Optimization of Max-Norm Objective Functions in Image Processing and Computer Vision

Filip Malmberg[1](✉), Krzysztof Chris Ciesielski[2,3], and Robin Strand[1]

[1] Centre for Image Analysis, Department of Information Technology, Uppsala University, Uppsala, Sweden
{filip.malmberg,robin.strand}@it.uu.se

[2] Department of Mathematics, West Virginia University, Morgantown, WV 26506-6310, USA
Krzysztof.Ciesielski@mail.wvu.edu

[3] Department of Radiology, MIPG, University of Pennsylvania, Philadelphia, PA 19104, USA

Abstract. 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.

We consider binary labeling problems where the objective function is defined as the max-norm over a set of variables. It is well known that for a limited subclass of such problems, globally optimal solutions can be found via watershed cuts, i.e., cuts by optimum spanning forests. Here, we propose a new algorithm for optimizing a broader class of such problems. We prove that the proposed algorithm returns a globally optimal labeling, provided that the objective function satisfies certain given conditions, analogous to the submodularity conditions encountered in min-cut/max-flow optimization. The proposed method is highly efficient, with quasi-linear computational complexity.

Keywords: Energy minimization · Pixel labeling · Minimum cut · Submodularity

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 this paper, we seek binary label assignments $\ell \colon V \to \{0,1\}$ that minimizes a given objective (energy) function E_∞ of the form

$$E_\infty(\ell) := \max\bigl\{\max_{s\in V} \phi_s(\ell(s)), \max_{\{s,t\}\in\mathcal{E}} \phi_{st}(\ell(s),\ell(t))\bigr\}. \tag{1}$$

In this equation, the image domain is identified with an undirected[1] graph $\mathcal{G} = (V, \mathcal{E})$, where the set V of vertices of the graph is the set of all pixels of the image, while $\mathcal{E}$ is the set of all its edges, that is, of pairs $\{s,t\}$ of vertices/pixels that are *adjacent* according to some given adjacency relation.

At first glance, the restriction to binary labeling may appear very limiting. Many successful methods for multi-label optimization, however, rely on iteratively minimizing binary labeling problems via *move-making* strategies [1]. Thus, the ability to find optimal solutions for problems with two labels has high relevance also for the multi-label case.

The functions $\phi_s(\cdot)$ are referred to as *unary* terms. Each unary term depends only on the label $\ell(s)$ assigned to the pixel s, and they are generally used to indicate the preference of an individual pixel to be assigned each particular label, typically based on some prior information.

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

Our main contribution is an algorithm for solving labeling problems of the form described above. Specifically, the algorithm is guaranteed to produce a labeling that is globally optimal with respect to the energy function E_∞, under the condition that all pairwise terms ϕ_{st} satisfy the condition

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

The proposed algorithm is very efficient, with an asymptotic time complexity bound by the time required to sort $\mathcal{O}(|V| + |\mathcal{E}|)$ values.[2]

1.1 Background and Related Work

In their seminal work, Kolmogorov and Zabih [5] considered binary labeling problems where the objective function has the form

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

[1] The energy formula (1) must be expressed in terms of undirected edges. But the algorithm can be used for directed graphs as well.

[2] Here, $|\cdot|$ denotes set cardinality.

and showed that a globally optimal solution can be computed by solving a max-flow/min-cut problem on a suitably constructed graph under the condition that all pairwise terms ϕ_{st} are *submodular*. A pairwise term ϕ_{st} is said to be submodular if

$$\phi_{st}(0,0) + \phi_{st}(1,1) \leq \phi_{st}(0,1) + \phi_{st}(1,0). \tag{4}$$

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

– A *local* error measure, in our case defined by the unary and pairwise terms.
– A *global* error measure, aggregating the local errors into a final score.

In the case of E_1, the global error measure is obtained by summing all the local error measures, and in the case of E_∞ the global error measure is taken to be the maximum of the local error measures. If we assume for a moment that all the 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 errors. Similarly, E_∞ can be interpreted as measuring the L_∞ or max norm of the vector. The L_1 and L_∞ norms are both special cases of L_p norms, and in this sense we can view both E_1 and E_∞ as special cases of a more general objective function

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

where $\phi_s^p(\cdot) = (\phi_s(\cdot))^p$ and $\phi_{st}^p(\cdot,\cdot) = (\phi_{st}(\cdot,\cdot))^p$. The value p 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.

Labeling problems with objective functions of the form E_p can be solved using minimal graph cuts, provided that all pairwise terms ϕ_{st}^p are submodular [6]. As shown by Malmberg and Strand [6], the submodularity of ϕ_{st}^p is guaranteed for any $p \geq 1$ if ϕ_{st} is submodular and satisfies (2).

