Noname manuscript No.
(will be inserted by the editor)

Two polynomial time graph labeling algorithms optimizing max-norm based objective functions

Filip Malmberg · Krzysztof Chris Ciesielski

Received: Draft date: February 17, 2020/ Accepted: date

Abstract Many problems in applied computer science can be expressed in a graph setting and solved by finding an appropriate vertex labeling of the associated graph. It is also common to identify the term "appropriate labeling" with a labeling that optimizes some application motivated objective function. The goal of this work is to present two algorithms that, for the objective functions in a general format motivated by image processing tasks, find such optimal labelings. Specifically, we consider a problem of finding an optimal binary labeling for the objective function defined as the max-norm over a set of *local costs* of a form that naturally appears in image processing.

It is well known that for a limited subclass of such problems, globally optimal solutions can be found via *watershed cuts*, that is, by the cuts associated with the optimal spanning forests of a graph. Here, we propose two new algorithms for optimizing a broader class of such problems. The first algorithm, that works for all considered objective functions, returns a globally optimal labeling in quadratic time with respect to the size of the graph (i.e., the number of its vertices and edges) or, for an image associated graph, the size of the image.

The second algorithm is more efficient, with quasi-linear time complexity, and returns a globally optimal labeling provided that the objective function satisfies certain given conditions. These conditions are analogous to the submodularity conditions encountered in max-flow/min-cut optimization, where the objective function is defined as sum of all local costs.

We will also consider a refinement of the max-norm measure, defined in terms of the lexicographical order, and examine the algorithms that could find minimal labelings with respect to this refined measure.

Keywords Energy Minimization · Pixel labeling · Minimum cut · NP-hard

Mathematics Subject Classification (2010) 68Q25 · 68W40 · 68R10

F. Malmberg
Centre for Image Analysis, Department of Information Technology, Uppsala University, Sweden
E-mail: filip.malmberg@it.uu.se

K.C. Ciesielski
Department of Mathematics, West Virginia University, Morgantown, WV 26506-6310, USA and Department of Radiology, MIPG, University of Pennsylvania, Philadelphia, PA 19104, USA E-mail: KCies@math.wvu.edu

1 Introduction

Many fundamental problems in image processing and computer vision, such as image filtering, segmentation, registration, and stereo vision, can naturally be formulated as optimization problems. Often, these optimization problems can be described as *labeling* problems in which we wish to assign to each image element (pixel, or vertex of an associated graph) $v \in V$ an element $\ell(v)$ from some finite K-element set of labels, usually $\{0, \ldots, K-1\}$. The interpretation of these labels depends on the optimization problem at hand. In image segmentation, the labels might indicate object categories. In registration and stereo disparity problems the labels represent correspondences between images, and in image reconstruction and filtering the labels represent intensities in the filtered image.

In what follows an undirected graph $\mathcal{G}$ is identified with a pair $\langle V, \mathcal{E} \rangle$, where V is its set of vertices and $\mathcal{E}$ is the set of its edges. Each edge connecting vertices s and t is identified with a pair $\{s, t\}$. We make the assumption that the vertices in V are linearly ordered, and let $\hat{\mathcal{E}} := \{\langle s, t \rangle \in V^2 \colon \{s, t\} \in \mathcal{E} \ \& \ s < t\}$.

Our new algorithms have no restriction on the format of the graph to which they can be applied. However, in what follows we will often treat $\mathcal{G}$ as associated with a digital image. In this case V is the set of all pixels of the image, while $\mathcal{E}$ is the set of pairs $\{s,t\}$ of vertices/pixels that are *adjacent* according to some given adjacency relation.

In this paper, we seek the vertex label assignments $\ell\colon V \to \{0,1,\ldots,K-1\}$ of the undirected[1] graphs $\mathcal{G} = (V,\mathcal{E})$ that minimize a given objective (energy) function E_∞ of the form

$$E_\infty(\ell) := \max\Big\{\max_{s\in V} \phi_s(\ell(s)),\ \max_{\langle s,t\rangle\in\hat{\mathcal{E}}} \phi_{st}(\ell(s),\ell(t))\Big\}. \quad (1)$$

The functions $\phi_s(\cdot)$ are referred to as *unary* terms. The value of $\phi_s(j)$ depends explicitly only on the label $j \in \{0,1,\ldots,K-1\}$,

but typically is also based on some prior information. These terms are used to indicate a preference for a vertex/pixel s to be assigned a particular label j.

The functions $\phi_{st}(\cdot,\cdot)$ are referred to as *pairwise* or *binary* terms. The value of $\phi_{st}(\cdot,\cdot)$ depends simultaneously on the labels assigned to the vertices/pixels s and t, and thus introduces a dependency between the labels of different pixels. Typically, this dependency between pixels is used to express an expectation that the desired solution should have some degree of smoothness or regularity.

The unary and pairwise terms taken together form the *local costs* error measures we mentioned in the abstract (and forming the functional Φ defined in Section 3). The same local costs are used in the L_1-norm energy E_1, that we discuss briefly in the next section.

Finding a labeling that globally minimizes an objective function of the form E_∞ is generally a challenging computational task—in Section 7 we show that this problem is in fact NP-hard in the general case, for $K > 2$. As we will see, however, there exist restricted classes of local cost functionals for which efficient algorithms can be formulated.

In the conference version of this paper [16], we introduced an algorithm for finding a binary labeling (i.e., with $K = 2$) and showed that the labeling it returns is always E_∞-optimal as long as all pairwise local cost terms ϕ_{st} are ∞-*submodular*, that is, that they satisfy the condition

$$\max\{\phi_{st}(0,0),\phi_{st}(1,1)\} \le \max\{\phi_{st}(1,0),\phi_{st}(0,1)\}. \quad (2)$$

This algorithm, presented in Section 6, is very efficient, with quasi-linear time complexity.[2] An important question left open in our previous work [16] was whether it is possible to optimize objective function E_∞ in polynomial time without any additional assumptions on the local cost functional, like that of ∞-submodularity needed for the algorithm from [16]. Here, we answer this question affirmatively by presenting in Section 5 an algorithm that produces, in $\mathcal{O}((|V|+|\mathcal{E}|)^2)$ time, a binary labeling that is globally E_∞-optimal for any local cost functional.

2 Background and related work

2.1 L_p norm objective functions and minimal graph cuts

While the main focus of this paper is to find efficient algorithms for the direct optimization of objective functions of the form E_∞, we will start by discussing the more general problems of optimizing L_p norm objective functions for $p \in [1,\infty]$.

In their seminal work, Kolmogorov and Zabih [13] considered binary labeling problems for the L_1-norm based objective function of the form

$$E_1(\ell) := \sum_{s\in V} \phi_s(\ell(s)) + \sum_{\langle s,t\rangle\in\hat{\mathcal{E}}} \phi_{st}(\ell(s),\ell(t)) \quad (3)$$

and showed that a globally optimal binary labeling can be found by solving a max-flow/min-cut problem on a suitably constructed graph under the condition that all pairwise terms ϕ_{st} are *submodular*, that is, that they satisfy the inequality

$$\phi_{st}(0,0) + \phi_{st}(1,1) \le \phi_{st}(0,1) + \phi_{st}(1,0). \quad (4)$$

Looking at the objective functions E_1 and E_∞, we can view them both as consisting of two parts:

- The *local* error measures, in our case expressed by the unary and pairwise terms.
- A *global* error measure, aggregating the local errors into a final score.

[1] Actually, the energy formula (1) is expressed in terms of the directed graph $\hat{\mathcal{G}} = \langle V,\hat{\mathcal{E}}\rangle$. But, for any $\{s,t\} \in \mathcal{E}$, we consider the value of $\phi_{st}(\ell(s),\ell(t))$ as depending only on $\ell \restriction \{s,t\} = \{\langle s,\ell(s)\rangle,\langle t,\ell(t)\rangle\}$—the restriction of ℓ to the indirect edge $\{s,t\}$. (See Section 3, where we explicitly express $E_\infty(\ell)$ in terms of the numbers $\Phi(\ell\restriction\{s,t\}) = \phi_{st}(\ell(s),\ell(t))$ and $\Phi(\ell\restriction\{s\}) = \phi_s(\ell(s))$.) So, the directedness of the graph is not really used in (1).

[2] Formally, the asymptotic time complexity is bounded by the time required to sort $\mathcal{O}(|V|+|\mathcal{E}|)$ values. Here, $|X|$ denotes the cardinality of the set X.

In the case of E_1, the global error measure is obtained by summing all local error measures; in the case of E_∞ the global error measure is taken to be the maximum of all local error measures. If we assume for a moment that all local error measurements are non-negative, then E_1 can be seen as measuring the L_1-norm of a vector[3] containing all local costs/errors. Similarly, E_∞ can be interpreted as the L_∞- (or max-) norm of the same vector. The L_1 and L_∞ norms are both the special cases of L_p norms, with $p \in [1,\infty]$, which for finite p are defined as

$$E_p(\ell) := \left(\sum_{s\in V} \phi_s^p(\ell(s)) + \sum_{\langle s,t\rangle\in\hat{\mathcal{E}}} \phi_{st}^p(\ell(s),\ell(t)) \right)^{1/p}, \quad (5)$$

where $\phi_s^p(\cdot) = (\phi_s(\cdot))^p$ and $\phi_{st}^p(\cdot,\cdot) = (\phi_{st}(\cdot,\cdot))^p$. The value $p \in [1,\infty]$ can be seen as a parameter controlling the balance between minimizing the overall cost versus minimizing the magnitude of the individual terms. For $p = 1$, the optimal labeling may contain arbitrarily large individual terms as long as the sum of the terms is small. As p increases, a larger penalty is assigned to solutions containing large individual terms. In the limit as p approaches infinity, E_p approaches E_∞ and the penalty assigned to a solution is determined by the largest individual term only. The limit behavior of L_p norm optimizers as p approaches ∞ has also been studied in, e.g., [20,8,18]. Abbas and Swoboda [1] considered optimization of mixed optimization problems, where the objective function contains both L_1 and L_∞ terms.

