K-smallest spanning tree segmentations

C. Straehle, S. Peter, U. Köthe, F. A. Hamprecht

HCI, University of Heidelberg

Abstract. Real-world images often admit many different segmentations that have nearly the same quality according to the underlying energy function. The diversity of these solutions may be a powerful uncertainty indicator. We provide the crucial prerequisite in the context of seeded segmentation with minimum spanning trees (i.e. edge-weighted watersheds). Specifically, we show how to efficiently enumerate the k smallest spanning trees that result in *different* segmentations; and we prove that solutions are indeed found in the correct order. Experiments show that about half of the trees considered by our algorithm represent unique segmentations. This redundancy is orders of magnitude lower than can be achieved by just enumerating the k-smallest MSTs, making the algorithm viable in practice.

1 Introduction

The most popular algorithms for interactive segmentation are Ising type Markov random fields (MRFs) [3], the random walker [11] and the seeded watershed [13]. Their smoothness terms penalize label changes with the L_1,L_2 and L_∞ norms respectively [6]. However, in the process of finding the single lowest energy solution to the graph partitioning problem a lot of information is lost and other modes of the solution space which may convey important aspects of the problem are ignored. An enumeration of more than one low energy solution allows to obtain a more robust segmentation, and can help defining an uncertainty of the resulting segmentation which may be used in downstream processing. Thus, recent work on these algorithms has focused on finding the M lowest energy solutions [9, 14, 20], ideally subject to a diversity constraint [1]. These references solve the problem for Ising-type MRFs. However, systematic empirical studies [17] show that the seeded watershed outperforms MRFs in certain datasets and offers computational advantages. The near linear runtime of a seeded watershed stems from its connection to the minimum spanning tree (MST) of the image graph [12, 7]. The present work presents the first viable algorithm that provides analogous M-best results for the seeded watershed cut. We build on the seminal work of Gabow [10] to enumerate only those spanning trees (ST) in an edge-weighted graph that lead to a change in the resulting segmentation. Furthermore we give a modification of Gabow's algorithm that allows to enumerate the M-best diverse solutions similar to [1], by enforcing a user specified distance between some of the generated segmentations. Such a diverse set of solutions can in turn be combined into a final segmentation [4].

2 Related work

The M-best solutions problem has been studied in the context of discrete graphical models [2] where it is known as the M-best MAP problem. Several types of algorithms have been proposed for the M-Best MAP problem: junction tree based exact algorithms [14, 16], dynamic programming based algorithms [15] and max marginal based algorithms [20]. An interesting extension of the M-best MAP problem was proposed and studied in [1]. Here, the authors give an algorithm to enumerate a *diverse* set of solutions. This is an attractive approach since the M-best solutions tend to be very similar to the MAP solution for a low M and thus important aspects of the solution space cannot be found. Generating such a set of diverse solutions was shown in [1] to remedy the problem of M-best solutions: when the initial M-best solutions are too close, important aspects of the solution space may only be found by enforcing a certain amount of diversity. The authors show that this diverse set of solutions which differ in a user specified amount from the MAP solution do not exhibit this problem.

The seeded watershed algorithm [13, 19] which we adapt enjoys great popularity in applications where large amounts of data have to be processed, including medical and biological 3D image analysis [17], and where the shrinking bias typical of MRFs is detrimental.

We rely on the equivalence of the edge weighted seeded watershed and the minimum spanning tree (MST) algorithm [12, 7, 8] and draw on the seminal work of Gabow [10] who solved the k-smallest minimum spanning trees problem.

