

# The Mutex Watershed and its Objective: Efficient, Parameter-Free Image Partitioning

Steffen Wolf\*, Alberto Bailoni\*, Constantin Pape, Nasim Rahaman, Anna Kreshuk, Ullrich Köthe, and Fred A. Hamprecht

**Abstract**—Image partitioning, or segmentation without semantics, is the task of decomposing an image into distinct segments, or equivalently to detect closed contours. Most prior work either requires seeds, one per segment; or a threshold; or formulates the task as multicut / correlation clustering, an NP-hard problem. Here, we propose a greedy algorithm for signed graph partitioning, the “Mutex Watershed”. Unlike seeded watershed, the algorithm can accommodate not only attractive but also repulsive cues, allowing it to find a previously *unspecified* number of segments without the need for explicit seeds or a tunable threshold. We also prove that this simple algorithm solves to global optimality an objective function that is intimately related to the multicut / correlation clustering integer linear programming formulation. The algorithm is deterministic, very simple to implement, and has empirically linearithmic complexity. When presented with short-range attractive and long-range repulsive cues from a deep neural network, the Mutex Watershed gives the best results currently known for the competitive ISBI 2012 EM segmentation benchmark.

**Index Terms**—Image segmentation, partitioning algorithms, greedy algorithms, optimization, integer linear programming, machine learning, convolutional neural networks.



## 1 INTRODUCTION

MOST image partitioning algorithms are defined over a graph encoding purely attractive interactions. No matter whether a segmentation or clustering is then found agglomeratively (as in single linkage clustering / watershed) or divisively (as in spectral clustering or iterated normalized cuts), the user either needs to specify the desired number of segments or a termination criterion. An even stronger form of supervision is in terms of seeds, where one pixel of each segment needs to be designated either by a user or automatically. Unfortunately, clustering with automated seed selection remains a fragile and error-fraught process, because every missed or hallucinated seed causes an under- or oversegmentation error. Although the learning of good edge detectors boosts the quality of classical seed selection strategies (such as finding local minima of the boundary map, or thresholding boundary maps), non-local effects of seed placement along with strong variability in region sizes and shapes make it hard for any learned predictor to place *exactly one* seed in every true region.

In contrast to the above class of algorithms, multicut / correlation clustering partitions vertices with both attractive and repulsive interactions encoded into the edges of a graph. Multicut has the great advantage that a “natural” partitioning of a graph can be found, without needing to specify a desired number of clusters, or a termination criterion, or one seed per region. Its great drawback is that its optimization is NP-hard.

The main insight of this paper is that when both attractive and repulsive interactions between pixels are available, then a generalization of the watershed algorithm can be devised that segments an image *without* the need for seeds or stopping criteria or thresholds. It examines all graph edges, attractive and repulsive,

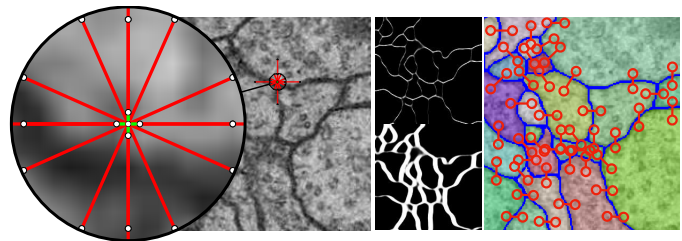


Fig. 1: Left: Overlay of raw data from the ISBI 2012 EM segmentation challenge and the edges for which attractive (green) or repulsive (red) interactions are estimated for each pixel using a CNN. Middle: vertical / horizontal repulsive interactions at intermediate / long range are shown in the top / bottom half. Right: Active mutual exclusion (mutex) constraints that the proposed algorithm invokes during the segmentation process.

sorted by their weight and adds these to an active set iff they are not in conflict with previous, higher-priority, decisions. The attractive subset of the resulting active set is a forest, with one tree representing each segment. However, the active set can have loops involving more than one repulsive edge. See Fig. 1 for a visual abstract.

In summary, our principal contributions are, first, a fast deterministic algorithm for signed graph partitioning that does not need prior specification of the number of clusters (section 4); and second, its theoretical characterization, including proof that it globally optimizes an objective related to the multicut correlation clustering objective (4).

Combined with a deep net, the algorithm also happens to define the state-of-the-art in a competitive neuron segmentation challenge (section 5).

This is an extended version version of [1], with the second principal contribution (section 4) being new.

\* Authors contributed equally

- All authors are with HCI/IWR, Heidelberg University, Germany. E-mail: <firstname>.<lastname>@iwr.uni-heidelberg.de
- A. Kreshuk and C. Pape are with EMBL, Heidelberg, Germany.

Manuscript received 2018-XX-XX; revised 2018-XX-XX.

## 2 RELATED WORK

In the original watershed algorithm [2], seeds were automatically placed at all local minima of the boundary map. Unfortunately, this leads to severe over-segmentation. Defining better seeds has been a recurring theme of watershed research ever since. The simplest solution is offered by the seeded watershed algorithm [3]: It relies on an oracle (an external algorithm or a human) to provide seeds and assigns each pixel to its nearest seed in terms of minimax path distance. In the absence of an oracle, automatic seed selection is challenging because *exactly one* seed must be placed in every region. Simple methods, e.g. defining seeds by connected regions of low boundary probability, don't work: The segmentation quality is usually insufficient because multiple seeds are in the same region and/or seeds leak through the boundary.

This problem is typically addressed by biasing seed selection towards over-segmentation (with seeding at all minima being the extreme case). The watershed algorithm then produces superpixels that are merged into final regions by more or less elaborate postprocessing. This works better than using watersheds alone because it exploits the larger context afforded by superpixel adjacency graphs. Many criteria have been proposed to identify the regions to be preserved during merging, e.g. region saliency [4], the waterfall transform [5], extinction values [6], region saliency [7], and  $(\alpha, \omega)$ -connected components [8]. A merging process controlled by criteria like these can be iterated to produce a hierarchy of segmentations where important regions survive to the next level. Variants of such hierarchical watersheds are reviewed and evaluated in [9].

These results highlight the close connection of watersheds to hierarchical clustering and minimum spanning trees/forests [10], [11], which inspired novel merging strategies and termination criteria. For example, [12] simply terminated hierarchical merging by fixing the number of surviving regions beforehand. [13] incorporate predefined sets of generalized merge constraints into the clustering algorithm. Graph-based segmentation according to [14] defines a measure of quality for the current regions and stops when the merge costs would exceed this measure. Ultrametric contour maps [15] combine the gPb (global probability of boundary) edge detector with an oriented watershed transform. Superpixels are agglomerated until the ultrametric distance between the resulting regions exceeds a learned threshold. An optimization perspective is taken in [16], which introduces  $h$ -increasing energy functions and builds the hierarchy incrementally such that merge decisions greedily minimize the energy. The authors prove that the optimal cut corresponds to a different unique segmentation for every value of a free regularization parameter.

An important line of research is based on the observation that superior partitionings are obtained when the graph has both attractive and repulsive edges [17]. Solutions that optimally balance attraction and repulsion do not require external stopping criteria such as predefined number of regions or seeds. This generalization leads to the NP-hard problem of correlation clustering or (synonymous) multicut (MC) partitioning. Fortunately, modern integer linear programming solvers in combination with incremental constraint generation can solve problem instances of considerable size [18], and good approximations exist for even larger problems [19], [20] Reminiscent of strict minimizers [21] with minimal  $L_\infty$ -norm solution, our work solves the multicut objective optimally when all graph weights are raised to a large power.

Related to the proposed method, the greedy additive edge

contraction (GAEC) [22] heuristic for the multicut also sequentially merges regions, but we handle attractive and repulsive interactions separately and define edge strength between clusters by a maximum instead of an additive rule. The greedy fixation algorithm introduced in [23] is closely related to the proposed method; it sorts attractive and repulsive edges by their absolute weight, merges nodes connected by attractive edges and introduces no-merge constraints for repulsive edges. However, similar to GAEC, it defines edge strength by an additive rule, which increases the algorithm's runtime complexity compared to the Mutex Watershed. Also, it is not yet known what objective the algorithm optimizes globally, if any.

Another beneficial extension relative to standard approaches is the introduction of additional long-range edges. The strength of such edges can often be estimated with greater certainty than is achievable for the local edges used by watersheds on standard 4- or 8-connected pixel graphs. Such repulsive long-range edges have been used in [24] to represent object diameter constraints, which is still an MC-type problem. When long-range edges are also allowed to be attractive, the problem turns into the more complicated lifted multicut (LMC) [25]. Realistic problem sizes can only be solved approximately [22], [26], but watershed superpixels followed by LMC postprocessing achieve state-of-the-art results on important benchmarks [27]. Long-range edges are also used in [28], as side losses for the boundary detection convolutional neural network (CNN); but they are not used explicitly in any downstream inference.

In general, striking progress in watershed-based segmentation has been achieved by learning boundary maps with CNNs. This is nicely illustrated by the evolution of neurosegmentation for connectomics, an important field we also address in the experimental section. CNNs were introduced to this application in [29] and became, in much refined form [30], the winning entry of the ISBI 2012 Neuro-Segmentation Challenge [31]. Boundary maps and superpixels were further improved by progress in CNN architectures and data augmentation methods, using U-Nets [32], FusionNets [33] or inception modules [27]. Subsequent postprocessing with the GALA algorithm [34], [35], conditional random fields [36] or the lifted multicut [27] pushed the envelope of final segmentation quality. MaskExtend [37] applied CNNs to both boundary map prediction and superpixel merging, while flood-filling networks [38] eliminated superpixels altogether by training a recurrent neural network to perform region growing one region at a time.

