^c we therefore set $x = h_1\mathbf{1}_{A_1} + \ldots + h_k\mathbf{1}_{A_k}$ with h_j the minimizer of $h \mapsto \sum_{i \in A_j} f_i(h)$. Note that each h_i could possibly be computed in parallel given the separability of f.

3.1.3 Backward step

In greedy algorithms for plain sparsity, backward steps remove variables to reduce the support of the solution. In our case, the appropriate notion of support is $S(x)$, which is formed as the union of the boundaries between pairs of components. A backward step is a step that reduces the total boundary length (or size). The most natural way to obtain this is by merging two adjacent components. Using the same ideas as the ones proposed in Soussen et al. (2011) for plain sparsity, we consider backward steps when the reduction of penalty obtained is larger than the increase of f.

Simple merge step: If a pair of adjacent components (A, B) is merged into a single constant component, $\Gamma(x)$ decreases by $w(A, B)$ and the merge is worth it if f increases by less than $\lambda\, w(A, B)$. It should be noted that the merge step considered does not in general not correspond to canceling a previous cut.

A shortcoming of the simple merge step is that while the removal of boundaries between components is considered, a shift or other type of remodeling of the created boundaries is not possible. But since the optimal binary computation only considers binary partitions, the shape of the components might be suboptimal without justifying, however, a complete removal. We therefore consider another kind of step:

Merge-resplit: This step is a combination of a merge step immediately followed by a new cut step on the merged components. It is a "backward-then-forward" step, which can be worth it even if the corresponding backward step taken individually is not decreasing the objective. It amounts to solve the corresponding min cut/max flow problem

$$\min_{z_i \in \{0,1\}, i \in A \cup B} \sum_{i \in A \cup B} z_i f_i(x_A) + (1 - z_i) f_i(x_B) + \frac{\lambda}{2} \sum_{(i,j) \in (A \times B) \cap E} w_{ij} \, |z_i - z_j|.$$

Note that finding the best way to resplit is very similar to what Boykov et al. (2001b) call an α-β *swap* in the context of energy minimization in Markov random fields: nodes assigned to other components[7] than A or B keep their current assignments to components, but the nodes of $A \cup B$ are reassigned to A or B so that the boundary between A and B minimizes the above energy. Note that the merge-resplit step includes the possibility of a simple merge step (without resplitting), since all elements can be "swapped" in the same set by the α-β swap , so that the new boundary is effectively empty. Note that during the merge-resplit step the value of x_A and x_B is held constant and only updated upon completion of the step. In fact, in a number of cases, it might be possible to iterate such steps for a given pair (A, B). We do not consider this computationally heavier possibility.

Remark: The work we presented in this section focussed on a formulation in which the Mumford-Shah total boundary size is penalized and not constrained. It is worth pointing out that trying to solve directly the constrained case seems difficult: indeed, designing algorithms that are only based on forward steps (e.g., in the style of OMP, OLS, etc) might not succeed, because of the dependence between the cuts that need to be introduced to form the final solution. Based on similar ideas as the ones used in ℓ_0-Cut Pursuit, we designed and tested an algorithm generalizing the FoBa algorithm (Zhang, 2009). The obtained algorithm tended to remain trapped in bad local minima and yielded solutions that were much worse than the ones based on the penalized formulation.

3.2 Implementation

Similarly as in the convex case, ℓ_0-Cut Pursuit maintains a current partition Π that is recursively split and computes optimal values for each of its components. It is comprised of three main steps: the splitting of the current partition, the computation of the connected components and their values, and a potential merging step, when necessary.

Splitting. For each component an optimal binary partition (B, B^c) is obtained by solving (8) as described in section 3.1.1: we alternatively minimize the objective with respect to B and with respect to (h, h') until either B does not change or a maximum number of iterations is reached. In practice, the algorithm converges

[7] In the context of MRFs the components correspond to a number of different classes fixed in advance and are in general not connected.

in 3 steps most of the time. The choice of an appropriate initialization for B is non trivial. Since the problem in which $\lambda = 0$ is often simpler, and can in a number of cases be solved analytically, we chose to use that solution to initialize our alternating minimization scheme. Indeed, for $\lambda = 0$, and when f is a squared Euclidean distance $f : x \mapsto \|x - x_0\|_2^2$ the objective of (8) is the same as the objective of one dimensional k-means with $k = 2$; in this particular setting, the problem reduces to a change-point analysis problem, and an exact solution can be computed efficiently by dynamic programming (Bellman, 1973). This can be generalized to the case of Bregman divergences and beyond (Nielsen and Nock, 2014).

As described in section 3.1.2, the partition Π is updated by computing its connected components after it is split by (B, B^c). Subroutine 1 gives the procedure algorithmically.
It is important to note that this is the only operation that involves the original graph G, and hence will be the computational bottleneck of the algorithm. Fortunately since f is separable, this procedure can be performed on each component in parallel.

Components saturation. We say that a component is *saturated* if the empty cut is an optimal binary cut. A *saturated* component will no longer be cut (because the separability of f entails that other cuts do not change the fact that it is saturated) unless it is first involved in a merge or merge-resplit step. A partition Π is said to be saturated if all its components are saturated.

Simple merge. This backward step consists in checking for each neighboring components A and B in Π whether merging them into a single component decreases the energy. If we denote $\Pi_-(A, B)$ the partition obtained by merging A and B, the corresponding decrease in energy $\delta_-(A, B)$ is

$$\delta_-(A, B) = f(x_\Pi) - f(x_{\Pi_-(A,B)}) + \lambda w(A, B),$$

with $\Pi_-(A, B) \doteq \Pi \setminus \{A, B\} \cup \{A \cup B\}$. This value is computed for each neighboring components, and stored in a priority queue. Each pair that provides a non negative decrease is merged, and δ_- is updated for the neighbors of A and B to reflect the change in value and graph topology. This operation scales with the size of the reduced graph only, and therefore can be performed efficiently for problems with a coarse solution.

Merge-resplit. This more complex backward steps, already described in 3.1.3 is computationally significantly more intensive as it is performed on the edges of the full graph, by contrast with the simple merge which only considers the edges of the reduced graph. As a consequence, while all potential simple merge steps can be precomputed and performed based on a priority queue by merging first the pair of component yielding the largest decrease in objective value, this would be too computationally heavy here and we perform boundary changes only once for each pair of neighbors in the graph $\mathcal{E}$. The pseudocode of the procedure is detailed in subroutine 3.

Algorithm structure: We present in Algorithm 2 and 3 implementations of the algorithm using respectively only simple merge or merge-resplit steps. We chose to alternate between splitting all component at once (possibly in parallel) and then iterating backward steps over all adjacent pairs of components. This allows for the splitting to be done in parallel directly on the original flow graph, thus avoiding the memory overheads associated with constructing a new flow graph for each new component. It would have been theoretically possible to be more greedy and to perform a single forward step (corresponding to splitting a single region) at a time or a single backward step at a time by maintaining a global priority queue and greedily choosing the most beneficial. However we did not implement this option because the overheads cost would

have been prohibitive.

Subroutine 1: $[\Pi, \mathcal{E}] \leftarrow$ `split`$(\Pi, \mathcal{E}, A)$

Split component A with a binary cut.
$\Pi \leftarrow \Pi \setminus \{A\}$
$B \leftarrow \arg\min_{B \subset A, h, h'} \sum_{i \in B} f_i(h) + \sum_{i \in B^c} f_i(h')$
while *not_converged* **do**
  $x \leftarrow \arg\min_h \sum_{i \in B} f_i(h)$
  $x' \leftarrow \arg\min_h \sum_{i \in A \setminus B} f_i(h)$
  $B \leftarrow \arg\min_{B \subset A} \sum_{i \in B} f_i(x) + \sum_{i \in B^c} f_i(x') + \lambda w(B, B^c)$
$[B_1, \cdots, B_k] \leftarrow$ connected components of B and $A \setminus B$
$\Pi \leftarrow \Pi \cup \{B_1, \cdots, B_k\}$
$\mathcal{E} \leftarrow$ updated adjacency structure return Π;

Subroutine 2: $[\Pi, \mathcal{E}] \leftarrow$ `simple_merge`$(\Pi, \mathcal{E}, A, B)$

Merges components A and B
$\Pi \leftarrow \Pi \setminus \{A, B\} \cup \{A \cup B\}$
$\mathcal{E} \leftarrow \mathcal{E} \setminus \big\{\{A, B\}\big\}$
for *C neighbors of A or B* **do**
  $\mathcal{E} \leftarrow \mathcal{E} \cup \big\{\{A \cup B, C\}\big\}$

Subroutine 3: $[\Pi, \mathcal{E}] \leftarrow$ `merge_resplit`$(\Pi, \mathcal{E}, A, B)$

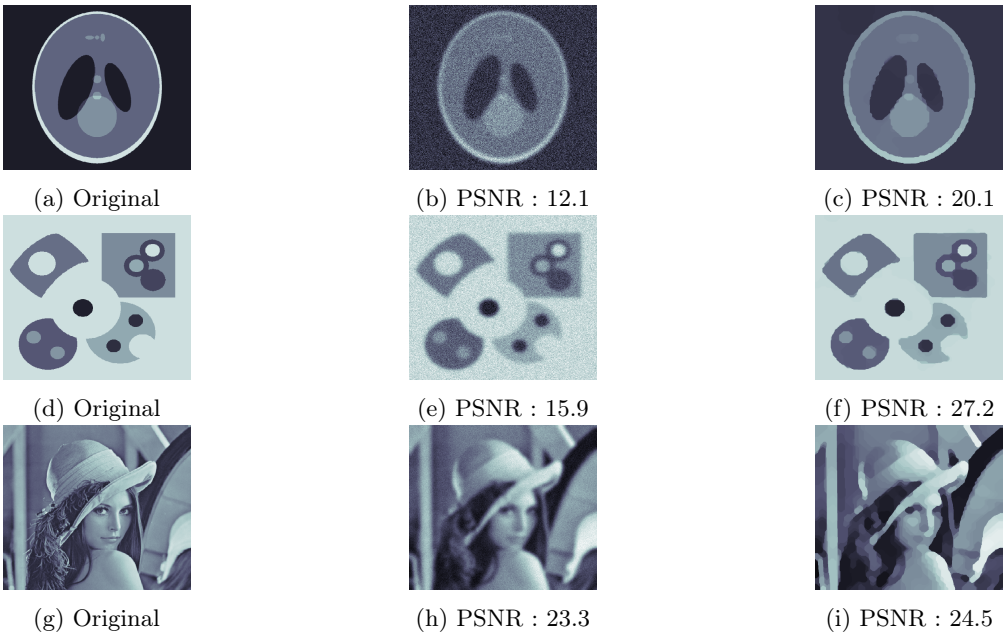
Perform a merge-resplit operation on components A and B.
$[\Pi, \mathcal{E}] \leftarrow$ `simple_merge`$(\Pi, \mathcal{E}, A, B)$
$\Pi \leftarrow \Pi \setminus \{A \cup B\}$
$x_A \leftarrow \arg\min_h \sum_{i \in A} f_i(h)$
$x_B \leftarrow \arg\min_h \sum_{i \in B} f_i(h)$
$C \leftarrow \arg\min_{C \subset A \cup B} \sum_{i \in C} f_i(x_A) + \sum_{i \in A \cup B \setminus C} f_i(x_B) + \lambda w(C, A \cup B \setminus C)$
$[C_1, \cdots, C_k] \leftarrow$ connected components of C and $A \cup B \setminus C$
$\Pi \leftarrow \Pi \cup \{C_1, \cdots, C_k\}$
$\mathcal{E} \leftarrow$ updated adjacency structure

