On the Topology of the Level Sets of a Scalar Field.

Valerio Pascucci
Lawrence Livermore National Laboratory

Abstract

This paper introduces the first algorithm for the construction of the Contour Tree of a 3D scalar field augmented with the Betti numbers of each contour component. The scheme has $O(N \log N)$ time complexity and $O(N)$ auxiliary storage, where N is the number of cells in the domain of the field. Previous schemes [11, 10, 3] computed only the Contour Tree which accounts for the number of components of each isosurface but does not make a distinction among isosurfaces of different topology. For example a toroidal isosurface is not differentiated from a spherical one.

1 Introduction

Scalar fields are used to represent data in different application areas like geographic information systems, medical imaging or scientific visualization.

One fundamental visualization technique for scalar fields is the display of isocontours: that is sets of points of equal scalar value. For example in terrain models isolines are used to highlight regions of equal elevation. In medical CT scans an isosurface can be used to show and reconstruct the separation between bones and soft tissues.

In molecular modeling the representation of the interaction energy between two molecules is a scalar field defined over a six dimensional configuration space. The six dimensions are the three translational and the three rotational degrees of freedom of the relative positions of the two molecules. The isocontours of the field represent all the configurations energetically equivalent.

The domain of a scalar field is typically a geometric mesh and the field is provided by associating each vertex in the mesh with a sampled scalar value. If the mesh is a simplicial complex then a piecewise linear field is naturally defined by interpolating linearly, within each simplex, the scalar values at the vertices.

The Contour Tree is a graph that represents the relations between the connected components of the isocontours in a scalar field. Two connected components that merge together (as one changes continuously the isovalue w) are represented as two arcs that join in a node of the graph. The pre-computation of the Contour Tree allows to collect structural information relative to the isocontours of the field. This can be used for example to speedup the computation of the isosurface [11]. The display [2] of the Contour Tree provides the user with direct insight on the structure of the field. Extensions beyond the 3D case [3] are particularly useful because for higher dimensional data [1] the traditional rendering schemes are less intuitive. Especially in those cases the complementary display of the Contour Tree would enhance the user interaction time necessary to "understand" the structure of the data.

One fundamental limitation of the Contour Tree is the lack of additional information regarding the topology of the contours. In high pressure chemical simulations [9] hydrogen bonds between the atoms cannot be represented in a traditional way, but can be characterized by isosurfaces of potential fields. The Contour Tree provides important information regarding the clustering of atoms into molecules but fails to discriminate between linear chains and closed rings (or more complex structures) that have different physical behaviors. The approach in [7] computes the Betti numbers of each isosurface extracted to characterize explicitly its topology. Computationally, such an approach becomes very expensive since it requires the explicit computation of the isosurface itself. If applied as a preprocessing stage to precompute all the Betti numbers of all the isocontours in a scalar field, it would have $O(N^2)$ complexity.

The first efficient technique for Contour Tree computation in 2D was introduced by de Berg and van Kreveld in [4]. The algorithm proposed has $O(N \log N)$ complexity. A simplified version with same complexity in 2D and $O(N^2)$ complexity in higher dimension was proposed by van Kreveld et al. in [11]. This new approach is also used as a preprocessing for an optimal isocontouring algorithm. It computes a small seed set from which any contour can be tracked in optimal running time. The approach has been improved by Tarasov and Vyalyi [10] achieving $O(N \log N)$ complexity in the 3D case by a three pass mechanism that allows to resolve the different types of criticalities. Recently Carr, Snoeyink and Axen [3] presented and elegant extension to the general dimensional case based on a two pass scheme that builds a merge-tree and a split-tree that are successively composed into a unique Contour Tree. The approach achieves $O(n \log n + N\alpha(N))$ time complexity in any dimension.

This paper introduces a new algorithm for the construction in 3D of an augmented Contour Tree where each arc of the tree is associated with the Betti numbers of each contour. The first Betti number β_0 of a mesh, in this case the isosurface, is the number of its connected components. The second number β_1 is the number of tunnels of the mesh. The third number β_2 is the number of voids enclosed in the mesh. With the Betti numbers one provides a complete topological characterization of all the isosurfaces. The algorithm provides this additional topological information within a $O(N \log N)$ worst case computation time.