Most networks mentioned so far learn boundary maps on pixels, but learning works equally well for edge-based watersheds, as was demonstrated in [39], [40] using edge weights generated with a CNN [41], [42]. Tailoring the learning objective to the needs of the watershed algorithm by penalizing critical edges along minimax paths [42] or end-to-end training of edge weights and region growing [43] improved results yet again.

Outside of connectomics, [44] obtained superior boundary maps from CNNs by learning not just boundary strength, but also its gradient direction. Holistically-nested edge detection [45], [46] couples the CNN loss at multiple resolutions using deep supervision and is successfully used as a basis for watershed segmentation of medical images in [47].

We adopt important ideas from this prior work (hierarchical single-linkage clustering, attractive and repulsive interactions, long-range edges, and CNN-based learning). The proposed efficient segmentation framework can be interpreted as a generalization of

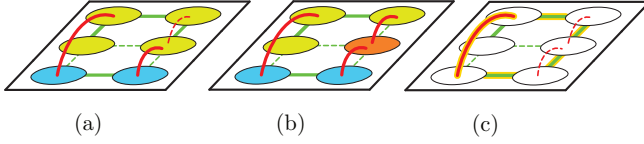


Fig. 2: In (a) and (b), possible active sets  $A$  (bold edges) with associated clusterings (arbitrary node colors), for a given graph with attractive (green) and repulsive (red) edges. In (c), the active set includes a frustrated  $\mathcal{C}_1(A)$  cycle (highlighted in yellow), therefore it cannot be associated with a consistent node clustering.

[13], because we also allow for soft repulsive interactions (which can be overridden by strong attractive edges), and constraints are generated on-the-fly.

### 3 THE MUTEX WATERSHED ALGORITHM

In this section we introduce the Mutex Watershed Algorithm, an efficient graph clustering algorithm that can ingest both attractive and repulsive cues. We first reformulate seeded watershed as a graph partitioning with infinitely repulsive edges and then derive the generalized algorithm for finitely repulsive edges, which obviates the need for seeds.

#### 3.1 Notation

We consider the problem of clustering a graph  $\mathcal{G}(V, E^+ \cup E^-, W^+ \cup W^-)$  with both attractive and repulsive edge attributes. The scalar attribute  $w_e^+ \in \mathbb{R}_0^+$  associated with edge  $e \in E^+$  is a merge affinity: the higher this number, the higher the inclination of the two incident vertices to be assigned to the same cluster. Similarly,  $w_e^- \in \mathbb{R}_0^+$  for  $e \in E^-$  is a split tendency: the higher this number, the greater the desire of the incident vertices to be in different clusters. In our application, each vertex corresponds to one pixel in the image to be segmented.

We describe a clustering of  $\mathcal{G}$  by the active set  $A = A^+ \cup A^-$  composed of two disjunct active sets  $A^+ \subseteq E^+$ ,  $A^- \subseteq E^-$ , that encode merges and mutual exclusions (must-not-link constraints) between clusters, respectively. We then define the set  $\mathcal{C}_i(A)$  of all cycles with exactly  $i$  active repulsive edges

$$\mathcal{C}_i(A) := \{c \in \text{cycles}(\mathcal{G}) \mid c \subseteq A \text{ and } |c \cap E^-| = i\}. \quad (1)$$

An active set is *consistent* when it can be associated to a clustering of  $\mathcal{G}$ . More specifically, in a consistent active set two vertices cannot be both mutually exclusive *and* connected. Thus, we enforce the consistency of the segmentation by forbidding all cycles with exactly one active repulsive edge, i.e. by allowing only active sets s.t.  $\mathcal{C}_1(A) = \emptyset$ . See Fig. 2c for an example of an inconsistent active set.

On the other hand, cycles with more than one active repulsive edge are allowed and  $\mathcal{C}_{i \geq 2}(A)$  can be nonempty in general. For instance, there can be consistent active sets s.t.  $\mathcal{C}_2(A) \neq \emptyset$  when there exist multiple mutual exclusion relations between two clusters (see Fig. 2a). The fact that  $\mathcal{C}_{i \geq 3}(A)$  can be nonempty reflects a fundamental asymmetry between attractive and repulsive edges, namely that attraction is transitive while mutual exclusion constraints are not (Fig. 2b).

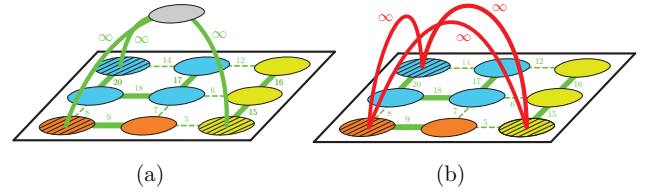


Fig. 3: Two equivalent representations of the seeded watershed clustering obtained using (a) a maximum spanning tree computation or (b) Algorithm 1. Both graphs share the weighted attractive (green) edges and seeds (hatched nodes). The infinitely attractive connections to the auxiliary node (gray) in (a) are replaced by infinitely repulsive (red) edges between each pair of seeds in (b). The two final clusterings are defined by the active sets (bold edges) and are identical. Node colors indicate the clustering result, but are arbitrary.

---

**Inputs:** weighted graph  $\mathcal{G}(V, E^+, W^+)$  and seeds  $S \subseteq V$ ;  
**Output:** clusters defined by spanning forest  $A \cap E^+$ ;  
**Initialization:**  $A = E^-$ , where  
 $E^- = \{(p, q) \mid p, q \in S; p \neq q\}$ ;  
**for**  $(i, j) = e \in E^+$  in descending order of  $w^+$  **do**  
    **if**  $\mathcal{C}_0(A \cup \{e\}) = \emptyset$  and  $\mathcal{C}_1(A \cup \{e\}) = \emptyset$  **then**  
         $A \leftarrow A \cup e$ ;  
    **end**  
**end**

---

**Algorithm 1:** Mutex version of seeded watershed algorithm. The set of cycles  $\mathcal{C}_0(A)$  and  $\mathcal{C}_1(A)$  are defined in Equation 1.

Finally, the constraint  $\mathcal{C}_0(A) = \emptyset$  can be enforced to ensure that the active set  $A^+ \subseteq E^+$  is a forest, similarly to a seeded watershed algorithm.

#### 3.2 Seeded watershed from a mutex perspective

One interpretation of the proposed method is in terms of a generalization of the edge-based watershed algorithm [10], [48], [49] or image foresting transform [50]. For didactic reasons, we here first reformulate this fundamental algorithm in a manner compatible with the proposed method.

Edge-based watershed can only ingest a graph with purely attractive interactions,  $\mathcal{G}(V, E^+, W^+)$ . Without further constraints, the algorithm would yield only the trivial result of a single cluster comprising all vertices. To obtain more interesting output, an oracle needs to provide seeds, namely precisely one node per cluster. These seed vertices are all connected to an auxiliary node (see Fig. 3a) by auxiliary edges with infinite merge affinity. A maximum spanning tree (MST) on this augmented graph can be found in linearithmic time; and the maximum spanning tree (or in the case of degeneracy: at least one of the maximum spanning trees) will include the auxiliary edges. When the auxiliary edges are deleted from the MST, a forest results, with each tree representing one cluster [48], [50], [51].

We now reformulate this well-known algorithm in a way that will later emerge as a special case of the proposed Mutex Watershed: we eliminate the auxiliary node and edges, and replace them by a set of infinitely repulsive edges, one for each pair of seeds (Fig. 3b).

Algorithm 1, applied to the seed mutex graph just defined, gives results identical to seeded watershed on the original graph.

The procedure closely resembles Kruskal’s MST algorithm, which iterates over a sorted list of edges and adds each edge only if it does not introduce a cycle.

Clearly, the modified algorithm has the same effect as the original seeded watershed algorithm, because the final set  $A^+$  is exactly the maximum spanning forest obtained after removing the auxiliary edges from the original solution.

In the sequel, we generalize this construction by admitting less-than-infinitely repulsive edges that do not need to coincide in seed points. This allows repulsive edges to be estimated on the same neighborhood for all pixels, favoring a fully convolutional learning and prediction with CNNs (section 5).

### 3.3 Mutex Watersheds

We now introduce the first core contribution: an algorithm that is empirically no more expensive than a MST computation; but that can ingest both attractive and repulsive cues and partition a graph into a number of clusters that does not need to be specified beforehand. There is no requirement of one seed per cluster, and not even of a hyperparameter that would implicitly determine the number of resulting clusters.

The Mutex Watershed, Algorithm 2, is illustrated in Fig. 4 for a toy example and proceeds as follows. Given a graph with sets of attractive and repulsive edges  $E^+$  and  $E^-$  with edge weights  $W^+$  and  $W^-$  respectively, do the following: sort all edges  $E^+ \cup E^-$ , attractive and repulsive, by their weight in descending order into a priority queue. Iteratively pop all edges from the priority queue and add them to the active set  $A$ , one by one, provided that a set of conditions are satisfied. More specifically, if the next edge popped from the priority queue is attractive, then it is added to the active set if and only if i) its incident vertices are not yet in the same tree of the forest  $A^+$  ( $\mathcal{C}_0(A \cup \{e\}) = \emptyset$ ) and ii) there are no mutual exclusion constraints between the two trees ( $\mathcal{C}_1(A \cup \{e\}) = \emptyset$ ). If, on the other hand, the popped edge is repulsive, then a mutual exclusion constraint is added to the active set if and only if the incident vertices of the edge are not yet in the same tree of  $A^+$  ( $\mathcal{C}_1(A \cup \{e\}) = \emptyset$ ).

The crucial difference to algorithm 1 is that mutex constraints are no longer pre-defined, but created dynamically whenever a repulsive edge is found. However, new exclusion constraints can never override earlier, high-priority merge decisions. In this case, the repulsive edge in question is simply ignored. Similarly, an attractive edge must never override earlier and thus higher-priority must-not-link decisions.