Algorithm 2: Simple merge variant (ℓ_0-CPm)

Initialization: $\Pi_0 = \{V\}$, $\mathcal{E} = \varnothing$
while Π *is not saturated* **do**
  for $A \in \Pi$ *in parallel* **do**
    if A *is not saturated* **then**
      $[\Pi, \mathcal{E}] \leftarrow$ `split` $(\Pi, \mathcal{E}, A)$
  Compute $\delta_-(A, B)$ for all $(A, B) \in \mathcal{E}$
  while $\max_{(A,B) \in \mathcal{E}} \delta_-(A, B) > 0$ **do**
    $(A, B) = \arg\max_{(A', B') \in \mathcal{E}} \delta_-(A', B')$
    $[\Pi, \mathcal{E}] \leftarrow$ `merge` $(\Pi, \mathcal{E}, A, B)$
    Update $\delta_-(A, B)$ for all $(A, B) \in \mathcal{E}$

Algorithm 3: Merge-resplit variant (ℓ_0-CPs)

Initialization: $\Pi_0 = \{V\}$, $\mathcal{E} = \varnothing$
while Π *is not saturated* **do**
  for $A \in \Pi$ *in parallel* **do**
    if A *is not saturated* **then**
      $[\Pi, \mathcal{E}] \leftarrow$ `split` $(\Pi, \mathcal{E}, A)$
  $\mathcal{E}' \leftarrow \mathcal{E}$
  for $\{A, B\} \in \mathcal{E}'$ **do**
    if $\{A, B\} \in \mathcal{E}$ **then**
      $[\Pi, \mathcal{E}] \leftarrow$ `merge_resplit`
      $(\Pi, \mathcal{E}, A, B)$



(a) Original (b) PSNR : 12.1 (c) PSNR : 20.1

(d) Original (e) PSNR : 15.9 (f) PSNR : 27.2

(g) Original (h) PSNR : 23.3 (i) PSNR : 24.5

Figure 5: Benchmark on the deblurring task. Left column : original images, middle column : blurred images, right column : images retrieved by Cut Pursuit (CP)

4 Experiments

4.1 Deblurring experiments with TV

To assess the performance in terms of speed of our working set algorithm for the total variation regularization, we compare it with several state-of-the-art algorithms on a deblurring task of the form presented in section 2.4. Specifically, given an image x, we compute $y = Hx + \epsilon$, where H is a Gaussian blur matrix, and ϵ is some Gaussian additive noise, and we solve (1) with a total variation regularization based on the 8-neighborhood graph built on image pixels. We use three 512×512 images of increasing complexity to benchmark the algorithms: the Shepp-Logan phantom, a simulated example and Lena, all displayed in Figure 5. For all images the standard deviation of the blur is set to 5 pixels.

4.1.1 Competing methods

Preconditionned Generalized Forward Backward (PGFB). As a general baseline, we consider a recent preconditioned generalized forward-backward splitting algorithm by Raguet and Landrieu (2015) whose prior non-preconditioned version was shown to outperform state-of-the art convex optimization on deblurring tasks in Raguet et al. (2013), including among others the algorithm of Chambolle and Pock (2011). Raguet and Landrieu (2015) demonstrate the advantages of the preconditioning strategy used over other adaptive metric approaches, such as the preconditioning proposed in Pock and Chambolle (2011) and the inertial acceleration developed in Lorenz and Pock (2014).

Accelerated forward-backward with parametric max-flows (FB+). Since efficient algorithms that solve the ROF problem have been the focus of recent work, and given that the ROF problem corresponds to the computation of the proximal operator of the total variation, we also compare with an implementation of the accelerated forward-backward algorithm of Nesterov (2007). To compute the proximal operator, we use an efficient solver of the ROF problem based on a reformulation as a parametric max-flow proposed by Chambolle and Darbon (2009). The solver we use is the one made publicly available by the authors, which is based on a divide and conquer approach that works through the resolution of a parametric max-flow

problem. This implies computing a sequence of max-flow problems, whose order make it possible to re-use the search trees in the Boykov et al. (2001b) algorithm, thereby greatly speeding up computations.

Cut Pursuit with Frank-Wolfe descent direction (CPFW). We consider an alternative to the steepest binary partition to split the existing components of the partial solution: Inspired by the conditional gradient algorithm for regularized problems proposed by Harchaoui et al. (2015), consider a variant of Cut Pursuit in which we replace the steepest binary cut by the cut (B, B^c) such that $\mathbf{1}_B$ is the Frank-Wolfe direction for the total variation, i.e. minimizing $w(B, B^c)^{-1}\langle \nabla f(x), \mathbf{1}_B \rangle$ (see the discussion at the end of Section 2.1 and appendix A). Note that the corresponding minimization of a ratio of combinatorial functions can in this setting be done efficiently using a slight modification of the algorithm of Dinkelbach (1967). See appendix C for more details. We chose not to make direct comparisons with the algorithms of Harchaoui et al. (2015) and of Bach (2013, Chap. 7.12), since it is clear that these algorithms will be outperformed by CPFW. Indeed, these algorithms include a single term of the form $\mathbf{1}_A$ in the expansion of x at each iteration, while CP and CPFW grow much faster the subspace in which x is sought (its dimension typically more than doubles at each iteration). This entails that these algorithms must be slower than CPFW, because for the former and for the latter, a single iteration requires to compute a Frank-Wolfe step, which require to solve several graph-cuts on the whole graph, and, as we discuss in Section 4.1.2 and illustrate on Figure 7, the cost of graph cuts already dominates the per iteration cost of CP and CPFW.

Cut Pursuit. To implement our algorithm (CP), we solve min-cut problems using the Kohli and Torr (2005) solver, which itself is based on Boykov et al. (2001b) and Kolmogorov and Zabih (2004). The problems on the reduced graph are solved using the PGFB algorithm. This last choice is motivated by the fact that the preconditioning is quite useful as it compensates for the fact that the weights on the reduced graph can be quite imbalanced.

4.1.2 Results

Figure 6 presents the convergence speed of the different approaches on the three test images on a quad-core CPU at 2.4 Ghz. Precisely, we represent the relative primal suboptimality gap $(Q_t - Q_\infty)/Q_\infty$ where Q_∞ is the lowest value obtained by CP in 100 seconds. We can see that our algorithm significantly speeds up the direct optimization approach PGFB when the solution is sparse, and that it remains competitive in the case of a natural image with strong regularization. Indeed since the reduced problems are of a much smaller size than the original, our algorithm can perform many more forward-backward iterations in the same allotted time.

The variant of Cut Pursuit using Frank-Wolfe directions (CPFW) is as efficient over the first few iterations but then stagnates. The issue is that the computation of a new Frank-Wolfe direction does not take into account the current support $S(x)$ which provides a set of edges that are "free"; this entails that the algorithm overestimates the cost of adding new boundaries, resulting in too conservative updates.

Accelerated forward-backward with parametric max-flow (FB+) is also slower than the Cut Pursuit approach in this setting. This can be explained by the fact that the calls to max-flow algorithms, represented by a mark on the curve, are better exploited in the cut pursuit setting. Indeed in the forward-backward algorithm the solutions of parametric max-flow problems are exploited by performing one (accelerated) proximal gradient step. By contrast, in the Cut Pursuit setting, the solution of each max-flow problem is used to optimize the reduced problem. Since the reduced graph is typically much smaller than the original, a precise solution can generally be obtained very quickly, yet providing a significant decrease in the objective function. Furthermore, as the graph is split into smaller and smaller independent connected components by Cut Pursuit, the call to the max-flow solver of Boykov et al. (2001b) are increasingly efficient because the augmenting paths search trees are prevented from growing too wide, which is the main source of computational effort.

Figure 7 presents the breakdown of computation time for each algorithm over 60 seconds of computation. In PGFB, the forward-backward updates naturally dominate the computation time, as well as the fast Fourier transform needed to compute the gradient at each iteration. In FB+, the computation of the proximal operator of the partial solution through parametric maximum flows is by far the costliest. Our approach and