Additionally, it turns out that in some problem instances the limit case E_∞ can be optimized for directly, using efficient greedy algorithms. For example, consider a labeling problem with objective function E_∞, where all pairwise terms satisfy the restriction $\phi_{st}(1,0) = \phi_{st}(0,1)$ and $\phi_{st}(0,0) = \phi_{st}(1,1) = 0$. In this simplified case, a globally optimal solution can be found by computing a cut by optimum spanning forest (MSF cut, or watershed cut) on a suitably constructed graph [2–4].This interesting result has a high practical value, since the computation time for finding an MSF cut is substantially lower than the

[3] This vector is identified with the function ϕ_ℓ defined in the next section.

computation time for solving a min-cut/max-flow problem, asymptotically as well as in practice [3]. An interesting question is therefore whether it is possible to use similar greedy techniques to optimize the objective function E_∞, beyond the special case outlined above. The results presented in this paper answers this question affirmatively.

We observe that our proposed algorithm has some structural similarity to Kruskal's algorithm for computing minimum spanning trees, and in this sense the algorithm can be seen as a generalization of the MSF/Watershed cut approach [4].

2 Preliminaries

The exposition of our proposed algorithm 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 represent 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.

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 $\{(s, i)\}$, with $i \in \{0, 1\}$. Similarly, each binary atom associated with an edge $\{s, t\} \in \mathcal{E}$ has the form $\{(s, i), (t, j)\}$, 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

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

For a given labeling ℓ, we define $\phi_\ell \colon \mathcal{D} \to [0, \infty)$, for every $D \in \mathcal{D}$, as

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

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

$$E_\infty(\ell) = \|\phi_\ell\|_\infty = \max_{D \in \mathcal{D}} \phi_\ell(D). \tag{6}$$

2.1 Global and Local Consistency, Incompatible Atoms

Conceptually, the proposed algorithm works 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: $\ell = \bigcup \mathcal{A}(\ell)$.

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

In general, determining whether a given set of atoms is globally consistent is difficult. Therefore we also introduce a seemingly weaker property of local consistency, which will be used in Sect. 4 to establish the correctness of our 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 $\{(s, i)\}$ and $\{(s, i), (t, j)\}$ both belong to $\mathcal{A}'$.

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 $\mathcal{A}'$, we say that an atom $A \in \mathcal{A}'$ is *incompatible* (w.r.t. $\mathcal{A}'$) if either

1. A is a unary atom so that $A = \{(v, i)\}$ for some vertex v, and there exists some edge $\{v, w\}$ adjacent to v such that $\mathcal{A}'$ contains neither $\{(v, i), (w, 0)\}$ nor $\{(v, i), (w, 1)\}$; or
2. A is a binary atom so that $A = \{(v, i), (w, j)\}$ for some edge $\{v, w\}$, and at least one of $\{(v, i)\}$ and $\{(w, j)\}$ is not in $\mathcal{A}'$.

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

3 Proposed Algorithm

In this section, we introduce the proposed algorithm 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 satisfy (2). Informally, the general outline of the proposed algorithm is as follows:

- Start with a set S consisting of all possible atoms.
- 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 incompatible atoms until S contains no more incompatible atoms.

Before we formalize this algorithm, we introduce a specific preordering relation $\succ$ on the atoms $\mathcal{A}$. For $A_0, A_1 \in \mathcal{A}$ we will write $A_0 \succ A_1$ 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 $\{(s, i), (t, i)\}$ (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 1.

Algorithm 1. Labeling Algorithm

```
Data: A graph $\mathcal{G} = (V, \mathcal{E})$ and associated $\Phi\colon \mathcal{A} \to [0, \infty)$ generating energy $E_\infty$
Result: A labeling $\ell\colon V \to \{0, 1\}$ minimizing energy $E_\infty$
Additional Structure: An array $\mathsf{A}$ of buckets of atoms, indexed by
$\mathcal{D} = \mathcal{V} \cup \mathcal{E}$; a list $\mathsf{H}$ of atoms; a queue $\mathsf{K}$ of vertices/edges such that every vertex
in $\mathsf{K}$ precedes any edge.
1  foreach vertex/edge $D \in \mathcal{D}$ do insert all $D$-atoms to $\mathsf{A}[D]$
2  create a list $\mathsf{H}$ of all atoms $\mathcal{A}$ such that $A_0$ precedes $A_1$ in $\mathcal{A}$ whenever $A_0 \succ A_1$
3  while $\mathsf{H} \neq \emptyset$ do
4      remove the first atom $A$ from $\mathsf{H}$
5      if $D \in \mathcal{D}$ is a vertex/edge of $A$ and $\mathsf{A}[D]$ has more than one element then
6          remove $A$ from $\mathsf{A}[D]$ and insert $D$ to (previously empty) $\mathsf{K}$
7          while $\mathsf{K} \neq \emptyset$ do
8              remove a vertex/edge $C$ from $\mathsf{K}$
9              foreach edge/vertex $D$ adjacent to $C$ do
10                 remove from $\mathsf{A}[D]$ and $\mathsf{H}$ all $A$ incompatible with $\bigcup_{D' \in \mathcal{D}} \mathsf{A}[D']$
11                 if any atom was removed from $\mathsf{A}[D]$ and $\mathsf{H}$ in line 10 then
12                     insert to $\mathsf{K}$ 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 $\ell = \bigcup_{D \in \mathcal{D}} \mathsf{A}[D]$
```