---

**Input:** weighted graph  $\mathcal{G}(V, E^+ \cup E^-, W^+ \cup W^-)$ ;  
**Output:** clusters defined by spanning forest  $A^* \cap E^+$ ;  
**Initialization:**  $A = \emptyset$ ;  
**for**  $(i, j) = e \in (E^+ \cup E^-)$  in descending order of  $W^+ \cup W^-$  **do**  
     **if**  $\mathcal{C}_0(A \cup \{e\}) = \emptyset$  **and**  $\mathcal{C}_1(A \cup \{e\}) = \emptyset$  **then**  
          $A \leftarrow A \cup e$ ;  
     **end**  
**end**  
 $A^* \leftarrow A$ ;  
**return**  $A^*$ ;

---

**Algorithm 2:** Mutex Watershed Algorithm. The set of cycles  $\mathcal{C}_0(A)$  and  $\mathcal{C}_1(A)$  are defined in Equation 1.

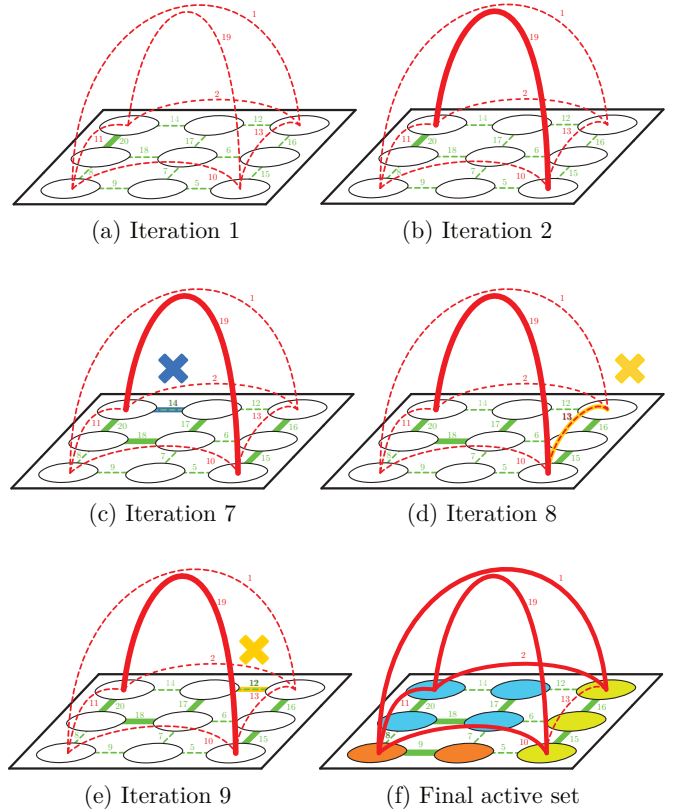


Fig. 4: Some iterations of the Mutex Watershed Algorithm 2 applied to a graph with weighted attractive (green) and repulsive (red) edges. Edges accumulated in the active set  $A$  after a given number of iterations are shown in bold. Once the algorithm terminates, the final active set (f) defines the final clustering (indicated using arbitrary node colors). Some edges that are not added to the active set because they would violate constraints  $\mathcal{C}_0$  or  $\mathcal{C}_1$  are highlighted in blue and yellow, respectively.

Fig. 4 illustrates the proposed algorithm: Fig. 4a and Fig. 4b show examples of an unconstrained merge and an added mutex constraint, respectively; Fig. 4c shows a  $\mathcal{C}_0$ -constrained attractive edge ( $w_e^+ = 14$ ) with incident vertices in the same tree of the forest  $A^+$ ; finally, Fig. 4d and Fig. 4e show, respectively, a repulsive edge ( $w_e^- = 13$ ) and an attractive edge ( $w_e^+ = 12$ ) ruled out by  $\mathcal{C}_1$  mutual exclusion relations.

## 4 THEORETICAL CHARACTERIZATION

This section summarizes our second core contribution, the proof that the Mutex Watershed Algorithm presented in the previous section globally optimizes a precise energy minimization problem. We begin by defining the Mutex Watershed Objective (Sec. 4.1) and prove that the Mutex Watershed Algorithm solves it optimally (Sec. 4.2).

Finally, we show how this objective is related to the NP-hard correlation clustering / multicut graph partitioning problem [52] (Sec. 4.3).

### 4.1 Mutex Watershed Objective

We now define the Mutex Watershed energy that is minimized by the Mutex Watershed Algorithm. First, we introduce the following property:

**Definition 4.1. Dominant power:** Let  $\mathcal{G} = (V, E, W)$  be an edge-weighted graph, with unique weights  $w_e \in \mathbb{R}_0^+$ ,  $\forall e \in E$ . We call  $p \in \mathbb{R}^+$  a dominant power if:

$$w_e^p > \sum_{t \in E, w_t < w_e} w_t^p \quad \forall e \in E, \quad (2)$$

Note that there exists a dominant power for any finite set of edges, since for any  $e \in E$  we can divide (2) by  $w_e^p$  and observe that the normalized weights  $w_t^p/w_e^p$  (and any finite sum of these weights) converges to 0 when  $p$  tends to infinity.

**Definition 4.2. Mutex Watershed Objective:** Let  $\mathcal{G} = (V, E, W)$  be a weighted graph, with unique weights  $w_e \in \mathbb{R}_0^+$  and  $p \in \mathbb{R}^+$  a dominant power. Then the Mutex Watershed Objective is defined as the integer linear program

$$\min_{a \in \{0,1\}^{|E|}} - \sum_{e \in E} a_e w_e^p \quad (3)$$

$$\text{s.t. } \mathcal{C}_0(A) = \mathcal{C}_1(A) = \emptyset, \quad (4)$$

$$\text{with } A := \{e \in E | a_e = 1\}$$

**Theorem 4.1.** Let  $\mathcal{G} = (V, E^+ \cup E^-, W^+ \cup W^-)$  be an edge-weighted graph and  $p \in \mathbb{R}^+$  a dominant power. The final active set of the Mutex Watershed Algorithm  $a_e^* = \mathbb{1}\{e \in A^*\}$  minimizes the Mutex Watershed Objective.

To prove Theorem 4.1 and show that the Mutex Watershed Algorithm 2 finds the optimum of the Mutex Watershed Objective, we prove its **greedy choice** and **optimal substructure property** [53]. For both proofs we rely on Equation 2, which guarantees that the greedy choice (i.e. the feasible edge with the highest weight) will always give a greater negative energy contribution than any combination of weaker edges.

## 4.2 Optimality of the Mutex Watershed Algorithm

In this section we prove Theorem 4.1, i.e. that the Mutex Watershed Objective defined in Equation 3 is solved to optimality by the Mutex Watershed Algorithm 2.

First, we generalize the Mutex Watershed (see algorithm 3) and objective (definition 4.3) such that an initial consistent set of active edges  $\tilde{A} \subseteq E$  is supplied and required to be a subset of the final solution. For readability, we use the set  $A = \{e \in E | a_e = 1\}$  of active edges instead of the indicator  $a_e$ .

**Definition 4.3. Energy optimization subproblem.** Let  $\mathcal{G} = (V, E^+ \cup E^-, W^+ \cup W^-)$  be an edge-weighted graph. Define the optimal solution of the subproblem as

$$S(\mathcal{G}, \tilde{A}) := \operatorname{argmin}_{A \subseteq (E \setminus \tilde{A})} T(A) \quad \text{with } T(A) := - \sum_{e \in A} w_e^p, \quad (5)$$

$$\text{s.t. } \mathcal{C}_0(A \cup \tilde{A}) = \mathcal{C}_1(A \cup \tilde{A}) = \emptyset, \quad (6)$$

where  $\tilde{A} \subseteq E$  is a set of initially activated edges such that  $\mathcal{C}_0(\tilde{A}) = \emptyset$  and  $\mathcal{C}_1(\tilde{A}) = \emptyset$ .

We note that for  $\tilde{A} = \emptyset$ , this optimization problem and its optimal solution  $S(\mathcal{G}, \emptyset)$  are equivalent to the Mutex Watershed Objective defined in Equation 3.

**Definition 4.4. Incomplete, feasible initial set:** For an edge-weighted graph  $\mathcal{G} = (V, E, W)$  a set of edges  $\tilde{A} \subseteq E$  is feasible if

$$\mathcal{C}_0(\tilde{A}) = \mathcal{C}_1(\tilde{A}) = \emptyset. \quad (7)$$

**Inputs:**

- weighted graph  $\mathcal{G}(V, E^+ \cup E^-, W^+ \cup W^-)$ ;
- initially activated set of edges  $\tilde{A}$  fulfilling  $\mathcal{C}_0(\tilde{A}) = \emptyset$  and  $\mathcal{C}_1(\tilde{A}) = \emptyset$ ;

**Output:** final set of activated edges  $A \subseteq E \setminus \tilde{A}$ ;

**Initialization:**  $A \leftarrow \emptyset$ ;

**for**  $e \in E \setminus \tilde{A}$  in descending order of weight **do**

if  $\mathcal{C}_0(A \cup \tilde{A} \cup \{e\}) = \emptyset$  and  $\mathcal{C}_1(A \cup \tilde{A} \cup \{e\}) = \emptyset$   
**then**  
 |  $A \leftarrow A \cup e$ ;  
**end**

**end**

**Algorithm 3:** Generalization of the Mutex Watershed algorithm defined in Algorithm 2. An initial set  $\tilde{A}$  of active edges is given as additional input and the final active set  $A \subseteq E \setminus \tilde{A}$ . Note that Algorithm 2 is a special case of this algorithm when  $\tilde{A} = \emptyset$ . Differences with Algorithm 2 are highlighted in blue.