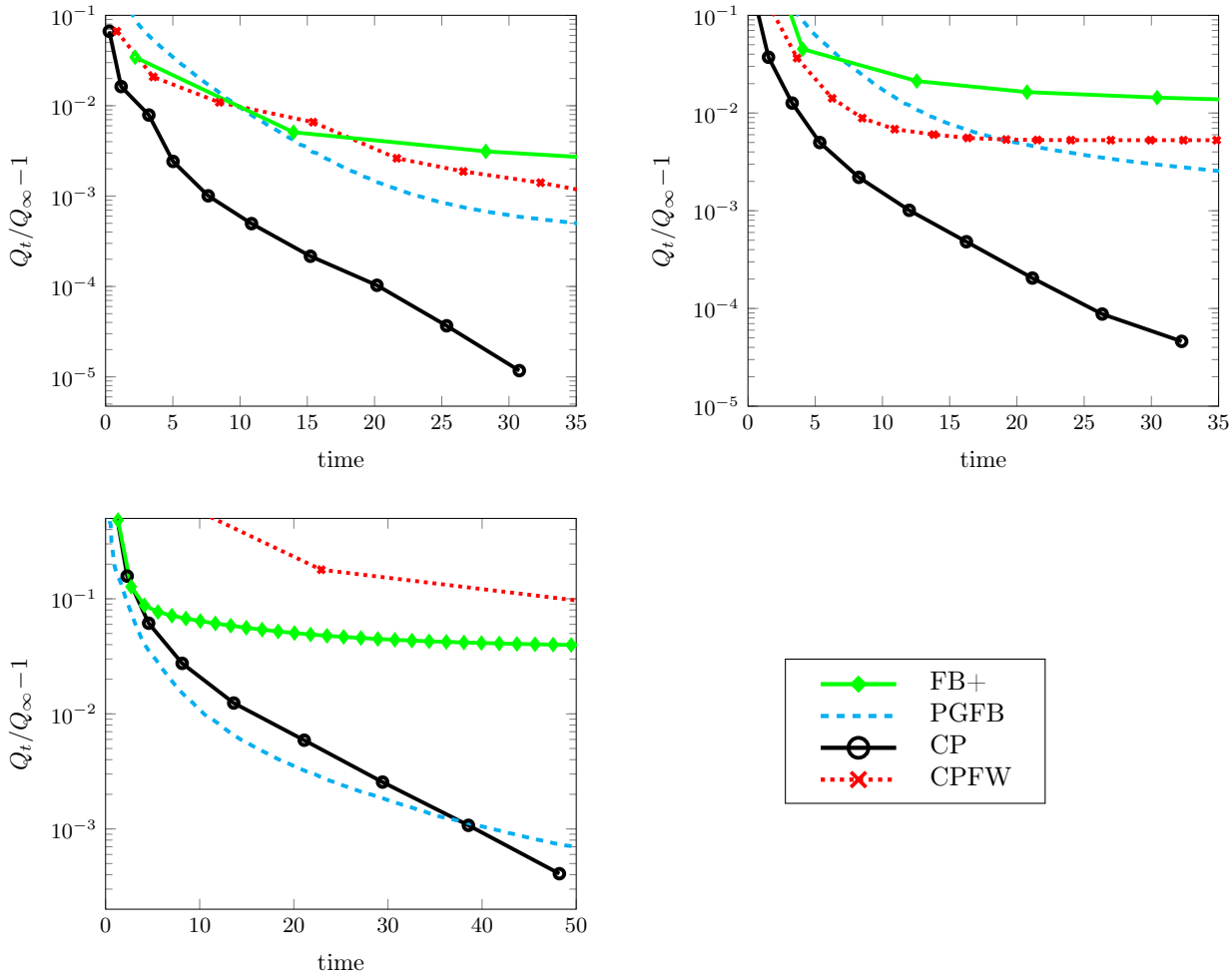


Figure 6: Relative primal suboptimality gap $Q_t/Q_\infty - 1$ at time t (in seconds) for different algorithms on the deblurring task: accelerated forward backward (FB+) , Preconditonned Generalized Forward Backward (PGFB) , Cut pursuit (CP) and a variant using Frank-Wolfe directions (CPFW) , and for different 512×512 images and different regularization values: Shepp-Logan phantom (left), our simulated example (middle) and Lena (right). The marks in (FB+), (CP) and (CPFW) corresponds to one iteration.

CPFW share a similar breakdown of computation time as their structures are similar. The maximum flow represents the highest cost, with the fast Fourier transform needed to compute $K^\intercal H^\intercal H K$ a close second. Finally diverse operations such as computing the reduced graph takes a small fraction of the time. More interestingly, solving the reduced problem (with the PGFB subroutine of CP) takes comparatively very little time (roughly 3%) when this is the only step that actually decreases the objective function. This is expected as, even at the last iteration, the reduced graph had only 300 components so that the associated problem is solved very rapidly.

4.1.3 Approximate regularization path

We now present the computation of an approximate regularization path for the ROF minimization, using warm-starts as described in Section 2.6. We consider the task of ROF-denoising on three natural images presented in Figure 9. For each image we pick 20 values of λ evenly distributed logarithmically in the range

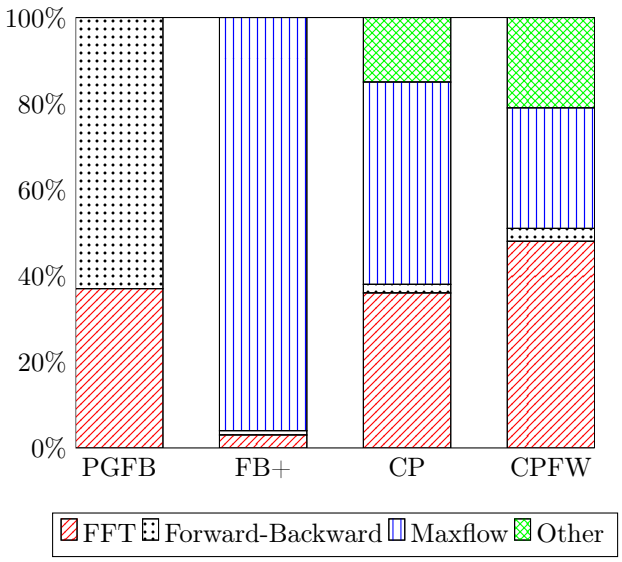


Figure 7: Time breakdown for the different algorithms over 60 seconds of optimization.

of parameters inducing from coarse to perfect reconstructions.

4.1.4 Competing methods

Parametric max-flows (PMF). We use the parametric max-flow based ROF solver of Chambolle and Darbon (2009) to compute each value. In our numerical experiments, it was the fastest of all available solvers, and moreover returns an exact solution.

Cut Pursuit (CP). We use the algorithm presented in this paper to separately compute the solutions for each parameter value. The algorithm stops when it reaches a relative primal suboptimality gap $Q_t/Q_\infty - 1$ of 10^{-5}, with Q_∞ the exact solution given by PMF. sec:mergings

Cut Pursuit Path (CPP). We use the warm start approach proposed in Section 2.6, with the same stopping criterion.

4.1.5 Results

We report in Figure 9 the time in seconds necessary to reach a primal suboptimality gap of 10^{-5} for the different approaches. We observe that, in general, cut pursuit (CP) is slightly faster than the parametric max-flow. It should be noted, however, that the latter finds an exact solution and remains from that point of view superior. Warm starts allow for a significant acceleration, needing at most two calls to the max-flow code to reach the desired gap. Unlike the deblurring task, for high noise levels, Cut Pursuit remains here very competitive for natural images which are not sparse, as illustrated in Table 10 and Figure. 8.

As the regularization strength decreases the coarseness of the solution decreases, and as a consequence the Cut Pursuit approaches CP and CPP become less and less efficient. This is because as the number of components increases, so does the time needed to solve the reduced problem. We note however that for the values provided with the peak PSNR, the warm start approach is faster than PMF.

PMF and CP perform significantly worse on sparse images and for high values of λ. This can be explained by the inner workings of the max-flow algorithm of Boykov et al. (2001b). Indeed for high values of λ or sparse images, the pairwise term of the corresponding Potts model will dominate, which forces the algorithm to build deep search trees to find augmenting paths. Indeed as the size of the regions formed by the cut increase, the combinatorial exploration of all possible augmenting paths drastically increases as well. The

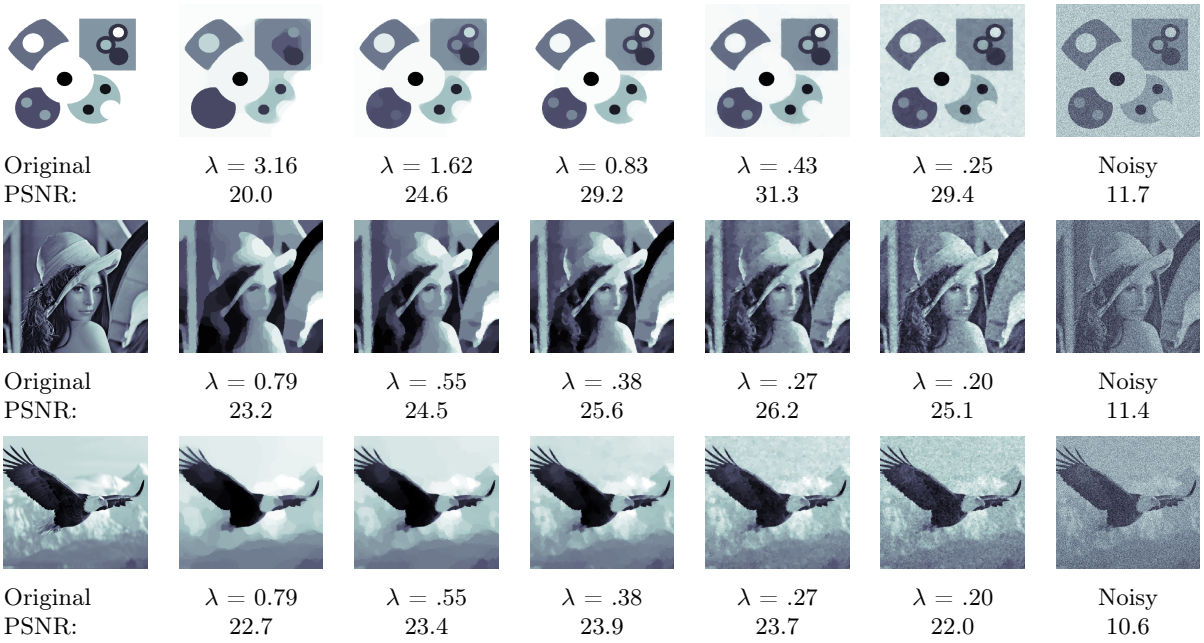


Figure 8: Illustration of the regularization path for the three images in the data set for 5 of the 20 values in the regularization parameters in the path. The peak PSNR is reached for $\lambda = 0.53$, 0.28 and 0.34 respectively.

warm-started path approach does not suffer from this problem because the graph is already split in smaller components at the warm-start initialization, which prevents the search trees from growing too large.

4.2 Experiments on minimal partitions

4.2.1 Denoising experiment

We now present experiments empirically demonstrating the superior performance of the ℓ_0-Cut pursuit algorithm presented in section 3. We assess its performance against two state-of-the art algorithms to minimize the Mumford-Shah energy of two noisy 512×512 images: the Shepp-Logan phantom (Shepp and Logan, 1974) and another simulated example. In order to illustrate the advantage of our algorithm over alternatives which discretize the value range, we add a small random shift of grey values to both images. We also test the algorithms on a spatial statistic aggregation problem using open-source data[8] which consists in computing the statistically most faithful simplified map of the population density in the Paris area over a regular grid represented in Figure 12. The raster is triangulated to obtain a graph with $252,183$ nodes and $378,258$ edges. We use the squared loss weighted by the surface of each triangle as a fidelity term.

