Generalized Hard Constraints for Graph Segmentation

Filip Malmberg, Robin Strand, and Ingela Nyström

Centre for Image Analysis, Uppsala University,
Uppsala, Sweden
{filip,robin,ingela}@cb.uu.se

Abstract. Graph-based methods have become well-established tools for image segmentation. Viewing the image as a weighted graph, these methods seek to extract a graph cut that best matches the image content. Many of these methods are *interactive*, in that they allow a human operator to guide the segmentation process by specifying a set of hard constraints that the cut must satisfy. Typically, these constraints are given in one of two forms: *regional constraints* (a set of vertices that must be separated by the cut) or *boundary constraints* (a set of edges that must be included in the cut). Here, we propose a new type of hard constraints, that includes both regional constraints and boundary constraints as special cases. We also present an efficient method for computing cuts that satisfy a set of generalized constraints, while globally minimizing a graph cut measure.

Keywords: Image segmentation, Graph cuts, Regional constraints, Boundary constraints.

1 Introduction

In recent years, several efficient methods for image segmentation have been formulated in the framework of *edge weighted graphs*. Common for these methods is that they seek to extract a *cut* from a *pixel adjacency graph*, i.e., a graph whose vertex set is the set of image elements, and whose edge set is given by an adjacency relation between the image elements. Informally, a cut in a connected graph is a set of edges such that if they are removed, the graph is separated into two or more connected components.

Generally, the goal of these methods is to find a cut that best matches some criterion based on image content, e.g., a cut that coincides with high contrast regions in the image. In interactive (or supervised) methods, the cut is additionally required to satisfy a set of *hard constraints*. These constraints may be specified by a human user in an interactive setting, or by an automated procedure. Typically, the constraints are given in one of two forms.

Regional constraints. The cut is required to separate all elements in a specified subset of the graph vertices.

A. Heyden and F. Kahl (Eds.): SCIA 2011, LNCS 6688, pp. 36–47, 2011.

Boundary constraints. The cut is required to include a specified subset of the graph edges.

Computing cuts with respect to regional constraints is a well-studied problem, and many efficient algorithms have been proposed for this purpose. See, e.g., [2, 15,6,9,4]. A unified theoretical framework, incorporating many of these methods, was recently proposed by Couprie et al. [3].

The most prominent example of image segmentation with respect to boundary constraints is the Live-wire method [7]. In its original form, this method is restricted to 2D image segmentation. Many attempts have been made to extend this paradigm to 3D, see, e.g., [12] and references therein. In general, these methods, unlike the 2D Live-wire method, do not guarantee optimality of the resulting segmentation. A notable exception is [10], where a method is presented for computing globally minimal discrete surfaces with prescribed boundary.

Here, we propose a new type of hard constraints for supervised graph segmentation. Informally, a generalized constraint is a pair of distinct vertices that any feasible cut must separate. We show that both boundary constraints and regional constraints may be viewed as special cases of the proposed generalized constraints. Moreover, we present an efficient method for finding cuts that satisfy a set of generalized constraints, and for which the maximal edge weight in the cut is globally minimal.

2 Preliminaries

In this Section, we present basic definitions of edge weighted graphs, graph cuts and vertex labelings. Moreover, we introduce the notion of graph cut segments.

2.1 Edge Weighted Graphs

We define a (undirected) *graph* as a pair $G = (V(G), E(G))$ where $V(G)$ is a set and $E(G)$ is composed of unordered pairs of distinct elements in V, i.e., E is a subset of $\{\{v, w\} \subseteq V \mid v \neq w\}$. The elements of V are called *vertices* of B, and the elements of E are called *edges* of G. In order to simplify the notation, the vertices and edges of a graph will be denoted V and E instead of $V(G)$ and $E(G)$ whenever it is clear from the context which graph they belong to. An edge spanning two vertices v and w is denoted $e_{v,w}$. If $e_{v,w}$ is an edge in E, the vertices v and w are *adjacent*. For the remainder of this paper, G denotes a graph (V, E), such that $|V|$ is finite.

