HAL open science

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

Loic Landrieu, Guillaume Obozinski

To cite this version:

Loic Landrieu, Guillaume Obozinski. Cut Pursuit: fast algorithms to learn piecewise constant functions on general weighted graphs. SIAM Journal on Imaging Sciences, 2017. hal-01306779v3

HAL Id: hal-01306779

https://hal.science/hal-01306779v3

Submitted on 31 Jul 2017 (v3), last revised 21 Aug 2017 (v4)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution - NonCommercial 4.0 International License

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

Loic Landrieu [‡†] and Guillaume Obozinski [‡]

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 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.

Key words. working-set, total variation, sparsity, Mumford-Shah, greedy algorithm

AMS subject classifications. 90C25, 90C27, 65K10, 90C59, 68U10, 90C99, 62H11

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 [25] 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 [69]. 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 [62], hereafter referred to as ROF, which 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 [46], which proposed to penalize the total perimeter of discontinuities of piecewise smooth functions. A fairly large literature is devoted to these formulations mainly in the fields of image processing and optimization. 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, [13, 30] use the fact that the problem can be formulated as a parametric max-flow. [23] use graph cuts to solve the formulation of Mumford and Shah for the case of two constant 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 the corresponding large-scale optimization problem with significant

*Submitted to the editors 24/01/2017.

† Université Paris-Est, LASTIG MATIS, IGN, ENSG, F-94160 Saint-Mande, France (loic.landrieu@ign.fr.)

‡Université Paris-Est, LIGM, Laboratoire d'Informatique Gaspard Monge (UMR 8049), CNRS, ENPC, ESIEE Paris, UPEM (guillaume.obozinski@enpc.fr).

This manuscript is for review purposes only.

speed-ups [3]. 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 algorithms proposed to exploit sparsity are working set[1] algorithms and the related (fully corrective) Frank-Wolfe algorithm [31]. In the non-convex case, forward selection algorithms such as OMP, FoBa and others have been proposed [45, 47, 70][2].

It is well understood that algorithms for the convex and non-convex cases 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 problems regularized by the total variation while the greedy strategy solves problems penalized by the boundary size for piecewise constant functions. In the convex case, our algorithms do not apply only to the cases in which the data fitting term is the MSE or a separable smooth convex function, for which some efficient algorithms implicitly exploiting sparsity exist [13, 2, 41], 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 $f : \mathbb{R}^n \to \mathbb{R}$ convex and 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

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

[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.

Energies of this form were first introduced by [29] for image regularization, and are widely used for their inducing spatial regularity as well as preserving discontinuities. In this paper, we consider the case ϕ equal to the absolute value, which corresponds to the total variation (denoted TV), and the case ϕ equal to one minus the Kronecker delta at 0, which leads to the *total boundary size* penalty for piecewise constant functions. For these functions, 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.

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 $\mathrm{span}(\Pi) = \mathrm{span}(\mathbf{1}_{A_1}, \cdots, \mathbf{1}_{A_k}) = \{\sum_{i=1}^k c_i \mathbf{1}_{A_i} \mid c \in \mathbb{R}^k\}$. We denote

$$x_\Pi \doteq \underset{z \in \mathrm{span}(\Pi)}{\arg\min}\ Q(z), \tag{3}$$

the solution of (1) when x is constrained to be in $\mathrm{span}(\Pi)$. Assuming that the regularization strength is such that the solution $x^\star$ decomposes over a coarse partition, and that the constrained problem (3) is easy to solve for such a partition, problem (1) boils down to finding an optimal partition $\Pi^\star$:

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

An additional motivation to consider a sequence of partitions and solve sequentially problems with x constrained to $\mathrm{span}(\Pi)$ 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 [17] 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. [46] describe an image as *simple* if it can be expressed as a piecewise-smooth function with few and small discontinuities, that is if the space can partitioned in a finite number regions with short contours and such that the image varies smoothly in each of these regions.

Given an observed noisy image modeled as a function $J : \Omega \to \mathbb{R}$ whose domain Ω is an open, bounded and connected subset of $\mathbb{R}^2$, and assuming $J \in L^\infty$, Mumford and Shah propose to obtain a denoised version I of the image via the minimization of an energy which we can write as

$$\int_\Omega \big(I(x) - J(x)\big)^2 dx + \mu \int_{\Omega \backslash \Gamma} \|\nabla I(x)\|^2\, dx + \lambda \mathcal{H}_1(\Gamma), \tag{MS}$$

where μ and λ are two nonnegative regularization coefficients. It is composed of three terms: a fidelity term quantifying the distortion between I and J, a term measuring the smoothness of I outside of a one-dimensional set of discontinuities Γ, and finally the one-dimensional Hausdorff measure of this set $\mathcal{H}_1(\Gamma)$. David Mumford and Jayant Shah conjectured that this problem admitted a solution (I^*, Γ^*) such that I^* was continuously differentiable on a finite number k of open sets R_i with $\Gamma^* = \Omega \backslash \bigcup_i R_i$ a one dimensional set consisting of points connected by rectifiable arcs.

In subsequent formalisations of the Mumford-Shah problem, I is constrained to the set $\mathcal{C}^1(\Omega\backslash\Gamma)$ of continuously differentiable functions on $\Omega\backslash\Gamma$, where Γ is a closed set of Hausdorff dimension 1. Ennio De Giorgi proposed a relaxed Mumford-Shah problem in which I is constrained to the set $\mathrm{SBV}(\mathbb{R}^2)$ of special bounded total variation functions and $\Gamma = \mathcal{S}I$ is the *jump set* of I (for detailed presentations of the different formulations of the Mumford-Shah problem and their connections, see [28, 5]). When $\mu \to \infty$, the smoothness term forces I to be constant on the connected components of $\Omega\backslash\Gamma$.

If the number k of regions R_i (also called *phases*) on which I is constant is fixed to k, the corresponding problem is referred to as the *piecewise constant Mumford-Shah problem* and can be reformulated as:

$$\text{(PC-MS)} \qquad \min_{\Gamma, I} \sum_{i=1}^{k} \int_{R_i} \big(I_i - J(x)\big)^2 \, dx + \lambda \, \mathcal{H}_1(\Gamma),$$

with I_i the constant value of I on R_i and $\Omega = R_1 \cup \ldots R_k \cup \Gamma$. Note that when k is fixed, the sets R_i are not necessarily connected sets. Note that both (MS) and (PC-MS) extend naturally to d-dimensional images by replacing $\mathcal{H}_1$ by the $d-1$-dimensional Hausdorff measure $\mathcal{H}_{d-1}$.

The setting in which $k = 2$ is known as the Chan-Vese problem and was first approached algorithmically using active contour methods [36, 1]. [16] 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 [68]. The resolution of those problems is substantially sped up by the introduction of graph-cut methods, for the binary phase [25] and in the multiphase setting [23].

Clearly, a counterpart of (PC-MS) in which the number of phases is not set a priori (and can possibly be infinite) is also of interest. It has been introduced in the discrete setting by [42] and has been studied in the continuous setting using the theory of *Caccioppoli partitions* [66, 43].

Independently of the work of Mumford and Shah, [62] proposed 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 [12]. The variant[3] of the problem of Rudin, Osher and Fatemi (ROF) where the total variation is used as a regularizer—corresponding to the proximal problem of the total variation—can be

[3] In [62] the TV is minimized under a constraint on the L_2 distance between I and J.

written

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

where TV is the total variation and 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] 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.

The ROF problem can be solved very efficiently for chain graphs using dynamic programming [35] or exploiting the structure of the optimality conditions [19]. See [38] for a broader discussion. In the general case, a first approach is to consider explicitly the set of edges presenting discontinuities and iteratively update this set using calculus of variations based on the Euler-Lagrange equations [1]. This class of methods is known as *active contours*. The level-sets approach [54, 67] takes an opposite point of view and defines the discontinuity set as the zero set of an auxiliary function. This allows for an indirect and continuous handling of the evolution of the curve, thereby avoiding complications associated to making discrete changes in the structure of the contours. In the recent literature, problems regularized with the total variation are typically solved using proximal splitting algorithms [14, 57].

Some of the connections between graph-cuts and the total variation were already known in [55] but some of these connections have been only fully exploited recently, when [13] and [30] among others, exploited the fact that the ROF model can be reformulated as a parametric maximum flow problem, which, in these papers, is moreover shown to be solved by a divide-and-conquer strategy: This algorithm entails 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 [2]. 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, [4] 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 [41, 34].

In the discrete setting, problems regularized by the total variation or the total boundary size 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 [32, 67]. A number of optimization techniques exist for CRFs [65]. One of the fastest is the α-expansion algorithm of [10], which relies on graph-cut algorithms [9].

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

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 include homotopy algorithms such as the LARS [21] or working set algorithms [52, 61, 26]. It should be noted that the Frank-Wolfe algorithm [33], 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, [31] have shown how a variant can be naturally constructed for the regularized setting, and can be applied 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 [45], orthogonal least squares [18] and related algorithms [47], (b) forward-backward selection approaches such as the Single Best Replacement (SBR) algorithm [64], based on an ℓ_0 penalization or the FoBa algorithm [70], which add backwards steps to remove previously introduced variables that are no longer relevant. See [3] for a review. [2] 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 [2, 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 convex, 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[5].

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. In Section 2.3, we will discuss how to solve the reduced problem efficiently, but first we present a criterion for refining 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 for choosing 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 $\min_{B\subset V} Q'(x_\Pi, u_B)$.

[5] In particular, this is the form taken by the anisotropic total variation for images if the weights are determined by the Cauchy-Crofton formula (see e.g. [30]).

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 $\langle \nabla f(x), \mathbf{1}_V \rangle = 0$ 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 $S = S(x_\Pi)$, $\nabla f(x_\Pi)$ is clearly orthogonal to $\mathrm{span}(\Pi)$ and thus to $\mathbf{1}_V$. 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 ignore the factor[6] $\gamma_B + \gamma_{B^c}$. 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 is well-known, at least since [55], 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

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

[6] γ_B and γ_{B^c} could otherwise be determined by requiring that $\langle \mathbf{1}_V, u_B \rangle = 0$. 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.

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 $y_{(i,j)} \doteq \text{sign}(x_i - x_j)$ with the convention that $\text{sign}(0) = 0$. As stated in the next proposition, finding a minimal cut in this graph provides

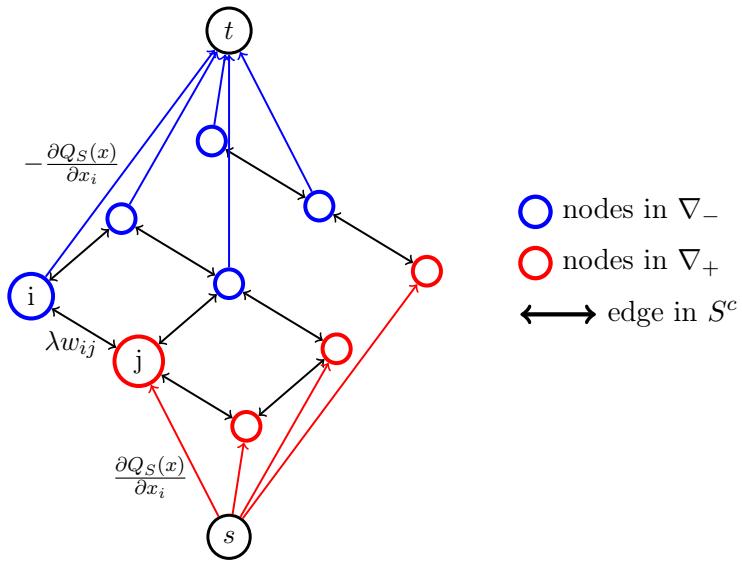


Figure 1: Directed graph for which finding a maximal flow is equivalent to solving (5). Neighboring nodes with different values of x in the original graph are linked by an undirected edge with capacity λw_{ij}, nodes with non-negative gradient are linked to the source, and nodes with negative gradient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

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 $B \mapsto Q'(x, \mathbf{1}_B)$.*

This result is a well-know result which was first discussed in [55]. We refer the reader to [39] 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* $Q'(x, \mathbf{1}_V) = 0$.