4 Analysis of the Algorithm

In this section, we analyze the computational complexity of the proposed algorithm and prove that it is guaranteed to return a globally optimal solution to the labeling problem given in the introduction.

4.1 Computational Complexity

We now analyze the asymptotic computational complexity of Algorithm 1. First, let $\eta := |A| = 2|V| + 4|\mathcal{E}|$. 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 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 bound by the time required to sort $\mathcal{O}(\eta)$ elements, i.e., at most $\mathcal{O}(\eta \log \eta)$.

4.2 Example Demonstrating the Need for Condition (2)

In Sect. 4.3 we will prove that if all binary terms satisfy (2), then this is a sufficient condition for the proposed algorithm to return an optimal labeling. In this section, we first give an example showing that when this condition is violated, the algorithm may indeed fail to produce a labeling.

Let $\mathcal{G}$ be a complete graph with three vertices $V = \{a, b, c\}$. Define ϕ for every $i \in \{0, 1\}$ via:

(i) $\phi(\langle a, i\rangle, \langle b, i\rangle) = \phi(\langle a, i\rangle, \langle c, i\rangle) = 1$;
(ii) $\phi(\langle a, i\rangle, \langle b, 1-i\rangle) = \phi(\langle a, i\rangle, \langle c, 1-i\rangle) = 9$;
(iii) $\phi(\langle b, i\rangle, \langle c, i\rangle) = 5$;
(iv) $\phi(\langle b, i\rangle, \langle c, 1-i\rangle) = 2$;
(v) $\phi(\langle v, i\rangle) = 0$ for all $v \in V$.

Consider the following possible steps in the execution of Algorithm 1.

Steps 1–4: Remove 4 atoms with $\phi(\langle a, i\rangle, \langle b, 1-i\rangle) = \phi(\langle a, i\rangle, \langle c, 1-i\rangle) = 9$.
Steps 5–6: Remove 2 atoms with $\phi(\langle b, 0\rangle, \langle c, 0\rangle) = \phi(\langle b, 1\rangle, \langle c, 1\rangle) = 5$.

Then, after these 6 steps, the system is inconsistent, since pairs $\langle a, b\rangle$ and $\langle a, c\rangle$ must have the same labels, while $\langle b, c\rangle$ must have different labels. So, this execution of Algorithm 1 fails to produce a valid labeling.

To motivate the introduction of the specific preordering relation $\succ$ for H, consider modifying the labeling problem given above so that

(iv) $\phi(\langle b, i\rangle, \langle c, 1-i\rangle) = 5$.

If we only require H to be ordered by decreasing Φ, steps 1–6 still represent a possible execution of Algorithm 1. By instead requiring H to be ordered according to $\succ$, we force the atoms $\{(b, 0), (c, 1)\}$ and $\{(b, 1), (c, 0)\}$ to be removed from H before $\{(b, 0), (c, 0)\}$ and $\{(b, 1), (c, 1)\}$, and the algorithm will in this case return a valid labeling.

4.3 Proof of Correctness

Theorem 1. *If all binary terms of the map $\Phi\colon \mathcal{D} \to [0,\infty)$ associated with graph $\mathcal{G} = (V, \mathcal{E})$ satisfy the condition (2), then ℓ returned by Algorithm 1 is indeed a labeling of V minimizing the objective function E_∞.*

Proof. 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 of 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 key element of this proof is to show, by induction, that the following properties hold for every $k \leq \mathsf{n}$.

(P0) For every edge $D = \{v, w\}$, if $\mathsf{A}_k[D]$ is missing either $\{(v,0),(w,0)\}$ or $\{(v,1),(w,1)\}$, then it must be also missing $\{(v,1),(w,0)\}$ or $\{(v,0),(w,1)\}$.
(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.