We assign to each edge $e \in E$ a non-negative real value $W(e)$, called a *weight*. The weight represents vertex affinity, i.e., two adjacent vertices are closely related if the weight of the edge connecting them is high. A common task in image segmentation applications is to segment an image into regions of homogeneous intensity. To this end, we may define edge weights as, e.g.,

$$W(e_{v,w}) = I_{max} - |I(v) - I(w)| \,, \tag{1}$$

where $I(v)$ is the intensity of the image element corresponding to the vertex v, and I_{max} is the maximum intensity value present in the image. The method proposed here is *contrast invariant*, i.e., applying a strictly monotonic transformation to the edge weights does not change the output of the method.

2.2 Graph Partitioning

A partitioning of a graph is commonly represented either as a *vertex labeling* or as a *graph cut*. The two representations are closely related, and the choice of one representation over the other is largely a matter of preference. Here, we use the graph cut representation in Sections 3 and 4 to derive our theoretical results. In Section 5, we switch to the vertex labeling representation, in order to formulate a practical algorithm for the proposed method. In this Section, we provide formal definitions of both representations, and clarify the relation between them.

A *path* in G is an ordered sequence of vertices $\pi = \langle v_1, v_2, \ldots, v_k \rangle$ such that $e_{v_i,v_{i+1}} \in E$ for all $i \in [1, k-1]$. Two vertices v and w are *linked* in G if there exists a path in G that starts at v and ends at w. The notation $v \underset{G}{\sim} w$ will here be used to indicate that v and w are linked on G. If all pairs of vertices in a graph are linked, then the graph is *connected*, otherwise it is *disconnected*. For the remainder of this paper, we assume that the graph G is connected.

If G and H are graphs such that $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$, then H is a *subgraph* of G. If H is a connected subgraph of G and $v \underset{G}{\not\sim} w$ for all vertices $v \in H$ and $w \notin H$, then H is a *connected component* of G.

Definition 1. *Let $S \subseteq E$, and $G' = (V, E \setminus S)$. If, for all $e_{v,w} \in S$, it holds that $v \underset{G'}{\not\sim} w$, then S is a* (graph) cut *on G.*

If S is a non-empty cut on G, then the removal of S from G separates G into two or more connected components. Note that E is a cut on G.

Definition 2. *A* (vertex) labeling $\mathcal{L}$ *of G is a map $\mathcal{L} : V \to L$, where L is an arbitrary set of* labels.

In the following, we assume that $|L| \geq |V|$. The *boundary*, $\partial\mathcal{L}$, of a vertex labeling $\mathcal{L}$ is defined as the edge set $\partial\mathcal{L} = \{e_{v,w} \in E \mid \mathcal{L}(v) \neq \mathcal{L}(w)\}$. The relation between labelings and cuts is summarized in Theorem 1.

Theorem 1. *For any graph $G = (V, E)$ and set of edges $S \subseteq E$, the following statements are equivalent:*

1. *There exists a labeling $\mathcal{L}$ of G such that $S = \partial\mathcal{L}$.*
2. *S is a cut on G.*

A proof of Theorem 1 can be found in [11]. Next, we introduce the concept of *graph cut segments*, which is central to the development of the proposed method in Section 4.

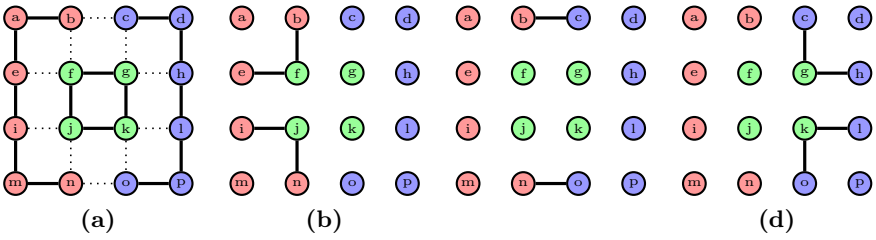

Fig. 1. Graph cut segments. (a) A graph cut, shown as dotted lines. The cut separates the graph into three connected components, shown in red, green, and blue. (b-d) The segments of the cut.