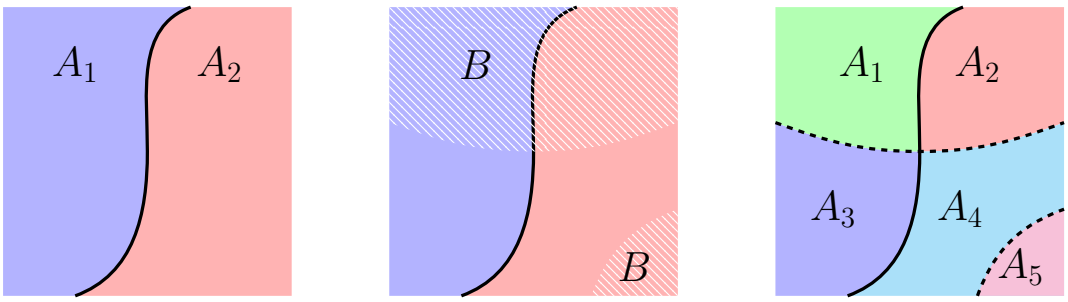
Proof. See Appendix B ∎

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 [60, 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 [31] for the regularized problem, and again the same as the direction that would be used in the primal active set algorithm of [2, Chap. 7.12] for generic Lovász extensions of submodular function, which is essentially a fully corrective and active-set version of the algorithm of [31]. 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 [17], which we discuss in Appendix A. For decomposable atomic norms [48] 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 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 2.7 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 equivalent to staying 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 $\text{span}(\Pi_{\text{new}}) = \mathcal{X}_{S_{\text{new}}}$. 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 $\text{span}(\Pi_{\text{new}})$. 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 is 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 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

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

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. ■

We summarize the obtained working set scheme as Algorithm 1, and illustrate its two first steps on a ROF problem in Figure 3. The following proposition provides a formal proof of convergence.

Proposition 5. *The scheme presented in Algorithm 1 converges to the a global minimum $x^\star$ of Q in a finite a finite amount of steps bounded by n.*

Algorithm 1 Cut pursuit

Initialize $\Pi \leftarrow \{V\}$, $x_\Pi \in \arg\min_{c\in\mathbb{R}} Q(c\mathbf{1}_V)$, $S \leftarrow \varnothing$
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 $x_\Pi \in \arg\min_{z\in\mathrm{span}(\Pi)} Q(z)$
  $S \leftarrow S(x_\Pi)$
end while
return (Π, x_Π)

(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)).

Proof. At the beginning of each iteration, if $\min_{B\subset V} Q'(x_\Pi, \mathbf{1}_B) < 0$ then the steepest binary partition is not trivial, that is $B_\Pi \neq \varnothing$. Consequently the new partition Π_{new} will have at least one more component than Π, and Proposition 4 states that the solution associated with Π_{new} will be strictly better than x_Π. This ensures that the objective function is strictly decreasing along iterations of the algorithm. If $\min_{B\subset V} Q'(x_\Pi, \mathbf{1}_B) = 0$, then Proposition 3 ensures that optimality is reached, because for each value of Π, by construction x_Π is such that $Q'(x_\Pi, 1_V) = 0$. Since the number of components of Π is strictly increasing and bounded by n, the algorithm converges in at most n steps, in the worst case scenario. Provided that each constrained problem $x_\Pi \in \arg\min_{z\in\mathrm{span}(\Pi)} Q(z)$ is solved exactly in finite time, this proves that x_Π converges to the optimum $x^\star$. In the next section we discuss how to exploit the sparse structure of x_Π to solve the reduced problem efficiently. ∎

Case of a non-convex function f. We assumed in all this section that f is a differentiable convex function. However, from a theoretical point of view, a number of results still hold even if f is non-convex provided it is assumed *strictly differentiable* in the sense of [8, Chapter 6.2], or more simply if f is assumed to be continuously differentiable, since continuous differentiability implies *strict differentiability*. Indeed, it can be shown in that case that the

calculations on subgradient and directional derivative that prove our results are still valid for such a function f for an appropriate generalization of the subgradient. As discussed in more details in Appendix C, Propositions 1 and 2 then still hold. In the non-convex case, Algorithm 1 has to be modified since it is no longer reasonable to assume that a global optimum can be found when solving the reduced problem (3), and we could assume instead that the solver called on the reduced problem finds a local optimum which strictly reduces the value of the objective. In the previous sections, proofs of Proposition 3 and Proposition 5 essentially showed that some first order subgradient conditions hold and relied on the fact that first order subgradient conditions are sufficient characterize minima of convex functions. For non-convex functions, the same first order subgradient conditions still hold (although they are no longer sufficient to characterize global minima) and these proposition can be extended, but new sufficient conditions are needed to guarantee that the algorithm converge to a local minimum of the objective (see the appendix for details).

From a practical point of view, we however do not recommend to use the algorithm for non-convex functions, because the low dimensional constraints of the active set algorithm could lead to find very suboptimal local minima of the function. Instead, we would recommend when possible to use majorization-minimization (MM) algorithms, based on convex upper bounds of f. For instance, it is of interest to be able to solve problem (1) for non-convex functions ϕ and in particular so-called concave penalties such as MCP, SCAD and others; for these formulations, MM schemes requiring to solve a sequence of TV are efficient ([53]) and can be advantageously combined with cut pursuit, since the latter will leverage the partition of the previous iterate as a warm-start for the next iteration. This is the scheme we use in Section 3.2.

2.3. A reduced graph for the reduced problem. Let Π be a coarse partition of V into connected components. We argue that $\min_{z\in\mathrm{span}(\Pi)} Q(z)$ can be solved efficiently on a smaller weighted graph whose nodes are associated with the elements of partition Π, and whose edges correspond to pairs of adjacent elements in the original graph. Indeed, consider the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ 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 $x \in \mathrm{span}(\Pi)$ we can indeed express $\mathrm{TV}(x)$ simply:

Proposition 6. *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.

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

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

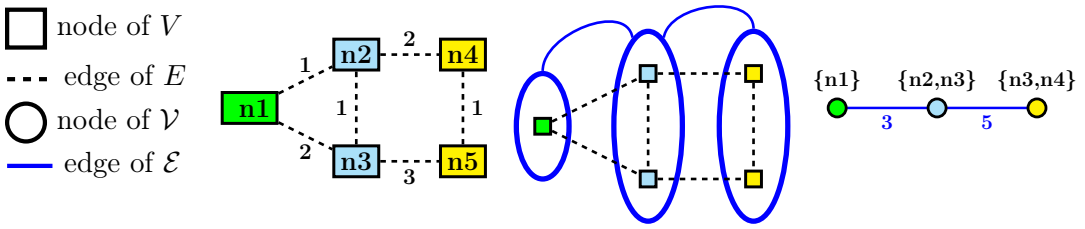


Figure 4: Example of reduced graph. Left: graph G with weights $(w_{ij})_{(i,j)\in E}$ on the edges, middle: partition Π of G into connected components, right: reduced graph $\mathcal{G}$ with weights $(w_{AB})_{(A,B)\in\mathcal{E}}$ on the edges.

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 $S(x_\Pi) = \bigcup \big\{\partial(A, A') \mid (A, A') \in \mathcal{E}, c_A \neq c_{A'}\big\}$.

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 [15], where a low TV prior on the image provides appropriate regularization. Typically if $x_0 \in \mathbb{R}^n$ is the original signal, H a $p \times n$ linear operator, ϵ additive noise, and $y = Hx_0 + \epsilon \in \mathbb{R}^p$ the degraded observed signal, this leads to problems of the form

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

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. However the reduced problem can be computed orders of magnitude faster provided that the current partition is coarse. 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 $x \in \mathrm{span}(\Pi)$ can be rewritten as Kc with $c \in \mathbb{R}^k$ and the gradient of the discrepancy function with respect to c then writes: $\nabla_c 1/2 \|HKc - y\|^2 = K^\intercal H^\intercal HKc - K^\intercal H^\intercal y$.

As a result, the reduced problem can be solved by a similar first-order scheme of much smaller size, with parameters $K^\intercal H^\intercal HK$ and $K^\intercal H^\intercal y$, which are of size $k\times k$ and k respectively. Given the sparsity of the matrix K, HK is computed in time $O(pn)$; consequently $K^\intercal H^\intercal HK$ can be precomputed in $\mathcal{O}(k^2 p + pn)$ and $K^\intercal H^\intercal y$ in $O(pn)$. Solving the reduced problem is then very quick provided k is small compared to n.

In the case of a blur operator H with adequate symmetry, for which $p = n$ is large, manipulating the matrices H or $H^\intercal$ directly should be avoided. However $x \mapsto Hx$ being a convolution, it

can be computed quickly using the fast Fourier transform and, in that case, $K^\intercal H^\intercal HK$ and $K^\intercal Hy$ can be precomputed in $\mathcal{O}(n \log n)$ time.

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 of 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 [10] 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 [58], which is proved to reach a primal suboptimality gap of ε in $\mathcal{O}(1/\varepsilon^2)$ 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 and H a blurr operator, 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.

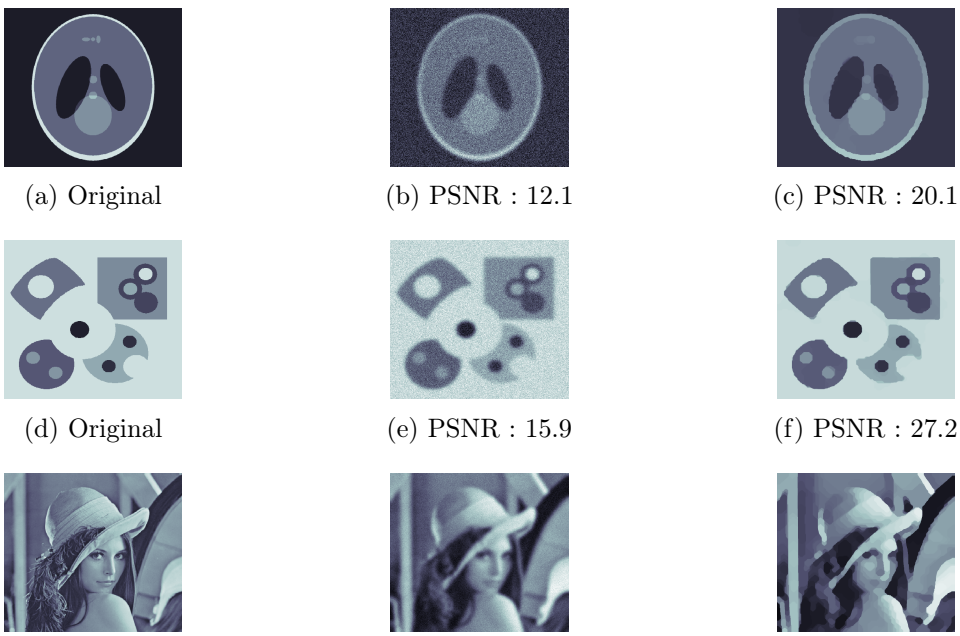
(c) Computing the reduced graph, requires computing the connected components of the graph obtained when removing the edges in S, and the weights $w(A, B)$ between all paris of components (A, B). This can be efficiently performed in $\mathcal{O}(m + n)$ through a depth-first exploration of the nodes of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

(d) The main factor determining the computation time is the number of global iterations needed. In the worst case scenario, 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, [21] 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 [61, 52]. The case of total variation has been studied as well for

1-dimensional signals in [7]. We propose a warm-start approach to compute an approximate[7] solution path for the total variation.