4.2.2 Competing methods

α-expansions on quantized models (CRFi). If the range of values of x_i is quantized, the MPP and TV problems reduce to a Potts model, in which each class c is associated with a (non necessarily connected) level-set (Ishikawa, 2003). In the MPP case, the pairwise terms are of the form $1_{\{c_i \neq c_j\}} w_{ij}$. We use α-expansions (Boykov et al., 2001b) to approximately minimize the corresponding energy. More precisely we use the α-expansions implementation of Fulkerson et al. (2009), which uses the same max-flow code (Boykov and Kolmogorov, 2004) as our algorithm. We denote the resulting algorithm CRFi where i is the number of levels of quantization of the observed image value range. While this algorithm is not theoretically guaranteed

[8] https://www.data.gouv.fr/fr/datasets/donnees-carroyees-a-200m-sur-la-population

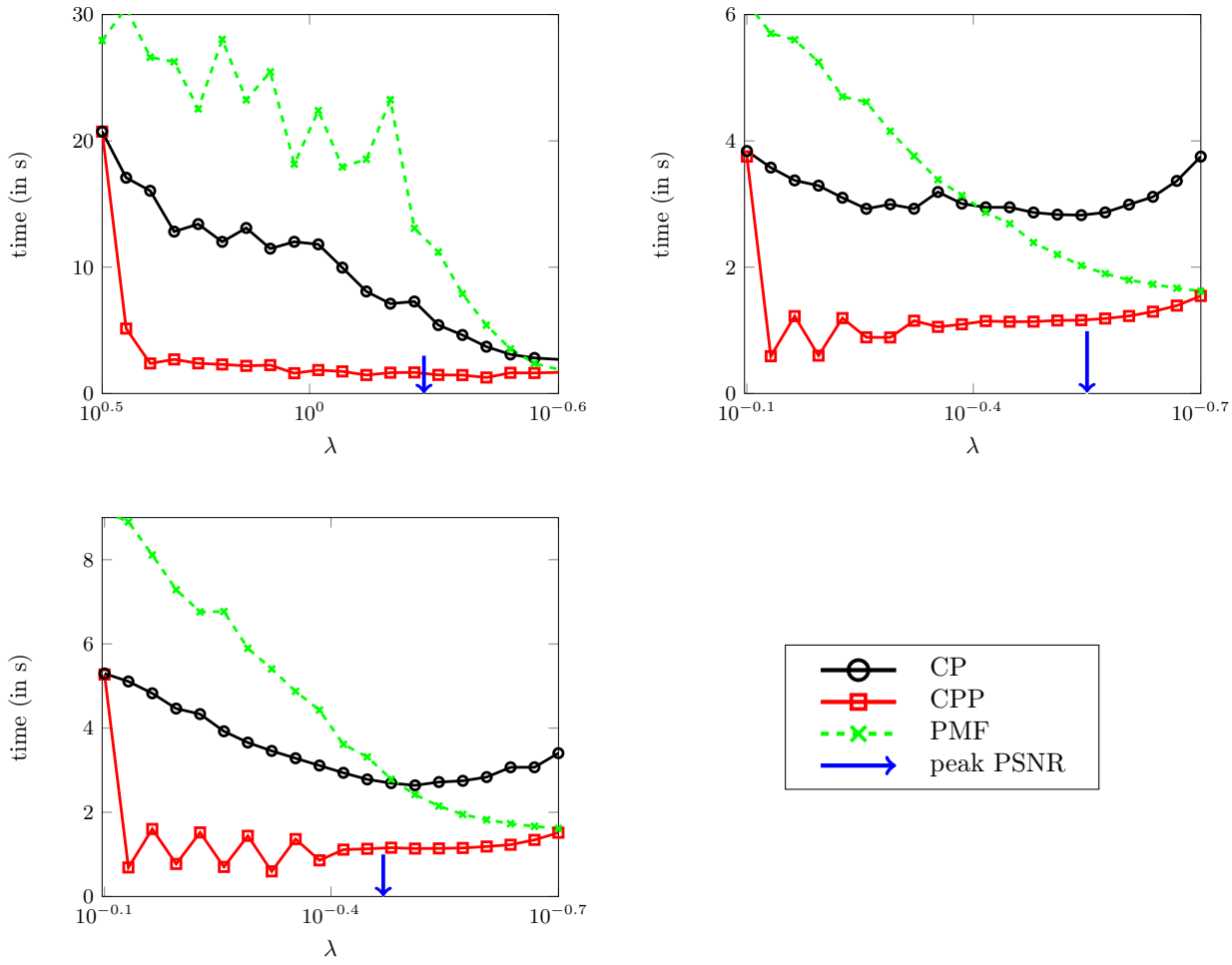


Figure 9: Time in seconds necessary to solve the problem regularized with a given λ (from the warm-start initialization when applicable) with a relative primal suboptimality gap of 10^{-5}, for regularly sampled values of λ along the regularization path. The competing methods are Cut Pursuit (CP), Cut Pursuit with warm start (CPP) and the parametric max-flow solver (PMF) for different 512×512 noisy images: simulated example (left), Lena (middle) and eagle (right). The computation times are averaged over 10 random degradations of the images by uniform noise. The blue arrow indicates the best PSNR value.

to converge, it does in practice and the local minima are shown by Boykov et al. (2001b) to be within a multiplicative constant of the global optimum.

Non-convex relaxation ($\mathbf{TV}_{0.5}$). We implemented a non-convex relaxation of the Mumford-Shah functional which is a "concave" version of the total variation, such as the adaptive Lasso (Zou, 2006) with $t \mapsto (\epsilon + t)^{\frac{1}{2}}$ in lieu of $t \mapsto |t|$. The resulting functional can be minimized locally using a reweighted TV scheme described in Ochs et al. (2015). We use our Cut Pursuit algorithm to solve each reweighted TV problems as it is the fastest implementation.

ℓ_0-Cut Pursuit We implemented three versions of ℓ_0 cut pursuit with different backward steps. In the simplest instantiation, ℓ_0-CPf, no backward step is used and the reduced graph can only increase in size. In ℓ_0-CPm, described in Algorithm 2, the simple merge step is performed after each round of cuts. Finally in ℓ_0-CPs, described in Algorithm 3, merge steps are replaced by merge-resplit steps but without priority queue.

Method	Simulated	Lena	Eagle
CPP	59	25	27
CP	194	62	70
PMF	356	67	91

Figure 10: Time in seconds necessary to compute the entire approximate regularization path at a relative primal suboptimality gap of 10^{-5} for the different algorithms, averaged over 10 samplings of the noise.

After a few preliminary experiments, we chose not to include either level-set methods (Chan and Vese, 2001) or active contour methods based on solving Euler-Lagrange equations (Kass et al., 1988) as their performances were much lower than the algorithms we consider.

Comparing speed results of code is always delicate as the degree of code optimization varies from one implementation to another. The α-expansion code uses the implementation of Fulkerson et al. (2009) which is a highly optimized code, ℓ_0-CPf and ℓ_0-CPm are implemented in C++, while ℓ_0-CPs and $\text{TV}_{0.5}$ are implemented in Matlab with a heavy use of mex-files. Even if minor improvements could be obtained on the latter, we believe that it would not change much the performances. In particular, a justification for direct time comparisons here is that computation time for each of the algorithms is mostly spent computing min cuts which is done in all codes using the same implementation of Boykov and Kolmogorov (2004) and which accounts for most of the computation time.

4.2.3 Results

Given that the MPP is hard, and that all the algorithms we consider only find local minima, we compare the different algorithms both in terms of running time and in terms of the objective value of the local minima found. The marks on the curves correspond to one iteration of each of the considered algorithms: For $\text{TV}_{0.5}$ there is a mark for each reweighted TV problem to solve, for CRFk, a mark corresponds to one α-expansion step, i.e. solving k max-flow problems. For ℓ_0-CP this corresponds to one forward (split) and one backward step. For clarity, the large number of marks were omitted in the third experiment, as well as for ℓ_0-CPs in the first experiment.

We report in Figure 11 the energy obtained by the different algorithms normalized by the energy of the best constant approximation. We can see that our algorithms find local optima that are essentially as good or better than α-expansions for the discretized problem in less time, as long as the solutions are sufficiently sparse. For the population density data the implementation ℓ_0-CPm with simple merge is faster and finds a better local minimum than CRF40, but is outperformed by CRF60. The implementation with swaps merge-resplit (ℓ_0-CPs) is on par with CRF60 when it comes to speed, and finds a slightly better minimum.

The simple merge step provides with a better solution than the purely forward approach at the cost of a slight increase in computational time. The merge-resplit backward step improves the quality of the solution further, but comes with a significant increase in computation.

We report in Figure 14 of appendix D the performance of approximations with CRFs solved with iterative α-expansions for different numbers of quantization levels, as compared to the performance of ℓ_0-CPm. We observe that although CRFs can outperform ℓ_0-CPm in term of quality of the local minima found for some of the higher numbers of quantization levels, the performances are very unstable with respect to this number. The fact that ℓ_0-CP does not rely on a priori quantized level leads to overall good performance, with significantly faster computation times. Plotting the corresponding PSNR shows that the smaller local minima of the objective found correlates well with gain in PSNR. It is interesting to note however that small improvements of the objective, which could be assessed as negligible, can yield unexpectedly high improvements in PSNR, as illustrated in Table 13.