Definition 3. *Let S be a cut on G, let $e \in S$, and let $G' = (V, E \setminus (S \setminus \{e\}))$. The* segment S_e *of S corresponding to e is defined as*

$$S_e = \{e_{v,w} \mid e_{v,w} \in S,\ v \underset{G'}{\sim} w\}\ . \tag{2}$$

Any cut $S \neq \emptyset$ consists of one or more segments. From Definition 3, it follows that $e_{v,w} \in S_{e_{x,y}} \Leftrightarrow e_{x,y} \in S_{e_{v,w}}$, and thus a segment can be uniquely identified by any of its constituent edges. Figure 1 illustrates the concept of graph cut segments.

3 Constrained Graph Cuts

In this Section, we introduce the proposed generalized hard constraints, hereinafter referred to as *constraints*.

Definition 4. *A* constraint *on G is an unordered pair of distinct elements in V, i.e., a constraint is an element in the set $\{\{v, w\} \subseteq V \mid v \neq w\}$.*

Note that while the definition of constraints is identical to the definition of graph edges in Section 2.1, we do not generally require a constraint to be an element of E. To differentiate between constraints and edges, a pair of vertices $\{v, w\}$ that represents a constraint is denoted $c_{v,w}$.

Definition 5. *Let $S \subseteq E$ and let C be a set of constraints. We say that S* satisfies *C if*

$$\forall c_{v,w} \in C, v \underset{(V,E\setminus S)}{\not\sim} w \tag{3}$$

and

$$\forall e \in S, \exists c_{v,w} \in C \text{ such that } v \underset{(V,E\setminus(S\setminus\{e\}))}{\sim} w\ . \tag{4}$$

If $S \subseteq E$ does not satisfy (3) then S is an *under-segmentation* with respect to C. If S does not satisfy (4), then S is an *over-segmentation* with respect to C. In

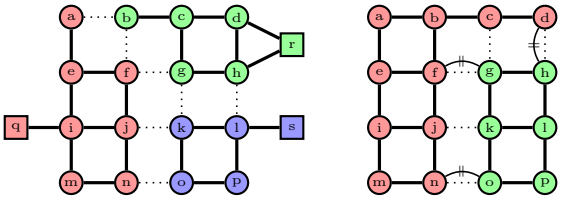

Fig. 2. Graph cuts with respect to regional and boundary constraints. (Left) A cut that satisfies the regional constraints $\{q, r, s\}$. The cut (dotted lines) separates the graph into three connected components. (Right) A cut that satisfies the boundary constraints $\{e_{d,h}, e_{f,g}, e_{n,o}\}$. The cut separates the graph into two connected components.

other words, a set of edges S satisfies the constraints C if it is neither an over- nor an under-segmentation with respect to C.[1]

Theorem 2. *Let C be a set of constraints and let $S \subseteq E$ such that S satisfies C. Then S is a cut on G.*

Proof. Let $G' = (V, E \setminus S)$. If S satisfies C, then for each edge $e_{v,w} \in S$ there exists a constraint $c_{x,y} \in C$ such that either $x \underset{G'}{\sim} v$ and $y \underset{G'}{\sim} w$ or $x \underset{G'}{\sim} w$ and $y \underset{G'}{\sim} v$. From (3), it thus follows that $v \underset{G'}{\not\sim} w$. □

In the remainder of this Section, we show that the proposed definition of constraints includes both boundary constraints and regional constraints as special cases.

3.1 Regional Constraints

A *regional constraint* on G is a vertex in V. Informally, a cut S satisfies the set of regional constraints $C_r \subseteq V$ if each connected component of $(V, E \setminus S)$ contains exactly one element in C_r. This is formalized in Definition 6.

Definition 6. *Let $C_r \subseteq V$ be a set of regional constraints and let $S \subseteq E$. We say that S* satisfies *C_r if*

$$\forall v, w \in C_r, v \underset{(V,E\setminus S)}{\not\sim} w \tag{5}$$

and

$$\forall e \in S, \exists v, w \in C_r \text{ such that } v \underset{(V,E\setminus(S\setminus\{e\}))}{\sim} w \ . \tag{6}$$

When regional constraints are used in image processing applications, the pixel adjacency graph is usually augmented with a set of *terminal vertices* that

[1] A similar definition of over- and under-segmentation was proposed by Felzenszwalb and Huttenlocher [8], in the context of unsupervised segmentation.

constitute the regional constraints [2]. Each terminal vertex represents an object category (e.g., object or background), and a vertex that is adjacent to a terminal vertex is called a *seed-point*. In this way, an object may be segmented from an image by specifying one or more seed-points corresponding to that object. See Figure 2.