3 Image segmentation with minimum spanning trees

We formulate the interactive image segmentation problem as a graph partitioning problem on the pixel neighborhood graph $G(E, V)$. All neighboring pixels $v \in V$ are connected with edges $(i, j) \in E$. All edges have an associated edge weight $w_{ij} \in \mathbb{R}$ which expresses the dissimilarity between the neighboring pixels. The edge weights w_{ij} can be computed for example from the color gradient or another suitable boundary indicator. It is well known [12, 7] that the seeded watershed cut on a graph G is equivalent to a minimum spanning tree computation on a suitably augmented graph $G'(V', E')$ that contains a supernode v_0 connected to seed nodes v_{-l} for each label class $l \in L$. These seed nodes are connected to the root node v_0 with zero weight edges. All labeled nodes (i.e. all supervoxels holding a user seed) are also connected to these seed nodes with zero-weight edges $w_{i,-l} = 0$, which are guaranteed to remain in the MST. Once the MST with root node v_0 has been constructed, subtrees originating from seed nodes v_{-l} form segments of the final segmentation. This graph construction is illustrated in Figure 1.

4 Gabow's algorithm for the k smallest spanning trees

The MST segmentation algorithm outlined in the previous section finds the single smallest spanning tree of the augmented graph. In the next section, we will

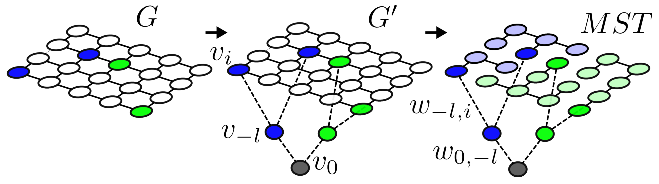

Fig. 1. Seeded image segmentation using minimum spanning trees. The image nodes which the user has seeded (blue, green) are connected to virtual seed nodes v_{-l} which in turn are connected to a virtual root node v_0. The associated edge weights are set to 0 which ensures that the edges belong to the minimum spanning tree of the augmented graph G'. The final label of an image node depends on the subtree to which the node is assigned in the resulting minimum spanning tree.

propose to generalize the algorithm by Gabow [10] to enumerate spanning trees that result in *different* segmentations. To lay the foundation for our extension, we start with a description of Gabow's original algorithm.

Gabow's algorithm starts with a minimum spanning tree for the graph generated by e.g. Kruskal's algorithm. This MST constitutes the first solution. The algorithm then enumerates different spanning trees in the order of increasing weight by swapping out an edge e belonging to the current spanning tree, and replacing it by another edge f which is currently not in the tree.

To obtain the smallest spanning tree under such a so-called e, f-exchange, it finds the pair e, f that gives the smallest weight increase $w(f) - w(e)$.

The main idea of the algorithm is to maintain a set of branchings and two lists associated with each branch, called IN and OUT, which prevent the algorithm from enumerating a spanning tree twice. All edges contained in the IN list have to stay in the spanning tree and all edges contained in the OUT list cannot enter the spanning tree.

To enumerate all spanning trees in order, the algorithm finds the smallest weight e, f-exchange which is feasible according to the IN and OUT lists of the current state and branches on this exchange. Branching is done by considering two different cases: one branch is constructed by adding the f edge to the OUT list, the other branch is established by adding f to the IN list. Any further branching which is executed in the two cases inherits the respective IN and OUT lists from its parent state. Thus, any spanning tree constructed in the first branch excludes edge f and any spanning tree constructed in the second branch includes edge f. By visiting all branchings strictly in the order of increasing tree weight, the first k minimum spanning trees are constructed in the correct order. This process is illustrated in Figure 2.

5 Enumerating changing segmentations

While Gabow's algorithm finds the k smallest spanning trees of a given graph, it cannot be used to find the different modes of a segmentation: a grid graph