2 The Contour Tree

Consider a scalar field $\mathcal{F}$ defined as a pair (f, M), where f is a real valued function and M is the domain of f. In the following the domain mesh M is assumed to be a simplicial complex with n vertices $\{v_1, \ldots, v_n\}$. The function f is restricted within each simplex $\sigma \in M$ to be the linear interpolation of its values at the vertices. In other words the field $\mathcal{F}$ is completely defined by the mesh M and the set of scalar values $\{f_1, \ldots, f_n\}$ where $f_i = f(v_i)$. Since M is connected (or processed one connected component at a time) the range of f is a simple closed interval $r = [f_{min}, f_{max}]$ with $f_{min} = \min\{f_1, \ldots, f_n\}$ and $f_{max} = \max\{f_1, \ldots, f_n\}$.

For simplicity of presentation M is also assumed to be a d-manifold with boundary (∂M is a (d-1)-manifold) homeomorphic to a d-ball. This assumption holds in most practical cases and technically can be removed (making the following discussion more complicated).

One fundamental way to study the field $\mathcal{F}$ is to extract its level sets. For a given scalar w the level set $L(w)$ of isovalue w is defined as the inverse image of w onto M through f:

$$L(w) \doteq f^{-1}(w).$$

We call each connected component of the level set $L(w)$ a *contour*. One aspect that is well understood in Morse theory [8] is the evolution of the homotopy types of the contours of $\mathcal{F}$ while w changes continuously in r. The points at which the topology of a contour changes are called critical points and the corresponding isovalue is called critical value. The critical points are usually assumed to be isolated. This assumption is not satisfied in general but can be enforced by a small perturbation of the function values $\{f_1, \ldots, f_n\}$. This perturbation procedure is here weakened by simply assuming that the function values $\{f_1, \ldots, f_n\}$ are sorted from the smallest to the largest so that $i < j \Rightarrow f_i \leq f_j$. This can be enforced with an $O(n \log n)$ preprocessing. In the following the order of the f_i is used to resolve non-isolated criticalities using the filtration technique of [5].

We follow the notation of [3] and define the Contour Tree as a graph $\mathcal{T}$ whose vertices are associated each with a function value f_i and whose connectivity represents the relation among the contours of $\mathcal{F}$ as follows.

- Each leaf of $\mathcal{T}$ represents a local extremum where a contour is created or destroyed, for continuous changes of w. The function value of the extremum is associated with the leaf nodes of $\mathcal{T}$.

- Each interior vertex of $\mathcal{T}$ represents the merging and/or splitting of two or more contours for continuous changes of w. The function value at which a split/merge occurs is associated with the node.

- Each arc of $\mathcal{T}$ represents a contour that remains isolated for w ranging between the function values associated with the extremes of the arc.

Figure 1 shows a 2D scalar field with the associated Contour Tree. Note that the Contour Tree is not a complete

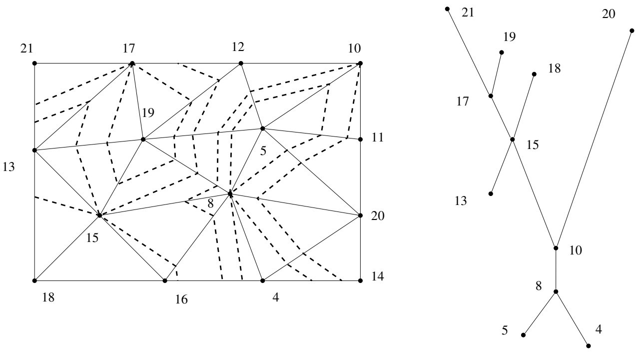


Figure 1: (a) 2D triangular mesh with the contours of the saddles. (b) Corresponding contour tree.