From Definitions 5 and 6, it becomes clear that regional constraints are a special case of the generalized hard constraints proposed here. For any set of regional constraints C_r there exists a set of constraints C, namely $C = \{c_{v,w} \mid v, w \in C_r\}$, such that $S \subseteq E$ satisfies C if and only if it satisfies C_r.

3.2 Boundary Constraints

A boundary constraint on G is an edge in E. Intuitively, a cut S satisfies the set of boundary constraints $C_b \subseteq E$ if $C_b \subseteq S$. This definition, however, is clearly not sufficient, since the cut $S = E$ satisfies this condition for any C_b. Instead, we propose the following definition.

Definition 7. *Let $C_b \subseteq E$ be a set of boundary constraints and let S be a cut on G. If $C_b \subseteq S$ and each segment of S contains at least one element of C_b, then we say that S* satisfies *C_b.*

From Theorem 3 and its Corollary below, it follows that Definition 7 coincides with Definition 5 in the special case that $C \subseteq E$. In this sense, boundary constraints are a special case of the proposed generalized constraints. See Figure 2.

Theorem 3. *Let $C \subseteq E$ be a set of constraints and let, $S \subseteq E$ such that S satisfies C. Then $C \subseteq S$.*

Proof. Assume to the contrary that S is a cut with respect to $C \subseteq E$ and $C \not\subseteq S$. Then there exists an edge $e_{v,w} \in C \setminus S$, and thus $v \underset{G'}{\sim} w$, where $G' = (V, E \setminus S)$. This contradicts the assumption that S is a cut with respect to C. □

Corollary 1. *Let $C \subseteq E$ be a set of constraints, let $S \subseteq E$ such that S satisfies C. For all $e \in S$, there exists a constraint $c \in C$ such that $c \in S_e$.*

Proof. Let $e_{v,w} \in S$. Since S satisfies C, there exists a constraint $c_{x,y} \in C$ such that $x \underset{(V, E \setminus (S \setminus \{e_{v,w}\}))}{\sim} y$. By Theorem 3 it holds that $c_{x,y} \in S$. Therefore $c_{x,y} \in S_e$. □

4 Strategies for Computing Constrained Graph Cuts

In this Section, we consider the problem of computing a cut that satisfies a set of constraints. We start by defining a general strategy for computing such cuts. Based on this general strategy, we show that for any set of constraints on G, there exists one or more cuts on G that satisfies the constraints. Additionally, we show that any such cut can be computed using the general strategy. Thereafter, we present a particular instance of the general strategy, that produces cuts such that the maximum edge weight in the cut is globally minimal.

4.1 A General Strategy for Computing Constrained Graph Cuts

If $S \subseteq E$ is a cut on G and $G' = (V, E \setminus S)$, then each segment of S forms a boundary between exactly two connected components of G'. Therefore, the removal of a segment from a cut is called a *merging operation*.

Definition 8. *Let S be a cut on G and let $e \in S$. The* merging operation $S \oslash e$ *is defined as*

$$S \oslash e = S \setminus S_e \ . \tag{7}$$

Note that merging operations preserve cuts, i.e., $S \oslash e$ is a cut on G.

Definition 9. *Let C be a set of constraints and let S be a cut on G. An edge $e \in S$ is* mergeable *with respect to C if $S \oslash e$ is not an under-segmentation with respect to C.*

The set of edges in S that are mergeable with respect to C is denoted $M_C(S)$.