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 of 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 [57] or [14]. 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.



(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)

2.7. Numerical experiments: deblurring with TV. To assess the performance in terms of speed of our working set algorithm for the total variation regularization, we compare it with

[7] 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 [21]. It should however 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.

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.

A `C++` implementation of the cut pursuit algorithm is available on the first author's page[8].

2.7.1. Competing methods.

Preconditioned Generalized Forward Backward (PGFB). As a general baseline, we consider a recent preconditioned generalized forward-backward splitting algorithm by [58] whose prior non-preconditioned version was shown to outperform state-of-the art convex optimization on deblurring tasks in [57], including among others the algorithm of [14]. [58] demonstrate the advantages of the preconditioning strategy used over other adaptive metric approaches, such as the preconditioning proposed in [56] and the inertial acceleration developed in [44].

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 [49]. 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 [13]. 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 [10] 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 [31], 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 [20]. See Appendix D for more details. We chose not to make direct comparisons with the algorithms of [31] and of [2, 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 requires solving several graph-cuts on the whole graph, and, as we discuss in Section 2.7.2 and

[8] https://github.com/loicland/cut-pursuit

illustrate in 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 [37] solver, which itself is based on [10] and [39]. 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.

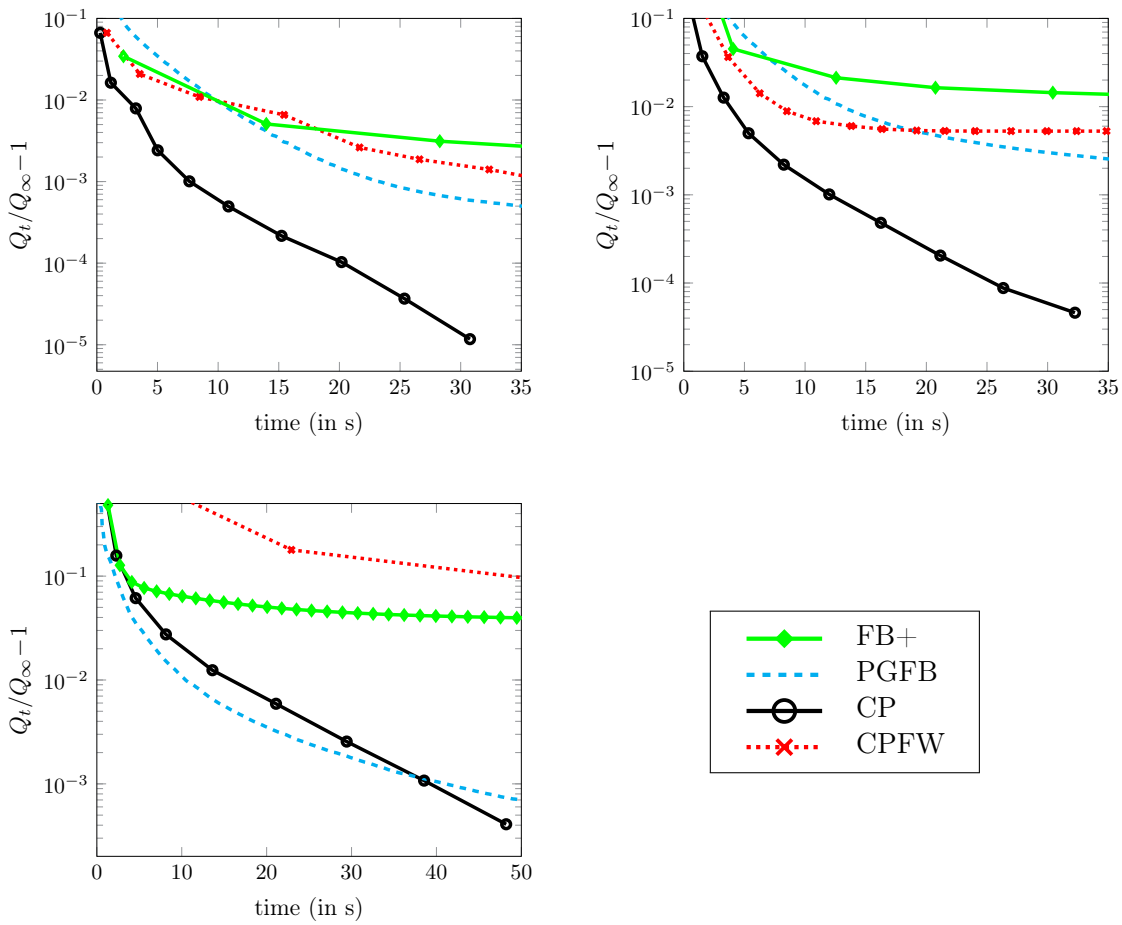


Figure 6: Relative primal suboptimality gap $Q_t/Q_\infty - 1$ at time t (in seconds) for different algorithms on the deblurring task: accelerated forward backward (FB+), Preconditoned Generalized Forward Backward (PGFB), Cut pursuit (CP) and a variant using Frank-Wolfe directions (CPFW), and for different 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.

2.7.2. 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 means that the algorithm overestimates the cost of adding new boundaries, resulting in overly-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 [10] 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 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 $K^\intercal H^\intercal HK$ 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.

2.8. Numerical experiments: approximate TV 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 of parameters inducing from coarse to perfect reconstructions.

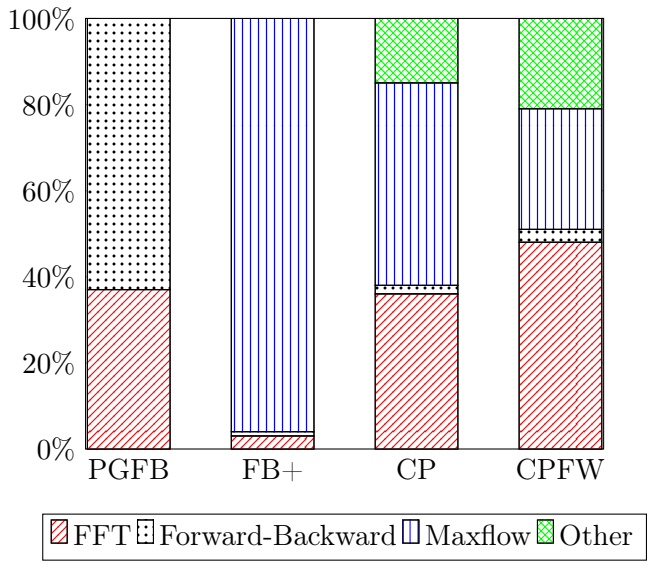


Figure 7: Time breakdown for the different algorithms over 60 seconds of optimization.

2.8.1. Competing methods. **Parametric max-flows (PMF).** We use the parametric max-flow based ROF solver of [13] 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.

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

2.8.2. 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 [10]. Indeed for high

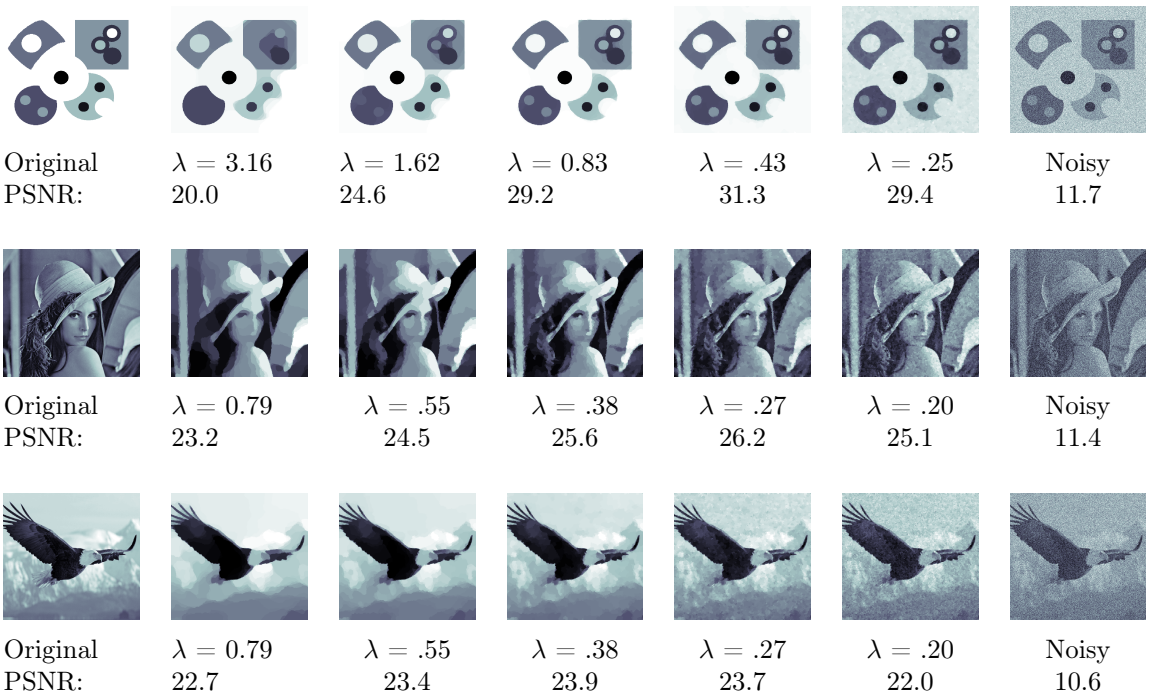


Figure 8: Illustration of the regularization path for the three images in the data set for 5 of the 20 values in the regularization parameters in the path. The peak PSNR is reached for $\lambda = 0.53$, 0.28 and 0.34 respectively.

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 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.