It is enough to prove that if for some $\kappa \leq \mathsf{n}$ these properties 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 $\{(v,1),(w,0)\}$ or $\{(v,0),(w,1)\}$, 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 still present in H and, by (2) and the choice of the ordering of H, the atoms $\{(v,1),(w,0)\}$ and $\{(v,0),(w,1)\}$ must precede in H any of the atoms $\{(v,0),(w,0)\}$ and $\{(v,1),(w,1)\}$. 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 $\{(v,i)\}$. But this means that $\mathsf{A}_{\kappa-1}[D]$ must have been missing both $\{(v,i),(w,0)\}$ and $\{(v,i),(w,1)\}$, 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 $k_j \le \kappa$. 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 $\nu \le \mu$, 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 $\{(v,0)\}$ and $\{(v,1)\}$. (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 $\mathsf{A}_{k_j+\nu}[\{v\}] = \big\{\{(v,i_v)\}\big\}$.

We will prove, by induction on $\nu \le \mu$, 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 atoms $\{(v,1),(w,0)\}$ and $\{(v,1),(w,1)\}$, cannot be locally inconsistent.

Note that such removal from locally consistent $\mathcal{A}_{k_j+\nu-1}$ 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 $\mathsf{A}_{k_j+\nu-1}[\{v\}] = \big\{\{(v,0)\}\big\}$, 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 $\{(v,0),(w,0)\}$ or $\{(v,0),(w,1)\}$. Hence, $\mathcal{A}_{k_j+\nu}$ 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

$$\mathsf{A}_{k_j-1}[\{w\}] \subsetneq \big\{\{(w,0)\},\{(w,1)\}\big\}. \tag{7}$$

Indeed, then $\mathsf{A}_{k_j-1}[\{w\}] = \big\{\{(w,i)\}\big\}$ 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 $\{(v,0),(w,1-i)\}$. Hence $\mathcal{A}_{k_j+\mu}$ must contain $\{(v,0),(w,i)\}$ and local consistency is preserved.

To finish the argument consider the following three cases.

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

$\mathsf{A}_{k_j+\nu}[\{w\}] = \big\{\{(w,1)\}\big\}$: Then also $\mathsf{A}_{k_j+\nu-1}[\{w\}] = \big\{\{(w,1)\}\big\}$ and w cannot be among $x_0, \ldots, x_{\nu-1}$, since this would contradict the second part of $(I_{\nu-1})$. In particular, (7) holds and so local consistency is preserved.

$\mathsf{A}_{k_j+\nu}[\{w\}] = \big\{\{(w,0)\}\big\}$: We can assume that (7) 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 $\{(v,0),(w,1)\}$. So, $\mathcal{A}_{k_j+\nu}$ must contain $\{(v,0),(w,0)\}$ and local consistency is preserved.

Before we proceed further, notice that for every $\nu \le \mu$,

(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 the case when x_ν is a vertex v. Then we must have had $\mathsf{A}_{k_j+\nu-1}[\{v\}] = \big\{\{(v,0)\},\{(v,1)\}\big\}$. Also, there exists a $p \in \{0, \ldots, \nu-1\}$ such that x_p is an edge $D = \{v,w\}$ and $\mathsf{A}_{k_j+p}[D] \subsetneq \mathsf{A}_{k_j+p-1}[D]$. Moreover, 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 $\mathsf{A}_{k_j+\nu}[\{v\}] \subset \mathsf{A}_{k_j+\nu-1}[\{v\}] = \big\{\{(v,0)\},\{(v,1)\}\big\}$ 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 $x_0, \dots, x_{\nu-1}$. 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 $x_0, \dots, x_{\nu-1}$. Indeed, if $p = 0$ then we must have $\nu = 2$ and $x_1 = w$. Since $i_w = 0$, we must have $\mathsf{A}_{k_j}[D] \subset \big\{\{(v,0),(w,0)\},\{(v,1),(w,0)\}\big\}$. Also, since $\mathsf{A}_{k_j+2}[\{v\}] \subsetneq \mathsf{A}_{k_j+1}[\{v\}]$, the bucket $\mathsf{A}_{k_j+1}[D] = \mathsf{A}_{k_j}[D]$ must contain precisely only one of the atoms $\{(v,0),(w,0)\}$ or $\{(v,1),(w,0)\}$. However, $\mathsf{A}_{k_j}[D]$ cannot be equal to $\big\{\{(v,1),(w,0)\}\big\}$, since, by (P0), this would mean that $\mathsf{A}_{k_j-1}[D] = \big\{\{(v,0),(w,1)\},\{(v,1),(w,1)\}\big\}$. But this contradicts (P3). So, $\mathsf{A}_{k_j+1}[D] = \big\{\{(v,0),(w,0)\}\big\}$, and indeed $i_v = 0$.