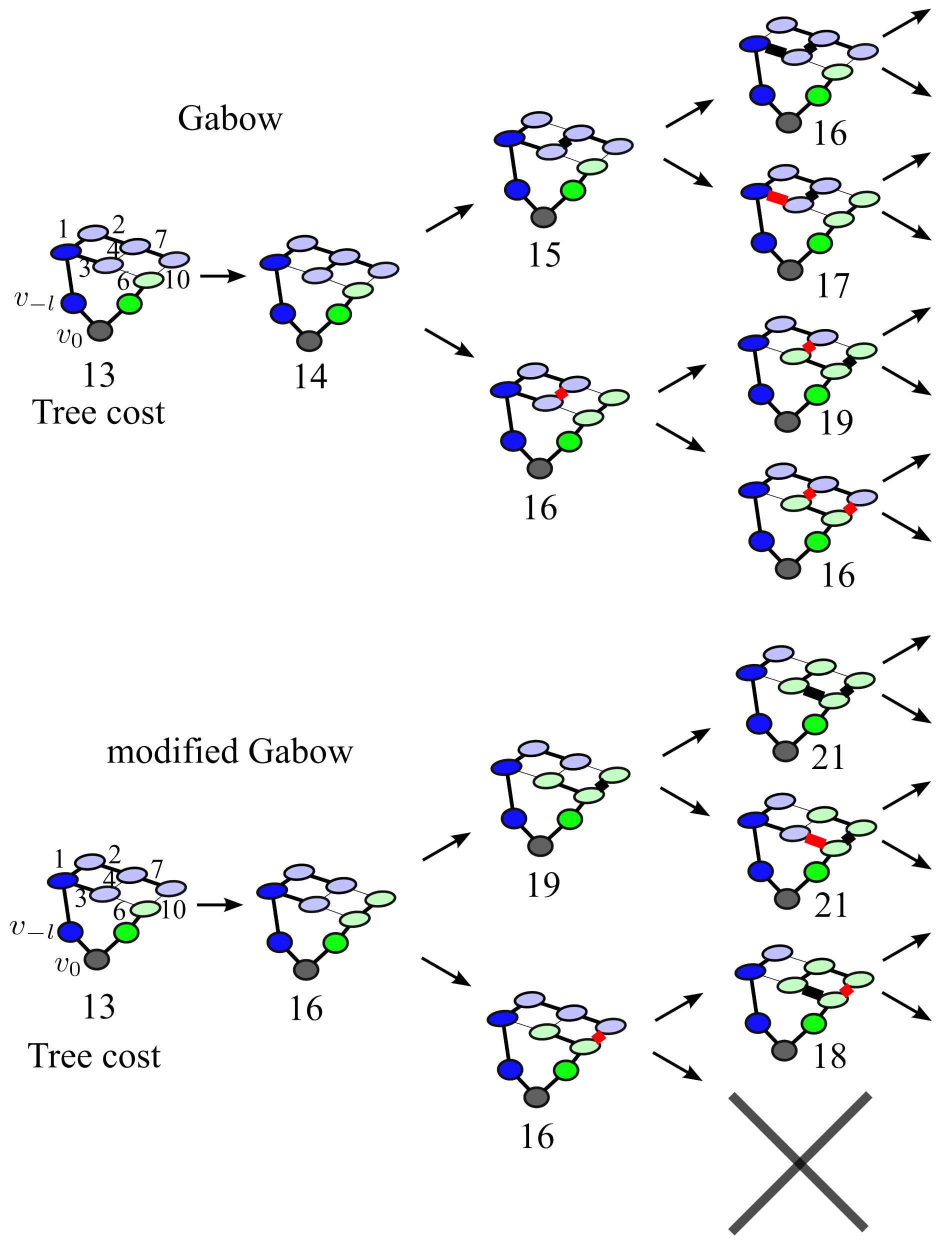

Fig. 2. Illustration of Gabow's algorithm and our modification (best viewed in color). The algorithm of Gabow branches after each e, f-exchange into an upper case where the edge f (indicated by thick black stroke) must stay in the tree and a lower case where the edge f (thick red stroke) must stay out of the tree. Our modified algorithm works in the same way, but only considers edges $f \in C(ST)$ which are part of the cut set for the current spanning tree. By definition, this induces a changed segmentation in each step. In contrast, in the original Gabow algorithm the segmentation often stays unchanged. Some of the k smallest spanning trees have a cut set fully contained in the OUT list and hence do not allow further branching: the set of viable edges for an exchange has been depleted (crossed out state).