$\tilde{A}$  is incomplete if it is not the final solution and there exists a feasible edge  $\tilde{e}$  that can be added to  $\tilde{A}$  without violating the constraints.

$$\exists \tilde{e} \in E \setminus \tilde{A} \quad \text{s.t.} \quad \mathcal{C}_0(\tilde{A} \cup \{\tilde{e}\}) = \mathcal{C}_1(\tilde{A} \cup \{\tilde{e}\}) = \emptyset \quad (8)$$

**Definition 4.5. First greedy step:** Let us consider an incomplete, feasible initial active set  $\tilde{A} \subseteq E$  on  $\mathcal{G} = (V, E, W)$ . We define

$$g := \operatorname{argmax}_{e \in (E \setminus \tilde{A})} w(e) \quad \text{s.t.} \quad \mathcal{C}_0(\tilde{A} \cup \{e\}) = \mathcal{C}_1(\tilde{A} \cup \{e\}) = \emptyset. \quad (9)$$

as the feasible edge with the highest weight, which is always the first greedy step of algorithm 3.

In the following two theorems, we prove that the Mutex Watershed problem has an *optimal substructure property* and a *greedy choice property* [53], which are sufficient to prove that the Mutex Watershed algorithm finds the optimum of the Mutex Watershed Objective.

**Theorem 4.2. Greedy-choice property.** For an incomplete, feasible initial active set  $\tilde{A}$  of the Mutex Watershed, the first greedy step  $g$  is always part of the optimal solution

$$g \in S(\mathcal{G}, \tilde{A}).$$

*Proof.* We will prove the theorem by contradiction by assuming that the first greedy choice is not part of the optimal solution, i.e.  $g \notin S(\mathcal{G}, \tilde{A})$ . Since  $g$  is by definition the feasible edge with highest weight, it follows that:

$$w(e) < w(g) \quad \forall e \in S(\mathcal{G}, \tilde{A}). \quad (10)$$

We now consider the alternative active set  $A' = \{g\}$ , that is a feasible solution, with

$$T(A') = -w_g^p \stackrel{(2)}{<} - \sum_{t \in S(\mathcal{G}, \tilde{A})} w_t^p = T(S(\mathcal{G}, \tilde{A})). \quad (11)$$

which contradicts the optimality of  $S(\mathcal{G}, \tilde{A})$ .  $\square$

**Theorem 4.3. Optimal substructure property.** Let us consider an initial active set  $\tilde{A}$ , the optimization problem defined in Equation 5, and assume to have an incomplete, feasible problem (see Definition 4.4). Then it follows that:

- 1) After making the first greedy choice  $g$ , we are left with a subproblem that can be seen as a new optimization problem of the same structure;
- 2) The optimal solution  $S(\mathcal{G}, \tilde{A})$  is always given by the combination of the first greedy choice and the optimal solution of the remaining subproblem.

*Proof.* After making the first greedy choice and selecting the first feasible edge  $g$  defined in Equation 9, we are clearly left with a new optimization problem of the same structure that has the following optimal solution:  $S(\mathcal{G}, \tilde{A} \cup \{g\})$ .

In order to prove the second point of the theorem, we now show that:

$$S(\mathcal{G}, \tilde{A}) = \{g\} \cup S(\mathcal{G}, \tilde{A} \cup \{g\}). \quad (12)$$

Since algorithm 3 fulfills the greedy-choice property,  $g \in S(\mathcal{G}, \tilde{A})$  and we can add the edge  $g$  as an additional constraint to the optimal solution:

$$\begin{aligned} S(\mathcal{G}, \tilde{A}) &= \operatorname{argmin}_{A \subseteq (E \setminus \tilde{A})} T(A) \\ \text{s. t. } &C_0(A \cup \tilde{A}) = C_1(A \cup \tilde{A}) = \emptyset; \quad g \in A \end{aligned} \quad (13)$$

Then it follows that:

$$\begin{aligned} S(\mathcal{G}, \tilde{A}) &= \{g\} \cup \operatorname{argmin}_{A \subseteq E \setminus (\tilde{A} \cup \{g\})} T(A) \\ \text{s. t. } &C_0(A \cup \{g\} \cup \tilde{A}) = C_1(A \cup \{g\} \cup \tilde{A}) = \emptyset \end{aligned} \quad (14)$$

which is equivalent to Equation 12.  $\square$

**Proof of Theorem 4.1.** In Theorems 4.2 and 4.3 we have proven that the optimization problem defined in 3 has the optimal substructure and a greedy choice property. It follows through induction that the final active set  $A^*$  found by the Mutex Watershed Algorithm 2 is the optimal solution for the Mutex Watershed objective (3) [53].  $\square$

### 4.3 Relation to multicut / correlation clustering

In this section, we show how the Mutex Watershed Objective (Def. 4.2) is related to the NP-hard multicut graph partition problem of [52], [54], [55]. The quick reader finds a summary in the last paragraph of this section.

Consider a graph  $\mathcal{G} = (V, E)$  with unique and finite weights  $\tilde{w}_e \in \mathbb{R}$ ,  $\forall e \in E$ . A large positive weight  $\tilde{w}_e$  is associated to a strong inclination of the two incident vertices to be assigned to the same cluster; on the other hand, a large negative weight  $\tilde{w}_e$  represents a great tendency of the incident vertices to be in different clusters. Small weights express (near) indifference. Solving the correlation clustering amounts to finding a graph multicut for which the sum of the weights of cut edges is minimal, which is equivalent to solving the following integer linear program for  $p = 1$ :

$$\min_{y \in \{0,1\}^{|E|}} \sum_{e \in E} \tilde{w}_e^p y_e \quad (15)$$

$$\text{s. t. } \forall c \in \text{Cycles in } G \quad \forall e^- \in c \quad \sum_{e \in c \setminus \{e^-\}} y_e \geq y_{e^-} \quad (16)$$

Optimizing this objective is NP-hard. Remarkably, we can derive a related objective that is optimally solved by the Mutex Watershed Algorithm, by choosing  $p$  sufficiently large.

In order to make the relation to the Mutex Watershed Objective explicit, let us separate the sum in the objective into repulsive

( $E^- := \{e \in E | \tilde{w}_e < 0\}$ ) and attractive edges ( $E^+ := \{e \in E | \tilde{w}_e \geq 0\}$ )

$$\sum_{e \in E^+} \tilde{w}_e^p y_e - \sum_{e \in E^-} |\tilde{w}_e|^p y_e, \quad (17)$$

subtract the constant sum of all positive edge weights

$$- \sum_{e \in E^+} \tilde{w}_e^p (1 - y_e) - \sum_{e \in E^-} |\tilde{w}_e|^p y_e \quad (18)$$

and substitute

$$a_e := \begin{cases} y_e, & \text{if } e \in E^- \\ 1 - y_e & \text{if } e \in E^+ \end{cases} \quad (19)$$

$$w_e := |\tilde{w}_e| \quad (20)$$

to obtain

$$- \sum_{e \in E} w_e^p a_e \quad (21)$$

Observe that for  $p = 1$  this objective is equivalent to the multicut, but for  $p$  large enough to become a dominant power, becomes the MWS energy (3).

Let us now consider the cycle inequalities (16). We find that the subset

$$\forall c \in \mathcal{C}_1 \quad \forall e^- \in c \cap E^- \quad \sum_{e \in c \setminus \{e^-\}} y_e \geq y_{e^-} \Leftrightarrow C_1(A) = \emptyset \quad (22)$$

of these inequalities over cycles from  $\mathcal{C}_1$  is equivalent to the MWS- $\mathcal{C}_1$  cycle constraints (for a detailed derivation see Appendix A.2). Therefore,

$$\min_{a \in \{0,1\}^{|E|}} - \sum_{e \in E} a_e w_e^p \quad (23)$$

$$\text{s. t. } C_1(A) = \emptyset \quad \text{with } A := \{e \in E | a_e = 1\}$$

is an integer linear program with multicut energy at  $p = 1$  and a subset of multicut constraints.

The Mutex Watershed objective differs from (23) only in the additional set of constraints  $C_0(A) = \emptyset$ . These are purely cosmetic and do not affect the clustering result, since only edges within a cluster are constrained. Their purpose is to restrict the solution to be a forest (a set of trees rather than a set of disconnected loopy graphs) w.r.t. the attractive edges which is motivated by the relation of MWS to minimum spanning trees. However, since the proof of Theorem 4.1 does not rely on  $C_0(A) = \emptyset$ , one can remove  $C_0(A \cup \{e\}) = \emptyset$  from Algorithm 2 and use identical arguments to prove that it yields the optimal solution to (23) for  $p$  large enough in the sense of Equation 2. This optimal solution is a consistent cut of  $G$  and thus lies within the tighter full set of multicut constraints. Therefore, it is also a solution to the modified multicut linear program (15-16) when  $p$  is chosen sufficiently large.

In summary, we make the following remarkable observation: in a graph with real (both positive and negative) edge weights in which the absolute weight of each edge is larger than the sum of absolute weights of all ‘‘lighter’’ edges (Equation 2), the normally NP-hard multicut problem becomes simpler and can be solved optimally using the greedy Mutex Watershed Algorithm.

## 5 EXPERIMENTS

We evaluate the Mutex Watershed on the challenging task of neuron segmentation in electron microscopy (EM) image volumes. This application is of key interest in connectomics, a field of neuroscience that strives to reconstruct neural wiring digrams spanning complete central nervous systems. The task requires segmentation of neurons from electron microscopy images of neural tissue – a challenging endeavor, since segmentation has to be based only on boundary information (cell membranes) and some of the boundaries are not very pronounced. Besides, cells contain membrane-bound organelles, which have to be suppressed in the segmentation. Some of the neuron protrusions are very thin, but all of those need to be preserved in the segmentation to arrive at the correct connectivity graph. While a lot of progress is being made, currently only manual tracing or proof-reading yields sufficient accuracy for correct circuit reconstruction [56].