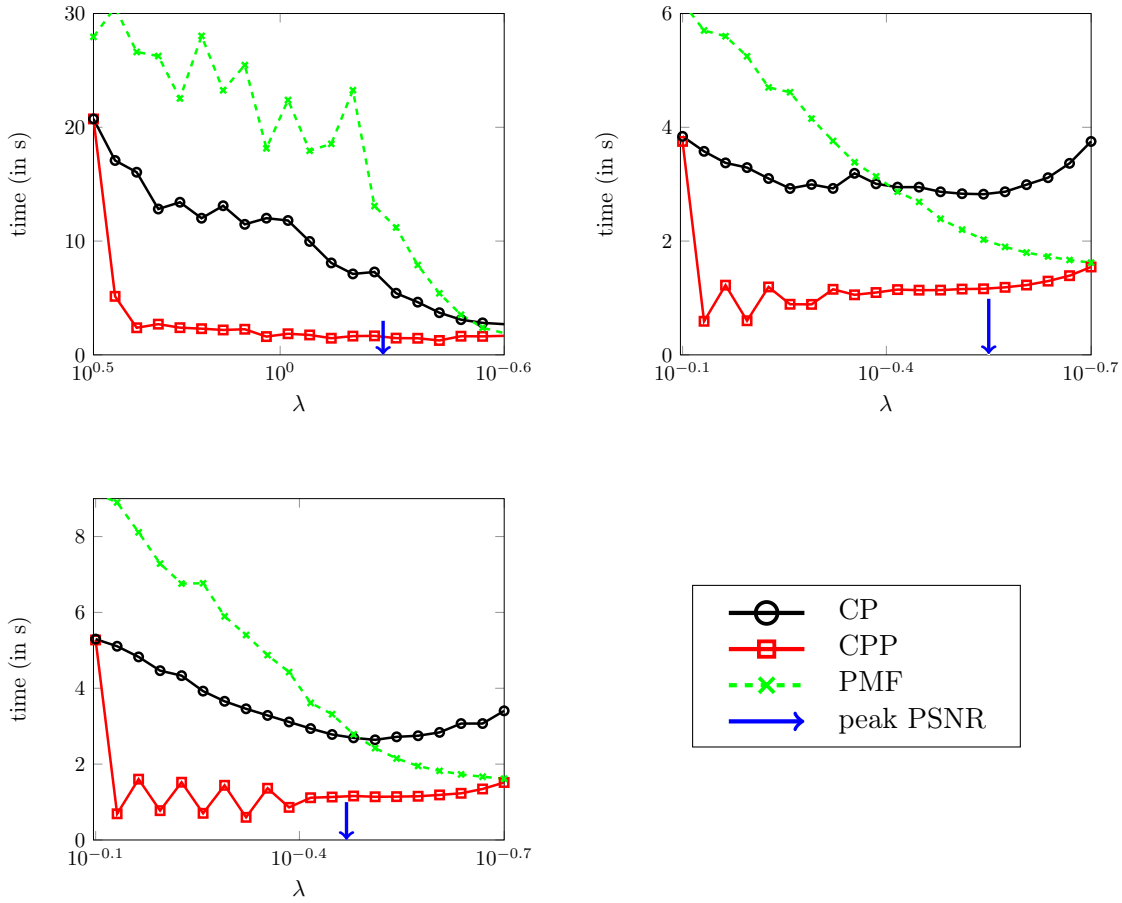


Figure 9: Time in seconds necessary to solve the problem regularized with a given λ (from the warm-start initialization when applicable) with a relative primal suboptimality gap of 10^{-5}, for regularly sampled values of λ along the regularization path. The competing methods are cut pursuit (CP), cut 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.

3. Generalized minimal partition. We consider now a generalization of the minimal partition problem $\min_{x \in \mathbb{R}^n} Q(x)$ with $Q(x) = f(x) + \lambda \Gamma(x)$ where $\Gamma(x) \doteq \frac{1}{2} \sum_{(i,j) \in S(x)} w_{ij}$ the total boundary size penalty for piecewise constant functions. 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 $f_i : \mathbb{R} \mapsto \mathbb{R}$ continuous[9].

[9]The algorithmic scheme we propose in this section does not require the functions f_i to be convex, but convexity will make subproblems easier to solve, and, as discussed later, can be helpful to establish sufficient conditions for convergence (see Section 3.1.1 and Appendix E.1.2)

Method	Simulated	Lena	Eagle
CPP	$\boxed{59}$	$\boxed{25}$	$\boxed{27}$
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.

Our formulation, like [42], but unlike most instances of the minimal partition problem in the literature, does not constrain the number of components in advance. 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 the assumption that the optimal partition Π^* 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 [22], 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 algorithms 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 [64] under the name Single Best Replacement (SBR), in reference to the similar Single Maximum Likelihood Replacement (SMLR) of [40]. At each iteration, the algorithm considers adding (forward step) or removing (backward step) a single variable, whichever reduces the value of the objective most. 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 section that a similar algorithm can be designed for the generalized minimal partition problem, using a general scheme which is similar to that of cut pursuit. We thus call this algorithm ℓ_0-cut pursuit. In particular, it follows a similar structure, in which a partition is successively split into its constant connected components. The main differences is

an adapted rationale to split elements of the partition, and the addition of a explicit backward step.

3.1. A greedy algorithm for generalized minimal partition. As in cut pursuit, we propose an algorithm which greedily splits the elements of the current partition $\Pi = (A_1, \cdots, A_k)$ in forward steps, reoptimizes the value taken by x on each of the A_j, then, in backward steps, possibly merges some regions (or moves some of the boundaries between regions), and iterates.

3.1.1. Forward step. 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 $B \subset V$, so as to minimize the global objective, i.e.

$$\min_{B\subset V} \min_{(h_j,h'_j)_{1\leq j\leq k}} \sum_{j=1}^{k} \Big[\sum_{i\in A_j\cap B} f_i(h_j) + \sum_{i\in A_j\cap B^c} f_i(h'_j) \Big] + \lambda \sum_{j=1}^{k} w(A_j \cap B, A_j \cap B^c)$$

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) the additional boundary perimeter incurred with the cut is simply the sum of the perimeters 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 between pre-existing components is "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 and the fact that $\Gamma(x)$ stays constant when the value of each of the regions is modified together entail that the optimization can be done separately on each set A_j. So the choice of an optimal cut reduces to independent choices of optimal cut on each set A_j as defined by the objective

$$\min_{B_j\subset A_j} \min_{(h,h')} \sum_{i\in B_j} f_i(h) + \sum_{i\in A_j\setminus B_j} f_i(h') + \lambda w(B_j, A_j\setminus B_j). \tag{8}$$

This optimization problem is difficult to solve globally, because even if the functions f_i were assumed convex, it would not be a convex optimization problem. However, for B_j fixed, the partial minimization with respect to h and h' is an optimization problem in $\mathbb{R}^2$, and, for (h, h') fixed, the optimisation with respect to B_j is solved as a min-cut/max-flow problem very similar to the one for the steepest binary cut of Section 2.1. We therefore propose the alternating minimization algorithm presented in pseudo-code as Subroutine 2. Under appropriate hypotheses on f detailed in Appendix E.1.2, this algorithm finds a local minimum of the objective. In particular, these hypotheses hold if each f_i is strictly convex and in general position so to as to avoid ties in assignments of i to B or B^c, for example if $f_i(\cdot) = (\cdot - x_i)^2$ with x_i drawn i.i.d. from a continuous distribution, which corresponds to our case of interest.

In this algorithm, since the minimization with respect to B_j can lead to more than two connected components, we use the same idea as presented in Section 2.2 and illustrated on Figure 2, which is to treat each connected component as a new element of the partition.

Further details on Subroutine 2 and initialization strategies are discussed in Appendix E.1.

3.1.2. Saturated sets. A particular situation occurs when the optimal solution B_j of problem (8) is equal to $\varnothing$ or A_j: in that case, any split of A_j would increase the objective.

We then say that the component A_j is *saturated*. The overall algorithm maintains a set Σ of *saturated* components which do not need to be processed anymore in the splitting steps.

For cut pursuit (i.e. in the TV case), it was essentially sufficient to design the splitting step to specify the algorithm: indeed, after splitting with a steepest binary cut, the problem solved on the reduced graph involved in that case a total variation term penalizing the difference of values between adjacent regions (cf Proposition 6), and this TV term could thus induce the merge of two adjacent regions. By contrast, for ℓ_0 cut pursuit, given that the optimization of the values on each region is independent and without any incidence on the definition of their contours, merge steps and other steps to modify the shape of the regions should be added explicitly. We discuss them in the next two sections.

3.1.3. Backward steps. 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 $S(x)$ (cf Equation 2), which is formed as the union of the boundaries between pairs of components. A backward step is a step that reduces the total boundary perimeter. The most natural way to obtain this is by merging two adjacent components.

Simple merge step: For a region C, let $f_C^\star := \min_h \sum_{i\in C} f_i(h)$. If a pair of adjacent components (A, B) is merged into a single constant component, and the value of $A \cup B$ is, reoptimized, the objective Q increases by

$$\delta_-(A, B) := f_A^\star + f_B^\star - f_{A\cup B}^\star + \lambda w(A, B).$$

A merge effectively decreases the value of the objective and is thus worth it if $\delta_-(A, B) > 0$ i.e. if $f_{A\cup B}^\star - (f_A^\star + f_B^\star) < \lambda w(A, B)$.

It should be noted that the merge step considered does not, in general, correspond to canceling exactly a previous cut, but can merge adjacent subregions that have each been obtained by splitting different regions. The merge step is described in Subroutine 4.

A shortcoming of the simple merge step is that while the removal of boundaries between components is considered, a simple change of the shape of the created boundaries that could reduce total boundary length is not possible. However, since the optimal binary computation only considers binary partitions, the shape of the components might be suboptimal. We therefore propose another kind of step.

Merge-resplit: This step is a combination of a merge step immediately followed by a new split 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. Given $h_A := \arg\min \sum_{i\in A} f_i(A)$ and $h_B := \arg\min \sum_{i\in B} f_i(B)$, the merge resplit step amounts to solve the corresponding

$$\min_{A',B'} \sum_{i\in A'} f_i(h_A) + \sum_{i\in B'} f_i(h_B) + \lambda w(A', B') \quad \text{s.t.} \quad B' = (A \cup B)\backslash A'.$$

But this problem can again be solved as a min-cut/max-flow problem on the region $A\cup B$.

Note that this merge-resplit step is very similar to what [10] call an α-β *swap* in the context of energy minimization in Markov random fields: nodes assigned to other components[10] than

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

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.

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. Finally, note that during the merge-resplit step the values of x_A and x_B are 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.

3.1.4. The ℓ_0 cut pursuit algorithm. Given definitions of forward and backward steps, different algorithms can be obtained by iterating and alternating these steps differently. We propose to alternate between splitting all components 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. This leads to two variants for the main algorithm which are presented as Algorithms 5 and 6, depending on whether only simple merge or merge-resplit steps are used. Implementation details of the algorithms and other possible variants are discussed in Appendix E.2.

Under mild assumptions, Algorithm 5 converges in a finite number of iterations and yields a partition $\Pi = (A_1, \cdots, A_n)$ such that $x_\Pi \doteq \arg\min_{z \in \mathrm{span}(\Pi)} Q(z)$ is a local minimum of Q. See in Appendix E.3 for a precise statement and a proof.