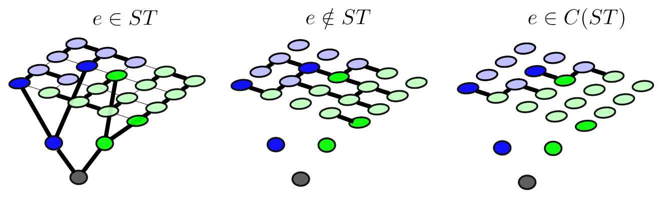

Fig. 3. Illustration of cut edges. On the left, all edges of a spanning tree (ST) are shown in bold. In the middle all edges not belonging to the spanning tree a shown. The subset of the middle edges which belong to the seeded watershed cut – i.e. the edges which connect the different segments – are shown on the right. We modify Gabow's algorithm by only considering these cut edges in any e, f-exchange. This enforces a changing segmentation between any two STs in the hierarchy of Figure 2, but does not guarantee that all resulting segmentations are unique.

has exponentially many spanning trees (10^{40} for a 4-connected grid graph of size 10×10, [18]). In typical images, exceedingly many spanning trees lead to the same segmentation. This effect is visible in the illustration of Gabow's algorithm in Figure 2: while the algorithm always produces a new spanning tree, the associated segmentation does not necessarily change. This is especially true when there are large basins, or areas with low edge weights, as are typical for graphs constructed from natural images.

Thus, when generating the k smallest spanning trees, many if not all (when k is small) of the trees correspond to the same segmentation result. Luckily we can identify the sufficient and necessary condition that leads to a different segmentation result (compared to the previous state) in an e, f-exchange in Gabow's algorithm.

In the case of a spanning tree segmentation, the assigned label (color) of a node depends on the subtree to which the node is connected in the spanning tree: the node is assigned the label of the virtual seed node v_{-l} of which it is a child in the spanning tree (see Figure 1). All edges in the spanning tree connect nodes of the *same* color. An edge connecting two nodes of *different* color cannot be part of the spanning tree segmentation.

Now, if an e, f-exchange removes an edge e from the tree and replaces it with an edge f that connects two nodes of the same color it is clear that the segmentation cannot change: the resulting spanning tree is merely a different way to express the same segmentation.

We now come to the core idea: to enforce a different segmentation, the edge f that is swapped in has to connect two nodes of *different* colors. After swapping in the edge f, both nodes belong to the same subtree and thus one of the two nodes changes its color. The resulting segmentation is different.

We call these edges f that connect nodes of different color in a ST segmentation "cut edges" $f \in C(ST)$. See Figure 3 for an illustration.

At this point, we are able to modify Gabow's algorithm to not only enumerate different spanning trees in order of increasing weight, but to enumerate *changing* segmentations in the order of increasing weight.

A small change is sufficient to reach the desired behavior: by adding all edges f which are not part of the cut set to a list OUTC $= \{f : f \notin C(ST)\}$, Gabow's algorithm can only consider edges $f \in C(ST)$ for any e, f-exchange since all edges in the OUT and OUTC list are not eligible. Thus the segmentation of any spanning tree that is generated differs from the previous segmentation. The OUTC list is always updated once a new spanning tree has been generated and ensures the desired behavior.

Our modification of Gabow's algorithm does not change its computational complexity, the scanning of the edges and the calculation of the OUTC list take time proportional to the number of edges in the graph which is of the same order as the normal Gabow algorithm takes in each iteration.

5.1 Algorithm correctness

Our algorithm may enumerate a segmentation (though not a spanning tree) more than one time and in that sense is an approximation. However, our modification of Gabow's algorithm generates all possible segmentations in their order of increasing weight and finds the lowest cost spanning tree that represents each segmentation. We first show that any spanning tree of minimum weight that represents a segmenation is found by our algorithm.

Definition: *a minimum spanning tree S of a given segmentation* can be found by computing the cut edges $C(S)$ of the desired segmentation: it contains all edges $c = (i, j), x_i \neq x_j$ that connect nodes of different color $x_i \neq x_j$. The minimum spanning tree S of this segmentation can then be found by removing these edges from the graph (or adding them to the *OUT* list) and computing the minimum spanning tree of the modified graph.