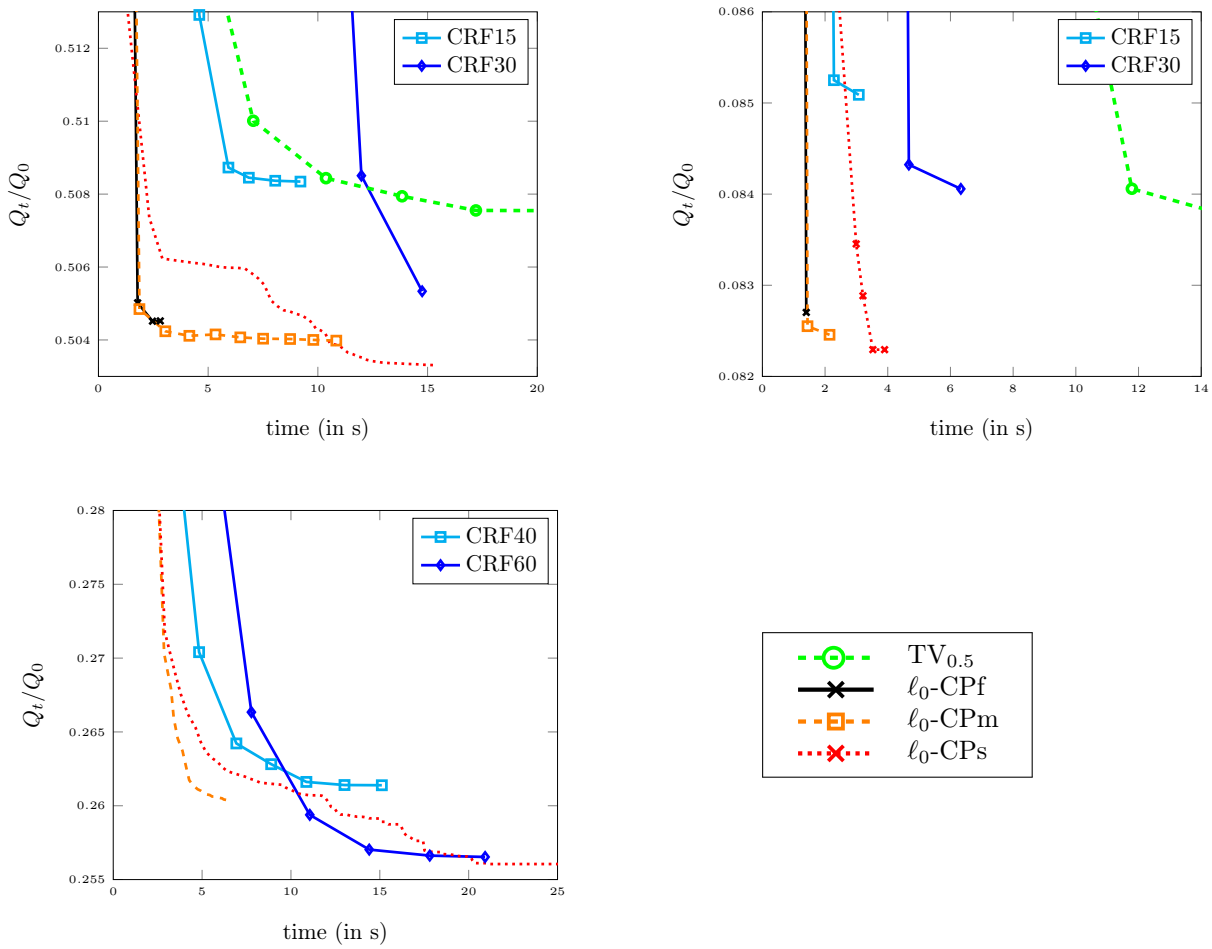
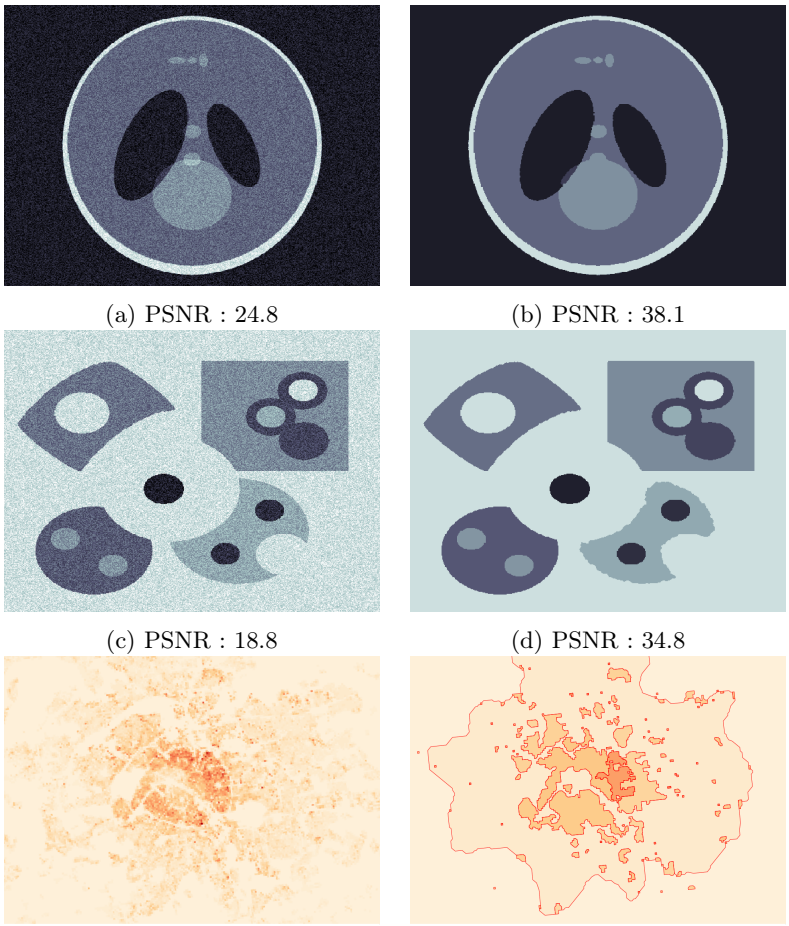


Figure 11: Mumford-Shah energy at time t (in seconds) divided by the same energy for the best constant approximation obtained by different algorithms: Non-convex relaxation ($\mathrm{TV}_{0.5}$), ℓ_0-CPf with no backward step, ℓ_0-CPm with simple merge step, ℓ_0-CPs with merge-resplit steps, and finally, α-expansions with different number of levels of quantization (see image legends), for different images: the Shepp-Logan phantom (left), our simulated example (middle) and the map simplification problem (right). Markers corresponds respectively to one reweighting, one α-expansion cycle and one cut for ($\mathrm{TV}_{0.5}$), (CRF) and (ℓ_0-CP).



(a) PSNR : 24.8 (b) PSNR : 38.1

(c) PSNR : 18.8 (d) PSNR : 34.8

(e) Population density of Paris (f) Simplified map

Figure 12: Benchmark on the denoising task. First two lines: (left) noisy images, (right) images retrieved by Cut Pursuit (CP). Last line: (left) rasterized population density of Paris area, (right) simplified map obtained by ℓ_0-Cut Pursuit with simple merge steps (ℓ_0-CPm): 69% of variance explained with 1.2% of contours length.

Experiment	Phantom		Simulated	
Algorithm	PSNR	time	PSNR	time
Noisy image	16.8	-	16.8	-
ℓ_0-CP	**33.5**	**4.3**	**37.0**	4.6
CRF20/CRF8	32.6	8.6	34.2	**4.0**
CRF40/CRF12	33.3	25.3	34.8	11.4
$TV_{0.5}$	32.2	16.4	33.6	18.0

Figure 13: PSNR at convergence and time to converge in seconds for the four algorithms as well as the noisy image for the first two denoising experiments.

5 Conclusion

We proposed two algorithms to minimize functions penalized respectively by the total variation and by the Mumford-Shah boundary size. They exploit computationally the fact that for sufficiently large regularization coefficients, the solution is typically piecewise constant with a small number of pieces, corresponding to a coarse partition. This is a consequence of the fact that, in the discrete setting, both the total variation and the Mumford-Shah boundary size penalize the size of the support of the gradient: indeed, functions with sparse gradients tend to have a small number of distinct level sets, that are moreover connected. The sparsity that is optimized is thus not exactly the same as the sparsity which is exploited computationally, although both are related.

By constructing a sequence of approximate solutions that are themselves piecewise constant with a small number of pieces, the proposed algorithms operate on reduced problems that can be solved efficiently, and perform only graph cuts on the original graph, which are thus the remaining bottleneck for further speed-ups. Like all working set algorithms, the cut pursuit variants are not competitive if the solution has too many connected level-sets.

In the convex case, cut pursuit outperforms all proximal methods for deblurring images with simple solutions. For denoising with a ROF energy, it outperforms the parametric maxflow approach when computing sequences of solutions for different regularization strengths. In the ℓ_0 case, our algorithm can find better solution in a shorter time than the non-convex continuous relaxation approach as well as the approach based on α-expansions. Furthermore, while the performance of the latter hinges critically on setting an appropriate number of level-sets in advance, cut pursuit needs no such parametrization.

Future developments will consider the case of Lovász extensions of other symmetric submodular functions (Bach, 2011) and to the multivariate case. It would also be interesting to determine the conditions under which the alternating scheme presented in 3.1.1 provides with a globally optimal solution of (8), as it would be a necessary step in order to prove approximation guarantees to the solution of ℓ_0-cut pursuit itself.

References

Aubert, G., Barlaud, M., Faugeras, O., and Jehan-Besson, S. (2003). Image segmentation using active contours: calculus of variations or shape gradients? *SIAM Journal on Applied Mathematics*, 63(6):2128–2154.

Bach, F. (2013). Learning with submodular functions: a convex optimization perspective. *Foundations and Trends in Machine Learning*, 6(2-3):145–373.

Bach, F., Jenatton, R., Mairal, J., and Obozinski, G. (2012). Optimization with sparsity-inducing penalties. *Foundations and Trends in Machine Learning*, 4(1):1–106.

Bach, F. R. (2011). Shaping level sets with submodular functions. In *Advances in Neural Information Processing Systems*, pages 10–18.

Bellman, R. (1973). A note on cluster analysis and dynamic programming. *Mathematical Biosciences*, 18(3):311–312.

Bleakley, K. and Vert, J.-P. (2011). The group fused Lasso for multiple change-point detection. *arXiv preprint arXiv:1106.4199*.

Boykov, Y. and Kolmogorov, V. (2004). An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137.

Boykov, Y., Veksler, O., and Zabih, R. (2001a). Efficient approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1222–1239.

Boykov, Y., Veksler, O., and Zabih, R. (2001b). Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239.

Bresson, X., Esedoḡlu, S., Vandergheynst, P., Thiran, J.-P., and Osher, S. (2007). Fast global minimization of the active contour/snake model. *Journal of Mathematical Imaging and Vision*, 28(2):151–167.

Chambolle, A., Caselles, V., Cremers, D., Novaga, M., and Pock, T. (2010). An introduction to total variation for image analysis. In *Theoretical foundations and numerical methods for sparse recovery*, pages 263–340. De Gruyter.

Chambolle, A. and Darbon, J. (2009). On total variation minimization and surface evolution using parametric maximum flows. *International Journal of Computer Vision*, 84(3):288–307.

Chambolle, A. and Pock, T. (2011). A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, 40(1):120–145.

Chan, T., Esedoḡlu, S., Park, F., and Yip, A. (2005). Recent developments in total variation image restoration. In *Mathematical Models of Computer Vision*, pages 17–31. Springer Verlag.

Chan, T. F. and Vese, L. A. (2001). Active contours without edges. *IEEE Transactions on Image Processing*, 10(2):266–277.

Chandrasekaran, V., Recht, B., Parrilo, P. A., and Willsky, A. S. (2012). The convex geometry of linear inverse problems. *Foundations of Computational mathematics*, 12(6):805–849.

Chen, S., Cowan, C. F., and Grant, P. M. (1991). Orthogonal least squares learning algorithm for radial basis function networks. *IEEE Transactions on Neural Networks*, 2(2):302–309.

Dinkelbach, W. (1967). On nonlinear fractional programming. *Management Science*, 13(7):492–498.

Efron, B., Hastie, T., Johnstone, I., Tibshirani, R., et al. (2004). Least angle regression. *The Annals of statistics*, 32(2):407–499.