Subroutine 2 $[\Pi, \mathcal{E}, \Sigma] \leftarrow \texttt{split}(\Pi, \mathcal{E}, \Sigma, A)$

[Splits the component A with a binary cut: updates the current partition Π, the component adjacency structure $\mathcal{E}$ and the set of saturated components Σ]

for $A \in \Pi$ **do**
  $\Pi \leftarrow \Pi \setminus \{A\}$
  $B \leftarrow \arg\min_{B \subset A, h, h'} \sum_{i \in B} f_i(h) + \sum_{i \in B^c} f_i(h')$
  while `not_converged` **do**
    $x \leftarrow \arg\min_h \sum_{i \in B} f_i(h)$
    $x' \leftarrow \arg\min_h \sum_{i \in A \setminus B} f_i(h)$
    $B \leftarrow \arg\min_{B \subset A} \sum_{i \in B} f_i(x) + \sum_{i \in B^c} f_i(x') + \lambda w(B, B^c)$
  end while
  if $B \in \{\varnothing, A\}$ **then**
    $\Sigma \leftarrow \Sigma \cup \{A\}$
  end if
  $[B_1, \cdots, B_k] \leftarrow$ connected components of B and $A \setminus B$
  $\Pi \leftarrow \Pi \cup \{B_1, \cdots, B_k\}$
  $\mathcal{E} \leftarrow$ updated adjacency structure
end for

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

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

Subroutine 4 $[\Pi, \mathcal{E}, \Sigma] \leftarrow$ `resplit`$(\Pi, \mathcal{E}, \Sigma, A, B)$

[Performs a merge-resplit step on components A and B.]
$x_A \leftarrow \arg\min_h \sum_{i \in A} f_i(h)$
$x_B \leftarrow \arg\min_h \sum_{i \in B} f_i(h)$
$C \leftarrow \arg\min_{C \subset A \cup B} \sum_{i \in C} f_i(x_A) + \sum_{i \in A \cup B \setminus C} f_i(x_B) + \lambda w(C, A \cup B \setminus C)$
if $C \notin \{A, B\}$ **then**
  $\Sigma \leftarrow \Sigma \setminus \{A, B\}$
else
  $[C_1, \cdots, C_k] \leftarrow$ connected components of C and $A \cup B \setminus C$
  $\Pi \leftarrow \Pi \setminus \{A, B\} \cup \{C_1, \cdots, C_k\}$
  $\mathcal{E} \leftarrow$ updated adjacency structure
end if

Algorithm 5 Simple merge variant (ℓ_0-CPm)

Initialization: $\Pi_0 = \{V\}$, $\mathcal{E} = \Sigma = \varnothing$
while $\Pi \neq \Sigma$ **do**
  for $A \in \Pi \setminus \Sigma$ in parallel **do**
    $[\Pi, \mathcal{E}, \Sigma] \leftarrow$ `split` $(\Pi, \mathcal{E}, A, \Sigma)$
  end for
  Compute $\delta_-(A, B)$ for all $(A, B) \in \mathcal{E}$
  while $\max_{\{A,B\} \in \mathcal{E}} \delta_-(A, B) > 0$ **do**
    $\{A, B\} = \arg\max_{\{A',B'\} \in \mathcal{E}} \delta_-(A', B')$
    $[\Pi, \mathcal{E}', \Sigma] \leftarrow$ `merge` $(\Pi, \mathcal{E}, \Sigma, A, B)$
    Update $\delta_-(A, B)$ for $\{A, B\} \in \mathcal{E}' \setminus \mathcal{E}$
    $\mathcal{E} \leftarrow \mathcal{E}'$
  end while
end while

Algorithm 6 Merge-resplit variant (ℓ_0-CPs)

Initialization: $\Pi_0 = \{V\}$, $\mathcal{E} = \Sigma = \varnothing$
while $\Pi \neq \Sigma$ **do**
  for $A \in \Pi \setminus \Sigma$ in parallel **do**
    $[\Pi, \mathcal{E}, \Sigma] \leftarrow$ `split` $(\Pi, \mathcal{E}, \Sigma, A)$
  end for
  $\mathcal{E}' \leftarrow \mathcal{E}$
  for $\{A, B\} \in \mathcal{E}'$ **do**
    if $\{A, B\} \in \mathcal{E}$ **then**
      $[\Pi, \mathcal{E}, \Sigma] \leftarrow$ `resplit`$(\Pi, \mathcal{E}, \Sigma, A, B)$
    end if
  end for
end while

3.2. Numerical experiments: denoising with ℓ_0 cut pursuit. 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 total boundary size of two noisy 512×512 images: the Shepp-Logan phantom [63] 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[11] which consists of 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.

A `C++` implementation of the ℓ_0-cut pursuit algorithm is available[12].

3.2.1. 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 [32]. In the MPP case, the pairwise terms are of the form $1_{\{c_i \neq c_j\}} w_{ij}$. We use α-expansions [10] to approximately minimize the corresponding energy. More precisely, we use the α-expansions implementation of [27], which uses the same max-flow code [9] 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 to converge, it does in practice and the local minima are shown by [10] to be within a multiplicative constant of the global optimum.

Non-convex relaxation (TV$_{0.5}$). We considered a non-convex counterpart of the total variation, similar to the formulations considered in [51] or [71], but 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 [53]. We use our cut pursuit algorithm to solve each reweighted TV problem 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 5, the simple merge step is performed after each round of cuts. Finally in ℓ_0-CPs, described in Algorithm 6, merge steps are replaced by merge-resplit steps but without priority queue.

After a few preliminary experiments, we chose not to include either level-set methods [16] or active contour methods based on solving Euler-Lagrange equations [36] 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 [27] which is a highly optimized code, ℓ_0-CPf and ℓ_0-CPm are implemented in C++, while ℓ_0-CPs and 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 the performances significantly. 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 [9] and which accounts for most of the computation time.

[11] https://www.data.gouv.fr/fr/datasets/donnees-carroyees-a-200m-sur-la-population
[12] https://github.com/loicland/cut-pursuit

3.2.2. 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 $\mathrm{TV}_{0.5}$ 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.

In Figure 11, we report 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 problems 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 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 Table 13 performance in PSNR that shows that ℓ_0-CP outperforms the CRF formulations for quantization levels that lead to comparable running time.

The comparison with CRF formulations is investigated in more details in Appendix F, where we report the performance of approximations with CRFs solved with iterative α-expansions for different numbers of quantization levels, as compared with the performance of ℓ_0-CPm. The results show that the running time for the CRF formulations grows linearly with the number of classes, although the performance in PSNR does not increase monotonically, and has oscillations which lead to results that are worse than ℓ_0-CPm for some number of classes.

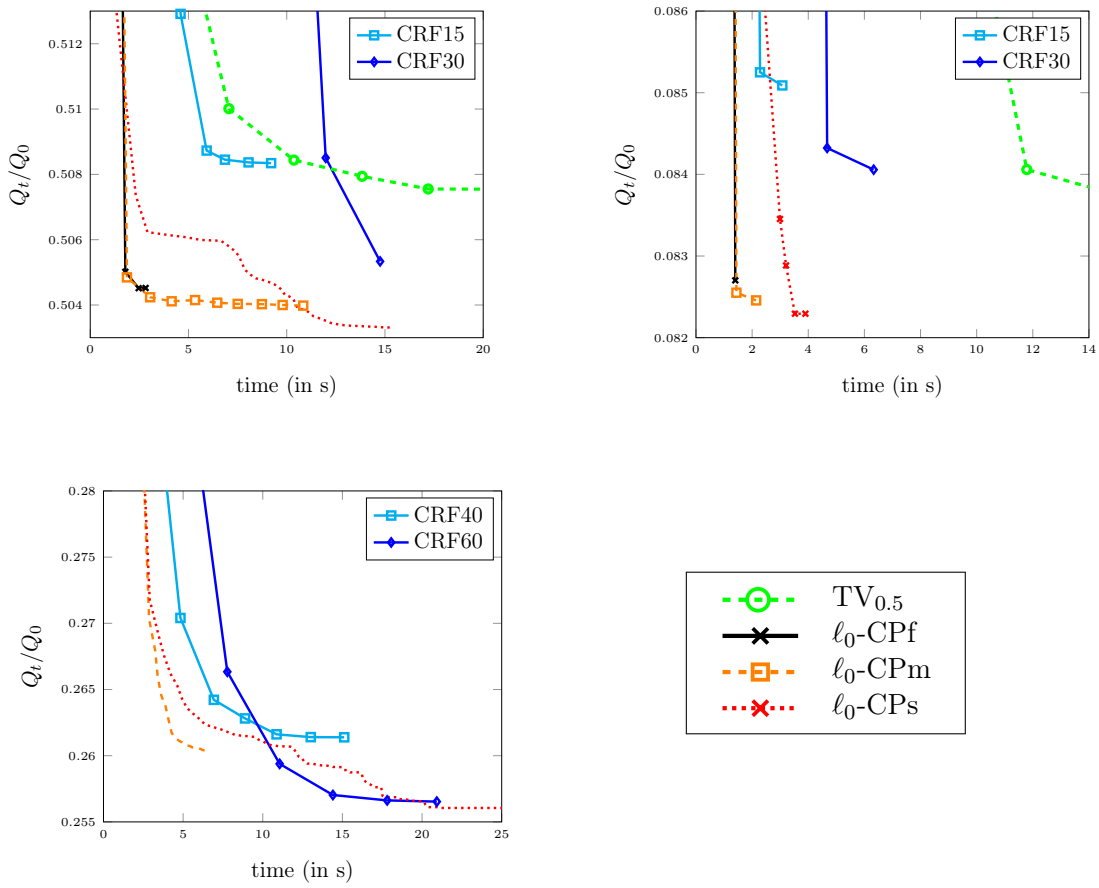
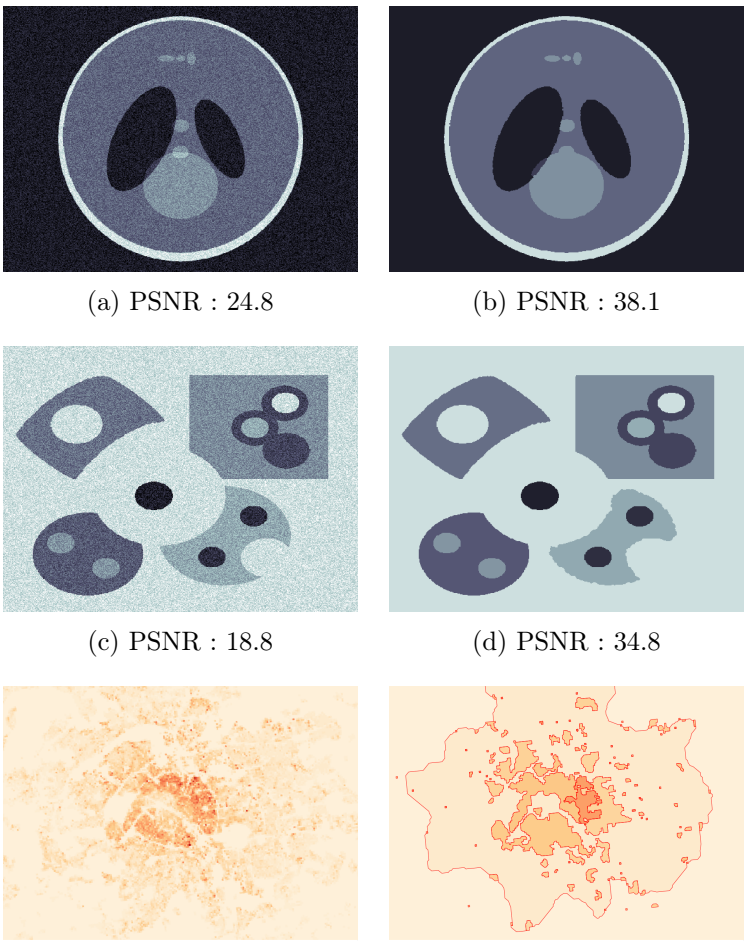


Figure 11: *Generalized minimal partition energy* at time t (in seconds) divided by the same energy for the best constant approximation obtained by different algorithms: Non-convex relaxation ($TV_{0.5}$), ℓ_0-CPf with no backward step, ℓ_0-CPm with simple merge step, ℓ_0-CPs with merge-resplit steps, 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 correspond respectively to one reweighting, one α-expansion cycle and one cut for ($TV_{0.5}$), (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 ℓ_0-cut pursuit with simple merge steps(ℓ_0-CPm). Last line: (left) rasterized population density of Paris area, (right) simplified map obtained by ℓ_0-CPm: 69% of variance explained with 1.2% of contours perimeter.