Lemma 1. *Let C be a set of constraints and let S be a cut on G such that S is not an under-segmentation with respect to C. Then the following statements are equivalent:*

1. *S is an over-segmentation with respect to C.*
2. *$M_C(S) \neq \emptyset$.*

Proof. S is an over-segmentation with respect to $C \Leftrightarrow \exists e \in S$ such that $\forall c_{v,w} \in C, \underset{(V,E\setminus(S\setminus\{e\}))}{v \not\sim w} \Leftrightarrow \exists e \in S$ such that $\forall c_{v,w} \in C, \underset{(V,E\setminus(S\oslash e))}{v \not\sim w} \Leftrightarrow \exists e \in M_C(S) \Leftrightarrow M_C(S) \neq \emptyset$. □

Definition 10. *Let S be a cut on G, and let $\sigma = \langle e_1, e_2, \ldots, e_k \rangle = \langle e_i \rangle_{i=1}^k$ be a sequence of edges in S. If $e_{n+1} \in S \oslash e_1 \oslash e_2 \oslash \ldots \oslash e_n$ for all $n \in [1, k-1]$, then σ is a* merging sequence *for S.*

If $\sigma = \langle e_i \rangle_{i=1}^k$ is a merging sequence for S, we define $S \oslash \sigma$ as

$$S \oslash \sigma = S \oslash e_1 \oslash e_2 \oslash \ldots \oslash e_k \ . \tag{8}$$

If C is a set of constraints and $e_{n+1} \in M_C(S \oslash \langle e_i \rangle_{i=1}^n)$ for all $n \in [1, k-1]$, then σ is *valid* with respect to C. If $M_C(S \oslash \sigma) = \emptyset$ then σ is *complete* with respect to C. If σ_1 and σ_2 are merging sequences for S, we denote by $\sigma_1 \cdot \sigma_2$ the concatenation of the two sequences. If σ_2 is a merging sequence for $S \oslash \sigma_1$, it holds that $\sigma_1 \cdot \sigma_2$ is a merging sequence for S and $S \oslash (\sigma_1 \cdot \sigma_2) = S \oslash (\sigma_2 \cdot \sigma_1)$.

Theorem 4. *Let C be a set of constraints, let S be a cut on G such that S is not an under-segmentation with respect to C, and let σ be a complete valid merging sequence for S with respect to C. Then $S \oslash \sigma$ satisfies C.*

Proof. Since $M_C(S \oslash \sigma) = \emptyset$, it holds by Lemma 1 that $S \oslash \sigma$ is not an over-segmentation with respect to C. Since S is not an under-segmentation with respect to C, it follows from the definition of mergeable edges that $S \oslash \sigma$ is also not an under-segmentation with respect to C. □

If S is a cut on G such that S is not an under-segmentation with respect to a set of constraints C, then there exists a merging sequence σ for S such that σ is valid and complete with respect to C. In particular, we can construct such a σ using the following procedure:

1. Let σ be an empty sequence
2. While $M_C(S \oslash \sigma) \neq \emptyset$, append an element from $M_C(S \oslash \sigma)$ to σ.

Thus, it follows from Theorem 4 that if S is a cut on G such that S is not an under-segmentation with respect to C, there exists one or more $S' \subseteq S$ such that S' satisfies C. We define the set $\Sigma(S, C)$ as

$$\Sigma(S, C) = \{S' \subseteq S \mid S' \text{ satisfies } C\} . \tag{9}$$

Note that if S satisfies C, then $\Sigma(S, C) = \{S\}$.

Theorem 5. *Let C be a set of constraints, let $S \subseteq E$ be a cut on G such that $\Sigma(S, C) \neq \emptyset$, and let $S' \in \Sigma(S, C)$. Then there exists a (possibly empty) merging sequence σ for S' such that $S' = S \oslash \sigma$ and σ is valid with respect to C.*

Proof. Consider the following procedure:

1. Let σ be an empty sequence
2. While $S \oslash \sigma \setminus S' \neq \emptyset$, append an element from $S \oslash \sigma \setminus S'$ to σ.