Labeling problems with objective functions of the form E_p, for $p \in [1,\infty)$, can be solved using minimal graph cuts, provided that all pairwise terms ϕ_{st}^p are p-submodular [17]. A binary term ϕ is said to be *p-submodular* if the corresponding term ϕ^p is submodular, which is equivalent to the condition

$$(\phi_{st}^p(0,0) + \phi_{st}^p(1,1))^{1/p} \le (\phi_{st}^p(0,1) + \phi_{st}^p(1,0))^{1/p}. \quad (6)$$

In the limit, as p goes to infinity, this inequality becomes

$$\max\{\phi_{st}(0,0), \phi_{st}(1,1)\} \le \max\{\phi_{st}(1,0), \phi_{st}(0,1)\},$$

that is, the ∞-submodularity condition (2). As observed by Malmberg and Strand [17], 1-submodularity does not neccesarily imply p-submodularity.[4] The following theorem was shown by Malmberg and Strand [17]:

[3] Formally, this vector is identified with the function ϕ_ℓ defined in the next section.

[4] As an example, consider the two-label pairwise term ϕ_{st} given by $\phi_{st}(0,0) = 3$, $\phi_{st}(1,1) = 0$, and $\phi_{st}(0,1) = \phi_{st}(1,0) = 2$. It is easily verified that ϕ_{st} is 1-submodular but not 2-submodular.

Theorem 1. *If a binary term ϕ is 1-submodular and ∞-submodular, then it is also p-submodular for any real $p \ge 1$.*

We note here that Theorem 1 implies also the following seemingly stronger result.

Corollary 1. *Let ϕ be a binary term. Then for every $\rho \in [1,\infty)$ the following conditions are equivalent.*

(i) ϕ is ρ-submodular and ∞-submodular.
(ii) ϕ is p-submodular for every $p \in [\rho,\infty)$.

Proof. To see that (ii) implies (i) notice that the p-submodularity inequality (6) can be written as

$$\|\langle \phi_{st}(0,0), \phi_{st}(1,1)\rangle\|_p \le \|\langle \phi_{st}(0,1), \phi_{st}(1,0)\rangle\|_p.$$

Since the L_p norm converges to the L_∞ norm, as p goes to infinity, the limit of both sides of the above inequality becomes

$$\|\langle \phi_{st}(0,0), \phi_{st}(1,1)\rangle\|_\infty \le \|\langle \phi_{st}(0,1), \phi_{st}(1,0)\rangle\|_\infty,$$

that is, the ∞-submodularity condition (2).

To see that (i) implies (ii) assume that ϕ_{st} satisfies (i). Then ϕ_{st}^ρ is both 1-submodular (raise both sides of the inequality (2) with $p = \rho$ to the power ρ) and ∞-submodular (as the map x^ρ is increasing on $(0,\infty)$). In particular, ϕ_{st} satisfies the assumptions of Theorem 1. Therefore, for every $p \in [\rho,\infty)$ it is $\frac{p}{\rho}$-submodular, that is, satisfies

$$(\phi_{st}^{\rho\frac{p}{\rho}}(0,0) + \phi_{st}^{\rho\frac{p}{\rho}}(1,1))^{\rho/p} \le (\phi_{st}^{\rho\frac{p}{\rho}}(0,1) + \phi_{st}^{\rho\frac{p}{\rho}}(1,0))^{\rho/p}.$$

But this clearly implies p-submodularity of ϕ_{st}. □

2.2 Optimization of E_∞ by classical algorithms

In Section 4 we will show that if the binary terms ϕ satisfy (i) of Corollary 1, then an optimal labeling for the associated energy E_∞ can be found by solving an appropriate max-flow/min-cut problem.

Moreover, it turns out that in some problem instances a labeling that is globally optimal with respect to E_∞ can be found using very efficient, greedy algorithms. Specifically, if

(D) all pairwise terms are such that $\phi_{st}(1,0) = \phi_{st}(0,1)$ and $\phi_{st}(0,0) = \phi_{st}(1,1) = 0$, while all unary terms have values in $\{0,\infty\}$,

then an optimal labeling for the associated energy E_∞ can be found by computing the partitioning induced by an optimum spanning forest on a suitably constructed

graph using, e.g., Prim's algorithm [19,7][5]. See more on this in Section 4. This property of optimum spanning forests has been observed by several authors [2,8,6]. This result has a high practical value since the computation time for constructing an optimal spanning forest is substantially lower than the computation time for solving a max-flow/min-cut problem, asymptotically as well as in practice [8].

Wolf et al. [23,22,21] recently proposed various extension of this greedy approach, and also reported state-of-the-art results on various image segmentation benchmarks. We note also that the notion of partitioning an image-induced graph by computing an optimum spanning forest is tightly connected to the classic *watershed* image segmentation method [9,10].

Based on the above, an interesting question is therefore whether it is possible to use similar greedy techniques to optimize the objective function E_∞ beyond the special case when the local costs satisfy property (D). The results presented in this paper answers this question affirmatively, and shows that the class of E_∞ optimization problems that are solvable by the efficient greedy algorithms is larger than what was previously known.

3 Algorithms for direct optimization of E_∞: preliminaries

In Sections 5 and 6, we will introduce two novel algorithms, each finding a binary labeling minimizing E_∞.

The exposition of these algorithms relies on the notion of unary and binary solution *atoms*, which we introduce in this section. Informally, a unary atom represents one possible label configuration for a single vertex, and a binary atom represents a possible label configuration for a pair of adjacent vertices. Thus, for a binary labeling problem, there are two atoms associated with every vertex and four atoms for every edge. The total number of atoms for a binary labeling problem is thus $\mathcal{O}(|V| + |\mathcal{E}|)$.

Formally, we let $\mathcal{V} = \{\{v\} \colon v \in V\}$, put $\mathcal{D} = \mathcal{V} \cup \mathcal{E}$, and let $\mathcal{A}$ be the family of all binary maps from $D \in \mathcal{D}$ into $\{0,1\}$. An atom, in this notation, is an element of $\mathcal{A}$. If we identify, as it is common, maps with their graphs then each unary atom associated with a vertex $s \in V$ has form $\{\langle s,i\rangle\}$, with $i \in \{0,1\}$. Similarly, each binary atom associated with an edge $\{s,t\} \in \mathcal{E}$ has the form $\{\langle s,i\rangle, \langle t,j\rangle\}$, with $i,j \in \{0,1\}$.

Notice, that the maps ϕ_s and ϕ_{st} used for the unary and binary terms in (1) can be combined to form a single function $\Phi \colon \mathcal{A} \to [0,\infty)$ defined, for every $A \in \mathcal{A}$, as

$$\Phi(A) := \begin{cases} \phi_s(i) & \text{for } A = \{\langle s,i\rangle\}, \\ \phi_{s,t}(i,j) & \text{for } A = \{\langle s,i\rangle, \langle t,j\rangle\}. \end{cases}$$

For a given labeling ℓ, we define $\phi_\ell \colon \mathcal{D} \to [0,\infty)$, for every $D \in \mathcal{D}$, as $\phi_\ell(D) := \Phi(\ell \restriction D)$, that is,

$$\phi_\ell(D) := \begin{cases} \phi_s(\ell(s)) & \text{for } D = \{s\} \in \mathcal{V}, \\ \phi_{s,t}(\ell(s),\ell(t)) & \text{for } D = \{s,t\} \in \mathcal{E}, \end{cases}$$

where $\ell \restriction D$ is the restriction of ℓ to D. With this notation, we may write the objective function E_∞ as

$$E_\infty(\ell) = \|\phi_\ell\|_\infty = \max_{D \in \mathcal{D}} \phi_\ell(D). \tag{7}$$

Similarly, $E_p(\ell) = \|\phi_\ell\|_p$ for any $p \in [1,\infty)$.

3.1 Consistency

Conceptually, both the proposed algorithms work as follows: starting from the set of all possible unary and binary atoms, the algorithm iteratively removes one atom at a time until the remaining atoms define a unique labeling. A key issue in this process is to ensure that, at all steps of the algorithm, at least one labeling can be constructed from the set of remaining atoms.

Let ℓ be a binary labeling. We define $\mathcal{A}(\ell)$, the *atoms for* ℓ, as the family

$$\mathcal{A}(\ell) := \{\ell \restriction D \colon D \in \mathcal{D}\}.$$

Notice that ℓ can be easily recovered from $\mathcal{A}(\ell)$ as its union: $\ell = \bigcup \mathcal{A}(\ell)$.

Definition 1. *Let $\mathcal{A}' \subset \mathcal{A}$ be a set of atoms. We say that $\mathcal{A}'$ is* consistent *if there exists at least one labeling ℓ such that $\mathcal{A}(\ell) \subseteq \mathcal{A}'$.*

We will now derive one of our main results, namely that the problem of determining whether a given set of atoms is consistent can be formulated as a 2-satisfiability problem. The 2-satisfiability problem is a well-studied problem in computer science, and several efficient algorithms exists for its solution. This result quite directly leads to Algorithm 1, presented in Section 5, for finding a labeling minimizing E_∞.

For a set $\mathcal{A}' \subseteq \mathcal{A}$ of atoms denote by $\bar{\mathcal{A}}'$ the complement of $\mathcal{A}'$ relative to $\mathcal{A}$, that is, $\bar{\mathcal{A}}' := \mathcal{A} \setminus \mathcal{A}'$. Then $\mathcal{A}'$ is consistent if, and only if, there exists a labeling ℓ such that $\mathcal{A}(\ell) \cap \bar{\mathcal{A}}' = \emptyset$. We will show that the existence of such labeling ℓ can be determined by solving a 2-satisfiability problem.

For this, let's treat any vertex $v \in V$ of our graph as a variable of propositional calculus, that is, a variable

[5] We note that this algorithm is also sometimes referred to as the Jarník-Prim-Dikstra algorithm, as it was independently discovered by these three authors [12,11,19]

that can take two possible values: TRUE, which will be identified with number 1, and FALSE, which will be identified with 0. Upon such identification, any labeling $\ell\colon V \to \{0,1\}$ can be treated as a truth functional.

Now, with any unary atom $A = \{\langle s,i\rangle\}$, with $i \in \{0,1\}$, we associate a propositional calculus formula in a very simple format known as *literal* (i.e., a variable or its negation):

$$\psi_A(s) := \begin{cases} \neg s & \text{if } i = 1, \\ s & \text{if } i = 0. \end{cases}$$

Less formally, but more concisely, $\psi_A(s) :=$ "$s \neq i$". Notice that $\ell\colon V \to \{0,1\}$ disagrees with A if, and only if, ψ_A is satisfied by ℓ treated as a truth functional.