Experiment	Phantom		Simulated	
Algorithm	PSNR	time	PSNR	time
Noisy image	16.8	-	16.8	-
ℓ_0-CPm	**33.5**	**4.3**	**37.0**	4.6
CRF15	32.6	8.6	34.2	**4.0**
CRF30	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.

4. Conclusion. We proposed two algorithms to minimize functions penalized respectively by the total variation and by the total boundary size. They computationally exploit 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 total 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, which 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 a 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 [4] and to the multivariate case. It would also be interesting to determine the conditions under which the alternating scheme presented in E.1 provides a globally optimal solution of (13), as it would be a necessary step in order to prove approximation guarantees to the solution of ℓ_0-cut pursuit itself.

REFERENCES

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

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

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

[4] F. R. Bach, *Shaping level sets with submodular functions*, in Advances in Neural Information Processing Systems, 2011, pp. 10–18.

[5] L. Bar, T. F. Chan, G. Chung, M. Jung, N. Kiryati, R. Mohieddine, N. Sochen, and L. A. Vese, *Mumford and Shah model and its applications to image segmentation and image restoration*, in Handbook of Mathematical Methods in Imaging, Springer, 2011, pp. 1095–1157.

[6] R. Bellman, *A note on cluster analysis and dynamic programming*, Mathematical Biosciences, 18 (1973), pp. 311–312.

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

[8] J. Borwein and A. S. Lewis, *Convex analysis and nonlinear optimization: theory and examples*, Springer Science & Business Media, 2010.

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

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

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

[12] A. Chambolle, V. Caselles, D. Cremers, M. Novaga, and T. Pock, *An introduction to total variation for image analysis*, in Theoretical foundations and numerical methods for sparse recovery, De Gruyter, 2010, pp. 263–340.

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

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

[15] T. Chan, S. Esedoḡlu, F. Park, and A. Yip, *Recent developments in total variation image restoration*, in Mathematical Models of Computer Vision, Springer Verlag, 2005, pp. 17–31.

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

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

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

[19] L. Condat, *A direct algorithm for 1D total variation denoising*, IEEE Signal Processing Letters, 20 (2013), pp. 1054–1057.

[20] W. Dinkelbach, *On nonlinear fractional programming*, Management Science, 13 (1967), pp. 492–498.

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

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

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

[24] N. El-Zehiry, P. Sahoo, and A. Elmaghraby, *Combinatorial optimization of the piecewise constant*

Mumford-Shah functional with application to scalar/vector valued and volumetric image segmentation, Image and Vision Computing, 29 (2011), pp. 365–381.

[25] N. Y. El-Zehiry and A. Elmaghraby, *Brain MRI tissue classification using graph cut optimization of the Mumford–Shah functional*, in Proceedings of the International Vision Conference of New Zealand, 2007, pp. 321–326.

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

[27] B. Fulkerson, A. Vedaldi, and S. Soatto, *Class segmentation and object localization with superpixel neighborhoods*, in Proceedings of the International Conference on Computer Vision, IEEE, October 2009, pp. 670–677.

[28] N. Fusco, *An overview of the Mumford-Shah problem*, Milan Journal of Mathematics, 71 (2003), pp. 95–119.

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

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

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

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

[33] M. Jaggi, *Revisiting Frank-Wolfe: projection-free sparse convex optimization*, in Proceedings of the 30th International Conference on Machine Learning, 2013, pp. 427–435.

[34] S. Jegelka, F. Bach, and S. Sra, *Reflection methods for user-friendly submodular optimization*, in Advances in Neural Information Processing Systems, 2013, pp. 1313–1321.

[35] N. A. Johnson, *A dynamic programming algorithm for the fused lasso and ℓ_0-segmentation*, Journal of Computational and Graphical Statistics, 22 (2013), pp. 246–260.

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

[37] P. Kohli and P. H. Torr, *Efficiently solving dynamic Markov random fields using graph cuts*, in International Conference on Computer Vision (ICCV), vol. 2, IEEE, 2005, pp. 922–929.

[38] V. Kolmogorov, T. Pock, and M. Rolinek, *Total variation on a tree*, SIAM Journal on Imaging Sciences, 9 (2016), pp. 605–636.

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

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

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

[42] Y. G. Leclerc, *Constructing simple stable descriptions for image partitioning*, International journal of computer vision, 3 (1989), pp. 73–102.

[43] G. P. Leonardi and I. Tamanini, *On minimizing partitions with infinitely many components*, Annali dell'Università di Ferrara, 44 (1998), pp. 41–57.

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

[45] S. Mallat and Z. Zhang, *Adaptive time-frequency decomposition with matching pursuits*, in Time-Frequency and Time-Scale Analysis, Proceedings of the IEEE-SP International Symposium, IEEE, 1992, pp. 7–10.

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

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

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

[49] Y. Nesterov, *Gradient methods for minimizing composite objective function*, tech. report, Université catholique de Louvain, Center for Operations Research and Econometrics (CORE), 2007.
[50] F. Nielsen and R. Nock, *Optimal interval clustering: Application to Bregman clustering and statistical mixture learning*, Signal Processing Letters, 21 (2014), pp. 1289–1292.
[51] M. Nikolova, M. K. Ng, and C.-P. Tam, *Fast nonconvex nonsmooth minimization methods for image restoration and reconstruction*, IEEE Transactions on Image Processing, 19 (2010), pp. 3073–3088.
[52] G. Obozinski, B. Taskar, and M. Jordan, *Multi-task feature selection*, Statistics Department, UC Berkeley, Technical report 743, (2006).
[53] P. Ochs, A. Dosovitskiy, T. Brox, and T. Pock, *On iteratively reweighted algorithms for nonsmooth nonconvex optimization in computer vision*, SIAM Journal on Imaging Sciences, 8 (2015), pp. 331–372.
[54] S. Osher and J. A. Sethian, *Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations*, Journal of Computational Physics, 79 (1988), pp. 12–49.
[55] J.-C. Picard and H. D. Ratliff, *Minimum cuts and related problems*, Networks, 5 (1975), pp. 357–370.
[56] T. Pock and A. Chambolle, *Diagonal preconditioning for first order primal-dual algorithms in convex optimization*, in Proceeding of the International Conference on Computer Vision (ICCV), IEEE, 2011, pp. 1762–1769.
[57] H. Raguet, J. Fadili, and G. Peyré, *A generalized forward-backward splitting*, SIAM Journal on Imaging Sciences, 6 (2013), pp. 1199–1226.
[58] H. Raguet and L. Landrieu, *Preconditioning of a generalized forward-backward splitting and application to optimization on graphs*, SIAM Journal on Imaging Sciences, 8 (2015), pp. 2706–2739.
[59] N. Rao, P. Shah, and S. Wright, *Forward–backward greedy algorithms for atomic norm regularization*, IEEE Transactions on Signal Processing, 63 (2015), pp. 5798–5811.
[60] R. T. Rockafellar, *Convex analysis*, Princeton University Press, 1970.
[61] V. Roth and B. Fischer, *The group-lasso for generalized linear models: uniqueness of solutions and efficient algorithms*, in Proceedings of the 25th international conference on Machine learning, ACM, 2008, pp. 848–855.
[62] L. I. Rudin, S. Osher, and E. Fatemi, *Nonlinear total variation based noise removal algorithms*, Physica D: Nonlinear Phenomena, 60 (1992), pp. 259 – 268, https://doi.org/http://dx.doi.org/10.1016/0167-2789(92)90242-F, http://www.sciencedirect.com/science/article/pii/016727899290242F.
[63] L. A. Shepp and B. F. Logan, *The Fourier reconstruction of a head section*, IEEE Transactions on Nuclear Science, 21 (1974), pp. 21–43.
[64] C. Soussen, J. Idier, D. Brie, and J. Duan, *From Bernoulli–Gaussian deconvolution to sparse signal restoration*, IEEE Transactions on Signal Processing, 59 (2011), pp. 4572–4584.
[65] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother, *A comparative study of energy minimization methods for Markov random fields*, in Proceeding of the European Conference in Computer Vision (ECCV), Springer, 2006, pp. 16–29.
[66] I. Tamanini and G. Congedo, *Optimal segmentation of unbounded functions*, Rendiconti del Seminario Matematico della Università di Padova, 95 (1996), pp. 153–174.
[67] Y.-H. R. Tsai and S. Osher, *Total variation and level set methods in image science*, Acta Numerica, 14 (2005), pp. 509–573.
[68] L. A. Vese and T. F. Chan, *A multiphase level set framework for image segmentation using the Mumford and Shah model*, International Journal of Computer Vision, 50 (2002), pp. 271–293.
[69] Y.-X. Wang, J. Sharpnack, A. Smola, and R. J. Tibshirani, *Trend filtering on graphs*, Journal of Machine Learning Research, 17 (2016), pp. 1–41.
[70] T. Zhang, *Adaptive forward-backward greedy algorithm for sparse learning with linear models*, in Advances in Neural Information Processing Systems, 2009, pp. 1921–1928.
[71] H. Zou, *The adaptive lasso and its oracle properties*, Journal of the American Statistical Association, 101 (2006), pp. 1418–1429.

Appendix A. The total variation as an atomic gauge. It is well known that the total variation is the Lovász extension of the submodular function $F : B \mapsto w(B, B^c)$ [2, 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*[13] 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 $\mathcal{B}_F$ [4]. 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. [17] 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

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

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 [33, 59]. 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 $\mathcal{A} = \left\{ w(B, B^c)^{-1} \mathbf{1}_B + \mu \mathbf{1}_V \right\}_{B \notin \{\varnothing, V\},\, \mu \in \mathbb{R}}$ or equivalently the set $\mathcal{A}' = \left\{ \frac{1}{2} w(B, B^c)^{-1} (\mathbf{1}_B - \mathbf{1}_{B^c}) + \mu' \mathbf{1}_V \right\}_{B \notin \{\varnothing, V\},\, \mu' \in \mathbb{R}}$. 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

$$\arg\max_{\xi : \mathrm{TV}(\xi) \leq 1} \langle s, \xi \rangle = \arg\max_{1_B : B \notin \{\varnothing, V\}} \frac{1}{w(B, B^c)} \langle s, \mathbf{1}_B \rangle, \tag{9}$$

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

Appendix B. Proof of Propositions 1 and 3.

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

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

[13] See [60] for definitions of *gauge*, *polar gauge* and *support function* of a set.

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:

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

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.