We validate the Mutex Watershed algorithm on the most popular neural segmentation challenge: ISBI2012 [31]. We estimate the edge weights using a CNN as described in Section 5.1 and compare with other entries in the leaderboard as well as with other popular post-processing methods for the same network predictions in Section 5.2.

### 5.1 Estimating edge weights with a CNN

The common first step to EM segmentation is to predict which pixels belong to a cell membrane using a CNN. Different post-processing methods are then used to obtain a segmentation, see section 2 for an overview of such methods. The CNN can either be trained to predict boundary pixels [27], [30] or undirected affinities [28], [57] which express how likely it is for a pixel to belong to a different cell than its neighbors in the 6-neighborhood. In this case, the output of the network contains three channels, corresponding to left, down and next imaging plane neighbors in 3D. The affinities do not have to be limited to immediate neighbors – in fact, [28] have shown that introduction of long-range affinities is beneficial for the final segmentation even if they are only used to train the network. Building on the work of [28], we train a CNN to predict short- and long-range affinities and then use those directly as weights for the Mutex Watershed algorithm.

We estimate the affinities / edge weights for the neighborhood structure shown in Figure 5. To that end, we define local attractive and long-range repulsive edges. When attractive edges are only short-range, the solution will consist of spatially connected segments that cannot comprise “air bridges”. This holds true for both (lifted) multicut and for Mutex Watershed. We use a different pattern for in-plane and between-plane edges due to the great anisotropy of the data set. In more detail, we pick a sparse ring of in-plane repulsive edges and additional longer-range in-plane edges which are necessary to split regions reliably (see Figure 5a). We also added connections to the indirect neighbors in the lower adjacent slice to ensure correct 3D connectivity (see Figure 5b). In our experiments, we pick a subset of repulsive edges, by using strides of 2 in the XY-plane in order to avoid artifacts caused by occasional very thick membranes. Note that the stride is not applied to local (attractive) edges, but only to long-range (repulsive) edges. The particular pattern used was selected after inspecting the size of typical regions. The specific pattern is the only one we have tried and was *not* optimized over.

In total,  $C^+$  attractive and  $C^-$  repulsive edges are defined for each pixel, resulting in  $C^+ + C^-$  output channels in the network.

We partition the set of attractive / repulsive edges into subsets  $H^+$  and  $H^-$  that contain all edges at a specific offset:  $E^+ = \bigcup_{c=1}^{C^+} H_c^+$  for attractive edges, with  $H^-$  defined analogously. Each element of the subsets  $H_c^+$  and  $H_c^-$  corresponds to a specific channel predicted by the network. We further assume that weights take values in  $[0, 1]$ .

### Network architecture and training

We use the 3D U-Net [32], [58] architecture, as proposed in [57].

Our training targets for attractive / repulsive edges  $w^{\pm}$  can be derived from a groundtruth label image  $L$  according to

$$w_{e=(i,j)}^+ = \begin{cases} 1, & \text{if } L_i = L_j \\ 0, & \text{otherwise} \end{cases} \quad (24)$$

$$w_{e=(i,j)}^- = \begin{cases} 0, & \text{if } L_i = L_j \\ 1, & \text{otherwise} \end{cases} \quad (25)$$

Here,  $i$  and  $j$  are the indices of vertices / image pixels. Next, we define the loss terms

$$\mathcal{J}_c^+ = - \frac{\sum_{e \in H_c^+} (1 - w_e^+) (1 - w_e^{*+})}{\sum_{e \in H_c^+} ((1 - w_e^+)^2 + (1 - w_e^{*+})^2)} \quad (26)$$

$$\mathcal{J}_c^- = - \frac{\sum_{e \in H_c^-} w_e^- w_e^{*-}}{\sum_{e \in H_c^-} ((w_e^-)^2 + (w_e^{*-})^2)} \quad (27)$$

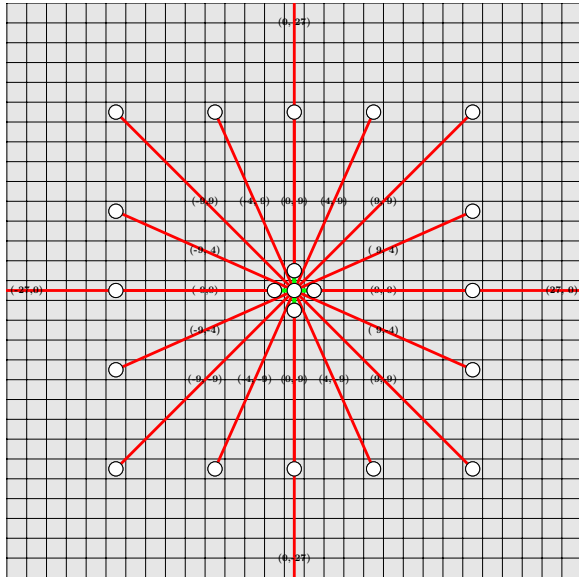
for attractive edges (i.e. channels) and repulsive edges (i.e. channels).

Equation 26 is the Sørensen-Dice coefficient [59], [60] formulated for fuzzy set membership values. During training we minimize the sum of attractive and repulsive loss terms  $\mathcal{J} = \sum_c^{C^+} \mathcal{J}_c^+ + \sum_c^{C^-} \mathcal{J}_c^-$ . This corresponds to summing up the channel-wise Sørensen-Dice loss. The terms of this loss are robust against prediction and / or target sparsity, a desirable quality for neuron segmentation: since membranes are locally two-dimensional and thin, they occupy very few pixels in three-dimensional the volume. More precisely, if  $w_e^+$  or  $w_e^{*+}$  (or both) are sparse, we can expect the denominator  $\sum_e ((w_e^+)^2 + (w_e^{*+})^2)$  to be small, which has the effect that the numerator is adaptively weighted higher. In this sense, the Sørensen-Dice loss at every pixel  $i$  is conditioned on the global image statistics, which is not the case for a Hamming-distance based loss like Binary Cross-Entropy or Mean Squared Error.

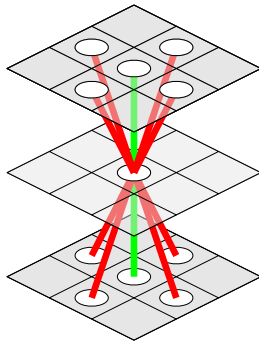
We optimize this loss using the Adam optimizer [61] and additionally condition learning rate decay on the Adapted Rand Score [31] computed on the training set every 100 iterations. During training, we augment the data set by performing in-plane rotations by multiples of 90 degrees, flips along the X- and Y-axis as well as elastic deformations. At prediction time, we use test time data augmentation, presenting the network with seven different versions of the input obtained by a combination of rotations by a multiple of 90 degrees, axis-aligned flips and transpositions. The network predictions are then inverse-transformed to correspond to the original image, and the results averaged.

### 5.2 ISBI Challenge

The ISBI 2012 EM Segmentation Challenge [31] is the neuron segmentation challenge with the largest number of competing entries. The challenge data contains two volumes of dimensions  $1.5 \times 2 \times 2$  microns and has a resolution of  $50 \times 4 \times 4$  nm per pixel.



(a) XY-plane neighborhood with local attractive edges (green) and sparse repulsive edges (red) with approximate radius 9 and further long-range connections with distance 27



(b) Due to the great anisotropy of the data we limit the Z-plane edges to a distance of 1. The direct neighbors are attractive, whereas the indirect neighbors are repulsive.

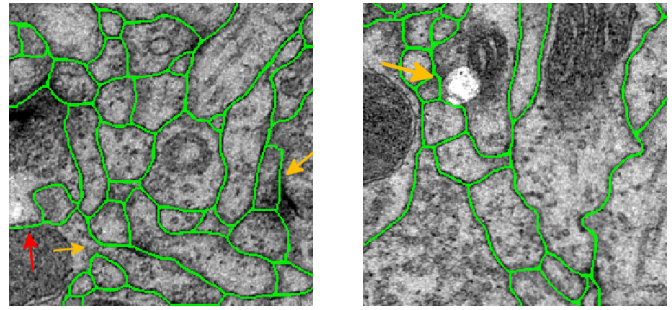
Fig. 5: Local neighborhood structure of attractive (green) and repulsive (red) edges in the Mutex Watershed graph.

The groundtruth is provided as binary membrane labels, which can easily be converted to a 2D, but not 3D segmentation. To train a 3D model, we follow the procedure described in [27].

The test volume has private groundtruth; results can be submitted to the leaderboard. They are evaluated based on the Adapted Rand Score (Rand-Score) and the Variation of Information Score (VI-Score) [31].