Morse graph of $\mathcal{F}$ since the topological changes of a single contour are not recorded. A more intuitive way to characterize the Contour Tree is the following informal definition:

Definition 1 *The Contour Tree of $\mathcal{F}$ is the graph obtained by contracting each contour of $\mathcal{F}$ to a single point.*

In the following we show how the Contour Tree can be efficiently computed and augmented in 3D with the complete topological information of each contour. In particular we mark each arc of the tree with the Betti numbers of the corresponding component specifying if it is open or closed and the number of its tunnels. This implies that additional nodes are added to $\mathcal{T}$ dividing each arc at the critical points where the topology of the corresponding contour changes. For example the event of a contour changing its topology from a sphere to a torus would not be noted in the basic Contour Tree but is registered in this augmented Contour Tree.

3 Construction of the Contour Tree

This section gives an outline of the algorithm for the construction of the augmented Contour Tree. More details can be found in [11]. Here we just remark the aspects that allow to avoid complexity degradation in the 3D case while computing the additional topological information.

3.1 Algorithm Outline

The algorithm used to build the augmented Contour Tree is based on two successive sweeps of the data (after sorting the vertices by function value).

Sweep 1. The mesh is processed from maximum function value to minimum function value. In this stage a set of "merge simplices" are marked.

Sweep 2. The mesh is processed from minimum function value to maximum function value. In this stage the Contour Tree is constructed and the Betti numbers of all the contours are reported.

We build a filtration of the domain mesh M such that processing the simplices in the filtration order is equivalent to sweeping the level sets of $\mathcal{F}$. Once this goal is achieved one can identify the set of simplices where contours merge with the filtration technique used in [5] and apply the original Contour Tree construction scheme in [11] maintaining $O(N \log N)$ running time in any dimension. This because the tandem search can be formed as in the 2D case starting from the simplex of the filtration where the contours would first merge.

Moreover the second sweep tracks the connectivity of the contours allowing one to determine their Betti numbers, as described below, without penalty on the time complexity of the scheme.

3.2 Tracking the Betti Numbers

Since we traverse and maintain the level sets $L(w)$ we also have the opportunity to count the number of the simplices of any dimension that they contain. In particular in 3D we count the number of vertices, edges and triangles in each contour. Moreover we count the number of boundary edges.

Each contour is by definition one connected component of $L(w)$, so its first Betti number β_0 is 1.

The last Betti number β_2 is 0 if the number of boundary edges is greater than 0 (one open surface). Otherwise the contour is a closed surface with $\beta_2 = 1$ (β_2 can be greater than 1 only at the critical values).

The second Betti number β_1 is be computed as in [7] from the linear equation given by the Euler number of the surface:

$$\sum_{i=0}^{d}(-1)^i \beta_i = \sum_{i=0}^{d}(-1)^i \nu_i \qquad (1)$$

where ν_i is the number of i-cells in $L(w)$. During the second sweep we maintain a count of the values ν_i by processing the vertices by increasing function value. At each step one vertex v_i is selected and the contours are updated within the set of simplices incident to v_i. In this way each simplex is processed one time for each of its vertices. Since we know β_0 and β_2, the only unknown in (1) is β_1. We determine from this equation the current vale of β_1 so that each contour is completely characterized topologically. Notice that this allows to determine indirectly what are the additional critical points of the field that do not cause the merge or split of any contour.

Figures 2 and 3 show an example of scalar field and corresponding augmented Contour Tree. Figures, 4, 5 and 6 show the same Contour Tree enlarged three times and divided in three pages. Attached to each arc in the tree you can see the three Betti numbers of each component (the first is always 1). The sequence of values on the left report the critical values of the scalar field.

If required the method allows also to compute the Betti numbers of contours at the critical values. Observe that, for some ϵ sufficiently small, the level set $L(w)$ is a *deformation retract* of the region $\mathcal{L}$ enclosed between $L(w - \epsilon)$ and $L(w + \epsilon)$. In other words $L(w)$ and $\mathcal{L}$ have the same Betti numbers. Consequently we use the algorithm in [5] to compute the Betti numbers of $\mathcal{L}$ incrementally with a filtration of $\mathcal{L}$ that starts from $L(w - \epsilon)$ and fills $\mathcal{L}$. The resulting output provides the Betti numbers of $L(w)$.

4 Conclusions

In this paper we have introduced a new efficient algorithm for the pre-computation of all the Betti numbers of all the contours in a scalar field. The algorithm generates the Contour Tree structure augmented with the combinatorial characterization of each contour allowing a better understanding of the structure of scientific datasets. The worst case complexity of the scheme is $O(N \log N)$ which is close to the $O(n \log n + N\alpha(N))$ of the currently best scheme that builds only the Contour Tree [3].