Since S' is a cut on G, it holds that $S' \subseteq S \oslash \sigma$ at each step of this procedure. In particular, this holds when the procedure terminates. By then it also holds that $S \oslash \sigma \setminus S' = \emptyset$, and so $S \oslash \sigma = S'$. The validity of σ follows from the fact that S' satisfies C. □

Any cut S on G is, by definition, a subset of E. Thus it follows from Theorem 5 that if S satisfies a set of constraints C, then $S = E \oslash \sigma$ for some merging sequence σ, i.e., any cut that satisfies C may be obtained by repeatedly performing merging operations on mergeable edges in E.

4.2 An Optimal Strategy for Computing Constrained Graph Cuts

In Section 4.1, we established that for any set of constraints C, there exists one or more cuts that satisfy C. In this Section we are interested in finding cuts, among all cuts that satisfy C, that represent "good" partitionings of the graph. Commonly, the goodness of a cut is measured by some function of the weight of the edges in the cut, i.e., the sum of the edge weights [2], the normalized sum

of the edge weights [14], or the maximum edge weight [13]. Here, we define the *weight* $W(S)$ of a cut S as the latter, namely

$$W(S) = \max_{e \in S}(W(e)) \, . \tag{10}$$

A cut S is considered to be good if it has a low weight. The relevance of this criterion for image segmentation has previously been demonstrated by others. For example, the popular *fuzzy connectedness* method has been shown to optimize the same criterion in the presence of regional constraints [13]. If S is a cut that is not an under-segmentation with respect to a set of constraints C, we define $\Sigma^*(S, C)$ as

$$\Sigma^*(S, C) = \operatorname*{argmin}_{S' \in \Sigma(S,C)} (W(S')) \, . \tag{11}$$

Since all cuts in $\Sigma^*(S, C)$ have the same (minimal) weight, $W(\Sigma^*(S, C))$ is well defined even if $|\Sigma^*(S, C)| > 1$. We now present a strategy for finding a merging sequence σ such that $E \oslash \sigma \in \Sigma^*(E, C)$, i.e., finding a cut $S = E \oslash \sigma$ for which $W(S)$ is globally minimal.

Lemma 2. *Let C be a set of constraints, let S be a cut on G such that S is not an under-segmentation with respect to C, and let $e \in M_C(S)$. If $W(e) = \max_{e' \in M_C(S)}(W(e'))$, then $W(\Sigma^*(S \oslash e, C)) = W(\Sigma^*(S, C)))$.*

Proof. Assume to the contrary that there exists a $S' \in \Sigma(S, C)$ such that $W(S') < W(\Sigma^*(S \oslash e, C))$. If $W(e) = \max_{e' \in M_C(S)}(W(e'))$, then $W(S') < W(e)$ and so $e \notin S'$. By Theorem 5, S' can thus be written as $S' = S \oslash \sigma$, where σ is a merging sequence of the form $\sigma_1 \cdot \langle e \rangle \cdot \sigma_2$. Since $S \oslash \sigma_1 \cdot \langle e \rangle \cdot \sigma 2 = S \oslash \langle e \rangle \cdot \sigma_1 \cdot \sigma_2$ it holds that $S' \in \Sigma(S \oslash e, C)$. This contradicts the assumption that $W(S') < W(\Sigma^*(S \oslash e, C))$. □

Definition 11. *Let S be a cut on G, let C be a set of constraints, and let $\sigma = \langle e_i \rangle_{i=1}^k$ be a merging sequence for S such that σ is valid with respect to C. If*

$$W(e_{n+1}) = \max_{e \in M_C(S \oslash \langle e_i \rangle_{i=1}^n)}(W(e)) \tag{12}$$

for all $n \in [1, k-1]$, then σ is maximal *with respect to C.*

Following the procedure for constructing a complete valid merging sequence in Section 4.1, it is straightforward to show that a complete maximal merging sequence exists for any S and C. From Lemma 2 it follows by induction that if σ is a merging sequence for S such that σ is maximal with respect to C, then $S \oslash \sigma \in \Sigma^*(S, C)$. Thereby, Theorem 6 follows.