Similarly, for every binary atom $A = \{\langle s,i\rangle, \langle t,j\rangle\}$ we define

$$\psi_A(s,t) := \psi_{\{\langle s,i\rangle\}}(s) \vee \psi_{\{\langle t,j\rangle\}}(t)$$

or, equivalently, as "$(s \neq i) \vee (t \neq j)$". Once again, $\ell\colon V \to \{0,1\}$ disagrees with A if, and only if, ψ_A is satisfied by ℓ treated as a truth functional.

Finally, for a set $\mathcal{A}' = \{A_1, A_2, \ldots, A_m\}$ of atoms define

$$\psi_{\mathcal{A}'} := \bigwedge_{i=1}^{m} \psi_{A_i} = \psi_{A_1} \wedge \cdots \wedge \psi_{A_m}.$$

Also, $\ell\colon V \to \{0,1\}$ disagrees with every $A \in \mathcal{A}'$ if, and only if, $\psi_{\mathcal{A}'}$ is satisfied by ℓ. Notice also that the formula $\psi_{\mathcal{A}'}$ is in the so called *2-conjunctive normal form*, that is, it is a conjunction of formulas ψ_{A_i}, each of which is a disjunction of at most two literals.

The above discussion leads to the following result.

Theorem 2. *A set $\mathcal{A}' \subseteq \mathcal{A}$ of atoms is consistent if, and only if, the 2-satisfiability problem for a formula $\psi_{\bar{\mathcal{A}}'}$ has a positive solution.*

Proof. This follows from the equivalence of the following conditions, each consecutive pair of which was argued above.

- $\mathcal{A}' \subseteq \mathcal{A}$ is consistent.
- $\mathcal{A}(\ell) \cap \bar{\mathcal{A}}' = \emptyset$ for some $\ell\colon V \to \{0,1\}$.
- There is an $\ell\colon V \to \{0,1\}$ which disagrees with every $A \in \bar{\mathcal{A}}'$.
- There is an $\ell\colon V \to \{0,1\}$ such that $\psi_{\bar{\mathcal{A}}'}$ is satisfied by ℓ.
- The 2-satisfiability problem for a formula $\psi_{\bar{\mathcal{A}}'}$ has a positive solution.

□

Recall, that the solution to the 2-satisfiability problem for a formula in the 2-conjunctive normal form that is a conjunction of n 2-disjunctions can be found in $\mathcal{O}(n)$ time, using, e.g., the algorithm by Aspvall et al. [3]. Thus, for any set $\mathcal{A}' \subseteq \mathcal{A}$ of atoms, the question

Is $\mathcal{A}'$ consistent?

can be answered in a linear time with respect to the number $n := |\bar{\mathcal{A}}'|$ of elements in $\bar{\mathcal{A}}' = \mathcal{A} \setminus \mathcal{A}'$ by deciding the satisfiability of $\psi_{\bar{\mathcal{A}}'}$.

4 Strict optimality

In this section we will introduce a refinement of the L_∞ norm measure. This will help us in the discussion of the two proposed algorithms, which will be introduced in the next two sections.

A potential drawback of the L_∞-norm is that it does not distinguish between solutions with high or low errors below the maximum error. To resolve this problem, Levi and Zorin introduced, in a 2014 paper [15], the concept of *strict minimizers*.[6] In this framework, two solutions are compared by ordering all elements (in our case, binary and unary terms) non-increasingly by their local error value, and then performing their lexicographical comparison.

Formally, using the notation from Section 3, let ℓ_1 and ℓ_2 be two labelings. Furthermore, let $\langle A_1, A_2, \ldots, A_k\rangle$ and $\langle B_1, B_2, \ldots, B_k\rangle$ be the sequences of all atoms in $\mathcal{A}(\ell_1)$ and $\mathcal{A}(\ell_2)$, respectively, each ordered by the decreasing costs of atoms, that is, with $\Phi(A_1) \geq \cdots \geq \Phi(A_k)$ and $\Phi(B_1) \geq \cdots \geq \Phi(B_k)$. We say that ℓ_1 *precedes ℓ_2 lexicographically*, and denote this as $\ell_1 \prec \ell_2$, provided there exists an $i \in \{1,2,\ldots,k\}$ such that $\Phi(A_i) \neq \Phi(B_i)$ and for the smallest such i we have $\Phi(A_i) < \Phi(B_i)$. Also, we write $\ell_1 \preceq \ell_2$ provided either $\ell_1 \prec \ell_2$ or $\Phi(A_i) = \Phi(B_i)$ for all $i \in \{1,2,\ldots,k\}$.

Definition 2. *A labeling ℓ is said to be* strictly minimal *provided $\ell \preceq \ell'$ for any other labeling ℓ'.*

From this definition, it is clear that any strict minimizer is also an L_∞-optimal solution. Thus, the set of all strict minimizers is a subset of all L_∞-norm optimal solutions. In fact, the limit, as $p \to \infty$, of L_p-norm minimizers discussed above, is not only an L_∞-minimizer but also a strict minimizer [15]. (For the local cost functions satisfying the property (D) it was proved earlier, in a 2012 paper [6] of Ciesielski et al.)[7]

[6] See also the 2010 paper by Ciesielski and Udupa [5] where strict optimization was earlier considered in a similar setting.

[7] Specifically, [6, theorem 5.3] states that for $q > 0$ large enough we have $\mathcal{P}^q(S,T) = \hat{\mathcal{P}}_{\max}(S,T)$, where parameters S

The above discussion indicates, that it would be desirable to have an efficient algorithm that not only finds L_∞-minimizers, but also strict minimizers. Unfortunately, in the general setting that we examine here, the problem of finding strict minimizers is NP-hard. We will show this at the end of this section. Nevertheless, there are two special situations in which efficient algorithms for finding strict minimizers do exist. The first case are described in the next subsection. The second one, discussed in Section 5.1 and solved by the algorithm presented there, is when all local terms have distinct weights.

4.1 When all ϕ_{st} are p-submodular for large enough p

For a finite set $Z \subset [0,\infty)$ and $k \geq 1$ let $\delta_Z^k := \log_b k$, where

$$b := \min\left\{\frac{s}{r} \colon 0 < r < s \;\&\; r,s \in Z\right\}.$$

We will use the following result, that identifies the strict optimality with the optimality with respect to E_p for p large enough. For the local costs maps satisfying (D) this was first proved in [6, theorem 5.3].

Proposition 1. *Let $|V| = k$ and assume that all local cost maps ϕ_s and $\phi_{s,t}$ have values in a finite set $Z \subset [0,\infty)$. If $p \geq \delta_Z^k$, then a binary labeling ℓ is strictly minimal if, and only if, it is minimal with respect to E_p.*

Proof. To see this, notice first that for every $p \geq \delta_Z^k$

$$\text{if } \ell_1 \prec \ell_2, \text{ then } E_p(\ell_1) < E_p(\ell_2). \tag{8}$$

Indeed, using the notation as in the definition of $\prec$, let i be the smallest such that $\Phi(A_i) < \Phi(B_i)$. If $\Phi(A_i) = 0$, then $E_p^p(\ell_1) = \sum_{j=1}^{i-1} \Phi^p(A_j) < \sum_{j=1}^{k} \Phi^p(B_j) = E_p^p(\ell_2)$ justifying (8). So, assume that $\Phi(A_i) > 0$. Then, for b defined as above, we have $b \leq \frac{\Phi(B_i)}{\Phi(A_i)}$ and

$$\log_b k = \delta_Z^k \leq p \leq p \log_b \frac{\Phi(B_i)}{\Phi(A_i)} = \log_b \frac{\Phi^p(B_i)}{\Phi^p(A_i)}$$

so that $k\Phi^p(A_i) < \Phi^p(B_i)$. Therefore,

$$E_p^p(\ell_1) \leq \sum_{j=1}^{i-1} \Phi^p(A_j) + k\Phi^p(A_i) < \sum_{j=1}^{k} \Phi^p(B_j) = E_p^p(\ell_2),$$

completing the argument for (8).

To prove the proposition, choose $p \geq \delta_Z^k$ and labelings ℓ_1 and ℓ_2. If ℓ_1 is strictly minimal, then either $\ell_1 \prec \ell_2$, in which case (8) implies that $E_p(\ell_1) < E_p(\ell_2)$, or $\langle \Phi(A_1), \ldots, \Phi(A_k)\rangle = \langle \Phi(B_1), \ldots, \Phi(B_k)\rangle$, in which case clearly $E_p(\ell_1) = E_p(\ell_2)$. Thus, strict minimality of ℓ_1 indeed implied its minimality with respect to E_p.

Conversely, if ℓ_1 is minimal with respect to E_p, then we must have $\ell_1 \preceq \ell_2$, since otherwise we would have $\ell_2 \prec \ell_1$ and, by (8), $E_p(\ell_2) < E_p(\ell_1)$, a contradiction. □

A number p for which the proposition holds is referred to by Wolf et al. [21] as a *dominant power*. Its existence is proved in that paper; however, no estimate similar to that of δ_Z is provided there. The estimate δ_Z can be found, in a similar settings, in [6, theorem 5.3]; however, this result does not explicitly relate this number with the lexicographical order.

The proposition immediately implies the next theorem.

Theorem 3. *Let δ_Z be as in Proposition 1 and assume that $p \in [\delta_Z, \infty)$ is such that all terms ϕ_{st} are submodular. Then any labeling ℓ minimizing E_p is a strict minimizer. In particular, if there is a $\rho \in [1,\infty)$ such that ϕ is ρ-submodular and ∞-submodular, then there is a $p \in [\rho,\infty)$ such that any E_p-optimizing label ℓ returned by max-flow/min-cut algorithm is a strict optimizer.*

We observe that in practice, the dominant power p may be large. This may give rise to numerical issues when solving the max-flow/min-cut problem, as each local cost is raised to the power p. The novel algorithms proposed in Sections 5 and 6 do not suffer from his potential issue.