A set of constraints IN_S, OUT_S that induce the minimum spanning tree S for a given segmentation can be found easily: the largest such set is $IN_S = \{e : e \in S\}$, $OUT_S = \{e : e \notin S\}$. Computing a spanning tree obeying these constraints will produce S. This shows the existence of at least one such set of constraints.

Theorem 1: Let T be a minimum spanning tree of graph G and let S be the minimum weight spanning tree that induces a given segmentation. Let IN_S, OUT_S be a set of constraints that induce spanning tree S. Then there exists a series of branchings that leads to IN_S, OUT_S.

Proof Let $C(T)$ be the cut edges of spanning tree T. Then any edge $f \in C(T)$ which is eligible for an e, f-exchange in our algorithm is either (a) $\notin S$ or (b) $\in S$. The algorithm now picks the $f \in C(T)$ that has the smallest exchange weight $w(f) - w(e)$ using their best respective exchange partner e. In case (a) if $f \notin S$ we follow the branch where f is added to the *OUT* list. This leads to a new state that has the same induced segmentation (and cut set) but a modified *OUT* list. In the new state the next best exchange pair e', f' with $f' \in C(T)$ will be considered and the same case consideration applies. In case (b) when $f \in S$

we follow the other branch by adding f to the IN list. This leads to a new state with a new segmentation (and new cut set), but starting from this new state the same consideration applies. Both cases (a) and (b) lead to a state which is closer to a set of constraints IN_S, OUT_S. After a finite number of branchings we end up with a set of constraints IN_S, OUT_S that define the minimum cost spanning tree that induces a given segmentation.

The proof shows that all segmentations that are representable by a spanning tree will be found by our algorithm, forbidden branchings as in Figure 2 do not pose a problem: they occur when the OUT list contains all edges in the current cut set. By construction of the proof, for all edges $f \in OUT$ we know $f \notin S$. Since the current cut set is fully contained in the OUT list no edge in the current cut set can be part of the segmentation inducing spanning tree S. But if no edge of the current cut set is $\in S$ then the current state already defines the correct segmentation.

Definition [10]: T-exchange. Let T be a spanning tree of graph G. A T-exchange is a pair of edges e,f where $e \in T$, $f \notin T$, and $T - e \cup f$ is a spanning tree.

Lemma 1 [10]: A spanning tree T has minimum weight if and only if no T-exchange has negative weight.

Theorem 2: Let T be a minimum spanning tree of graph G and let f be an edge not in T. Let e, f be a T-exchange having the smallest weight of all exchanges e', f. Then $T - e \cup f$ is a minimum weight spanning tree of graph G under the constraint that f is in this tree.

Proof Let $S = T - e \cup f$. Suppose S does not have minimum weight. By Lemma 1, there is a S-exchange g, h having negative weight. We derive a contradiction below. Let $T - e$ consist of the two trees U, V. Edge e joins U and V. Edge f must also join U and V since e, f is a T-exchange. Edge h also joins U and V. For if not, assume without loss of generality that h joins two vertices in U. Since g, h is an S-exchange g is also in subtree U and thus g, h is also a T-exchange. Thus, by Lemma 1 it has positive weight which violates our assumption, thus h must join U and V. Edge $g \neq f$ is in $U \cup V$ since exchange g, h cannot remove edge f from S due to the constraint. Assume without loss of generality that $g \in U$. Now let $U - g$ consist of the two trees W, X. Edge e is incident to one of those trees, say W. Since $T - e - g \cup f \cup h$ is a spanning tree, either f or h is incident to X. If h is incident to X then g, h is also a T-exchange which would violate Lemma 1 and leads to a contradiction. If f is incident to X then g, f is also a T-exchange, but since $w(g) \leq w(e)$ this would imply the T-exchange e, h being negative since $w(h) \leq w(g) \leq w(e)$: also a contradiction.

Induction and Theorem 2 gives us the property that our algorithm always produces spanning trees of increasing weight. No permitted e, f-exchange is of negative weight.

From Theorem 1 and Theorem 2 we have that our algorithm produces all possible segmentations in the order of increasing weight.