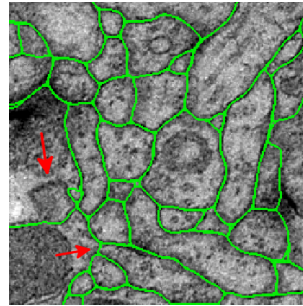
Our method holds the top entry in the challenge’s leaderboard<sup>1</sup> at the time of submission, see Table 1a. This is especially remarkable insofar as it is simpler than the methods holding the other top entries. Three out of four rely on a CNN to predict boundary locations and postprocess its output with the complex pipeline described in [27]. This post-processing first generates superpixels via distance transform watersheds. Then it computes a merge cost for local and long-range connections between superpixels. Based on this, it defines a lifted multicut partitioning problem that is solved approximately. In contrast, our

1. [http://brainiac2.mit.edu/isbi\\_challenge/leaders-board-new](http://brainiac2.mit.edu/isbi_challenge/leaders-board-new)



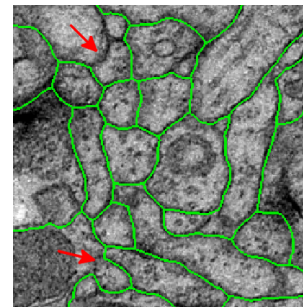
(a) Mutex Watershed

(b) Mutex Watershed

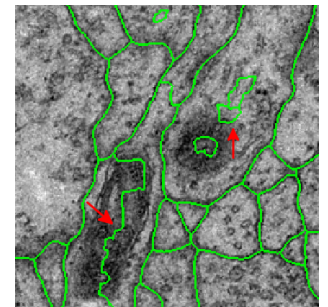


(c) Multicut partitioning based segmentation (MC-FULL)

(d) Thresholding of local boundary maps (THRESH)



(e) Watershed, seeded at local minima of the smoothed input map (WS)



(f) Distance Transform Watershed (WSDT)

Fig. 6: Mutex Watershed and baseline segmentation algorithms applied on the ISBI Challenge test data. Red arrows point out major errors. Orange arrows point to difficult, but correctly segmented regions. All methods share the same input maps.

method operates purely on the pixel level and does not involve a NP-hard partitioning step.

### Comparison with other segmentation methods

The weights predicted by the CNN described above can be post-processed directly by the Mutex Watershed algorithm. To ensure a fair comparison, we transform the same CNN predictions into a segmentation using basic and state-of-the-art post-processing methods. We start from simple thresholding (THRESH) and seeded watershed. Since these cannot take long-range repulsions into account, we generate a boundary map by taking the maximum<sup>2</sup> values over the attractive edge channels. Based on this boundary map, we introduce seeds at the local minima (WS) and at the maxima of the smoothed distance transform (WSDT). For both variants, the degree of smoothing was optimized such that each

2. The maximum is chosen to preserve boundaries.



region receives as few seeds as possible, without however causing severe under-segmentation. The performance of these three baseline methods in comparison to Mutex Watershed is summarized in Table 1b. The methods were applied only in 2D, because the high degree of anisotropy leads to inferior results when applied in 3D. In contrast, the Mutex Watershed can be applied in 3D out of the box and yields significantly better 2D segmentation scores.

Qualitatively, we show patches of results in Figure 6. The major failure case for WS (Figure 6e) and WSDT (Figure 6f) is over-segmentation caused by over-seeding a region. The major failure case for THRESH is under-segmentation due to weak boundary evidence (see Figure 6d). In contrast, the Mutex Watershed produces a better segmentation, only causing minor over-segmentation (see Figure 6a, Figure 6b).

Note that, in contrast to most pixel-based postprocessing methods, our algorithm can take long range predictions into account. To compare with methods which share this property, we turn to the multicut and lifted multicut-based partitioning for neuron segmentations as introduced in [18] and [25]. As proposed in [62], we compute costs corresponding to edge cuts from the affinities estimated by the CNN via:

$$s_e = \begin{cases} \log \frac{w_e^+}{1-w_e^+}, & \text{if } e \in E^+ \\ \log \frac{1-w_e^-}{w_e^-}, & \text{otherwise,} \end{cases} \quad (28)$$

We set up two multicut problems: the first is induced only by the short-range edges (MC-LOCAL), the other by short- and long-range edges together (MC-FULL). Note that the solution to the full connectivity problem can contain “air bridges”, i.e. pixels that are connected only by long-range edges, without a path along the local edges connecting them. However, we found this not to be a problem in practice. In addition, we set up a lifted multicut (LMC) problem from the same edge costs.

Both problems are NP-hard, hence it is not feasible to solve them exactly on large grid graphs. For our experiments, we use the approximate Kernighan Lin [22], [63] solver. Even this allows us to only solve individual 2D problems at a time. The results for MC-LOCAL and MC-FULL can be found in Table 1b. The MC-LOCAL approach scores poorly because it under-segments heavily. This observation emphasizes the importance of incorporating the longer-range edges. The MC-FULL and LMC approaches perform well. Somewhat surprisingly, the Mutex Watershed yields a better segmentation still, despite being much cheaper in inference. We note that both MC-FULL, LMC and the Mutex Watershed are evaluated on the same long-range affinity maps (i.e. generated by the same CNN with the same set of weights).

## 6 CONCLUSION AND DISCUSSION

We have presented a fast algorithm for the clustering of graphs with both attractive and repulsive edges. The ability to consider both obviates the need for the kind of stopping criterion or even seeds that all popular algorithms except for multicut / correlation clustering need. The proposed method has low computational complexity in imitation of its close relative, Kruskal’s algorithm. We have shown which objective this algorithm optimizes exactly, and that this objective emerges as a specific case of the multicut objective. It is possible that recent interesting work [67] on partial optimal solutions may open an avenue for an alternative proof.

Finally, we have found that the proposed algorithm, when presented with informative edge costs from a good neural network,

Method	Rand-Score	VI-Score
UNet + MWS	<b>0.98792</b>	<b>0.99183</b>
ResNet + LMC [64]	0.98788	0.99072
SCN + LMC [65]	0.98680	0.99144
M2FCN-MFA [66]	0.98383	0.98981
FusionNet + LMC [33]	0.98365	0.99130

(a) Top five entries at time of submission. Our Mutex Watershed (MWS) is state-of-the-art without relying on the complex lifted multicut postprocessing used by most other top entries.

Method	Rand-Score	VI-Score	Time [s]
MWS	<b>0.98792</b>	<b>0.99183</b>	43.3
MC-FULL	0.98029	0.99044	9415.8
LMC	0.97990	0.99007	966.0
THRESH	0.91435	0.96961	0.2
WSDT	0.88336	0.96312	4.4
MC-LOCAL	0.70990	0.86874	1410.7
WS	0.63958	0.89237	4.9

(b) Comparison to other segmentation strategies, all of which are based on our CNN.

TABLE 1: Results on the ISBI 2012 EM Segmentation Challenge.

outperforms all known methods on a competitive bioimage partitioning benchmark, including methods that operate on the very same network predictions.

## 7 ACKNOWLEDGMENTS

This work was partially supported by the grants DFG HA 4364/8-1, DFG SFB 1129 from the Deutsche Forschungsgemeinschaft and the Baden-Württemberg Stiftung Elite PostDoc Program.

## REFERENCES

- [1] S. Wolf, C. Pape, A. Bailoni, N. Rahaman, A. Kreshuk, U. Köthe, and F. Hamprecht, “The mutex watershed: Efficient, parameter-free image partitioning,” *Proc. ECCV’18*, 2018.
- [2] L. Vincent and P. Soille, “Watersheds in digital spaces: an efficient algorithm based on immersion simulations,” *IEEE Trans. Pattern Analysis Machine Intelligence*, no. 6, pp. 583–598, 1991.
- [3] S. Beucher and F. Meyer, “The morphological approach to segmentation: the watershed transformation,” *Optical Engineering*, vol. 34, pp. 433–433, 1992.
- [4] M. Grimaud, “New measure of contrast: the dynamics,” in *Proc. Image Algebra and Morphological Processing*, ser. SPIE Conf. Series, P. D. Gader, E. R. Dougherty, & J. C. Serra, Ed., vol. 1769, 1992, pp. 292–305.
- [5] S. Beucher, “Watershed, hierarchical segmentation and waterfall algorithm,” in *Proc. ISMM’94*, vol. 94, 1994, pp. 69–76.
- [6] C. Vachier and F. Meyer, “Extinction value: a new measurement of persistence,” in *Worksh. Nonlinear Signal and Image Processing*, vol. 1, 1995, pp. 254–257.
- [7] L. Najman and M. Schmitt, “Geodesic saliency of watershed contours and hierarchical segmentation,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 18, no. 12, pp. 1163–1173, 1996.
- [8] P. Soille, “Constrained connectivity for hierarchical image decomposition and simplification,” *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 30, no. 7, pp. 1132–1145, 2008.
- [9] B. Perret, J. Cousty, S. J. F. Guimaraes, and D. S. Maia, “Evaluation of hierarchical watersheds,” *IEEE Transactions on Image Processing*, vol. 27, no. 4, pp. 1676–1688, 2018.
- [10] F. Meyer, “Morphological multiscale and interactive segmentation,” in *NSIP*, 1999, pp. 369–377.
- [11] L. Najman, “On the equivalence between hierarchical segmentations and ultrametric watersheds,” *J. of Mathematical Imaging and Vision*, vol. 40, no. 3, pp. 231–247, 2011.
- [12] P. Salembier and L. Garrido, “Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval,” *IEEE Trans. Image Proc.*, vol. 9, pp. 561–576, 2000.