Efroymson, M. (1960). Multiple regression analysis. *Mathematical methods for digital computers*, 1:191–203.

El-Zehiry, N. and Grady, L. (2011). Discrete optimization of the multiphase piecewise constant Mumford-Shah functional. In *Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 233–246. Springer.

El-Zehiry, N., Sahoo, P., and Elmaghraby, A. (2011). Combinatorial optimization of the piecewise constant Mumford-Shah functional with application to scalar/vector valued and volumetric image segmentation. *Image and Vision Computing*, 29(6):365–381.

El-Zehiry, N. Y. and Elmaghraby, A. (2007). Brain MRI tissue classification using graph cut optimization of the Mumford–Shah functional. In *Proceedings of the International Vision Conference of New Zealand*, pages 321–326.

Friedman, J., Hastie, T., and Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22.

Fulkerson, B., Vedaldi, A., and Soatto, S. (2009). Class segmentation and object localization with superpixel neighborhoods. In *Proceedings of the International Conference on Computer Vision*, pages 670–677. IEEE.

Geman, D. and Reynolds, G. (1992). Constrained restoration and the recovery of discontinuities. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 14(3):367–383.

Goldfarb, D. and Yin, W. (2009). Parametric maximum flow algorithms for fast total variation minimization. *SIAM Journal on Scientific Computing*, 31(5):3712–3743.

Harchaoui, Z., Juditsky, A., and Nemirovski, A. (2015). Conditional gradient algorithms for norm-regularized smooth convex optimization. *Mathematical Programming*, 152(1–2):75–112.

Ishikawa, H. (2003). Exact optimization for Markov random fields with convex priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(10):1333–1336.

Jaggi, M. (2013). Revisiting Frank-Wolfe: projection-free sparse convex optimization. In *Proceedings of the 30th International Conference on Machine Learning*, pages 427–435.

Jegelka, S., Bach, F., and Sra, S. (2013). Reflection methods for user-friendly submodular optimization. In *Advances in Neural Information Processing Systems*, pages 1313–1321.

Kass, M., Witkin, A., and Terzopoulos, D. (1988). Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331.

Kohli, P. and Torr, P. H. (2005). Efficiently solving dynamic Markov random fields using graph cuts. In *International Conference on Computer Vision (ICCV)*, volume 2, pages 922–929. IEEE.

Kolmogorov, V. and Zabih, R. (2004). What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):147–159.

Kormylo, J. J. and Mendel, J. M. (1982). Maximum likelihood detection and estimation of Bernoulli-Gaussian processes. *IEEE Transactions on Information Theory*, 28(3):482–488.

Kumar, K. and Bach, F. (2015). Active-set methods for submodular optimization. *arXiv preprint arXiv:1506.02852*.

Lorenz, D. A. and Pock, T. (2014). An inertial forward-backward algorithm for monotone inclusions. *Journal of Mathematical Imaging and Vision*, 51(2):311–325.

Mallat, S. and Zhang, Z. (1992). Adaptive time-frequency decomposition with matching pursuits. In *Time-Frequency and Time-Scale Analysis, Proceedings of the IEEE-SP International Symposium*, pages 7–10. IEEE.

Mumford, D. and Shah, J. (1989). Optimal approximations by piecewise smooth functions and associated variational problems. *Communications on pure and applied mathematics*, 42(5):577–685.

Needell, D. and Tropp, J. A. (2009). CoSaMP: Iterative signal recovery from incomplete and inaccurate samples. *Applied and Computational Harmonic Analysis*, 26(3):301–321.

Negahban, S., Yu, B., Wainwright, M. J., and Ravikumar, P. K. (2009). A unified framework for high-dimensional analysis of m-estimators with decomposable regularizers. In *Advances in Neural Information Processing Systems*, pages 1348–1356.

Nesterov, Y. (2007). Gradient methods for minimizing composite objective function. Technical report, Université catholique de Louvain, Center for Operations Research and Econometrics (CORE).

Nielsen, F. and Nock, R. (2014). Optimal interval clustering: Application to Bregman clustering and statistical mixture learning. *Signal Processing Letters*, 21(10):1289–1292.

Obozinski, G., Taskar, B., and Jordan, M. (2006). Multi-task feature selection. *Statistics Department, UC Berkeley, Tech. Rep.*

Ochs, P., Dosovitskiy, A., Brox, T., and Pock, T. (2015). On iteratively reweighted algorithms for nonsmooth nonconvex optimization in computer vision. *SIAM Journal on Imaging Sciences*, 8(1):331–372.

Osher, S. and Sethian, J. A. (1988). Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79(1):12–49.

Picard, J.-C. and Ratliff, H. D. (1975). Minimum cuts and related problems. *Networks*, 5(4):357–370.

Pock, T. and Chambolle, A. (2011). Diagonal preconditioning for first order primal-dual algorithms in convex optimization. In *Proceeding of the International Conference on Computer Vision (ICCV)*, pages 1762–1769. IEEE.

Raguet, H., Fadili, J., and Peyré, G. (2013). A generalized forward-backward splitting. *SIAM Journal on Imaging Sciences*, 6(3):1199–1226.

Raguet, H. and Landrieu, L. (2015). Preconditioning of a generalized forward-backward splitting and application to optimization on graphs. *SIAM Journal on Imaging Sciences*, 8(4):2706–2739.

Rao, N., Shah, P., and Wright, S. (2015). Forward–backward greedy algorithms for atomic norm regularization. *IEEE Transactions on Signal Processing*, 63(21):5798–5811.

Rockafellar, R. T. (1970). *Convex analysis*. Princeton University Press.

Roth, V. and Fischer, B. (2008). The group-lasso for generalized linear models: uniqueness of solutions and efficient algorithms. In *Proceedings of the 25th international conference on Machine learning*, pages 848–855. ACM.

Rudin, L. I., Osher, S., and Fatemi, E. (1992). Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60:259 – 268.

Santner, J., Pock, T., and Bischof, H. (2011). Interactive multi-label segmentation. In *Proceedings of the Asian Conference on Computer Vision*, pages 397–410. Springer.

Shepp, L. A. and Logan, B. F. (1974). The Fourier reconstruction of a head section. *IEEE Transactions on Nuclear Science*, 21(3):21–43.

Soussen, C., Idier, J., Brie, D., and Duan, J. (2011). From Bernoulli–Gaussian deconvolution to sparse signal restoration. *IEEE Transactions on Signal Processing*, 59(10):4572–4584.

Szeliski, R., Zabih, R., Scharstein, D., Veksler, O., Kolmogorov, V., Agarwala, A., Tappen, M., and Rother, C. (2006). A comparative study of energy minimization methods for Markov random fields. In *Proceeding of the European Conference in Computer Vision (ECCV)*, pages 16–29. Springer.

Tsai, Y.-H. R. and Osher, S. (2005). Total variation and level set methods in image science. *Acta Numerica*, 14:509–573.

Vese, L. A. and Chan, T. F. (2002). A multiphase level set framework for image segmentation using the Mumford and Shah model. *International Journal of Computer Vision*, 50(3):271–293.

Wang, Y.-X., Sharpnack, J., Smola, A., and Tibshirani, R. J. (2014). Trend filtering on graphs. *arXiv preprint arXiv:1410.7690*. To appear in JMLR.

Zhang, T. (2009). Adaptive forward-backward greedy algorithm for sparse learning with linear models. In *Advances in Neural Information Processing Systems*, pages 1921–1928.

Zou, H. (2006). The adaptive lasso and its oracle properties. *Journal of the American Statistical Association*, 101(476):1418–1429.

A The total variation as an atomic gauge

It is well known that the total variation is the Lovász extension of the submodular function $F : B \mapsto w(B, B^c)$ (see Bach, 2013, chap. 6.2). The *base polytope* associated with F is the set $\mathcal{B}_F \doteq \{s \in \mathbb{R}^n \mid s(B) \leq F(B), B \subset V,\ s(V) = F(V)\}$, where $s(B) \doteq \sum_{i \in B} s_i$. For any submodular function F such that $F(\varnothing) = F(V) = 0$, which is true in particular for all symmetric submodular functions, the Lovász extension γ_F is a *gauge function* which is the *support function*[9] of $\mathcal{B}_F$: $\gamma_F(x) = \max_{s \in \mathcal{B}_F} \langle s, x \rangle$ and its *polar gauge* is the gauge of $\mathcal{B}_F$ (Bach, 2011). The total variation is thus a gauge function and its polar gauge is TV° with

$$\mathrm{TV}^\circ(s) = \begin{cases} \max\limits_{\varnothing \subsetneq B \subsetneq V} \dfrac{s(B)}{w(B, B^c)} & \text{if } s(V) = 0 \\ +\infty & \text{else.} \end{cases}$$

Chandrasekaran et al. (2012) have recently introduced the concept of *atomic gauge*. Given a closed set $\mathcal{A} \subset \mathbb{R}^n$ whose elements are called *atoms*, the associated atomic gauge is the gauge $\gamma_{\mathcal{A}}$ of the convex hull $C_{\mathcal{A}}$ of $\mathcal{A} \cup \{0\}$, i.e. $\gamma_{\mathcal{A}}(x) \doteq \inf\{t \,|\, x \in t\, C_{\mathcal{A}}\}$. The polar gauge is the support function of $\mathcal{A} \cup \{0\}$, that is $\gamma^\circ_{\mathcal{A}}(s) = \sup_{a \in \mathcal{A} \cup \{0\}} \langle a, s \rangle$. Given that $\mathcal{A} \subset \mathbb{R}^n$, using Caratheodory's theorem, we have that

$$\gamma_{\mathcal{A}}(x) = \inf \big\{ \textstyle\sum_{a \in \mathcal{A}} c_a \mid \forall a \in \mathcal{A}, c_a \geq 0,\ \sum_{a \in \mathcal{A}} c_a\, a = x \big\}.$$

Regularizing with an atomic gauge thus favors solutions that are sparse combinations of atoms, which motivated the use of algorithms that exploit the sparsity of the solution computationally (Jaggi, 2013; Rao et al., 2015). It is clear from previous definitions that Lovász extensions are atomic gauges. In particular the total variation is the atomic gauge associated with the set of atoms $\mathcal{A} = \big\{w(B, B^c)^{-1}\mathbf{1}_B + \mu \mathbf{1}_V\big\}_{B \notin \{\varnothing, V\},\, \mu \in \mathbb{R}}$ or equivalently the set $\mathcal{A}' = \big\{\frac{1}{2} w(B, B^c)^{-1}(\mathbf{1}_B - \mathbf{1}_{B^c}) + \mu' \mathbf{1}_V\big\}_{B \notin \{\varnothing, V\},\, \mu' \in \mathbb{R}}$. Expressing solutions to problem regularized with the total variation as combinations of set indicators or cuts as we propose to do in this paper is thus very natural from this perspective.