4.2 NP-hardness of finding strict optimizers

We will now show that, in the general case, the problem of finding strict optimizers is indeed NP-hard. This is justified by an example from Kolmogorov and Zabih [13, Appendix A] that shows that L_1-optimality for nonsubmodular energies is NP-hard.

Recall, that the set U of vertices of a graph $\mathcal{G} = \langle V, \mathcal{E}\rangle$ is independent when it contains no two vertices connected by an edge. It is known, that the problem of finding maximal independent set of vertices of an arbitrary graph is NP-hard [7, chapter 34].

In the example, associate the following local costs:

- for every vertex v of label i, give the cost $1 - i$;
- for every edge with both vertices of label 1, let the cost be $N := |V| + 1$;
- with any other edge, associate the cost 0.

and T indicate that the unary local cost maps ensure that for any optimal label ℓ we have $S \subset \ell^{-1}(1)$ and $T \subset \ell^{-1}(0)$ (i.e., $\psi_s(i) = \infty$ if, and only if, either $i = 0$ and $s \in S$ or else $i = 1$ and $s \in T$), $\mathcal{P}^q(S,T)$ is the set of all labelings minimizing E_q, while $\hat{\mathcal{P}}_{\max}(S,T)$ is the set of all strictly optimal labelings.

Notice that the max-cost of any labeling ℓ is $< N$ if, and only if, the set $U := \ell^{-1}(1)$ is independent. Among all labelings ℓ associated with an independent U, the max cost is 1. Moreover, the labeling ℓ is a strict minimizer when the number of cost 1 atoms for U, which is $|V| - |U|$, is minimal, that is, when the size of U is maximal.

In other words, if for a graph $\mathcal{G}$ we use the local costs assignments as above, then ℓ is a strict minimizer if, and only if, $U := \ell^{-1}(1)$ is a maximal independent set of vertices. So, our problem is indeed NP-hard, similarly as the problem of finding maximal independent set of vertices.

5 A quadratic time algorithm for direct optimization of E_∞

With these preliminaries in place, we are now ready to introduce a general method for finding a binary labeling that globally optimizes E_∞. Pseudocode for this method is given in Algorithm 1.

Algorithm 1: Labeling Algorithm, general case

Data: A graph $\mathcal{G} = \langle V, \mathcal{E}\rangle$ and associated $\Phi\colon \mathcal{A} \to [0,\infty)$ generating energy E_∞
Result: A labeling $\ell\colon V \to \{0,1\}$ minimizing E_∞
Additional Structure: A max-priority queue H, a set L of atoms approximating ℓ.

1 $\mathsf{H} \leftarrow \mathcal{A}$ and $\mathsf{L} \leftarrow \emptyset$
2 **while** $\mathsf{H} \neq \emptyset$ **do**
3 **remove** the first atom A **from** H
4 **if** $\mathsf{H} \cup \mathsf{L}$ is not consistent **then** insert A to L
5 return $\ell \leftarrow \bigcup \mathsf{L}$

If n is the number of elements, atoms, in $\mathcal{A}$, then Algorithm 1 terminates after $O(n^2)$ operations. This is the case, since the execution of line 1 has complexity $O(n \ln n)$ (as it requires ordering of H) while the loop 2-4 is executed n times and each its execution requires $O(n)$ operations, as we indicated after Theorem 2.

Theorem 4. *An ℓ returned by Algorithm 1 is a labeling minimizing energy E_∞.*

Proof. The main loop 2-4 is executed precisely n-times, where $n := |\mathcal{A}|$.

For every $k \in \{0, 1, \ldots, n\}$ let H_k and L_k be the states of H and L, respectively, directly after the kth execution of the loop 2-4. First notice that, for every $k \in \{0, 1, \ldots, n\}$,

(C_k) $\mathsf{H}_k \cup \mathsf{L}_k$ is consistent.

Clearly $\mathsf{H}_0 \cup \mathsf{L}_0 = \mathcal{A}$, is consistent. Also, for every $k < n$, if $\mathsf{H}_k \cup \mathsf{L}_k$ is consistent, then so is $\mathsf{H}_{k+1} \cup \mathsf{L}_{k+1}$. Indeed, if during the $(k+1)$st execution of line 3 an atom A is removed from H_k, then $\mathsf{H}_{k+1} = \mathsf{H}_k \setminus \{A\}$. If $\mathsf{H}_{k+1} \cup \mathsf{L}_k$ is consistent, then $\mathsf{L}_{k+1} = \mathsf{L}_k$ and (C_{k+1}) holds. Otherwise, line 4 ensures that $\mathsf{L}_{k+1} = \mathsf{L}_k \cup \{A\}$ and $\mathsf{H}_{k+1} \cup \mathsf{L}_{k+1} = \mathsf{H}_k \cup \mathsf{L}_k$ is consistent by (C_k).

The above shows that $\mathsf{H}_n \cup \mathsf{L}_n = \mathsf{L}_n$ is consistent, that is, there exists a labeling $\ell'\colon V \to \{0,1\}$ so that $\mathcal{A}(\ell') \subseteq \mathsf{L}_n$. To finish the proof that $\ell = \bigcup \mathsf{L}_n$ is a labeling we need to show that $\mathcal{A}(\ell') = \mathsf{L}_n$.

So see this, first notice that $\mathsf{H}_{k+1} \cup \mathsf{L}_{k+1} \subseteq \mathsf{H}_k \cup \mathsf{L}_k$ for every $k < n$. So, $\mathcal{A}(\ell') \subseteq \mathsf{L}_n \subseteq \mathsf{H}_k \cup \mathsf{L}_k$. To see that $\mathsf{L}_n \subseteq \mathcal{A}(\ell')$, assume by way of contradiction that there is an $A \in \mathsf{L}_n \setminus \mathcal{A}(\ell')$. Then, A is removed from H during some, say kth, execution of line 2. So, $A \notin \mathsf{H}_{k+1}$. Also, if $A \notin \mathcal{A}(\ell')$, then $\mathsf{H}_{k+1} \cup \mathsf{L}_k$ is consistent, as it contains $\mathcal{A}(\ell')$. Therefore, $\mathsf{L}_{k+1} = \mathsf{L}_k$ and $A \notin \mathsf{H}_{k+1} \cup \mathsf{L}_{k+1} \supset \mathsf{L}_n$, a contradiction. This means that $\mathcal{A}(\ell') = \mathsf{L}_n$.

Finally, by way of contradiction, assume that $\ell = \bigcup \mathsf{L}_n$ does not minimize E_∞, that is, that there is a labeling ℓ' with $c := E_\infty(\ell') < E_\infty(\ell)$. Then, there is an $A \in \mathcal{A}(\ell)$ of cost $> c$. Let $k \leq n$ be such that A is removed from H during the kth execution of line 2. Then $A \notin \mathsf{H}_{k+1}$. Also, by the ordering of H, we have $\mathcal{A}(\ell') \subset \mathsf{H}_{k+1}$. So, $\mathsf{H}_{k+1} \cup \mathsf{L}_k$ is consistent and $\mathsf{L}_{k+1} = \mathsf{L}_k$. In particular, $A \notin \mathsf{H}_{k+1} \cup \mathsf{L}_{k+1} \supset \mathsf{L}_n = \mathcal{A}(\ell)$, contradicting the fact that $A \in \mathcal{A}(\ell)$. □

5.1 Atoms with unique weights

We say that the atoms (in $\mathcal{A}$) *have unique weights* provided the map $\Phi\colon \mathcal{A} \to [0,\infty)$ is injective, that is, when $\Phi(A_1) \neq \Phi(A_2)$ for every distinct $A_1, A_2 \in \mathcal{A}$. Our main result here is the following

Theorem 5. *If the atoms in $\mathcal{A}$ have unique weights, then the labeling ℓ returned by Algorithm 1 is the unique strict optimizer.*

First we prove the uniqueness part of the theorem, in form of the following lemma.

Lemma 1. *If the atoms in $\mathcal{A}$ have unique weights, then the strictly optimal labeling is unique.*

Proof. Let ℓ_1 and ℓ_2 be strictly optimal labelings. We will show that $\ell_1 = \ell_2$.

To see this, consider the sequences of the atoms in $\mathcal{A}(\ell_1)$ and $\mathcal{A}(\ell_2)$, respectively, each ordered by decreasing cost. Then, since both labelings are strictly optimal, the decreasing sequences of the costs of the atoms in $\mathcal{A}(\ell_1)$ and $\mathcal{A}(\ell_2)$ must be identical. However, since every atom has a unique weight, this means that the sets of atoms in $\mathcal{A}(\ell_1)$ and in $\mathcal{A}(\ell_2)$ must themselves

be identical. In particular $\mathcal{A}(\ell_1) = \mathcal{A}(\ell_2)$ and therefore $\ell_1 = \bigcup \mathcal{A}(\ell_1) = \bigcup \mathcal{A}(\ell_2) = \ell_2$, as needed. □

Proof of Theorem 5. We will use the same notation as in the proof of Theorem 4. Let ℓ and ℓ' be distinct labelings such that ℓ is strictly optimal and, by way of contradiction, assume that Algorithm 1 returns labeling ℓ' rather than ℓ. Fix the sequences $\langle A_1, A_2, \ldots, A_m\rangle$ and $\langle B_1, B_2, \ldots, B_m\rangle$ of all atoms in $\mathcal{A}(\ell)$ and $\mathcal{A}(\ell')$, respectively, each ordered by the decreasing costs of atoms. By Lemma 1 , we have $\ell \prec \ell'$. Therefore, there exists an $i \in \{1, 2, \ldots, m\}$ such that $\Phi(A_i) < \Phi(B_i)$ and $\Phi(A_j) = \Phi(B_j)$ for all $j < i$.