6 Enumerating diverse segmentations

When enumerating the M-best solutions and choosing a sensible M, say 50, there is a certain danger that the returned solution set is very similar and only differs marginally from the lowest energy solution. To remedy this problem in the M-best MAP setting the authors in [1] propose to enumerate diverse solutions S that obey a given minimum distance $\Delta(MAP, S)$ to the MAP solution. We now show that a similar constraint can be incorporated into our algorithm without increasing computational complexity. We will modify our algorithm that enumerates changing segmentations in such a way that it returns a changing segmentation which differs in at least Δ nodes from the previous segmentation.

The core part of our modified Gabow algorithm from the previous section is the e, f-exchange with a restriction that only allows edges f from the current cut set to be moved into the tree. This restriction of f enforces that at least one node changes its color because it is attached (via edge f) to a differently colored subtree. The exact number of nodes that change their color depends on the edge e which is removed from the tree: all nodes and edges below edge e are reconnected by edge f to another subtree with different color. Thus, by also restricting the edges e for an e, f-exchange we can control how many nodes will change their color in that exchange.

The exact implementation is straightforward: in each step of our modified algorithm, we compute for each edge e the number of nodes $\#(e)$ below this edge. This can be done in time linear in the number of edges. Then, we add all edges e which do not fullfill the user specified diversity requirement Δ to an IND list: IND $= \{e : \#(e) < \Delta\}$. The algorithm is adapted to use both lists, IN and IND. Thus, the IND edges are not considered for an e, f-exchange in the current iteration since they must stay in the tree and any eligible e, f-exchange must change the color of at least Δ nodes. The IND list is updated in each iteration of the algorithm.

7 Experiments

The proposed algorithm is a heuristic because it may produce multiple spanning trees that induce the same segmentation, see section 5.1. To study how many unique segmentations are generated, we choose a segmentation task that is suitable for a minimum spanning tree segmentation [17]: the segmentation of single cells in neural tissue. We ran Gabow's original algorithm and our modified version and compared the induced segmentation of each generated spanning tree to all previous generated segmentations to see how many unique segmentations each of the algorithm generates. As can be seen in Figure 5, Gabow's original algorithm fails to generate even two unique segmentations in $k' = 300$ iterations. Each generated spanning tree differs, but all spanning trees induce an equivalent segmentation. The same effect can be observed in the toy example in Figure 2. Our modification ensures that each generated spanning tree induces a different segmentation compared to the previous state. It works well in practice: more than half of the generated MST's induce a unique segmentation.

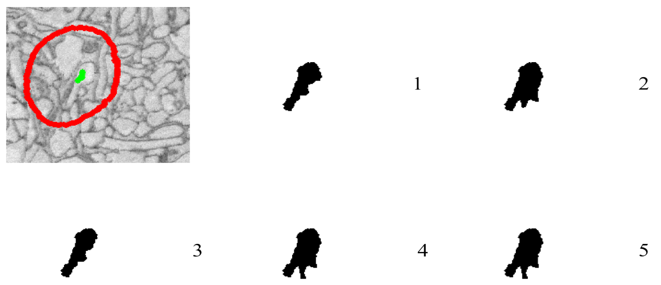

Fig. 4. Modified Gabow example. The top-left image shows an electron micropscopy image of cells in neural tissue [5] and user given seeds (red,green). The $k = 1, 2, 3, 4, 5$ first segmentations generated by our modified Gabow algorithm are shown in black and white. The algorithm successfully finds different modes of the segmentation.

8 Conclusion

Recently a way to enumerate the diverse M-best solutions for Markov random fields has been proposed in [1]. We present an algorithm that allows to enumerate locally changing segmentations in order of increasing spanning tree weight. We prove that the algorithm finds the smallest weight spanning tree that represents a given segmentation. We experimentally validate the algorithm and show that it can be used to effectively enumerate different smallest spanning tree segmentations. Furthermore we show how a diversity constraint can be incorporated into the algorithm that allows to enumerate segmentations which differ in a user specified number of nodes.

We expect the proposed algorithm to be of value in the pursuit of meaningful uncertainty measures as well as user guidance in a 3D segmentation setting. We also trust that more robust segmentations can be obtained by taking into account the information contained in a diverse set of solutions.