For the total variation, the Frank-Wolfe direction associated to $s = -\nabla f(x)$ such that $\langle s, \mathbf{1}_V \rangle = 0$ is

$$\operatorname*{arg\,max}_{\xi : \mathrm{TV}(\xi) \leq 1} \langle s, \xi \rangle = \operatorname*{arg\,max}_{1_B : B \notin \{\varnothing, V\}} \frac{1}{w(B, B^c)} \langle s, \mathbf{1}_B \rangle, \tag{10}$$

since the maximizer is necessarily an extreme point of the set $\{\xi \mid \mathrm{TV}(\xi) \leq 1\}$ and therefore among the atoms.

B Proof of Propositions 1 and 2

Proposition 1. *For $x \in \mathbb{R}^n$, if we set $S = S(x)$ then*

$$Q'(x, \mathbf{1}_B) = \langle \nabla Q_S(x), \mathbf{1}_B \rangle + \lambda w_{S^c}(B, B^c).$$

Moreover if $\langle \nabla f(x), \mathbf{1}_B \rangle = 0$ then

$$Q'(x, u_B) = (\gamma_B + \gamma_{B^c})\, Q'(x, \mathbf{1}_B).$$

Proof. For $B \subset V$ we have that $Q'(x, \mathbf{1}_B) = \langle \nabla Q_S(x), \mathbf{1}_B \rangle + \sup_{\epsilon \in \partial \mathrm{TV}|_{S^c}(x)} \langle \epsilon, \mathbf{1}_B \rangle$. This can be shown using the chain rule for subgradients that we have:

$$\partial \mathrm{TV}|_{S^c}(x) = \big\{ \tfrac{1}{2} D^\intercal \delta \mid \delta_S = 0,\ \|\delta_{S^c}\|_\infty \leq 1,\ \forall (i,j) \in E, \delta_{ij} = -\delta_{ji} \big\},$$

with $D \in \mathbb{R}^{2m \times n}$ the matrix whose only non-zero entries are $D_{(i,j),i} = w_{ij}$ and $D_{(i,j),j} = -w_{ij}$ for all $(i,j) \in E$, and with the notations $\delta_S \in \mathbb{R}^{2m}$ and $\delta_{S^c} \in \mathbb{R}^{2m}$ for the vectors whose entries are equal to those of δ respectively on S and S^c and equal to zero otherwise.

[9] See Rockafellar (1970) for definitions of *gauge*, *polar gauge* and *support function* of a set.

Therefore if $\epsilon = \frac{1}{2} D^\intercal \delta_{S^c}$ then

$$\langle \epsilon, \mathbf{1}_B \rangle = \langle \tfrac{1}{2}\delta_{S^c}, D\mathbf{1}_B \rangle = \frac{1}{2} \sum_{(i,j)\in S^c} \delta_{ij} w_{ij} ([\mathbf{1}_B]_i - [\mathbf{1}_B]_j)$$

so that $\sup_{\epsilon \in \partial \,\mathrm{TV}|_{S^c}(x)} \langle \epsilon, \mathbf{1}_B \rangle = w_{S^c}(B, B^c)$.

For the second statement, we have that

$$Q'(x, u_B) = \langle \nabla Q_S(x), u_B \rangle + \sup_{\epsilon \in \partial \mathrm{TV}|_{S^c}(x)} \langle \epsilon, u_B \rangle.$$

But letting $g = Q_S(x)$, and given that $\langle g, 1\rangle$=0 implies that $\langle g, \mathbf{1}_{B^c}\rangle = \langle g, \mathbf{1} - \mathbf{1}_B\rangle = -\langle g, \mathbf{1}_B\rangle$, we have

$$\langle g, u_B\rangle = \gamma_B \langle g, \mathbf{1}_B\rangle - \gamma_{B^c}\langle g, \mathbf{1}_{B^c}\rangle = (\gamma_B + \gamma_{B^c})\langle g, \mathbf{1}_B\rangle.$$

Similarly, $\langle \epsilon, u_B\rangle = \langle \frac{1}{2}\delta_{S^c}, Du_B\rangle = \frac{1}{2}\gamma_B \langle \delta_{S^c}, D\mathbf{1}_B\rangle - \frac{1}{2}\gamma_{B^c}\langle \delta_{S^c}, D\mathbf{1}_{B^c}\rangle = \frac{1}{2}(\gamma_B + \gamma_{B^c})\langle \delta_{S^c}, D\mathbf{1}_B\rangle$ because $D\mathbf{1}_B = -D\mathbf{1}_{B^c}$. Taking the supremum over ϵ then proves the result. □

Proposition 2. *We have* $x = \arg\min_{z\in\mathbb{R}^n} Q(z)$ *if and only if* $\min_{B\subset V} Q'(x, \mathbf{1}_B) = 0$ *and* $Q'(x, \mathbf{1}_V) = 0$.

Proof. ($\Rightarrow$) If x is the solution of problem (1), the directional derivative of Q along any direction must be nonnegative, which implies that $Q'(x, \mathbf{1}_B) \geq 0$ for all B. But $\min_{B\subset V} Q'(x, \mathbf{1}_B) \leq Q'(x, \mathbf{1}_\varnothing) = 0$, which proves the first part. Then since $w(V, \varnothing) = 0$ we have $Q'(x, \mathbf{1}_V) = \langle \nabla Q_S(x), \mathbf{1}_V\rangle$, and, in fact, since all elements of the subgradient of $\mathrm{TV}|_{S^c}$ are orthogonal to $\mathbf{1}_V$ we also have $Q'(x, -\mathbf{1}_V) = -\langle \nabla Q_S(x), \mathbf{1}_V\rangle$. So $0 \leq Q'(x, -\mathbf{1}_V) = -Q'(x, \mathbf{1}_V) \leq 0$.

($\Leftarrow$) Conversely we assume that $\min_{B\subset V} Q'(x, \mathbf{1}_B) = 0$ and $Q'(x, \mathbf{1}_V) = 0$.
Since $Q'(x, \mathbf{1}_V) = 0$ and since $w_{S^c}(V, \varnothing) = 0$ we have $\langle \nabla Q_S(x), \mathbf{1}_V\rangle = 0$. Now, for any set A which is a maximal connected component of $G|_{S^c} \doteq (V, S^c)$, we also have $w_{S^c}(A, A^c) = 0$ so that $0 \leq Q'(x, \mathbf{1}_A) = \langle \nabla Q_S(x), \mathbf{1}_A\rangle$ but the same holds for the complement A^c and $\langle \nabla Q_S(x), \mathbf{1}_A\rangle + \langle \nabla Q_S(x), \mathbf{1}_{A^c}\rangle = \langle \nabla Q_S(x), \mathbf{1}_V\rangle = 0$ so that $\langle \nabla Q_S(x), \mathbf{1}_A\rangle = 0$.

As a consequence the capacities of the graph G_{flow} defined in (3) of the article are such that, for any set A which is a maximal connected component of $G|_{S^c}$, we have

$$\sum_{i\in\nabla_+\cap A} c_{si} = \sum_{i\in\nabla_-\cap A} c_{it}. \tag{11}$$

Then since $Q'(x, \mathbf{1}_\varnothing) = 0$ and since $\min_{B\subset V} Q'(x, \mathbf{1}_B) = 0$ it is a minimizing argument. The characterization of the steepest partition as a minimal cut then guarantees that there exists a minimal cut in G_{flow} which does not cut any edge in S^c and isolates the source and the rest of the graph. Given equality (11), the set of minimal cuts are the cuts that remove indifferently for each maximal connected component A either all edges $\{(s, i)\}_{i\in A}$ or the edges $\{(i, t)\}_{i\in A}$.

A consequence of the max-flow/min-cut duality is that to this cut corresponds a maximal flow $e \in \mathbb{R}^{2m}$ in G_{flow}. This flow is such that it is saturated at the minimal cut, and we thus have $e_{si} = c_{si}$ for all $i \in \nabla_+$ and $e_{it} = c_{it}$ for all $i \in \nabla_-$, again because of equation (11).

Writing flow conservation yields

$$\begin{cases} e_{si} + \sum_{j\in N_i}(e_{ji} - e_{ij}) = 0 & \forall i \in \nabla_+ \\ -e_{it} + \sum_{j\in N_i}(e_{ji} - e_{ij}) = 0 & \forall i \in \nabla_-, \end{cases} \tag{12}$$

with $N_i = \{j | (i, j) \in S^c\}$.

By replacing e_{si} and e_{it} by their value, the flow conservation (12) at node i rewrites

$$\begin{aligned} \nabla_i Q_S(x) + \sum_{j \in N_i} \lambda w_{ij} \delta_{ij} &= 0 \\ \nabla_i Q_S(x) + \frac{1}{2} \sum_{j \in N_i} \lambda w_{ij} \left(\delta_{ij} - \delta_{ji}\right) &= 0, \end{aligned} \tag{13}$$

with $\delta_{ij} = \frac{e_{ji} - e_{ij}}{\lambda w_{ij}}$ for $(i,j) \in S^c(x)$ and $\delta_{ij} = \delta_{ji} = 0$ for all edges $(i,j) \in S(x)$. The flow e must respect the capacity at all edges and hence $0 \leq e_{ij} \leq c_{ij} = \lambda w_{ij}$ for all edges in $S^c(x)$. Since the flow is maximal, only one of e_{ij} or e_{ji} is non zero. Hence δ we naturally have $\delta_{ij} = -\delta_{ji}$, and $|\delta_{ij}| \leq 1$. But we can rewrite (13) as $\nabla Q_S(x) = \frac{1}{2}\lambda D^\intercal \delta$ with $\delta_S = 0$ and $\|\delta_{S^c}\| \leq 1$ with D as in the characterization of the subgradient of $\mathrm{TV}|_{S^c}$ which shows that $-\frac{1}{\lambda}\nabla Q_S(x) \in \partial \mathrm{TV}|_{S^c}(x)$ thus that $0 \in \partial Q(x)$, and finally that x minimizes Q. □