The difference of our scheme from the previous approaches is that here we use the filtration technique for tracking the topological changes of the contours in a mesh instead of performing explicit resolution of the singularities.

Previous schemes computing only the Contour Tree account only for the number of components of each isosurface but do not make a distinction among isosurfaces of different topology. We plan to exploit this added information for improved interfaces oriented to the visualization of scientific data.

5 Acknowledgments

Special thanks go to Ken Joy and Herbert Edelsbrunner for the interesting discussions that helped developing this work. Thanks also go to Peter Lindstrom for his review and constructive comments that helped to improve the presentation of this paper.

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract no. W-7405-Eng-48.

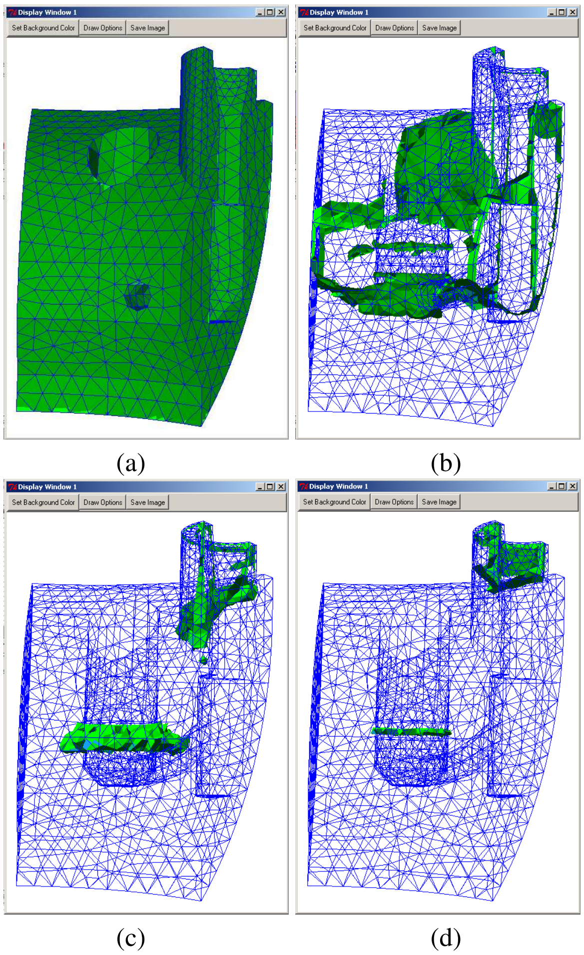


Figure 2: (a) Tetrahedral mesh, (b-d) three isosurfaces of isovalues 0.264, 0.456 and 0.765