- [13] F. Malmberg, R. Strand, and I. Nyström, “Generalized hard constraints for graph segmentation,” in *Scandinavian Conference on Image Analysis*. Springer, 2011, pp. 36–47.
- [14] P. F. Felzenszwalb and D. P. Huttenlocher, “Efficient Graph-Based Image Segmentation,” *Int. J. Comput. Vision*, vol. 59, no. 2, pp. 167–181, 2004.
- [15] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, “Contour detection and hierarchical image segmentation,” *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 33, no. 5, pp. 898–916, 2011.
- [16] B. R. Kiran and J. Serra, “Global–local optimizations by hierarchical cuts and climbing energies,” *Pattern Recognition*, vol. 47, no. 1, pp. 12–24, 2014.
- [17] M. Keuper, S. Tang, Y. Zhongjie, B. Andres, T. Brox, and B. Schiele, “A multi-cut formulation for joint segmentation and tracking of multiple objects,” *arXiv preprint arXiv:1607.06317*, 2016.
- [18] B. Andres, T. Kröger, K. L. Briggmann, W. Denk, N. Norogod, G. Knott, U. Köthe, and F. A. Hamprecht, “Globally optimal closed-surface segmentation for connectomics,” in *Proc. ECCV’12, part 2*, no. 7574, 2012, pp. 778–791.
- [19] J. Yarkony, A. Ihler, and C. C. Fowlkes, “Fast planar correlation clustering for image segmentation,” in *Proc. ECCV’12*, 2012, pp. 568–581.
- [20] C. Pape, T. Beier, P. Li, V. Jain, D. D. Bock, and A. Kreshuk, “Solving large multicut problems for connectomics via domain decomposition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1–10.
- [21] Z. Levi and D. Zorin, “Strict minimizers for geometric optimization,” *ACM Trans. Graph.*, vol. 33, no. 6, pp. 185:1–185:14, Nov. 2014.
- [22] M. Keuper, E. Levinkov, N. Bonneel, G. Lavoué, T. Brox, and B. Andres, “Efficient decomposition of image and mesh graphs by lifted multicuts,” in *Proc. ICCV’15*, 2015, pp. 1751–1759.
- [23] E. Levinkov, A. Kirillov, and B. Andres, “A comparative study of local search algorithms for correlation clustering,” in *German Conference on Pattern Recognition*. Springer, 2017, pp. 103–114.
- [24] C. Zhang, J. Yarkony, and F. A. Hamprecht, “Cell detection and segmentation using correlation clustering,” in *Proc. MICCAI’14*, 2014, pp. 9–16.
- [25] A. Horňáková, J.-H. Lange, and B. Andres, “Analysis and optimization of graph decompositions by lifted multicuts,” in *International Conference on Machine Learning*, 2017, pp. 1539–1548.
- [26] T. Beier, B. Andres, U. Köthe, and F. A. Hamprecht, “An efficient fusion move algorithm for the minimum cost lifted multicut problem,” in *European Conference on Computer Vision*. Springer, 2016, pp. 715–730.
- [27] T. Beier, C. Pape, N. Rahaman, and T. e. a. Prange, “Multicut brings automated neurite segmentation closer to human performance,” *Nature Methods*, vol. 14, no. 2, pp. 101–102, 2017.
- [28] K. Lee, J. Zung, P. Li, V. Jain, and H. S. Seung, “Superhuman accuracy on the SNEMI3D connectomics challenge,” *arXiv preprint arXiv:1706.00120*, 2017.
- [29] V. Jain, J. F. Murray, F. Roth, S. Turaga, V. Zhigulin, K. L. Briggman, M. N. Helmstaedter, W. Denk, and H. S. Seung, “Supervised learning of image restoration with convolutional networks,” *Proc. ICCV’07*, pp. 1–8, 2007.
- [30] D. C. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber, “Deep neural networks segment neuronal membranes in electron microscopy images,” *Proc. NIPS’12*, 2012.
- [31] I. Arganda-Carreras, S. Turaga, D. Berger *et al.*, “Crowdsourcing the creation of image segmentation algorithms for connectomics,” *Front. Neuroanatomy*, vol. 9, p. 142, 2015.
- [32] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional networks for biomedical image segmentation,” *Proc. MICCAI’15*, pp. 234–241, 2015.
- [33] T. M. Quan, D. G. Hilderbrand, and W.-K. Jeong, “FusionNet: a deep fully residual convolutional neural network for image segmentation in connectomics,” *arXiv:1612.05360*, 2016.
- [34] J. Nunez-Iglesias, R. Kennedy, T. Parag, J. Shi, and D. Chklovskii, “Machine learning of hierarchical clustering to segment 2D and 3D images,” *PLoS one*, vol. 8, p. e71715, 2013.
- [35] S. Knowles-Barley, V. Kaynig, T. R. Jones, A. Wilson, J. Morgan, D. Lee, D. Berger, N. Kasthuri, J. W. Lichtman, and H. Pfister, “RhoanaNet pipeline: Dense automatic neural annotation,” *arXiv:1611.06973*, 2016.
- [36] M. G. Uzunbaş, C. Chen, and D. Metaxas, “Optree: a learning-based adaptive watershed algorithm for neuron segmentation,” in *Int. Conf. Medical Image Computing and Computer-Assisted Intervention (MICCAI’14)*, 2014, pp. 97–105.
- [37] Y. Meirovitch, A. Matveev, H. Saribekyan, D. Budden, D. Rolnick, G. Odor, S. K.-B. T. R. Jones, H. Pfister, J. W. Lichtman, and N. Shavit, “A multi-pass approach to large-scale connectomics,” *arXiv preprint:1612.02120*, 2016.
- [38] M. Januszewski, J. Kornfeld, P. H. Li, A. Pope, T. Blakely, L. Lindsey, J. Maitin-Shepard, M. Tyka, W. Denk, and V. Jain, “High-precision automated reconstruction of neurons with flood-filling networks,” *Nature methods*, p. 1, 2018.
- [39] A. Zlateski and H. S. Seung, “Image segmentation by size-dependent single linkage clustering of a watershed basin graph,” *arXiv:1505.00249*, 2015.
- [40] T. Parag, F. Tschopp, W. Grisaitis, S. C. Turaga, X. Zhang, B. Matejek, L. Kamentsky, J. W. Lichtman, and H. Pfister, “Anisotropic EM segmentation by 3d affinity learning and agglomeration,” *arXiv preprint 1707.08935*, 2017.
- [41] S. C. Turaga, J. F. Murray, V. Jain, F. Roth, M. Helmstaedter, K. Briggman, W. Denk, and H. S. Seung, “Convolutional networks can learn to generate affinity graphs for image segmentation,” *Neural Computation*, vol. 22, no. 2, pp. 511–538, 2010.
- [42] K. Briggman, W. Denk, S. Seung, M. N. Helmstaedter, and S. C. Turaga, “Maximin affinity learning of image segmentation,” in *Advances in Neural Information Processing Systems*, 2009, pp. 1865–1873.
- [43] S. Wolf, L. Schott, U. Köthe, and F. Hamprecht, “Learned watershed: End-to-end learning of seeded segmentation,” *Proc. ICCV’17*, 2017.
- [44] M. Bai and R. Urtasun, “Deep watershed transform for instance segmentation,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017, pp. 2858–2866.
- [45] S. Xie and Z. Tu, “Holistically-nested edge detection,” in *Proc. ICCV’15*, 2015, pp. 1395–1403.
- [46] I. Kokkinos, “Pushing the boundaries of boundary detection using deep learning,” *arXiv:1511.07386*, 2015.
- [47] J. Cai, L. Lu, Z. Zhang, F. Xing, L. Yang, and Q. Yin, “Pancreas segmentation in MRI using graph-based decision fusion on convolutional neural networks,” in *Proc. MICCAI*, 2016.
- [48] F. Meyer, “Topographic distance and watershed lines,” *Signal processing*, vol. 38, no. 1, pp. 113–125, 1994.
- [49] —, “Minimum spanning forests for morphological segmentation,” in *Mathematical morphology and its applications to image processing*, 1994, pp. 77–84.
- [50] A. X. Falcão, J. Stolfi, and R. de Alencar Lotufo, “The image foresting transform: Theory, algorithms, and applications,” *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 26, no. 1, pp. 19–29, 2004.
- [51] J. Cousty, G. Bertrand, L. Najman, and M. Couprie, “Watershed cuts: Minimum spanning forests and the drop of water principle,” *IEEE Trans. Patt. Anal. Mach. Intell.*, 2009.
- [52] J. H. Kappes, M. Speth, B. Andres, G. Reinelt, and C. Schn, “Globally optimal image partitioning by multicuts,” in *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*. Springer, 2011, pp. 31–44.
- [53] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Third Edition*, 3rd ed. The MIT Press, 2009.
- [54] S. Chopra and M. R. Rao, “On the multiway cut polyhedron,” *Networks*, vol. 21, no. 1, pp. 51–89, 1991.
- [55] B. Andres, “Lifting of multicuts,” *CoRR*, vol. abs/1503.03791, 2015.
- [56] P. Schlegel, M. Costa, and G. S. X. E. Jefferis, “Learning from connectomics on the fly,” *Current opinion in insect science*, vol. 24, pp. 96–105, 2017.
- [57] J. Funke, F. D. Tschopp, W. Grisaitis, A. Sheridan, C. Singh, S. Saalfeld, and S. C. Turaga, “Large scale image segmentation with structured loss based deep learning for connectome reconstruction,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.
- [58] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, “3D U-Net: learning dense volumetric segmentation from sparse annotation,” in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2016, pp. 424–432.
- [59] L. R. Dice, “Measures of the amount of ecologic association between species,” *Ecology*, vol. 26, no. 3, pp. 297–302, 1945.
- [60] T. Sørensen, “A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on danish commons,” *Biol. Skr.*, vol. 5, pp. 1–34, 1948.
- [61] D. Kinga and J. B. Adam, “A method for stochastic optimization,” in *International Conference on Learning Representations (ICLR)*, vol. 5, 2015.
- [62] B. Andres, T. Kroeger, K. L. Briggman, W. Denk, N. Korogod, G. Knott, U. Koethe, and F. A. Hamprecht, “Globally optimal closed-surface segmentation for connectomics,” in *European Conference on Computer Vision*. Springer, 2012, pp. 778–791.
- [63] B. W. Kernighan and S. Lin, “An efficient heuristic procedure for partitioning graphs,” *The Bell system technical journal*, vol. 49, no. 2, pp. 291–307, 1970.