Finally, assume that $p > 0$. Then $w = x_q$ for some $q \in \{0, \dots, p-1\}$ and so $\mathsf{A}_{k_j+q}[\{w\}] = \big\{\{(w,0)\}\big\}$. Thus, $\mathsf{A}_{k_j+p}[D] \subset \big\{\{(v,0),(w,0)\},\{(v,1),(w,0)\}\big\}$ 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 $\mathsf{A}_{k_j+p}[D] = \big\{\{(v,1),(w,0)\}\big\}$ is impossible. Indeed, this would imply that $\mathsf{A}_{k_j+q-1}[D] \subset \big\{\{(v,1),(w,0)\},\{(v,0),(w,1)\},\{(v,1),(w,1)\}\big\}$ and using property (P0), that $\mathsf{A}_{k_j+q-1}[D] \subset \big\{\{(v,1),(w,0)\},\{(v,1),(w,1)\}\big\}$. But 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 $x_0, \dots, x_q$, while $q < \nu$ contradicts this. This finishes the proof of (P1)-(P3).

To finish the proof of Theorem 1, 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 $\{0,1\}$.

Finally we prove 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

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

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

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

once again ensuring optimality of ℓ.

5 Conclusions

We have presented an efficient algorithm for finding optimal solutions of binary labeling problem with objective functions of the form E_∞, related to the L_∞ norm. We have showed that the algorithm is guaranteed to find a globally optimal labeling, under the condition that all binary terms satisfy the condition (2).

As we observed in Sect. 1.1, there is an interesting similarity between our results and the work by Boykov et al. [1] and Kolmogorov and Zabih [5] on optimizing binary labeling problem via minimal graph cuts. Specifically, the condition (2) strongly resembles the submodularity condition required for the minimal graph cut approach to be applicable. We note that the condition (2) also appeared in a different context in the recent work by Malmberg and Strand [6], regarding minimization of L_p norm objective functions by minimal graph cuts. This connection should be investigated further to determine if the similarity is only superficial, or the result of a deeper connection between these problems.

As discussed in Sect. 1.1, special cases of the max norm labeling problems considered here can be solved by computing a MSF/watershed cut on a suitably constructed graph. The set of problems that are solvable by this approach appears to be a strict subset of the problems solvable by our proposed algorithm. We note, however, that the extensions to the MSF-cut concept proposed by Malmberg et al. [7] and Wolf et al. [8] can be used to solve a subclass of max norm optimization problems where the binary terms do not satisfy (2). Thus, an interesting direction for future work is to determine the precise class of max-norm problems that can be solved via efficient greedy algorithms.

References

1. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. IEEE Trans. Pattern Anal. Mach. Intell. **23**(11), 1222–1239 (2001)
2. 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. J. Math. Imaging Vis. **44**(3), 375–398 (2012)
3. Couprie, C., Grady, L., Najman, L., Talbot, H.: Power watershed: a unifying graph-based optimization framework. IEEE Trans. Pattern Anal. Mach. Intell. **33**(7), 1384–1399 (2011)
4. Cousty, J., Bertrand, G., Najman, L., Couprie, M.: Watershed cuts: minimum spanning forests and the drop of water principle. IEEE Trans. Pattern Anal. Mach. Intell. **31**(8), 1362–1374 (2009)
5. Kolmogorov, V., Zabih, R.: What energy functions can be minimized via graph cuts? IEEE Trans. Pattern Anal. Mach. Intell. **26**(2), 147–159 (2004)
6. Malmberg, F., Strand, R.: When can l_p-norm objective functions be minimized via graph cuts? In: Barneva, R.P., Brimkov, V.E., Tavares, J.M.R.S. (eds.) IWCIA 2018. LNCS, vol. 11255, pp. 112–117. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-05288-1_9

7. Malmberg, F., Strand, R., Nyström, I.: Generalized hard constraints for graph segmentation. In: Heyden, A., Kahl, F. (eds.) SCIA 2011. LNCS, vol. 6688, pp. 36–47. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21227-7_4
8. Wolf, S., Pape, C., Bailoni, A., Rahaman, N., Kreshuk, A., Köthe, U., Hamprecht, F.A.: The mutex watershed: efficient, parameter-free image partitioning. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11208, pp. 571–587. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01225-0_34