Remark: We proved Proposition 2 using directly the flow formulation and the simplest possible arguments. It is also possible to prove the result more directly using more abstract results. We actually used the fact that x is a minimum of Q if and only if, for $S = S(x)$, $-\frac{1}{\lambda}\nabla Q_S(x) \in \partial \mathrm{TV}|_{S^c}(x)$. But it is possible to give another representation of $\partial \mathrm{TV}|_{S^c}(x)$ using that the subgradient of a gauge γ at x is $\partial\gamma(x) = \{s \mid \langle x, s\rangle,\ \gamma^\circ(s) \leq 1\}$. Indeed, for $\gamma = \mathrm{TV}$, the set $\{\gamma^\circ(s) \leq 1\}$ is simply the submodular polytope $\mathcal{P}_F$ of $F : B \mapsto w(B, B^c)$. As a result $\partial \mathrm{TV}|_{S^c}(x) = \{s \in \mathbb{R}^n \mid \langle s, x\rangle = 1, \forall B,\ s(B) \leq w_{S^c}(B,B)\}$. But having that $\min_{B \subset V} \langle \nabla Q_S(x), \mathbf{1}_B\rangle + \lambda w_{S^c}(B, B^c) = 0$ is equivalent to having $-\frac{1}{\lambda}\nabla Q_S(x) \in \{s \in \mathbb{R}^n \mid \forall B,\ s(B) \leq w_{S^c}(B,B)\}$. There thus just remains to show that $\langle \nabla Q_S(x), x\rangle = \mathrm{TV}(x)$. Let Π_S denote the set of maximal connected components of $G|_{S^c} = (V, S^c)$, so that we have $x = \sum_{A \in \Pi_S} c_A \mathbf{1}_A$. Since $w_{S^c}(V, \varnothing) = 0$, we have $0 = Q'(x, \mathbf{1}_V) = \langle \nabla Q_S(x), 1_V\rangle$. Similarly for $A \in \Pi_S$, we have $w_{S^c}(A, A^c) = 0$, which entails that $\langle \nabla Q_S(x), 1_A\rangle \geq 0$. But then $-\langle \nabla Q_S(x), 1_A\rangle = \langle \nabla Q_S(x), 1_{A^c}\rangle \geq 0$ also, which proves $\langle \nabla Q_S(x), 1_A\rangle = 0$. Finally by linearity $\langle \nabla Q_S(x), x\rangle = \sum_{A \in \Pi_S} c_A \langle \nabla Q_S(x), 1_A\rangle = 0 = \mathrm{TV}|_{S^c}(x)$ which proves the result.

C Computation of the Frank-Wolfe direction

The computation of the Frank-Wolfe direction defined in (10) requires to optimize a ratio of combinatorial functions. More precisely, it requires to solve

$$\max_{B \notin \{\varnothing, V\}} \frac{N(B)}{D(B)} \quad \text{with} \quad N(B) \doteq -\langle \nabla f(x), \mathbf{1}_B\rangle, \quad \text{and} \quad D(B) \doteq w(B, B^c).$$

But $B \mapsto \frac{N(B)}{D(B)}$ it is the ratio of a supermodular function (in fact a modular function) and a nonnegative submodular function, which, as we explain in this appendix, can thus be minimized using a natural extension to combinatorial functions of the algorithm proposed by Dinkelbach (1967) to minimize the ratio of a convex function to a positive concave function.

We first consider the case where D is a positive function (which is not the case for the cut function since $D(\varnothing) = D(V) = 0$). We then have:

Lemma 6. *Let $N : 2^V \to \mathbb{R}$ and $D : 2^V \to \mathbb{R}_+ \setminus \{0\}$. We have that $\lambda_0 \doteq \frac{N(A_0)}{D(A_0)} = \max_{A \subset V} \frac{N(A)}{D(A)}$ if and only if $N(A_0) - \lambda_0 D(A_0) = \max_{A \subset V} N(A) - \lambda_0 D(A) = 0$.*

Proof. Let us define $A_0 \doteq \arg\min_{A \subset V} \frac{N(A)}{D(A)}$ and $\lambda_0 = \frac{N(A_0)}{D(A_0)}$. Since D is positive, we have

$$\begin{cases} N(A) - \lambda_0 D(A) & \leq 0, \quad \text{for all } A \subset V, \\ N(A_0) - \lambda_0 D(A_0) & = 0. \end{cases}$$

We conclude that A_0 is a maximizer of $N(A) - \lambda_0 D(A)$.

Conversely, let A_0 be such that $N(A_0) - \lambda_0 D(A_0) = \arg\max_{A \subset V} N(A) - \lambda_0 D(A) = 0$, and so, for all $A \subset V$ we have that $\frac{N(A)}{D(A)} \leq \lambda_0 = \frac{N(A_0)}{D(A_0)}$. □

This lemma from Dinkelbach (1967), shows that, up to the determination of λ_0, the original maximization problem is equivalent to the maximization of G_{λ_0} for $G_\lambda : A \mapsto N(A) - \lambda D(A)$. Moreover it is immediate that $\lambda \mapsto \max_A G_\lambda(A)$ is a nondecreasing function which is equal to 0 for λ_0, it is therefore easy to find λ_0 with a bisection algorithm.

The problem $\max_{A \subset V} G_\lambda(A)$ is easy to solve if G_λ is a supermodular function (Dinkelbach's paper considers the case of functions of real vectors and focusses on the case in which G is convex). But G_λ is supermodular for all $\lambda \in \mathbb{R}$ if and only if N is supermodular and D is submodular. In that case, the algorithm proposed by Dinkelbach is immediately applicable to our setting and we have the following result:

Proposition 7. *If N and D are respectively supermodular and submodular and if D is positive them Algorithm 4 is finitely convergent and converges to* $\arg\max_{A \subset V} \frac{N(A)}{D(A)}$.

Proof. The proof of this proposition follows the same arguments as the ones of Dinkelbach (1967). □

Algorithm 4: Dinkelbach's algorithm

Initialization: $\lambda_0 = 1, \lambda_{-1} = 0,\ t = 0$
while $\lambda_t \neq \lambda_{t-1}$ **do**
  $A_t \leftarrow \arg\max_{A \subset V} N(A) - \lambda_t D(A)$
  $\lambda_{t+1} \leftarrow \frac{N(A_t)}{D(A_t)}$
  $t \leftarrow t + 1$
return A_t

Proposition 8. *If $N : 2^V \to \mathbb{R}$, $D : 2^V \to \mathbb{R}_+$ and if there exists a set $Z \subset 2^V$ such that $Z = \{A \mid D(A) = 0\} = \{A \mid N(A) = 0\}$, if then $M \doteq \operatorname{Arg\,max}_{A \notin Z} \frac{N(A)}{D(A)}$, $M^\star \doteq \operatorname{Arg\,max}_{A \in M} N(A)$, and $A^\star \in M^\star$, if $\frac{N(A^\star)}{D(A^\star)} > 0$ then*

$$M^\star = \operatorname*{Arg\,max}_A \frac{N(A)}{D(A) + \eta} \quad \text{for all} \quad \forall \eta \ \text{ s.t. } \ 0 < \eta < \min_{B:\, N(B) < N(A^\star)} \frac{D(A^\star)D(B)}{N(A^\star) - N(B)} \left(\frac{N(B)}{D(B)} - \frac{N(A^\star)}{D(A^\star)} \right).$$

Proof. For any such η, it is easy to check that $N(A^\star)D(B) - N(B)D(A^\star) + \eta\,(N(A^\star) - N(B)) > 0$ for any $B \notin M^\star$, which yields the result by dividing this inequality by $(D(A^\star) + \eta)(D(B) + \eta)$ and noting that for any $A' \in M^\star$ we must have $N(A') = N(A^\star)$ and $D(A') = D(A^\star)$. □

By setting $Z = \{\varnothing, V\}$ in the previous proposition, we see that it is applies to the computation of the Frank-Wolfe direction for any point x such that $\langle \nabla f(x), \mathbf{1}_V \rangle = 0$, because $N(B) = -N(B^c)$ and $D(B) = D(B^c)$, which guarantees that the maximum is strictly positive. Proposition 7 then shows that the maximization is obtained by solving a sequence of problems of the form $\max_{B \in V} -\langle \nabla f(x), \mathbf{1}_B \rangle - \lambda w(B, B^c)$ which are of the exact same general form as (5) and are thus solved as max-flow problems. In practice the algorithm converges in a few iterations.

D CRF formulation and number of quantization levels

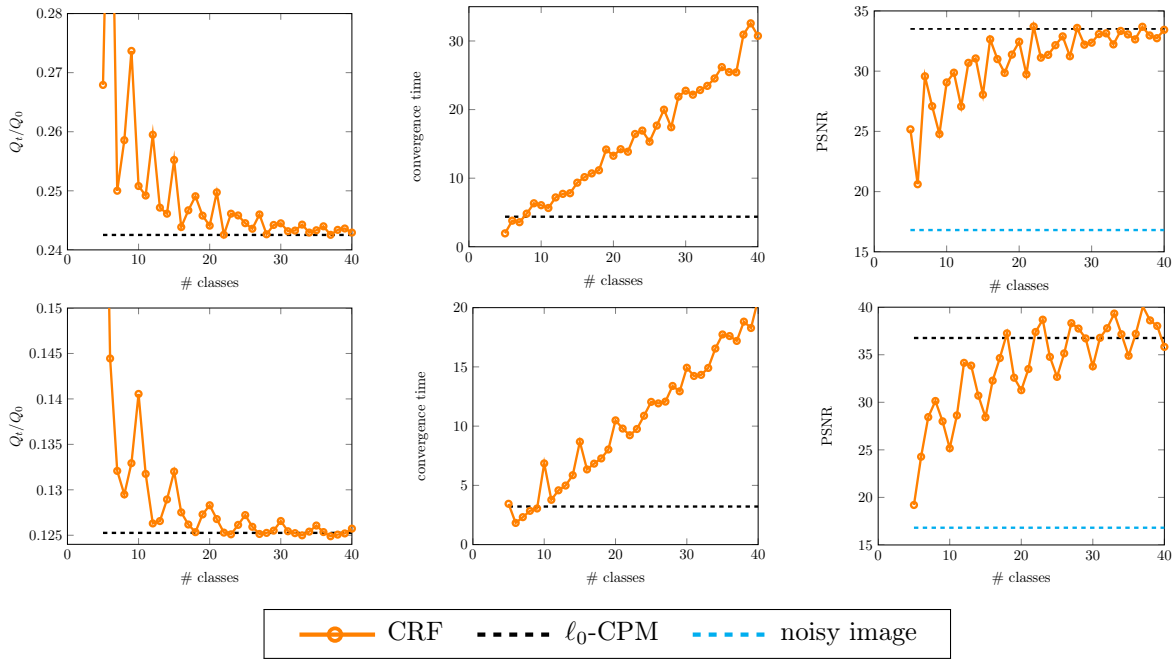


Figure 14: **Behavior of the CRF algorithm for different number of quantization levels** for the phantom (top) and the simulated data (bottom) averaged on 10 denoising experiments: (left) ratio between the energy Q at convergence and the energy at time 0, (middle) running time, (right) corresponding PSNRs. The two algorithms represented are α-expansions (CRF) for a varying number of quantization levels and ℓ_0-CPM.