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

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

The supremum is reached for $\delta_{ij} = \mathrm{sign}([\mathbf{1}_B]_i - [\mathbf{1}_B]_j)$ for $(i,j) \in S_c$, 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

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

Letting $g = \nabla Q_S(x)$, and since $\langle \nabla f, \mathbf{1} \rangle = 0$, we have $\langle g, \mathbf{1} \rangle = 0$. Consequently $\langle g, \mathbf{1}_{B^c} \rangle = \langle g, \mathbf{1} - \mathbf{1}_B \rangle = -\langle g, \mathbf{1}_B \rangle$, and 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, $\langle \epsilon, u_B \rangle = \langle \frac{1}{2} \delta_{S^c}, D u_B \rangle = \frac{1}{2} \gamma_B \langle \delta_{S^c}, D\mathbf{1}_B \rangle - \frac{1}{2} \gamma_{B^c} \langle \delta_{S^c}, D\mathbf{1}_{B^c} \rangle = \frac{1}{2} (\gamma_B + \gamma_{B^c}) \langle \delta_{S^c}, D\mathbf{1}_B \rangle$ because $D\mathbf{1}_B = -D\mathbf{1}_{B^c}$. Taking the supremum over ϵ then proves the result. ∎

Proposition 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. ($\Rightarrow$) 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 since $\min_{B \subset V} Q'(x, \mathbf{1}_B) \leq Q'(x, \mathbf{1}_\varnothing) = 0$, this 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 (6) of the article are such that, for any set A which is a maximal connected component of $G|_{S^c}$, we have

$$\sum_{i \in \nabla_+ \cap A} c_{si} = \sum_{i \in \nabla_- \cap A} c_{it}. \tag{10}$$

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 or the sink from the rest of the graph. Given equality (10), 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 (10).

Writing flow conservation yields

$$\begin{cases} e_{si} + \sum_{j\in N_i}(e_{ji} - e_{ij}) = 0 & \forall i \in \nabla_+ \\ -e_{it} + \sum_{j\in N_i}(e_{ji} - e_{ij}) = 0 & \forall i \in \nabla_-, \end{cases} \tag{11}$$

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

By replacing e_{si} and e_{it} by their value, the flow conservation (11) at node i rewrites

$$\nabla_i Q_S(x) + \sum_{j\in N_i} \lambda w_{ij}\delta_{ij} = 0$$

$$\nabla_i Q_S(x) + \frac{1}{2}\sum_{j\in N_i} \lambda w_{ij}\left(\delta_{ij} - \delta_{ji}\right) = 0, \tag{12}$$

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 $0 \le e_{ij} \le c_{ij} = \lambda w_{ij}$ 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 $|\delta_{ij}| \le 1$. But we can rewrite (12) as $\nabla Q_S(x) = \frac{1}{2}\lambda D^\intercal \delta$ with $\delta_S = 0$ and $\|\delta_{S^c}\| \le 1$ 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 3 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 $\partial\gamma(x) = \{s \mid \langle x, s\rangle = \gamma(x),\ \gamma^\circ(s) \le 1\}$. Indeed, for $\gamma = \mathrm{TV}$, the set $\{\gamma^\circ(s) \le 1\}$ is simply the submodular polytope $\mathcal{P}_F$ of $F : B \mapsto w(B, B^c)$. As a result $\partial \mathrm{TV}|_{S^c}(x) = \{s \in \mathbb{R}^n \mid \langle s, x\rangle = 1,\ \forall B,\ s(B) \le w_{S^c}(B,B)\}$. 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 $-\frac{1}{\lambda}\nabla Q_S(x) \in \{s \in \mathbb{R}^n \mid \forall B,\ s(B) \le w_{S^c}(B,B)\}$. 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 $\langle \nabla Q_S(x), 1_A\rangle \ge 0$. But then $-\langle \nabla Q_S(x), 1_A\rangle = \langle \nabla Q_S(x), 1_{A^c}\rangle \ge 0$ 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.

Appendix C. Theoretical results for cut pursuit with a non-convex function f.

This appendix discusses how the propositions of Section 2 can be extended to the case of non-convex functions f.

It relies on the fact that notions of directional derivative and subgradient can be extended to non-convex functions. This presents some difficulties in general and different definitions of directional derivatives and subgradient have been introduced by Dini, by Clarke, and by Michel and Penot [8, Chap. 6.1]. These extended subgradients do not behave like usual subgradients in general and some of the rules of the calculus of subgradient are no longer valid. Fortunately, for so-called *regular* functions, that is functions for which the Dini, Clarke and Michel-Penot subgradient all coincide, the usual subgradient calculus applies [8, Chap. 6.2]. In particular, a function $Q = f + g$ with f *strictly differentiable*[14] and g convex is *regular* at any point x of the interior of its domain and $\partial Q(x) = \nabla f(x) + \partial g(x)$, where $\partial\cdot$ denotes here the generalized subgradient for regular function (that coincides with the usual subgradient if the function is convex). This is in particular true for $g = \mathrm{TV}$. As a consequence, the proof of Propositions 1 and 2 only require f to be *strictly* differentiable. Similarly, Proposition 3 no longer holds as stated because the first order subgradient condition is not sufficient for optimality, but we still have

Proposition 7. *For $Q = f + \mathrm{TV}$ with f strictly differentiable, $0 \in \partial Q(x)$ if and only if $\min_{B \subset V} Q'(x, \mathbf{1}_B) = 0$ and $Q'(x, \mathbf{1}_V) = 0$.*

Proof. Since f is strictly differentiable, Q is regular so that the usual subgradient calculus applies and the proof is the same as that of Proposition 3. ∎

If f is non-convex, solving the subproblem on the reduced graph is more difficult, even if only a local minimum is sought. To extend Algorithm 1 to the non-convex setting, it seems appropriate to assume that reoptimizing on the reduced graph (at the end of the main loop) yields a vector x_{Π^t} which is a local minimum of the reduced objective and such that $Q(x_{\Pi^t}) < Q(x_{\Pi^{t-1}})$.

With that modification Proposition 4 remains true, and instead of Proposition 5, we have that the algorithm converges in a finite number of iterations to a point x^*, which is a local minimum of Q in the subspace $\mathrm{span}(\Pi)$ and satisfies $0 \in \partial Q(x^*)$. This is not sufficient in general for x^* to be a minimum of Q. However, if $T(x^*)$ denotes the tangent cone of Q at x^*, that is $T(x^*) := \{h \in \mathbb{R}^n \mid Q'(x^*, h) = 0\}$ (since there are no directions such that $Q'(x^*, h) < 0$), and if $\nabla^2 f(x^*)$ denotes the Hessian of f at x^* then, by standard arguments, the condition $\forall h \in T(x^*)$, $\langle h, \nabla^2 f(x^*) h\rangle > 0$ is sufficient to guarantee that x^* is a local minimum of Q.

Appendix D. Computation of the Frank-Wolfe direction.

The computation of the Frank-Wolfe direction defined in (9) requires to optimize a ratio of combinatorial functions. More precisely, it requires to solve

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

[14] f is strictly differentiable at x if there exists $\varphi \in \mathbb{R}^n$ such that $\forall h \in \mathbb{R}^n, \lim_{y \to x, t \downarrow 0} \frac{f(y+th) - f(y)}{t} = \langle \varphi, h \rangle$.

Given that $B \mapsto \frac{N(B)}{D(B)}$ it is the ratio of a supermodular function (in fact a modular function) and a nonnegative submodular function, it can be maximized efficiently by Algorithm 7 as proved in Proposition 8.

Algorithm 7 Computation of $\max_A N(A)/D(A)$

Initialization: $\lambda_0 = 1, \lambda_{-1} = 0,\ t = 0$
while $\lambda_t \neq \lambda_{t-1}$ **do**
  $\mathcal{S}_t \leftarrow \operatorname{Arg\,max}_{A \subset V} N(A) - \lambda_t D(A)$
  $A_t \leftarrow \arg\min_{A \subset \mathcal{S}_t} D(A)$
  $\lambda_{t+1} \leftarrow \frac{N(A_t)}{D(A_t)}$
  $t \leftarrow t + 1$
end while
return A_t

Proposition 8. *The sequence $(\lambda_t)_t$ generated by Algorithm 7 is monotonically increasing and converges in a finite number of iterations to $\max_{\varnothing \subsetneq A \subset V} \frac{N(A)}{D(A)}$.*

Proof. As the maximum of a finite number non-increasing linear functions of a scalar argument, the function $\varphi : \lambda \mapsto \max_{A \subset V} \left[N(A) - \lambda D(A)\right]$ is a non-increasing, continuous, piecewise linear convex function. It is also non negative because $N(\varnothing) - \lambda D(\varnothing) = 0$. It is immediate to check that $\lambda^* := \min\{\lambda \mid \varphi(\lambda) = 0\} = \max_{\varnothing \subsetneq A \subset V} \frac{N(A)}{D(A)}$. At each iteration, if $\varphi(\lambda_t) \neq 0$, we must have $\lambda_{t+1} > \lambda_t$, because the function $\lambda \mapsto N(A^t) - \lambda D(A_t)$ is strictly positive for $\lambda = \lambda_t$ and equal to 0 for $\lambda = \lambda_{t+1}$. Moreover by construction, the sets A_t are all distinct, as long as $\varphi(\lambda_t) \neq 0$. As a consequence we must reach $\varphi(\lambda_T) = 0$ after a finite number of iterations T. At the end of the algorithm, $\varphi(\lambda_T) = 0$ entails that $\forall A \subset V$, $N(A) \leq \lambda_T D(A)$, which entails that for all $A \neq \varnothing$, $D(A)^{-1}N(A) \leq \lambda_T = D(A_{T-1})^{-1}N(A_{T-1})$. This shows that $\lambda_T = \max_{\varnothing \subsetneq A \subset V} \frac{N(A)}{D(A)}$. This concludes the proof. The choice of taking the maximizer with smallest value of $D(A)$ on line 4 of the algorithm is not key to convergence of the algorithm, but aims at computing the right-derivative which maximizes the step size in λ. ∎

Note that this algorithm is closely related to the algorithm of [20] to maximize a ratio of functions, and in fact applies to any functions N and D; but the minimization of the function $(A \mapsto \lambda D(A) - N(A))$ can be done in polynomial here because, since D and N are respectively sub- and super-modular, their difference is submodular. Moreover, when D is submodular and N is modular, the number of iterations may be bounded by d, because the algorithm may be reinterpreted as the divide-and-conquer algorithm to maximise submodular functions over the submodular polytope [2, p. 160] (for the general case, it may only be bounded in general by 2^d).

Appendix E. Details of the derivation, technical elements and proofs for ℓ_0 cut pursuit.

E.1. Splitting step.

Since in Section 3.1.1 the problem of finding an optimal binary cut of the component A_j is decoupled from the same problem on other components and leads to formulation (8), we discuss the splitting step for the case of the optimal binary cut of the

initial component V.

In the same way that we defined the steepest binary cut in cut pursuit for the convex formulation, 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 described in Section E.1.3. Note however that the penalization of the perimeter of the boundary between B and B^c should strongly discourage the choice of sets B with many connected components.