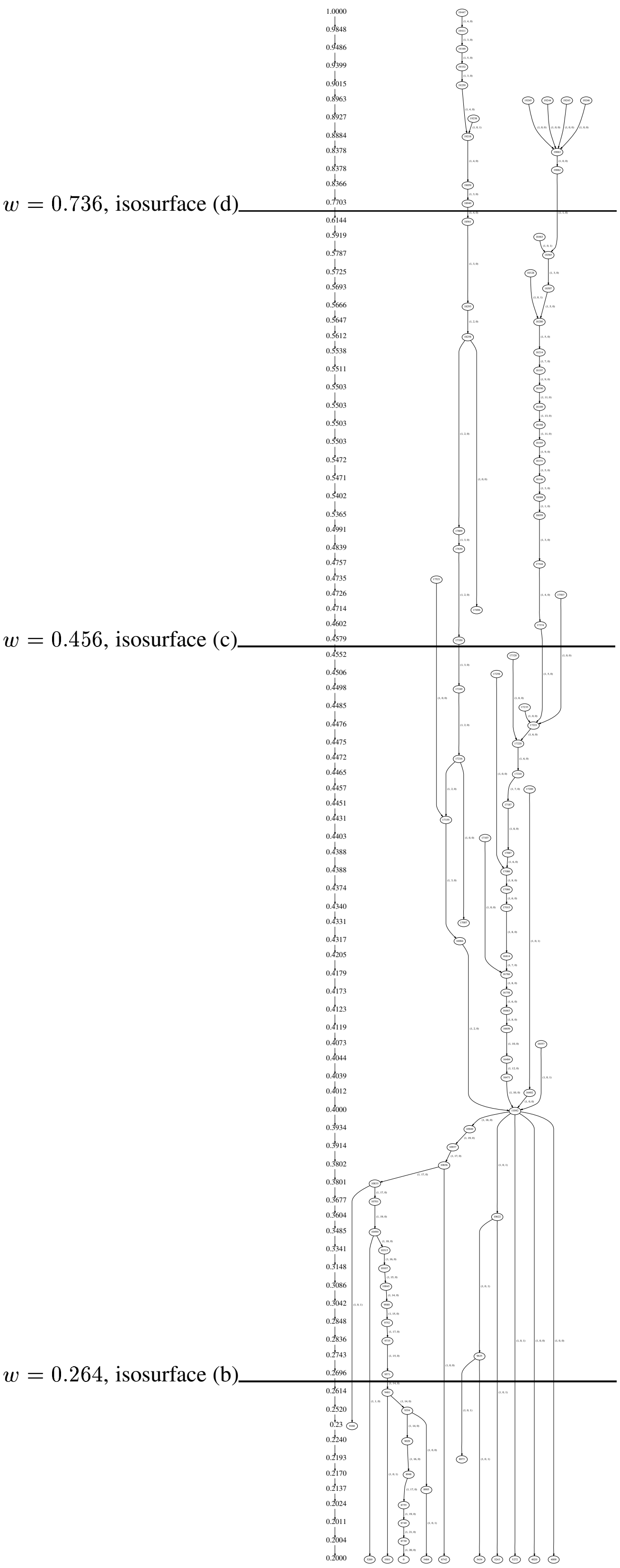


Figure 3: Global view of the Contour Tree of the scalar field shown in Figure 2. On the left the sequence of critical values of the field.

References

[1] C. L. Bajaj, V. Pascucci, G. Rabbiolo, and D. R. Schikore. Hypervolume visualization: a challenge in simplicity (color plate p. 172). In *Proceedings of the 1998 Symposium on Volume Visualization (VOLVIS-98)*, pages 95–102, New York, October 19–20 1998. ACM Press.

[2] C. L. Bajaj, V. Pascucci, and D. R. Schikore. The contour spectrum. In Roni Yagel and Hans Hagen, editors, *IEEE Visualization ́97*, pages 167–175. IEEE, November 1997.

[3] H. Carr, J. Snoeyink, and U. Axen. Computing contour trees in all dimensions. In *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, pages 918–926, January 2000.

[4] M. de Berg and M. van Kreveld. Trekking in the alps without freezing or getting tired. *Algorithmica*, 18(3):306–323, July 1997.

[5] C. J. A. Delfinado and H. Edelsbrunner. An incremental algorithm for betti numbers of simplicial complexes on the 3-sphere. *Computer Aided Geometric Design*, 12(7):771–784, 1995. ISSN 0167-8396.

[6] H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. In *Proceeding of The 41st Annual Symposium on Foundations of Computer Science*. IEEE, November 2000.

[7] S. F. Konkle, P. J. Moran, B. Hamann, and K. I. Joy. Fast methods for computing isosurface topology with betti numbers. Submitted to Proceedings of Dagstuhl 2000, F. Post, G. Neilson and G.-P. Bonneau, eds.

[8] J. Milnor. *Morse Theory*, volume 51 of *Annals of Mathematics Studies*. Princeton University Press, 1963.

[9] E. Schwegler, G. Galli, and F. Gygi. Water under pressure. *Physical Review Letters*, 84(11):2429–2432, 2000.

[10] S. P. Tarasov and M. N. Vyalyi. Construction of contour trees in 3d in o(n log n) steps. In *Proceedings of the fourteenth annual symposium on Computational geometry*, pages 68–75, Minneapolis, June 1998. ACM.