Let $k \leq n$ be such that B_i is removed from H during the kth execution of line 2. Then, $\{B_1, B_2, \ldots, B_m\} = \mathcal{A}(\ell') \subset \mathsf{L}_n \subset \mathsf{H}_k \cup \mathsf{L}_k$. In fact, by the ordering principle of H we have $\{A_1, \ldots, A_{i-1}\} = \{B_1, \ldots, B_{i-1}\} \subset \mathsf{H}_k$ and $\{A_i, \ldots, A_n\} \subset \mathsf{L}_k$. In particular, $\mathsf{H}_{k+1} \cup \mathsf{L}_k$ is consistent since it contains $\{A_1, A_2, \ldots, A_m\} = \mathcal{A}(\ell)$. Thus, $\mathsf{L}_{k+1} = \mathsf{L}_k$ and $B_i \notin \mathsf{H}_{k+1} \cup \mathsf{L}_{k+1} \supset \mathsf{L}_n = \mathcal{A}(\ell')$, a contradiction that finishes the proof of Theorem 5. □

The requirement in Theorem 5 (and the forthcoming Theorem 7) that all atoms in $\mathcal{A}$ have unique weights may appear restrictive, and for real world problems this condition may or may not hold. We will therefore now discuss how these theorems may be interpreted when all atoms weights are not unique. First we observe that when all atom weights are not unique, it is straightforward to define a new local cost function $\hat{\Phi}$ with unique weights and such that, for any atoms $A, A' \in \mathcal{A}$, $\Phi(A) < \Phi(A')$ implies $\hat{\Phi}(A) < \hat{\Phi}(A')$. Such weights may, e.g., be defined by the following simple procedure:

- Fix, by some method (e.g., a sorting algorithm), an increasing order of the atoms in $\mathcal{A}$ by weight, i.e., find a map $O\colon \mathcal{A} \to Z$ such that $O(A_1) \neq O(A_2)$ for every distinct $A_1, A_2, \in \mathcal{A}$ and $O(A_1) < O(A_2) \Rightarrow \Phi(A_1) \leq \Phi(A_j)$ for all $A_1, A_2 \in \mathcal{A}$.
- For all $A \in \mathcal{A}$, define $\hat{\Phi}(A) := O(A)$.

By design, all atoms associated with the local costs $\hat{\phi}$ have unique weights and thus running Algorithm 1 (or Algorithm 2 in case of Theorem 7) with these weights will return a strict optimizer with respect to the local costs $\hat{\Phi}$.

We observe that if the original atom weights are all unique, then the ordering O is also unique and running either of our new algorithms with the new local costs $\hat{\Phi}$ induced by O would yield an identical result as with the original weights. Furthermore, we observe that the procedure above is essentially what happens during the execution of the algorithms: by ordering the max-priority queue H, we are establishing a specific (implementation dependent) ordering of the atoms that is increasing by weight just like the ordering O defined in the procedure above. Thus, even when all atoms do not have unique weights, the algorithms will return labelings that are strictly optimal with respect to *some* increasing order of the atoms by weight. When all atom weights are not unique, however, this ordering will not be unique but will depend on the specific implementation of the max-priority queue H.

6 A quasi-linear time algorithm for direct optimization of E_∞ when all binary terms are ∞-submodular

We now present a more efficient algorithm, previously reported in the conference version of this manuscript [16], for the case when all binary terms are ∞-submodular. Superficially, this algorithm is slightly more complicated than Algorithm 1. We emphasize however, that both algorithms have a very similar structure – starting from the set of all possible atoms, both algorithms iteratively remove one atom at a time until the remaining atoms define a unique labeling. The main difference between the algorithms is the steps taken to ensure the consistency of the set of remaining atoms.

6.1 Local consistency, incompatible atoms

We introduce a property of local consistency, which will be used to establish the correctness of our second proposed algorithm. A set of atoms $\mathcal{A}'$ is said to be *locally consistent* if, for every vertex $s \in V$ and edge $\{s, t\} \in \mathcal{E}$ there are $i, j \in \{0, 1\}$ such that the atoms $\{\langle s, i\rangle\}$ and $\{\langle s, i\rangle, \langle t, j\rangle\}$ both belong to $\mathcal{A}'$ (i.e., that $\mathcal{A}'$ still allows that s will have some label). Clearly, any consistent set of atoms is also locally consistent. However, in general, local consistency does not imply consistency. [8]

Furthermore we introduce the notion of an incompatible atom, which will be needed for the exposition of the proposed algorithm. For a given set of $\mathcal{A}'$ of atoms, we say that an atom $A \in \mathcal{A}'$ is (locally) *incompatible* (w.r.t. $\mathcal{A}'$) if either

1. A is a unary atom so that $A = \{\langle v, i\rangle\}$ for some vertex v, and there exists some edge $\{v, w\}$ adjacent to v such that $\mathcal{A}'$ contains neither $\{\langle v, i\rangle, \langle w, 0\rangle\}$ nor $\{\langle v, i\rangle, \langle w, 1\rangle\}$; or

[8] For example, if g is a complete graph with three vertices $V = \{a, b, c\}$ and $\mathcal{A}$ consists of all unary atoms and the binary atoms $\{\langle a, i\rangle, \langle b, i\rangle\}$, $\{\langle a, i\rangle, \langle c, i\rangle\}$, $\{\langle b, i\rangle, \langle c, 1 - i\rangle\}$ for $i \in \{0, 1\}$, then $\mathcal{A}$ is locally consistent, but not (globally) consistent.

2. A is a binary atom so that $A = \{\langle v, i\rangle, \langle w, j\rangle\}$ for some edge $\{v, w\}$, and at least one of $\{\langle v, i\rangle\}$ and $\{\langle w, j\rangle\}$ is not in $\mathcal{A}'$.

Note that a locally consistent set of atoms may still contain incompatible atoms.

6.2 The second algorithm

We now introduce the proposed algorithm, with quasi-linear time complexity, for finding a binary label assignment $\ell\colon V \to \{0, 1\}$ that globally minimizes the objective function E_∞ given by (1), under the condition that all pairwise terms in the objective function are ∞-submodular. If, additionally, all atoms have unique weights then the labeling returned by the algorithm is also the strict minimizer. Informally, the general outline of the proposed algorithm is as follows:

- Start with a set S consisting of all possible atoms and an initially empty set I of atoms identified as incompatible. (Recall that the total number of atoms is $\mathcal{O}(|V| + |\mathcal{E}|)$.)
- For each atom A, in order of decreasing cost $\Phi(A)$:
  - If A is still in S, and is not the only remaining atom for that vertex/edge, remove A from S.
  - After the removal of A, S may contain incompatible atoms. Iteratively remove all such incompatible atoms until S contains no more incompatible atoms.

Before we formalize this algorithm, we introduce a specific preordering relation $\gg$ on the atoms $\mathcal{A}$. For $A_0, A_1 \in \mathcal{A}$ we will write $A_0 \gg A_1$ if either $\Phi(A_0) > \Phi(A_1)$, or else $\Phi(A_0) = \Phi(A_1)$ and A_1 is a binary atom of the form $\{\langle s, i\rangle, \langle t, i\rangle\}$ (equal labeling) while A_0 is not in this form.

With these preliminaries in place, we are now ready to introduce the proposed algorithm, for which pseudocode is given in Algorithm 2.

Algorithm 2: Labeling Algorithm, for the ∞-submodular objective functions

Data: A graph $\mathcal{G} = \langle V, \mathcal{E}\rangle$ and associated $\Phi\colon \mathcal{A} \to [0, \infty)$ generating ∞-submodular energy E_∞
Result: A labeling $\ell\colon V \to \{0, 1\}$ minimizing energy E_∞
Additional Structure: An array A of buckets of atoms, indexed by $\mathcal{D} = \mathcal{V} \cup \mathcal{E}$; a list H of atoms; a queue K of vertices/edges such that every vertex in K precedes any edge.

```
1  foreach vertex/edge D ∈ 𝒟 do insert all D-atoms to A[D]
2  create a list H of all atoms 𝒜 such that A_0 precedes A_1 in 𝒜 whenever A_0 ≫ A_1
3  while H ≠ ∅ do
4      remove the first atom A from H
5      if D ∈ 𝒟 is a vertex/edge of A and A[D] has more than one element then
6          remove A from A[D] and insert D to (previously empty) K
7          while K ≠ ∅ do
8              remove a vertex/edge C from K
9              foreach edge/vertex D adjacent to C do
10                 remove from A[D] and H all A incompatible with ⋃_{D'∈𝒟} A[D']
11             if any atom was removed from A[D] and H in line 10 then
12                 insert to K any vertex/edge C' adjacent to D: to its top, when C' is a vertex and its bottom when C' is an edge
13 return ℓ = ⋃_{D∈𝒟} A[D]
```

6.3 Computational complexity

We now analyze the asymptotic computational complexity of Algorithm 2. First, let $\eta := |\mathcal{A}| = 2|V| + 4|\mathcal{E}|$. In image processing applications the graph $\mathcal{G}$ is commonly sparse, in the sense that $\mathcal{O}(|V|) = \mathcal{O}(|\mathcal{E}|)$. In this case, we have $\mathcal{O}(\eta) = \mathcal{O}(|V|)$.

Creating the list H requires us to sort all atoms in $\mathcal{A}$. The sorting can be performed in $\mathcal{O}(\eta \log \eta)$ time. In some cases, e.g., if all unary and binary terms are integer valued, the sorting may be possible to perform in $\mathcal{O}(\eta)$ time using, e.g., *radix* or *bucket* sort.

We make the reasonable assumption that the following operations can all be performed in $\mathcal{O}(1)$ time:

- Remove an atom from H.
- Remove an atom from $\mathsf{A}(D)$.
- Remove or insert elements in K.
- Given an atom, find its corresponding edge or vertex.
- Given a vertex, find all edges incident at that vertex.
- Given an edge, find the vertices spanned by the edge.

The combined number of the executions of the main loop, lines 3-12, and of the internal loop, lines 7-12, equals to $|\mathcal{A}|$, that is, $\mathcal{O}(\eta)$. This is so, since any insertion of an atom into K requires its prior removal from the list H. If the assumptions above are satisfied, it is easily seen that only $\mathcal{O}(1)$ operations are needed between consecutive removals of an atom from H. Therefore, the amortized cost of the execution of the main loop is $\mathcal{O}(\eta)$.

Thus, the total computational cost of the algorithm is bounded by the time required to sort $\mathcal{O}(\eta)$ elements, i.e., at most $\mathcal{O}(\eta \log \eta)$.