Theorem 6. *Let C be a set of constraints, and let S_1 and S_2 be cuts that satisfy C. If $S_1 = E \oslash \sigma$, where σ is a maximal sequence with respect to C, then $W(S_1) \leq W(S_2)$.*

5 A Practical Algorithm

In this Section, we re-formulate the approach presented in Section 4.2 in terms of a practical algorithm, listed in pseudo-code in Algorithm 1. Given a set of constraints C, the algorithm computes a vertex labeling $\mathcal{L}$ such that $\partial\mathcal{L} \in \Sigma^*(E, C)$. The algorithm is based on the observation that, given a vertex labeling such that each component has a unique label, a merging operation between two adjacent components is equivalent to replacing all labels in one component with the label of the other component.

The computational efficiency of Algorithm 1 depends on a number of implementational choices. Below, we highlight some implementational details that we have found greatly improves the speed of the algorithm.

Pre-sorting edges. At each step of the outermost loop of Algorithm 1, an edge with maximum weight is selected (line 4). This can be implemented efficiently by pre-sorting all edges of the graph in a non-increasing order by weight. If the edge weights are integer valued, this can be performed in $\mathcal{O}(|E|)$ operations using count sorting. In interactive segmentation applications, this sorting step need only be performed once, prior to user interaction.

Efficient region merging. A naive implementation of the region merging step of Algorithm 1 (lines 6–7) involves updating the label of all elements for which $\mathcal{L}(z) = \mathcal{L}(v)$. This becomes prohibitively slow even for modestly sized images. Instead, we have used a look-up table to keep track of the label of each region. This table maps the initial, unique, label of each vertex to its label in the final segmentation. Thus, lines 6–7 of Algorithm 1 is replaced by an operation that can be performed in constant time.

With the above techniques implemented, the most computationally expensive operation within the outermost loop of Algorithm 1 is to check the existence of a constraint that makes $e_{v,w}$ non-mergeable (line 5). In our current implementation, this check is performed by iterating over all constraints in C. The outermost loop is repeated $|E|$ times and so, in total, the algorithm requires $\mathcal{O}(|E||C|)$ operations.

Algorithm 1: Computing minimum weight cuts satisfying generalized hard constraints.

Input: A weighted graph $G = (V, E)$ and a set C of constraints.
Output: A vertex labeling $\mathcal{L}$ such that $\partial\mathcal{L} \in \Sigma^*(E, C)$.
Auxiliary: A set of edges E'.

```
1 Initialize L so that each vertex has a unique label;
2 Set E' ← E;
3 while E' ≠ ∅ do
4     Select an edge e_{v,w} ∈ E' such that W(e_{v,w}) is maximal;
5     if L(v) ≠ L(w) and ∄ c_{x,y} ∈ C s. t. L(v) = L(x) and L(w) = L(y) then
6         foreach z ∈ V s.t. L(z) = L(w) do
7             Set L(z) ← L(v);
8     Set E' ← E' \ {e_{v,w}};
```

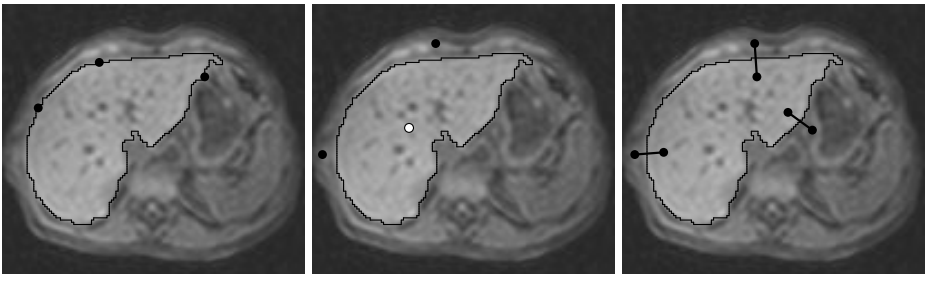

Fig. 3. Interactive segmentation of the liver in a slice from an MR volume image, using three different interaction paradigms. All segmentations were computed using Algorithm 1. (Left) Segmentation using boundary constraints. The black dots indicate graph edges that must be included in the segmentation boundary. (Middle) Segmentation using regional constraints. Black and white dots indicate background and object seeds, respectively. (Right) Segmentation using generalized constraints. Each constraint is displayed as two black dots connected by a line. [MRI data courtesy of Dr. Olof Dahlqvist-Leinhard at CMIV, Linköping University, Sweden.]