[11] M. van Kreveld, R. van Oostrum, C. L. Bajaj, V. Pascucci, and D. R. Schikore. Contour trees and small seed sets for isosurface traversal. In *Proceedings of the 13th International Annual Symposium on Computational Geometry (SCG-97)*, pages 212–220, June 1997. Extended version. Techincal report UCRL-JC-132016 Lawrence Livermore National Laboratory.

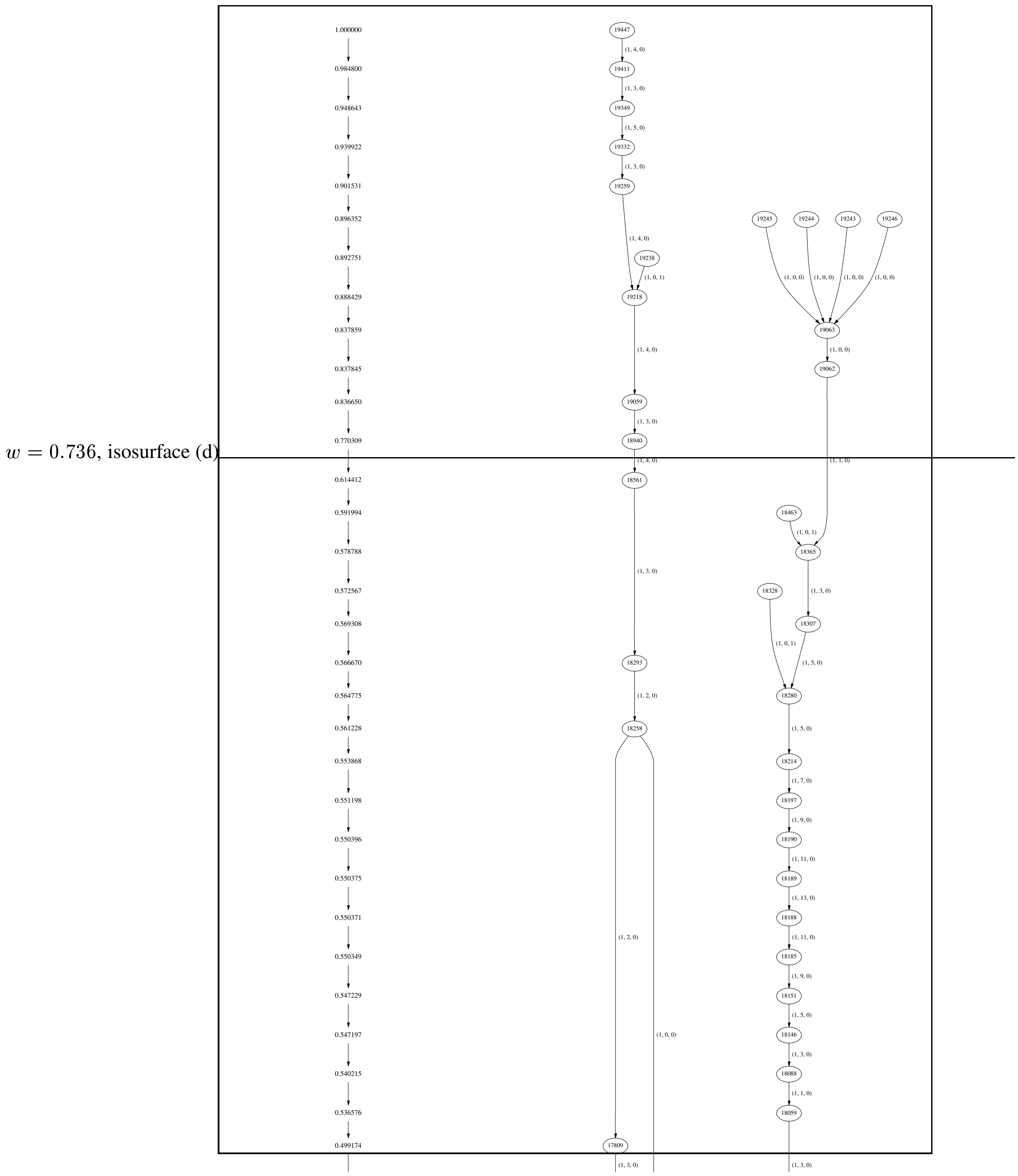


Figure 4: Top part of the Contour Tree in Figure 3.