6.4 Proof of correctness

Theorem 6. *If all binary terms of the cost function $\Phi\colon \mathcal{D} \to [0,\infty)$ associated with graph $\mathcal{G} = \langle V, \mathcal{E}\rangle$ are ∞-submodular, then ℓ returned by Algorithm 2 is a labeling of V minimizing the objective function E_∞.*

Let $\mathsf{n} := |V| + 3|\mathcal{E}|$, the number of removals of an atom from A. For every $D \in \mathcal{D}$ and $k \in \{0,\dots,\mathsf{n}\}$ let $\mathsf{A}_k[D]$ be equal to the value of $\mathsf{A}[D]$ directly after the k-th removal of some atom(s) from A, which can happen only as a result of execution of either line 6 or line 10. (For $k = 0$ we mean, directly after the execution of line 2.) Let $\mathcal{A}_k = \bigcup_{D\in\mathcal{D}} \mathsf{A}_k[D]$.

Let $1 = k_1 < \cdots < k_m$ be the list of all values of $k \in \{1,\dots,\mathsf{n}\}$ such that $\mathcal{A}_k$ is a proper refinement of $\mathcal{A}_{k-1}$ resulting from the execution of line 6. Note that it is conceivable that the numbers k_j and k_{j+1} are consecutive—this happens when the execution of loop 8-12 directly after the execution of line 5 has been used to create $\mathcal{A}_{k_j}$ resulted in removal of no atoms from $\mathcal{A}_{k_j}$.

The proof of Theorem 6 is based on the following Lemma, for which a proof is given in Appendix Section.

Lemma 2. *During the execution of Algorithm 2, the following properties hold for every $k \le \mathsf{n}$.*

(P0) For every edge $D = \{v, w\}$, if $\mathsf{A}_k[D]$ is missing either $\{\langle v,0\rangle, \langle w,0\rangle\}$ or $\{\langle v,1\rangle, \langle w,1\rangle\}$, then it must be also missing $\{\langle v,1\rangle, \langle w,0\rangle\}$ or $\{\langle v,0\rangle, \langle w,1\rangle\}$.
(P1) $\mathsf{A}_k[D]$ contains at least one atom for every $D \in \mathcal{D}$.
(P2) $\mathcal{A}_k$ is locally consistent.
(P3) $\mathcal{A}_k$ has no incompatible atoms directly before any execution of line 4.

Proof of Theorem 6. Beside Lemma 2, we still need to argue for two facts. First notice that the algorithm does not stop until all buckets $\mathsf{A}_\mathsf{n}[D]$, $D \in \mathcal{D}$, have precisely one element. Thus, since $\mathcal{A}_\mathsf{n}$ is locally consistent, $\ell = \bigcup_{D\in\mathcal{D}} \mathsf{A}[D]$ is indeed a function from V into $\{0,1\}$.

To finish the proof, we need to show that ℓ indeed minimizes energy E_∞. For this, first notice that at any time of the execution of the algorithm, any atom in H is also in $\bigcup_{D\in\mathcal{D}} \mathsf{A}[D]$. Indeed, these sets are equal immediately after the initialization and we remove from $\bigcup_{D\in\mathcal{D}} \mathsf{A}[D]$ only those atoms, that have been already removed from H. Now, let $L\colon V \to \{0,1\}$ be a labeling minimizing E_∞. We claim, that the following property holds any time during the execution of the algorithm:

(P) if $\Phi(A') > E_\infty(L)$ for some $A' \in \bigcup_{D\in\mathcal{D}} \mathsf{A}[D]$, then $\mathcal{A}[L] \subset \bigcup_{D\in\mathcal{D}} \mathsf{A}[D]$.

Indeed, it certainly holds immediately after the initialization. This cannot be changed during the execution of line 6 when the assumption is satisfied, since then A considered there has just been removed from $\mathsf{H} \supset \bigcup_{D\in\mathcal{D}} \mathsf{A}[D]$ and

$$\Phi(A) \ge \max_{H\in\mathsf{H}} \Phi(H) \ge \max_{H\in\bigcup_{D\in\mathcal{D}} \mathsf{A}[D]} \Phi(H) \ge \Phi(A') > E_\infty(L) = \max_{H\in\mathcal{A}[L]} \Phi(H),$$

so $A \notin \mathcal{A}[L]$. Also, (P) is not affected by an execution of line 10, since the inclusion $\mathcal{A}[L] \subset \bigcup_{D\in\mathcal{D}} \mathsf{A}[D]$ is not affected by it: no atom in $\mathcal{A}[L]$ is incompatible with $\mathcal{A}[L]$ so also with $\bigcup_{D\in\mathcal{D}} \mathsf{A}[D]$. This concludes the proof of (P).

Now, by the property (P), after the termination of the main loop, we have either $\mathcal{A}[L] \subset \bigcup_{D\in\mathcal{D}} \mathsf{A}[D]$, in which case $\ell = L$ have minimal E_∞ energy, or else

$$E_\infty(L) \ge \max_{H\in\bigcup_{D\in\mathcal{D}} \mathsf{A}[D]} \Phi(H) = \max_{H\in\mathsf{H}} \mathcal{A}[\ell] = E_\infty(\ell)$$

once again ensuring optimality of ℓ. □

Theorem 7. *If the atoms in $\mathcal{A}$ have unique weights, then the labeling ℓ returned by Algorithm 2 is the unique strict optimizer.*

Proof. The uniqueness part of the theorem is already shown in Lemma 1. The rest of the argument is essentially identical to that used in the proof of Theorem 5. □

7 NP-hardness of multi-label E_∞-optimization

We will now show that, for a number of labels $K > 2$, the problem of finding a labeling that minimizes E_∞ is NP-hard in the general case.

Recall that a K-coloring of a graph is a mapping $c : V \to \{1, 2, \dots, K\}$ such that $c(s) \neq c(t)$ for every edge $\{s,t\} \in \mathcal{E}$. The *K-coloring problem* consists of determining whether a given undirected graph admits a K-coloring. Recall also that already 3-coloring problem is NP-complete [7, chapter 34].

To see that optimization of E_∞ is NP-hard for $K > 2$ labels, consider 3 labelings, where we associate the costs:

- for every vertex v the cost of any label assignment is 0;
- for any edge with distinct labeling of its vertices the cost is 0;

– for any edge with the same labeling of its vertices the cost is 1.

For such assignments, the E_∞-energy of a labeling is ≤ 0 if, and only if, the labeling is a 3-coloring. The same argument can be repeated also for $K > 3$. Thus, the problem of E_∞-optimization with $K > 2$ labels is indeed NP-hard.

8 Conclusions

We have presented two algorithms for finding a binary vertex labeling of a graph that globally minimizes objective functions of the form E_∞. It is well known that for a limited subclass of such problems, globally optimal solutions can be found by computing an optimal spanning forest on a suitably constructed graph. Such optimal spanning forests can, in turn, be computed using very efficient, greedy algorithms. Despite the fact that this optimum spanning forest approach is commonly used in many image processing applications, the potential and limitations of this method in terms of more general optimization problems is, to the best of our knowledge, largely unexplored. The exact class of max-norm optimization problems that can be solved using efficient greedy algorithms, or even in polynomial time, has remained unknown. By the introduction of the two proposed algorithms, we show that the class of such problems that can be solved in (low order) polynomial time is indeed larger than what was previously known. In Table 1, we provide a summary of the various subclasses of the general optimization problem considered in this paper, and algorithms for solving them.

An important observation here is the following: Optimization binary labeling problems with objective functions of the form E_1 frequently occur in image processing and computer vision applications. The max-flow/min-cut approach proposed by Kolmogorov and Zabih [13] still remains one of the primary methods for solving such problems when all pairwise terms are submodular. When the local cost functionals include non-submodular terms, however, the same problem becomes NP-hard. As concluded in our discussion in Section 2.1, similar submodularity requirements hold also for the generalized objective functions E_p for any finite p. Practitioners looking to solve such optimization problems must therefore first verify that their local cost functional satisfies the appropriate submodularity conditions. If this is not the case, they must resort to approximate optimization methods that may or may not produce satistfactory results for a given problem instance. Here we show, by the introduction of Algorithm 1, that in the limit as p goes to infinity, the requirement for submodularity of the pairwise terms disappears. Indeed Algorithm 1 returns, in low order polynomial time, a E_∞-minimal binary labeling for any local cost functional. Thus, even when the local costs are such that the problem of minimizing E_p is NP-hard for some or all finite p, a labeling minimizing E_∞ can be found in low order polynomial time.

The motivation for our work comes from image processing applications, and the local cost functionals we consider naturally occurs in many image processing problems. The two proposed algorithms, however, are formulated for general graphs and may thus also have applications to other applied problems in computer science. Structurally, both the proposed algorithms resemble Kruskal's algorithm [14, 7], and in this sense the proposed algorithms can be seen as generalizations of the optimum spanning forest approach to optimization.

Algorithm 1 has quadratic time complexity, and is thus less efficient than Algorithm 2. It appears likely, however, that the time complexity of Algorithm 1 could be reduced further. Specifically, Algorithm 1 operates by solving a series of n 2-satisfiability problem. In the proposed algorithm each such problem is solved in isolation, but we observe that there is a high degree of similarity between each consecutive problem – each 2-satisfiability problem differs from the previous one only by the introduction of one additional disjunction of two literals. Exploring whether this redundancy can be utilized to formulate a more efficient version of Algorithm 1 is an interesting direction for future work.

Another natural extension of the work presented here is to consider optimization with more than two labels. In Section 7 we showed that for more than two labels, finding a labeling that is optimal according to E_∞ is NP-hard in the general case. Nevertheless, as can be seen in Table 1, there are special cases of multilabel max-norm problems that can be solved using Prim's algorithm. Determining the class of multilabel problems that can be solved in low order polynomial time is an interesting direction for future work.

At first glance, the restriction to binary labeling may appear very limiting. We note, however, that many successful methods for approximate multi-label optimization rely on iteratively minimizing binary labeling problems via *move-making* strategies [4]. Thus, the ability to find optimal solutions for problems with two labels potentially has a high relevance also for the multi-label case.

Acknowledgements The authors would like to thank Robin Strand for valuable discussions on the ideas presented in this manuscript.