References

1. Batra, D., Yadollahpour, P., Guzman-Rivera, A., Shakhnarovich, G.: Diverse M-best solutions in Markov random fields. In: ECCV (2012)
2. Blake, A., Kohli, P., Rother, C.: Markov random fields for vision and image processing. MIT Press (2011)
3. Boykov, Y., Jolly, M.: Interactive graph cuts for optimal boundary and region segmentation of objects in ND images. In: ICCV (2001)
4. Brendel, W., Todorovic, S.: Segmentation as maximumweight independent set. In: NIPS. vol. 4 (2010)
5. Briggman, K.L., Denk, W., et al.: Towards neural circuit reconstruction with volume electron microscopy techniques. Current opinion in neurobiology (2006)
6. Couprie, C., Grady, L., Najman, L., Talbot, H.: Power watershed: A unifying graph-based optimization framework. IEEE PAMI (2010)

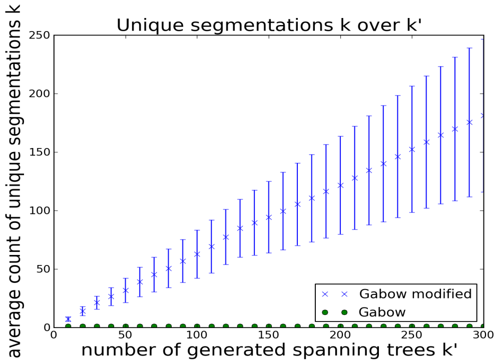

Fig. 5. This plot shows how many unique segmentations are generated when varying the parameter k' that determines how many spanning trees are generated. Shown are the results for Gabow's original algorithm and for our modified version. Gabow's original algorithm fails to generate different segmentations: the spanning trees differ, but the induced segmentation is always the same. In contrast, in the proposed algorithm, more than half of the generated spanning trees induce a unique segmentation.

7. Cousty, J., Bertrand, G., Najman, L., Couprie, M.: Watershed cuts: minimum spanning forests and the drop of water principle. IEEE PAMI pp. 1362–1374 (2009)
8. Falcão, A.X., Stolfi, J., Lotufo, R.A.: The image foresting transform: Theory, algorithms, and applications. IEEE PAMI 26 (2004)
9. Fromer, M., Globerson, A.: An LP view of the M-best MAP problem. NIPS (2009)
10. Gabow, H.: Two algorithms for generating weighted spanning trees in order. SIAM Journal on Computing (1977)
11. Grady, L.: Random walks for image segmentation. IEEE PAMI 28 (2006)
12. Meyer, F.: Minimum spanning forests for morphological segmentation. In: Mathematical morphology and its applications to image processing, pp. 77–84. Springer (1994)
13. Meyer, F., Beucher, S.: Morphological segmentation. Journal of visual communication and image representation 1(1), 21–46 (1990)
14. Nilsson, D.: An efficient algorithm for finding the M most probable configurations in probabilistic expert systems. Statistics and Computing 8(2), 159–173 (1998)
15. Rollon, N.E., Dechter, R.: Inference schemes for M-best solutions for soft CSPs. In: Proceedings of Workshop on Preferences and Soft Constraints. vol. 2 (2011)
16. Seroussi, B., Golmard, J.: An algorithm directly finding the K most probable configurations in bayesian networks. International Journal of Approximate Reasoning 11(3), 205–233 (1994)
17. Straehle, C.N., Köthe, U., Knott, G., Hamprecht, F.A.: Carving: Scalable interactive segmentation of neural volume electron microscopy images. In: MICCAI (2011)
18. Tzeng, W.J., Wu, F.: Spanning trees on hypercubic lattices and nonorientable surfaces. Applied Mathematics Letters 13 (2000)
19. Vincent, L., Soille, P.: Watersheds in digital spaces: an efficient algorithm based on immersion simulations. IEEE PAMI (1991)
20. Yanover, C., Weiss, Y.: Finding the AI most probable configurations using loopy belief propagation. NIPS (2004)