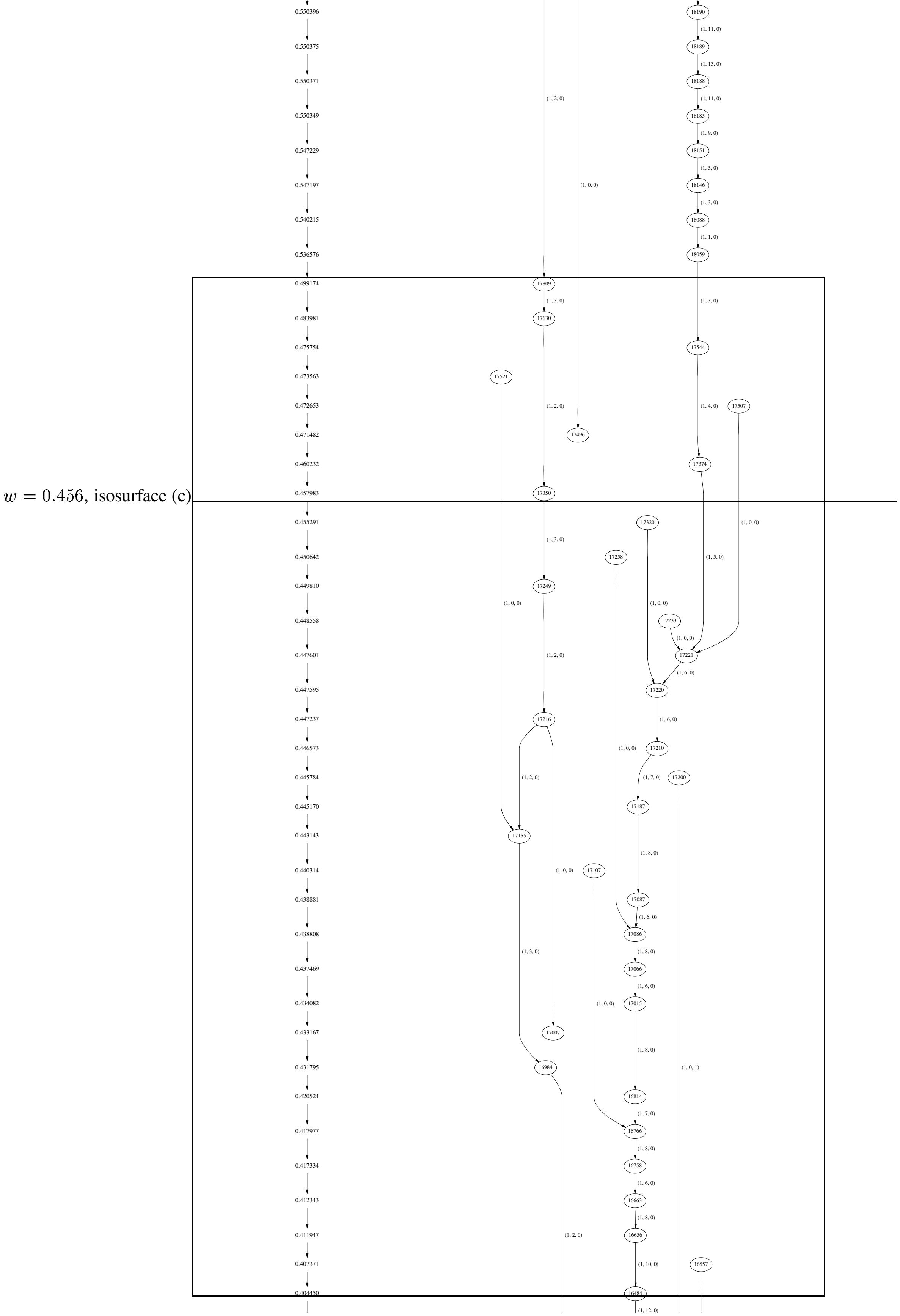
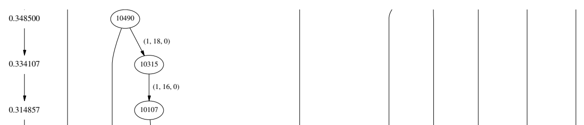


Figure 5: Central part of the Contour Tree in Figure 3.