E.1.1. Optimal binary cut with alternating minimization. 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

$$\min_{B \subset V} \min_{h,h' \in \mathbb{R}} \sum_{i \in B} f_i(h) + \sum_{i \in B^c} f_i(h') + \lambda w(B, B^c). \tag{13}$$

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:

$$E_{flow} = \begin{cases} (s, i), \forall i \in \nabla_+, & \text{with} \quad c_{si} = f_i(h) - f_i(h'), \\ (i, t), \forall i \in \nabla_-, & \text{with} \quad c_{it} = f_i(h') - f_i(h), \\ (i, j), \forall (i, j) \in E, & \text{with} \quad c_{ij} = \lambda w_{ij}, \end{cases} \tag{14}$$

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

The regularity and convexity of f with respect to h and h' guarantee that the objective can be minimized efficiently with respect to these variables. As suggested by [11] or [24], $\psi(B, h, h') = \sum_{i \in B} f_i(h) + \sum_{i \in B^c} f_i(h') + \lambda w(B, B^c)$ can be efficiently minimized by alternatively minimizing with respect to B and (h, h') separately.

E.1.2. Proof of convergence of the alternating minimization scheme. The alternating scheme used to compute the optimal binary cut provide a local minimum of $\psi(B, h, h') = \sum_{i \in B} f_i(h) + \sum_{i \in B^c} f_i(h') + \lambda w(B, B^c)$ with the following assumptions:

- (A0): the functions f_i are continuous,
- (A1): the solution of $\min_{(h,h')} \psi(h, h', B)$ exists and is unique for all sets B
- (A2): the minimizer with respect to B of $\psi(h_A, h'_A, B)$ is unique for all A.

Note that (A1) holds if for example all functions f_i are strictly convex. (A2) can be shown to hold with probability one if f_i is appropriately random, for example if $f_i(\cdot) = (\cdot - x_i)^2$ with x_i drawn i.i.d. from a continuous distribution, which corresponds to our case of interest.

Proposition 9. *Assuming that the assumptions (A0), (A1) and (A2) hold, the alternate minimization scheme converges in a finite number of iterations to a local minimum of $\psi(h, h', B)$ in*

the sense that there exists a neighborhood $\mathcal{N}_B$ *of* (h_B, h'_B) *such that for all* $(h, h', A) \in \mathcal{N}_B \times 2^V$, *we have* $\psi(h, h', A) \geq \psi(h_B, h'_B, B)$.

Proof. Let $\psi(B) = \min_{h,h'} \psi(h, h', B)$. By construction and with assumption (A1), the sequence $(\psi(B^t))_t$ is strictly decreasing until minimization with respect to either (h, h') or B yields no progress, i.e. until a partial minimum with respect to both blocks is attained. Since the set 2^V is finite, the algorithm must converge in a finite number of iterations.

The point B attained must be a local minimum in the sense above: indeed for any set A different than B, we must have $\phi(h_B, h'_B, B) < \phi(h_B, h'_B, A)$ because the algorithm stopped (which excludes $\phi(h_B, h'_B, B) > \phi(h_B, h'_B, A)$) and because an equality is excluded by (A2). But then by assumption (A0), ϕ is continuous with respect to (h, h') so that in a neighborhood $\mathcal{N}_B$ of (h_B, h'_B) we must have $\phi(h, h', A)$ sufficiently close to $\phi(h_B, h'_B, A)$ for the inequality characterizing a local minimum to hold. ■

E.1.3. 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 (13) 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 perimeter 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.

E.2. Implementation. 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.

E.2.1. Splitting. For each component an optimal binary partition (B, B^c) is obtained by solving (13) as described in section E.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 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 (13) 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 [6]. This can be generalized to the case of Bregman divergences and beyond [50].

As described in section E.1.3, the partition Π is updated by computing its connected components after it is split by (B, B^c). Subroutine 2 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.

E.2.2. Simple merge.. This backward step consists of 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 $\Pi_-(A,B) \doteq \Pi \setminus \{A,B\} \cup \{A \cup B\}$.

The exact implementation of the while loop described in Algorithm 5 is in fact based on a priority-queue. The value $\delta_-(A,B)$ is computed for each neighboring components, and stored in the priority queue. Each pair that provides a nonnegative 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 in which the partition Π does not get too large.

E.2.3. Merge-resplit.. This more complex backward step, already described in 3.1.3 is significantly computationally 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 components yielding the largest decrease in objective value, it would be too computationally heavy in the merge-resplit case and we thus perform boundary changes only once for each pair of neighbors in the graph $\mathcal{E}$. The pseudocode of the procedure is detailed in subroutine 4

E.2.4. Other algorithmic variants. We discuss here the relevance of constructing more greedy algorithms and of variants to tackle the problem in which the total boundary size is constrained instead of penalized.

It would have been theoretically possible to implement a more greedy version of ℓ_0 cut pursuit in which one performs 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 one greedily chooses the most beneficial, but the overhead costs would have been prohibitive.

The ℓ_0 cut pursuit algorithms constructed in Section 3 is a greedy algorithm to solve a formulation in which the 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 [70]. 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.

E.3. Convergence to a local minimum of the generalized MP problem. We now prove the local optimality of the solution provided by Algorithm 5.

Proposition 10. *If assumption (A0) holds, then the ℓ_0 cut pursuit algorithm provides in a*

finite number of iterations a partition $\Pi = (A_1, \cdots, A_n)$ *such that* $x_\Pi \doteq \arg\min_{z \in span(\Pi)} Q(z)$ *is a local minimum of* Q.

Proof. The fact that f is separable ensures that x_Π can be minimized separately over each connected component. We denote $x_{A_i} \in \arg\min_z \sum_{i \in A_i} f_i(z)$.

We denote Π^t the partition at iteration t, and x_Π^t the associated solution. We first prove that the sequence $Q(x_\Pi^t)$ is strictly decreasing. Indeed if the stopping criterion for the algorithm is not met, then there exists at least one component A_j which is not saturated, i.e. such that there exists a binary partitions $B \subsetneq A_j$ such that $\min_{h,h'} \sum_{i \in B} f_i(h) + \sum_{i \in B^c} f_i(h') + \lambda w(B, B^c) < \sum_{i \in A_j} f_i(x_{A_j})$. Consequently this component will be split in the next partition to yield a strict decrease of the objective function Q, at least equal to the one provided by the minimizing arguments (h, h'). Since the set of all partition is a finite set, the algorithm stops in a finite number of steps. We now prove that the partition Π attained when the algorithm stops is such that the corresponding variable x_Π is a local minimum of Q. Let $\mathcal{E}$ be the set of pairs of adjacent components of Π. We can assume that $x_A \neq x_B$ for any $(A, B) \in \mathcal{E}$. If it is not the case we replace Π by the partition in which such components are merged, without changing x_Π. Consequently there exists $\delta_1 > 0$ such that $|x_A - x_B| > \delta_1$ for any $(A, B) \in \mathcal{E}$.

As all edge weights are assumed strictly non negative we have that $w_{min} = \min_{(i,j) \in E} w_{i,j} > 0$. Since (A0) states that f is continuous, there exists an Euclidean ball centered at x_Π and of radius δ_2 in which all elements are strictly greater than $f(x_\Pi) - \min_{(i,j) \in E} w_{i,j}$.

We now prove by contradiction that x_Π is a local minimum of Q. Let x' be an element of the euclidian ball $\mathcal{B}$ centered at x_Π and of radius $\min(\frac{1}{3}\delta_1, \delta_2)$ such that $Q(x') < Q(x_\Pi)$. We can first recognize that since the values of x_Π associated to each connected component differs by at least δ, x' cannot have two connected components of Π sharing a common value. Consequently the boundary perimeter can only increase $\Gamma(x') \geq \Gamma(x_\Pi)$.

If we first assume that $\Gamma(x') = \Gamma(x_\Pi)$, then x' must be piecewise constant with respect to Π, and be such that $f(x') < f(x_\Pi)$, which is a contradiction with the definition of x_Π. We must then assume that $\Gamma(x') > \Gamma(x)$. Since the smallest increment in Γ is w_{min}, we have $\Gamma(x') \geq \Gamma(x) + w_{min}$. Since the radius of $\mathcal{B}$ is smaller than δ_2, we have that $f(x) \geq f(x_\Pi) - w_{min}$, and consequently $Q(x_\Pi) \geq Q(x)$, which is a contradiction.

Appendix F. CRF formulation and number of quantization levels. In this appendix, we report the performance of ℓ_0-cut pursuit and α-expansions for different numbers of quantization levels for denoising an image. The regularization strentgh is chosen by cross-validation to maximize the PSNR.

The fact that ℓ_0-CPm does not rely on an a priori quantized level leads to overall good performance, with significantly faster computation times. By contrast, the running time for the α-expansions based algorithms has a complexity which empirically grows linearly with the number of classes, and the performance whether measured in terms of the original objective or in PSNR does not increase monotonically as a function of the number of classes.

Plotting the corresponding PSNRs shows that the smaller local minima of the objective found correlate well with gains in PSNR, and that the corresponding gains can be quite substantial as illustrated as well in Table 13. The fact that the level of performance for CRF is highly sensitive to the exact number of classes is a shortcoming of the method, especially given its computational cost.

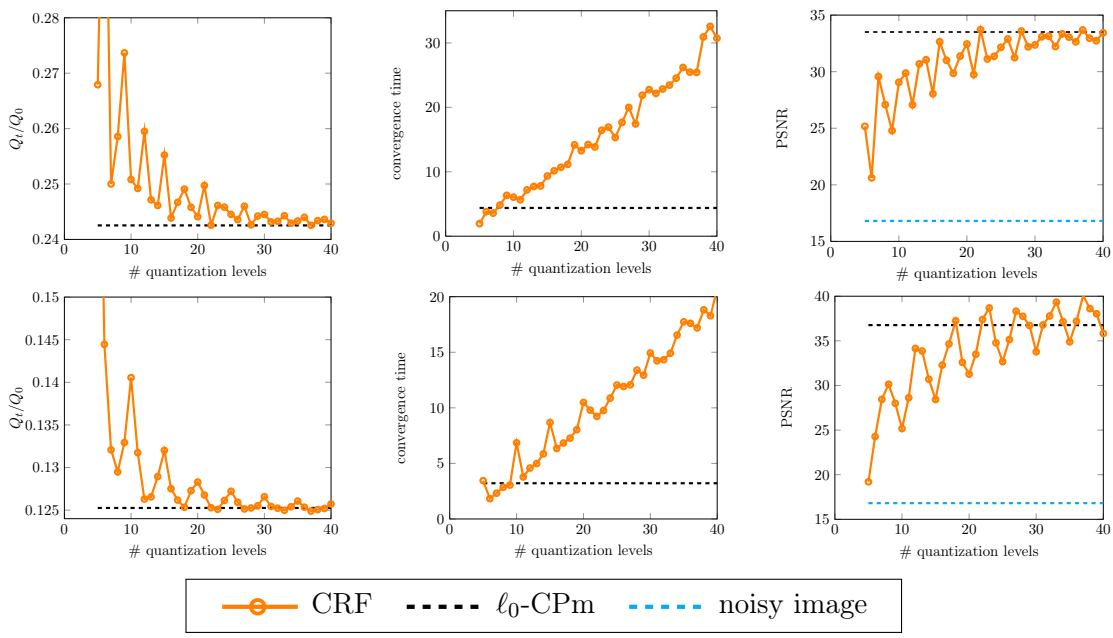


Figure 14: **Behavior of the α-expansion based algorithm on CRF formulations for different number of quantization levels** for the phantom (top) and the simulated data (bottom) averaged on 10 denoising experiments: (left) ratio between the energy Q at convergence and the energy at time 0, (middle) running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