6 Conclusions

We have defined graph cuts with respect to generalized hard constraints, and shown that this new type of constraints include boundary constraints and regional constraints as special cases. In previous work on supervised graph segmentation, different computational strategies have typically been required to compute cuts that satisfy regional and boundary constraints, respectively. This work unifies and generalizes the two paradigms. Figure 3 illustrates the ability of Algorithm 1 to handle different types of constraints. We emphasize that while this example shows segmentation of a 2D image, our results are derived for arbitrary undirected graphs and thus directly applicable to images of any dimension.

All interactive segmentation methods are subject to variations in user input. It is therefore desirable for a segmentation method to be invariant to "small" changes in the set of constraints [1]. Initial experiments indicate that Algorithm 1 indeed satisfies such a property. Note, e.g., that all three segmentations shown in Figure 3 are identical, despite being computed from different sets of constraints. In future work, the precise nature of this robustness property will be investigated.

In Section 5, we presented a practical algorithm for computing minimum weight cuts that satisfy a set of constraints. In future work, we intend to explore further the computational aspects of the proposed method. In particular, *differential* algorithms are of interest [5].

References

1. Audigier, R., Lotufo, R.A.: Seed-relative segmentation robustness of watershed and fuzzy connectedness approaches. In: Falcão, A.X., Lopes, H. (eds.) Proceedings of the 20th Brazilian Symposium on Computer Graphics and Image Processing, pp. 61–68. IEEE Computer Society, Los Alamitos (2007)

2. Boykov, Y., Jolly, M.-P.: Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images. In: Proceedings of the 8th IEEE International Conference on Computer Vision (ICCV), vol. 1, pp. 105–112 (2001)
3. Couprie, C., Grady, L., Najman, L., Talbot, H.: Power watersheds: A unifying graph-based optimization framework. IEEE Transactions on Pattern Analysis and Machine Intelligence 99(PrePrints) (2010), doi:10.1109/TPAMI.2010.200.
4. 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 (2010)
5. Falcão, A.X., Bergo, F.P.: Interactive volume segmentation with differential image foresting transforms. IEEE Transactions on Medical Imaging 23(9), 1100–1108 (2004)
6. Falcão, A.X., Stolfi, J., Lotufo, R.A.: The image foresting transform: Theory, algorithms, and applications. IEEE Transactions on Pattern Analysis and Machine Intelligence 26(1), 19–29 (2004)
7. Falcão, A.X., Udupa, J.K., Miyazawa, F.K.: An ultra-fast user-steered image segmentation paradigm: Live wire on the fly. IEEE Transactions on Medical Imaging 19(1), 55–62 (2000)
8. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient graph-based image segmentation. International Journal of Computer Vision 59(2) (2004)
9. Grady, L.: Random walks for image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence 28(11), 1768–1783 (2006)
10. Grady, L.: Minimal surfaces extend shortest path segmentation methods to 3D. IEEE Transactions on Pattern Analysis and Machine Intelligence 32(2), 321–334 (2010)
11. Malmberg, F., Lindblad, J., Sladoje, N., Nyström, I.: A graph-based framework for sub-pixel image segmentation. Theoretical Computer Science (2010), doi:10.1016/j.tcs.2010.11.030.
12. Malmberg, F., Vidholm, E., Nyström, I.: A 3D live-wire segmentation method for volume images using haptic interaction. In: Kuba, A., Nyúl, L.G., Palágyi, K. (eds.) DGCI 2006. LNCS, vol. 4245, pp. 663–673. Springer, Heidelberg (2006)
13. Miranda, P.A., Falcão, A.X.: Links between image segmentation based on optimum-path forest and minimum cut in graph. Journal of Mathematical Imaging and Vision 35(2), 128–142 (2009)
14. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence 22(8), 888–905 (2000)
15. Udupa, J.K., Saha, P.K., Lotufo, R.A.: Relative fuzzy connectedness and object definition: Theory, algorithms, and applications in image segmentation. IEEE Transactions on Pattern Anaysis and Machine Intelligence 24(11), 1485–1500 (2002)