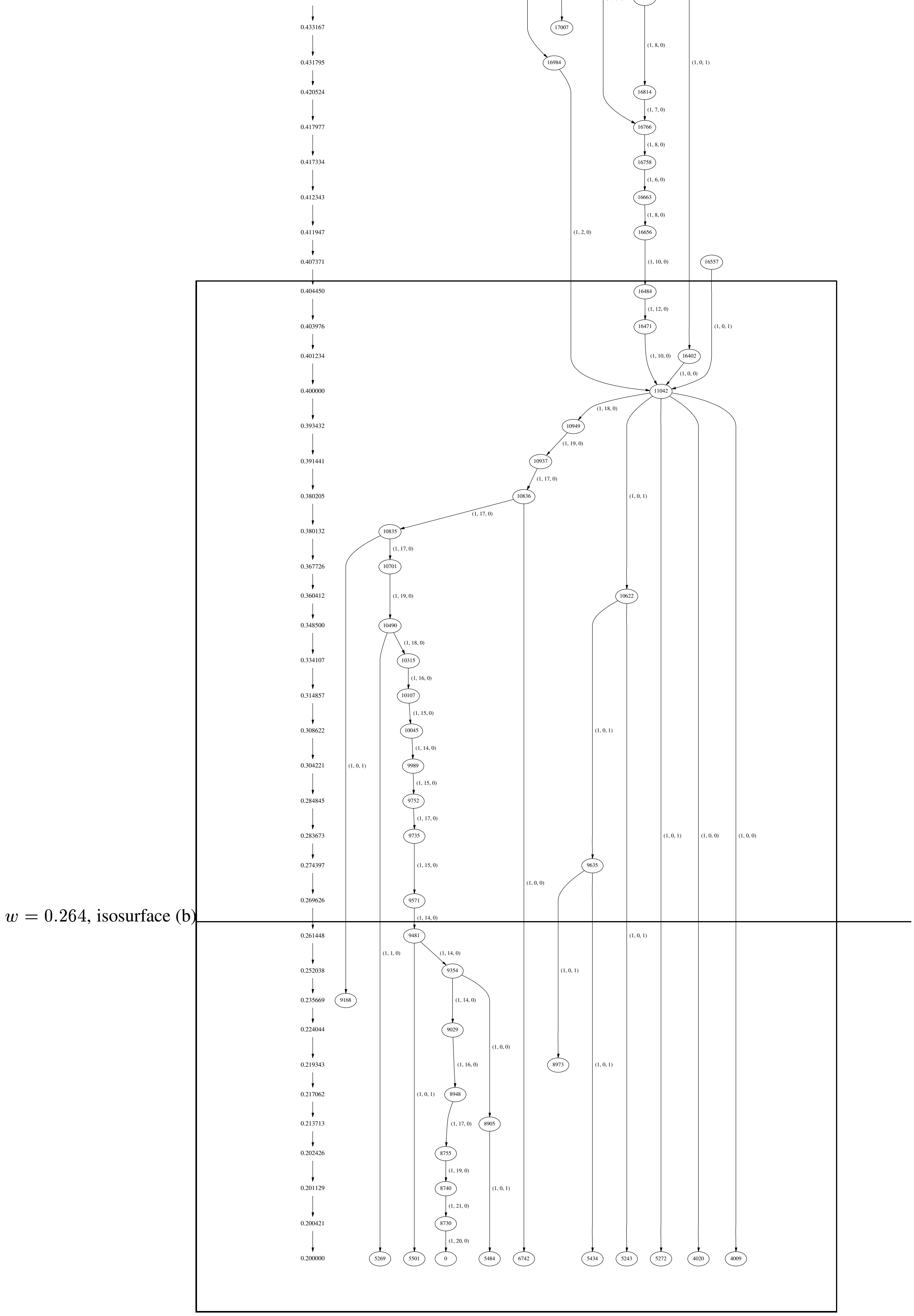


Figure 6: Bottom part of the Contour Tree in Figure 3.