Optimization of E_∞	2 labels	≥ 3 labels
general case	$\mathcal{O}(n^2)$, **Algorithm 1**	**NP-hard problem**, Sec. 7
∞-submodular binary terms	$\mathcal{O}(n \log n)$, **Algorithm 2**	-
Local costs satisfy property (D)	$\mathcal{O}(n \log n)$, Prim's algorithm	$\mathcal{O}(n \log n)$, Prim's algorithm
Strict optimization	2 labels	≥ 3 labels
general case	**NP-hard problem**, Sec. 4.2	NP-hard problem
unique weights	$\mathcal{O}(n^2)$, **Algorithm 1**	-
Binary terms satisfy property (i) of Corollary 1	max-flow/min cut algorithm, Sec. 4.1	-

Table 1 Summary of results: subclasses of the general max-norm optimization problem considered here, and algorithms for solving them. When indicating computational complexity, we let $n = |V| + |\mathcal{E}|$. Novel results proposed in this paper are marked in bold.

Appendix: Proof of Lemma 2

In this appendix, we provide a proof of Lemma 2. It is enough to prove that if for some $\kappa \leq \mathsf{n}$ the properties (P0)-(P3) hold for every $k < \kappa$, then they also hold for κ. Clearly, these properties hold immediately after the execution of line 2, that is, for $\kappa = 0$. So, we can assume that $\kappa > 0$. We need to show that (P0)-(P3) are preserved by each operation of the algorithm. More specifically, by the execution of lines 6 or 10, since the status of each of these properties can change only when an atom is removed from A during their execution.

Proof of (P0): Fix an edge $D = \{v, w\}$ and assume that (P0) holds for this D and all $k < \kappa$. Now, if $\mathsf{A}_{\kappa-1}[D]$ has less than 4 elements, then by the inductive assumption it must be already missing either $\{\langle v, 1\rangle, \langle w, 0\rangle\}$ or $\{\langle v, 0\rangle, \langle w, 1\rangle\}$, and so the same will be true for $\mathsf{A}_\kappa[D]$, as needed. So, assume that $\mathsf{A}_{\kappa-1}[D]$ has still all 4 elements. This means, that these 4 elements are present in H and, by (2) and the choice of the ordering of H, the atoms $\{\langle v, 1\rangle, \langle w, 0\rangle\}$ or $\{\langle v, 0\rangle, \langle w, 1\rangle\}$ must precede in H any of the atoms $\{\langle v, 0\rangle, \langle w, 0\rangle\}$ or $\{\langle v, 1\rangle, \langle w, 1\rangle\}$. In particular, if $\kappa = k_j$ for some j, then $\mathsf{A}_\kappa[D]$ is obtained as a result of execution of line 3 and the ordering of H ensures that $\mathsf{A}_\kappa[D]$ still satisfies (P0). So, assume that $\kappa = k_j$ for no j; that is, that $\mathsf{A}_\kappa[D]$ is obtained from $\mathsf{A}_{\kappa-1}[D]$ by the execution of line 10. Since one of the atoms from $\mathsf{A}_{\kappa-1}[D]$ was removed as a result of this execution, for one of vertices of D, say v, the bucket $\mathsf{A}_{\kappa-1}[\{v\}]$ must be missing one of its atoms, say $\{\langle v, i\rangle\}$. But this means that $\mathsf{A}_{\kappa-1}[D]$ must have been missing both $\{\langle v, i\rangle, \langle w, 0\rangle\}$ or $\{\langle v, i\rangle, \langle w, 1\rangle\}$, so indeed $\mathsf{A}_\kappa[D]$ satisfies (P0).

Proof of (P1)-(P3): This will be proved by the simultaneous induction on κ.

(P1) must be preserved by the execution of line 10, by the inductive assumption (P2) that $\mathcal{A}_{\kappa-1}$ is locally consistent. It also cannot be destroyed by the execution of line 6, since this is prevented by the condition of line 5. Thus, $\mathsf{A}_\kappa[D]$ still has the property (P1).

To see (P3) we can assume that $\kappa = k_j$ for some $j > 0$. Clearly (P3) holds for $k = k_{j-1}$. Thus, we need only to show that removal of an atom A in line 6 and consecutive execution of loop 7-12 preserves (P3). Indeed, the potential incompatibility can occur only in relation of the vertices associated with the atoms removed from $\bigcup_{D \in \mathcal{D}} \mathsf{A}[D]$. However, each time such an atom is removed, all adjacent atoms are inserted into the queue K and the execution of the loop 7-12 does not end until all such potential incompatibilities are taken care off.

The proof of the preservation of (P2) is more involved. Let j be the largest such that $k_j \leq \kappa$. First notice that if $\kappa = k_j$, then (P2) holds. Indeed, by the inductive assumptions (P2) and (P3), $\mathcal{A}_{\kappa-1}$ is locally consistent and has no incompatible atoms. Since $\mathcal{A}_\kappa \neq \mathcal{A}_{\kappa-1}$, the bucket $\mathsf{A}[D]$ must have contained two or more atoms prior to the removal of A in line 6. Since $\mathcal{A}_{\kappa-1}$ did not contain any incompatible atoms, $\mathcal{A}_\kappa = \mathcal{A}_{\kappa-1} \setminus \{A\}$ must remain locally consistent. So, we can assume that $\mu := \kappa - k_j$ is non-zero. We will examine families $\mathcal{A}_{k_j}, \mathcal{A}_{k_j+1}, \ldots, \mathcal{A}_{k_j+\mu} = \mathcal{A}_\kappa$.

Let $A = A_0, \ldots, A_\mu$ be the order in which the atoms were removed from K during of this time execution of loop 8-12. Also, let $x_0, \ldots, x_\mu$ be the vertices/edges associated with the atoms $A_0, \ldots, A_\mu$, respectively. We will show, by induction on $\nu \leq \mu$, the following property (I_ν), which in particular imply that $\mathcal{A}_{k_j+\nu}$ is locally consistent.

To state (I_ν) first notice that if an atom for a vertex v is among $x_0, \ldots, x_{\nu-1}$, then $\mathcal{A}_{k_j+\nu}$ must contain precisely one of two atoms $\{\langle v, 0\rangle\}$ and $\{\langle v, 1\rangle\}$. (Must contain at least one, by (P1). It cannot contain both, since this would mean that no v-atom was removed so far and hence $A_{k_j+\nu}$ could not have been removed from $\mathcal{A}_{k_j+\nu-1}$.) In particular, this means that there is an $i_v \in \{0, 1\}$ for which $\mathcal{A}_{k_j+\nu}$ already ensures that the final value of $\ell(v)$ is i_v. This means, that $\mathsf{A}_{k_j+\nu}[\{v\}] = \big\{\{\langle v, i_v\rangle\}\big\}$.

We will prove, by induction on $\nu \leq \mu$, that

(I_ν) $\mathcal{A}_{k_j+\nu}$ is locally consistent and if vertices v and w are among $x_0, \ldots, x_\nu$, then $i_v = i_w$.

Of course, this will finish the proof of (P2).

Clearly, (I_0) holds, as we already shown that $\mathcal{A}_{k_j}$ is locally consistent, and the other condition is satisfied in void. So, fix $\nu \in \{1, \ldots, \mu\}$ such that (I_ξ) holds for all $\xi < \nu$. We will show that (I_ν) holds as well.

For this, assume first that x_ν is an edge $\{v, w\}$. We need to show only that $\mathcal{A}_{k_j+\nu}$ remains locally consistent, the other part of (I_ν) being ensured in this case by $(I_{\nu-1})$. Since $x_\nu = \{v, w\}$, there must exist a $j < \nu$ such that x_j is a vertex and $x_j \in \{v, w\}$. For simplicity we assume that $x_j = v$ and that $i_v = 0$, the other cases being similar.

We need to show that $\mathcal{A}_{k_j+\nu}$, obtained from $\mathcal{A}_{k_j+\nu-1}$ by removing from it the atoms $\{\langle v, 1\rangle, \langle w, 0\rangle\}$ and $\{\langle v, 1\rangle, \langle w, 1\rangle\}$, cannot be locally inconsistent.

Note that such removal from locally consistent set $\mathcal{A}_{k_j+\nu-1}$ can potentially influence local consistency of $\mathcal{A}_{k_j+\nu}$ only of $\{v, w\}$ with respect to the vertices v and w. However, since $\mathsf{A}_{k_j+\nu-1}[\{v\}] = \big\{\{\langle v, 0\rangle\}\big\}$, this is also equal to $\mathsf{A}_{k_j+\nu}[\{v\}]$. Also, both $\mathcal{A}_{k_j+\nu-1}$ and $\mathcal{A}_{k_j+\nu}$ must contain either $\{\langle v, 0\rangle, \langle w, 0\rangle\}$ or $\{\langle v, 0\rangle, \langle w, 1\rangle\}$. So, $\mathcal{A}_{k_j+\nu}$ it cannot have local inconsistency of $\{v, w\}$ with v. Therefore, we must show only that $\mathcal{A}_{k_j+\nu}$ contains no local inconsistency between $\{v, w\}$ and w.

To see this, first notice that there will be no such inconsistency when

$$\mathsf{A}_{k_j-1}[\{w\}] \subsetneq \big\{\{\langle w, 0\rangle\}, \{\langle w, 1\rangle\}\big\}. \tag{9}$$

Indeed, then $\mathsf{A}_{k_j-1}[\{w\}] = \big\{\{\langle w, i\rangle\}$ for some $i \in \{0, 1\}$ and, by the property (P3), $\mathcal{A}_{k_j-1} \supset \mathcal{A}_{k_j+\mu}$ cannot contain atom $\{\langle v, 0\rangle, \langle w, 1-i\rangle\}$. Hence $\mathcal{A}_{k_j+\mu}$ must contain $\{\langle v, 0\rangle, \langle w, i\rangle\}$ and local consistency is preserved.

To finish the argument consider the following three cases.

$\mathsf{A}_{k_j+\nu}[\{w\}] = \big\{\{\langle w, 0\rangle\}, \{\langle w, 1\rangle\}\big\}$: Then $\mathcal{A}_{k_j+\nu}$ is indeed locally consistent, since it contains either $\{\langle v, 0\rangle, \langle w, 0\rangle\}$ or $\{\langle v, 0\rangle, \langle w, 1\rangle\}$.

$\mathsf{A}_{k_j+\nu}[\{w\}] = \big\{\{\langle w, 1\rangle\}\big\}$: Then also $\mathsf{A}_{k_j+\nu-1}[\{w\}] = \big\{\{\langle w, 1\rangle\}\big\}$ and w cannot be among $x_0, \ldots, x_{\nu-1}$, since this would contradict the second part of $(I_{\nu-1})$. In particular, (9) holds and so local consistency is preserved.