- [64] C. Xiao, J. Liu, X. Chen, H. Han, C. Shu, and Q. Xie, “Deep contextual residual network for electron microscopy image segmentation in connectomics,” in *Biomedical Imaging (ISBI 2018), 2018 IEEE 15th International Symposium on*. IEEE, 2018, pp. 378–381.
- [65] M. Weiler, F. A. Hamprecht, and M. Storath, “Learning steerable filters for rotation equivariant CNNs,” 2018, pp. 849–858.
- [66] W. Shen, B. Wang, Y. Jiang, Y. Wang, and A. L. Yuille, “Multi-stage multi-recursive-input fully convolutional networks for neuronal boundary detection,” *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2410–2419, 2017.
- [67] J.-H. Lange, A. Karrenbauer, and B. Andres, “Partial optimality and fast lower bounds for weighted correlation clustering,” in *International Conference on Machine Learning*, 2018, pp. 2898–2907.

## APPENDIX A EFFICIENT IMPLEMENTATION OF THE MUTEX WATERSHED ALGORITHM

In this section we propose an efficient implementation of the MWS, derive its theoretical average runtime complexity and show that in practice the MWS scales with  $\mathcal{O}(E \log E)$ .

### Notation:

Let us explicitly define clusters using the “connected” predicate  $\forall i, j \in V$ :

$$\begin{aligned} \text{connected}(i, j|A) &:= \mathbb{I}\{\exists \text{ path } \pi \subseteq A^+ \text{ from } i \text{ to } j\} \\ \text{cluster}(i|A) &= \{i\} \cup \{j : \text{connected}(i, j|A)\} \end{aligned}$$

Conversely, the active subset  $A^- \subseteq E^-$  of repulsive edges defines mutual exclusion relations by using the following predicate:

$$\text{mutex}(i, j|A) := \mathbb{I}\{\exists \text{ path } \pi \subseteq A \text{ from } i \text{ to } j \text{ s.t. } |\pi \cap A^-| = 1\}$$

To implement the MWS efficiently we reformulate the algorithm in terms of the predicates connected and mutex using the following equalities (for an arbitrary set  $A \subseteq E$  with  $C_{0/1}(A) = \emptyset$ )

$$\begin{aligned} \forall (i, j) \in E^+ \setminus A : \text{connected}(i, j|A) &\Leftrightarrow C_0(A \cup \{(i, j)\}) \neq \emptyset \\ \text{mutex}(i, j|A) &\Leftrightarrow C_1(A \cup \{(i, j)\}) \neq \emptyset \\ \forall (i, j) \in E^- \setminus A : \text{connected}(i, j|A) &\Leftrightarrow C_1(A \cup \{(i, j)\}) \neq \emptyset \end{aligned}$$

to rewrite the Mutex Watershed Algorithm as Algorithm 4. In the following analysis we omit the dependence on  $A$  in all predicates to improve readability.

### A.1 Time Complexity Analysis

Before analyzing the time complexity of algorithm 4 we first review the complexity of Kruskal’s algorithm. Using a union-find data structure (with path compression and union by rank) the time complexity of  $\text{merge}(i, j)$  and  $\text{connected}(i, j)$  is  $\mathcal{O}(\alpha(V))$ , where  $\alpha$  is the slowly growing inverse Ackerman function, and the total runtime complexity is dominated by the initial sorting of the edges  $\mathcal{O}(E \log E)$  [53].

To check for mutex constraints efficiently, we maintain a set of all active mutex edges

$$M[C_i] = \{(u, v) \in A^- | u \in C_i \vee v \in C_i\}$$

for every  $C_i = \text{cluster}(i)$  using hash tables, where insertion of new mutex edges (i.e.  $\text{addmutex}$ ) and search have an average complexity of  $\mathcal{O}(1)$ . Note that every cluster can be efficiently identified by its union-find root node. For  $\text{mutex}(i, j)$  we check if  $M[C_i] \cap M[C_j] = \emptyset$  by searching for all elements of the smaller hash table in the larger hash table. Therefore  $\text{mutex}(i, j)$

---

**Input:** weighted graph  $\mathcal{G}(V, E^+ \cup E^-, W^+ \cup W^-)$ ;  
**Output:** clusters defined by active set  $A^+$ ;  
**Initialization:**  $A^+ = \emptyset$ ;  $A^- = \emptyset$ ;  
**for**  $(i, j) = e \in (E^+ \cup E^-)$  in descending order of  $W^+ \cup W^-$  **do**  
  **if**  $e \in E^+$  **then**  
    **if not**  $\text{connected}(i, j)$  **and not**  $\text{mutex}(i, j)$  **then**  
       $\text{merge}(i, j): A^+ \leftarrow A^+ \cup e$ ;  
      ▷  $\text{merge } i \text{ and } j \text{ and inherit the mutex constraints of the parent clusters}$   
    **end**  
  **else**  
    **if not**  $\text{connected}(i, j)$  **then**  
       $\text{addmutex}(i, j): A^- \leftarrow A^- \cup e$ ;  
      ▷  $\text{add mutex constraint between } i \text{ and } j$   
    **end**  
  **end**  
**end**

---

**Algorithm 4:** Efficient implementation of Mutex Watershed. The connected predicate can be efficiently evaluated.

has an average complexity of  $\mathcal{O}(\min(|M[C_i]|, |M[C_j]|))$ . Similarly, during  $\text{merge}(i, j)$ , mutex constraints are inherited by merging two hash tables, which also has an average complexity  $\mathcal{O}(\min(|M[C_i]|, |M[C_j]|))$ .

In conclusion, the average runtime contribution of attractive edges  $\mathcal{O}(|E^+| \cdot \alpha(V) + |E^+| \cdot M)$  (checking mutex constraints and possibly merging) and repulsive edges  $\mathcal{O}(|E^-| \cdot \alpha(V) + |E^-|)$  (insertion of one mutex edge) result in a total average runtime complexity of algorithm 4:

$$\mathcal{O}(E \log E + E \cdot \alpha(V) + EM). \quad (29)$$

where  $M$  is the expected value of  $\min(|M[C_i]|, |M[C_j]|)$ . Using  $\alpha(V) \in \mathcal{O}(\log V) \in \mathcal{O}(\log E)$  this simplifies to<sup>3</sup>

$$\mathcal{O}(E \log E + EM). \quad (30)$$

In the worst case  $\mathcal{O}(M) \in \mathcal{O}(E)$ , the Mutex Watershed Algorithm has a runtime complexity of  $\mathcal{O}(E^2)$ . Empirically, we find that  $\mathcal{O}(EM) \approx \mathcal{O}(E \log E)$  by measuring the runtime of Mutex Watershed for different sub-volumes of the ISBI challenge (see Figure 7), leading to a

$$\text{Empirical Mutex Watershed Complexity: } \mathcal{O}(E \log E) \quad (31)$$

### A.2 Multicut Cycle Inequalities

Here we prove equation (22)

$$\forall c \in \mathcal{C}_1 \quad \forall e^- \in c \cap E^- \quad \sum_{e \in c \setminus \{e^-\}} y_e \geq y_{e^-} \quad \Leftrightarrow \quad \mathcal{C}_1(A) = \emptyset.$$

Using the indicator

$$a_e := \begin{cases} y_e, & \text{if } e \in E^- \\ 1 - y_e & \text{if } e \in E^+ \end{cases}$$

we can decompose the r.h.s. of equation (22) into

$$\mathcal{C}_1(A) = \emptyset \quad \Leftrightarrow \quad \sum_{e \in c} a_e < |c| \quad \forall c \in \mathcal{C}_1 \quad (32)$$

3. In the worst case  $G$  is a fully connected graph, with  $|E| = |V|^2$ , hence  $\log |V| = \frac{1}{2} \log |E|$  for graphs without parallel edges.

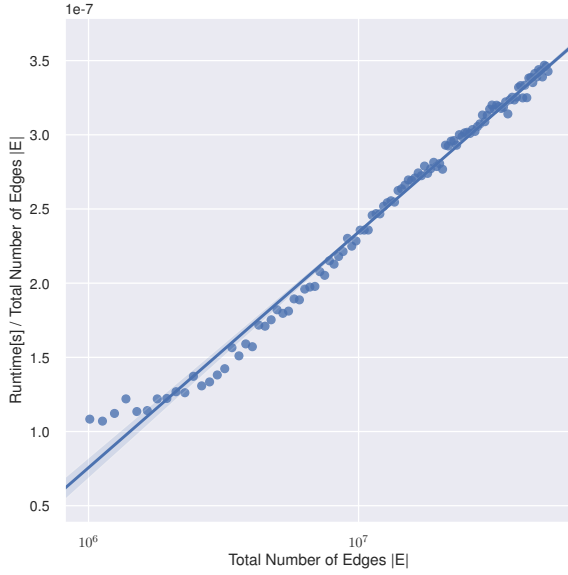


Fig. 7: Runtime  $T$  of Mutex Watershed (without sorting of edges) measured on sub-volumes of the ISBI challenge of different sizes (thereby varying the total number of edges  $E$ ). We plot  $\frac{T}{|E|}$  over  $|E|$  in a logarithmic plot, which makes  $T \sim |E| \log(|E|)$  appear as straight line. A logarithmic function (blue line) is fitted to the measured  $\frac{T}{|E|}$  (blue circles) with ( $R^2 = 0.9896$ ). The good fit suggests that empirically  $T \approx \mathcal{O}(E \log E)$ .

and show the equivalence for any cycle  $c \in \mathcal{C}_1$  with repulsive edge  $e^- \in c \cap E^-$ :

$$\begin{aligned}
 \sum_{e \in c} a_e < |c| &\Leftrightarrow \sum_{e \in c} a_e \leq |c| - 1 \\
 &\Leftrightarrow \sum_{e \in c} (a_e - 1) \leq -1 \\
 &\Leftrightarrow \sum_{e \in c} (1 - a_e) - 1 \geq 0 \\
 &\Leftrightarrow \sum_{e \in c \setminus \{e^-\}} (1 - a_e) \geq a_{e^-} \\
 &\Leftrightarrow \sum_{e \in c \setminus \{e^-\}} y_e \geq y_{e^-}
 \end{aligned}$$