$\mathsf{A}_{k_j+\nu}[\{w\}] = \big\{\{\langle w, 0\rangle\}\big\}$: We can assume that (9) does not hold. Then there exists $p \in \{0, \ldots, \nu-1\}$ such that $x_j = w$. Therefore, $\mathcal{A}_{k_j+p} \supset \mathcal{A}_{k_j+\nu}$ cannot contain $\{\langle v, 0\rangle, \langle w, 1\rangle\}$. So, $\mathcal{A}_{k_j+\nu}$ must contain $\{\langle v, 0\rangle, \langle w, 0\rangle\}$ and local consistency is preserved.

Before we proceed further, note that for every $\nu \leq \mu$,

(J_ν) for every vertex v there is at most one edge $D = \{v, w\}$ such that $\mathsf{A}_{k_j+\nu}[\{v\}]$ contains an atom incompatible with all atoms in $\mathsf{A}_{k_j+\nu}[D]$.

Indeed, by (P3), this clearly holds for $\nu = 0$. Also, if x_ν is an edge, than the ordering conditions we imposed on the queue K ensure that the atoms of no other edge can be added to K and subsequently modified, before each vertex (adjacent to x_ν) that can have incompatible atoms with that for x_ν is added to K and subsequently modified, so that the potential incompatibilities are removed.

Finally, consider x_ν being a vertex v. Then we must have had $\mathsf{A}_{k_j+\nu-1}[\{v\}] = \big\{\{\langle v, 0\rangle\}, \{\langle v, 1\rangle\}\big\}$. Moreover, $\mathsf{A}_{k_j+p}[D] \subsetneq \mathsf{A}_{k_j+p-1}[D]$. Also, by (J_ν), such p is unique. Therefore, $\mathcal{A}_{k_j+\nu}$ must be locally consistent, since the only potential local inconsistency in $\mathcal{A}_{k_j+\nu}$ could be between v and $\{v, w\}$. But our choice of $\mathsf{A}_{k_j+\nu}[\{v\}] \subset \mathsf{A}_{k_j+\nu-1}[\{v\}] = \big\{\{\langle v, 0\rangle, \langle v, 1\rangle\}\big\}$ ensures that such inconsistency cannot occur.

Notice also that the second part of (I_ν) holds as well. Indeed, this is satisfied in void when there is no vertex among $x_0, \ldots, x_{\nu-1}$. So, assume that such vertex exists. Then, w, the second vertex of the above chosen edge $x_p = D = \{v, w\}$, must be among such $x_0, \ldots, x_{\nu-1}$. Indeed, if $p = 0$ then we must have $\nu = 2$ and $x_1 = w$. Since $i_w = 0$, we must have $\mathsf{A}_{k_j}[D] \subset \big\{\{\langle v, 0\rangle, \langle w, 0\rangle\}, \{\langle v, 1\rangle, \langle w, 0\rangle\}\big\}$. Also, as $\mathsf{A}_{k_j+2}[\{v\}] \subsetneq \mathsf{A}_{k_j+1}[\{v\}]$, the bucket $\mathsf{A}_{k_j+1}[D] = \mathsf{A}_{k_j}[D]$ must contain precisely only one of the atoms $\{\langle v, 0\rangle, \langle w, 0\rangle\}$ or $\{\langle v, 1\rangle, \langle w, 0\rangle\}$. However, $\mathsf{A}_{k_j}[D]$ cannot be equal to the set $\big\{\{\langle v, 1\rangle, \langle w, 0\rangle\}\big\}$, since, by (P0), this would mean that $\mathsf{A}_{k_j-1}[D] = \big\{\{\langle v, 0\rangle, \langle w, 1\rangle\}, \{\langle v, 1\rangle, \langle w, 1\rangle\}\big\}$. But this contradicts (P3). So, $\mathsf{A}_{k_j+1}[D] = \big\{\{\langle v, 0\rangle, \langle w, 0\rangle\}\big\}$, and indeed $i_v = 0$.

Finally, assume that $p > 0$. Then $w = x_q$ for some $q \in \{0, \ldots, p-1\}$ and so $\mathsf{A}_{k_j+q}[\{w\}] = \big\{\{\langle w, 0\rangle\}\big\}$. Thus, $\mathsf{A}_{k_j+p}[D] \subset \big\{\{\langle v, 0\rangle, \langle w, 0\rangle\}, \{\langle v, 1\rangle, \langle w, 0\rangle\}\big\}$ and $\mathsf{A}_{k_j+\nu-1}[D]$ must contain precisely one of these atoms to ensure that the inclusion $\mathsf{A}_{k_j+\nu}[\{v\}] \subsetneq \mathsf{A}_{k_j+\nu-1}[\{v\}]$ holds. We need to show that the equality $\mathsf{A}_{k_j+p}[D] = \big\{\{\langle v, 1\rangle, \langle w, 0\rangle\}\big\}$ is impossible. Indeed, this would imply that $\mathsf{A}_{k_j+q-1}[D] \subset \big\{\{\langle v, 1\rangle, \langle w, 0\rangle\}, \{\langle v, 0\rangle, \langle w, 1\rangle\}, \{\langle v, 1\rangle, \langle w, 1\rangle\}\big\}$ and using the property (P0), also that $\mathsf{A}_{k_j+q-1}[D] \subset \big\{\{\langle v, 1\rangle, \langle w, 0\rangle\}, \{\langle v, 1\rangle, \langle w, 1\rangle\}\big\}$. However, this means that $\mathcal{A}_{k_j+q-1}$ already decided the value of $\lambda(v)$ as 1. Since the value of $\lambda(w)$ was previously decided, the reasoning as for (J_ν) shows that v should appear already in $x_0, \ldots, x_q$, while $q < \nu$ contradicts this. This finishes the proof of (P1)-(P3). □

References

1. Abbas, A., Swoboda, P.: Bottleneck potentials in markov random fields. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 3175–3184 (2019)
2. Allène, C., Audibert, J.Y., Couprie, M., Cousty, J., Keriven, R., et al.: Some links between min-cuts, optimal

spanning forests and watersheds. Mathematical Morphology and its Applications to Image and Signal Processing pp. 253–264 (2007)
3. Aspvall, B., Plass, M.F., Tarjan, R.E.: A linear-time algorithm for testing the truth of certain quantified boolean formulas. Information Processing Letters **8**(3), 121–123 (1979)
4. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. IEEE Transactions on Pattern Analysis and Machine Intelligence **23**(11), 1222–1239 (2001)
5. Ciesielski, K.C., Udupa, J.K.: Affinity functions in fuzzy connectedness based image segmentation I: Equivalence of affinities. Computer Vision and Image Understanding **114**(1), 146–154 (2010)
6. Ciesielski, K.C., Udupa, J.K., Falcão, A.X., Miranda, P.A.: Fuzzy connectedness image segmentation in graph cut formulation: A linear-time algorithm and a comparative analysis. Journal of Mathematical Imaging and Vision **44**(3), 375–398 (2012)
7. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to algorithms. MIT press (2009)
8. Couprie, C., Grady, L., Najman, L., Talbot, H.: Power watershed: A unifying graph-based optimization framework. IEEE Transactions on Pattern Analysis and Machine Intelligence **33**(7), 1384–1399 (2011)
9. Cousty, J., Bertrand, G., Najman, L., Couprie, M.: Watershed cuts: Minimum spanning forests and the drop of water principle. IEEE Transactions on Pattern Analysis and Machine Intelligence **31**(8), 1362–1374 (2009)
10. Cousty, J., Bertrand, G., Najman, L., Couprie, M.: Watershed cuts: Thinnings, shortest path forests, and topological watersheds. IEEE Transactions on Pattern Analysis and Machine Intelligence **32**(5), 925–939 (2009)
11. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numerische mathematik **1**(1), 269–271 (1959)
12. Jarnìk, V.: O jistém problému minimálím (On a certain problem of minimization). Práce moravské přírodovědecké společnosti **6**(4), 57–63 (1930)
13. Kolmogorov, V., Zabih, R.: What energy functions can be minimized via graph cuts? IEEE Transactions on Pattern Analysis and Machine Intelligence **26**(2), 147–159 (2004)
14. Kruskal, J.B.: On the shortest spanning subtree of a graph and the traveling salesman problem. Proceedings of the American Mathematical society **7**(1), 48–50 (1956)
15. Levi, Z., Zorin, D.: Strict minimizers for geometric optimization. ACM Transactions on Graphics (TOG) **33**(6), 185 (2014)
16. Malmberg, F., Ciesielski, K.C., Strand, R.: Optimization of max-norm objective functions in image processing and computer vision. In: International Conference on Discrete Geometry for Computer Imagery, pp. 206–218. Springer (2019)
17. Malmberg, F., Strand, R.: When can l_p-norm objective functions be minimized via graph cuts? In: International Workshop on Combinatorial Image Analysis. Springer (2018)
18. Najman, L.: Extending the power watershed framework thanks to γ-convergence. SIAM Journal on Imaging Sciences **10**(4), 2275–2292 (2017)
19. Prim, R.C.: Shortest connection networks and some generalizations. The Bell System Technical Journal **36**(6), 1389–1401 (1957)
20. Sinop, A.K., Grady, L.: A seeded image segmentation framework unifying graph cuts and random walker which yields a new algorithm. In: 2007 IEEE 11th International Conference on Computer Vision, pp. 1–8. IEEE (2007)
21. Wolf, S., Bailoni, A., Pape, C., Rahaman, N., Kreshuk, A., Köthe, U., Hamprecht, F.A.: The mutex watershed and its objective: Efficient, parameter-free image partitioning. arXiv preprint arXiv:1904.12654 (2019)
22. Wolf, S., Pape, C., Bailoni, A., Rahaman, N., Kreshuk, A., Kothe, U., Hamprecht, F.: The mutex watershed: efficient, parameter-free image partitioning. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 546–562 (2018)
23. Wolf, S., Schott, L., Kothe, U., Hamprecht, F.: Learned watershed: End-to-end learning of seeded segmentation. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2011–2019 (2017)