Watersheds on hypergraphs for data clustering

Fabio Dias[1], Moussa R. Mansour[2,3], Paola Valdivia[2,4], Jean Cousty[5], and Laurent Najman[5]

[1] New York University - Tandon School of Engineering, New York, USA.
fabio.dias@nyu.edu
[2] Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, Brazil.
[3] Jack's Ventures, Perth, Australia.
[4] INRIA, Saclay, France.
[5] Université Paris-Est, LIGM, Equipe A3SI, ESIEE, France.

Abstract. We present a novel extension of watershed cuts to hypergraphs, allowing the clustering of data represented as an hypergraph, in the context of data sciences. Contrarily to the methods in the literature, instances of data are not represented as nodes, but as edges of the hypergraph. The properties associated with each instance are used to define nodes and feature vectors associated to the edges. This rich representation is unexplored and leads to a data clustering algorithm that considers the induced topology and data similarity concomitantly. We illustrate the capabilities of our method considering a dataset of movies, demonstrating that knowledge from mathematical morphology can be used beyond image processing, for the visual analytics of network data. More results, the data, and the source code used in this work are available at `https://github.com/015988/hypershed`.

Keywords: Data clustering, Hypergraphs, Watershed algorithm.

1 Introduction

Data clustering is one of the most fundamental operations for the exploration of large amounts of information, allowing the identification of similarities and the highlight of differences, reducing the amount of cognitive effort required to gain gist information using visual analytics. Therefore, several methods for clustering data exist in the literature, including methods using graph clustering.

Our interest lies on network data, defined as data that includes relationships between its portions, usually modelled as digital structures, enabling the representation of more detailed nuances. Indeed, when the relationships are not derived from similarities in the data, but represent a different facet of the information, a clustering method needs to consider both to obtain meaningful results.

Interestingly, this is the exact context in which image segmentation methods are developed, considering both the information on the pixels and the neighboring relationship between them. In particular, the watershed algorithm is fast,

easy to implement, and was recently extended to several digital structures, including graphs. However, to improve the flexibility for data representation, we adopted hypergraphs as base digital structure, and we introduce a trivial extension of the watershed algorithm for hypergraphs. However, our method aims to cluster the edges of the hypergraph, not its nodes. This slight, but crucial, difference is more suitable to represent data relationships where one data point, represented as an edge, is related to several entities, represented as nodes.

In summary, the main contributions of this work are:

- An extension of the watershed algorithm to hypergraphs.
- A novel framework to represent and cluster data represented as an hypergraph.
- An application of the watershed algorithm outside of image processing.

2 Related work

Since clustering is a crucial step for data sciences, several methods have been proposed [12]. Traditionally, the data itself is composed of points, and the objective is to identify clusters of similar points, considering some metric. Some methods build a similarity graph, where the data points are represented as nodes and the similarity as the weights on the edges [1]. This approach then leverages graph clustering, where the objective is to separate the graph structure into strongly connected clusters [21,10]. However, seldom additional data is considered [25], and the clusters reflect only the topology induced by the similarity function. When available, this application dependent data may lead to a finer, more accurate, clustering result.

Of course, hypergraphs can also be considered, when the relationships in the data cannot be accurately expressed using only pairwise links [14,13]. Several different clustering methods have been proposed, including random walks [9], spectral clustering [24], game theory [4], among others.

However, data clustering using a topology that is not derived from data similarity is not nearly as explored [22], at least not with this interpretation. Indeed, image segmentation is an equivalent problem, where both the data and its relationships need to be concomitantly considered; the objective is to identify portions of pixels (data) that are similar and connected (linked).

While a myriad of methods for image segmentation have been proposed in the literature, including the use hypergraphs for image representation [3,9], the watershed algorithm [17,23,18], and the family of derived methods [19,20,2,6], is one of the most used, because of its simplicity and robust results. Moreover, the algorithm was extended to digital structures as well [16,8,5,7].

Despite the adequacy of the watershed algorithm for the clustering of network data, its use is not properly explored outside of image processing. This is the exact context of this work, where we explore the watershed algorithm for the visual analytics of network data. Visual analytics methods are generally used to systematically explore unknown datasets, combining the perception of an operator with the numerical capabilities of a computer.

Moreover, our method can be implemented without constructing a matricial representation of the digital structure, increasing its scalability and allowing the processing of massive datasets, while maintaining the low computational cost of watershed cuts.

3 Watershed on hypergraphs

Our objective is to partition the edges of an hypergraph into groups such that the edges in each group have similar data associated with them and a connected through one or more nodes. Between the several possible definitions for the watershed operator, we follow the framework of watershed-cuts [8], adapted where necessary to hypergraphs, including the semantic difference of clustering the edges instead of the nodes.

3.1 Hypergraphs

We define an hypergraph as $H = (V, E, \mathcal{D})$, where V is a finite set of vertices, E is the set of edges such that $\forall e \in E, e \subseteq V$, and $\mathcal{D}$ is a function that associates data to the edges of the hypergraph, in the form of a representative *feature vector*: $\mathcal{D} : E \rightarrow \mathbb{R}^m$, with $m \in \mathbb{N}^+$. While we assume that the feature vector is an array of numbers, any information can be considered, as long as a distance metric between two instances can be defined. Two of the possible visual representations of this structure are illustrated in Figure 1, considering a small hypergraph with six edges and five nodes. Each edge is associated to a two dimensional real vector that characterizes the data point. This definition is different of the common practice, where data points are represented as nodes and the edges have associated weights.

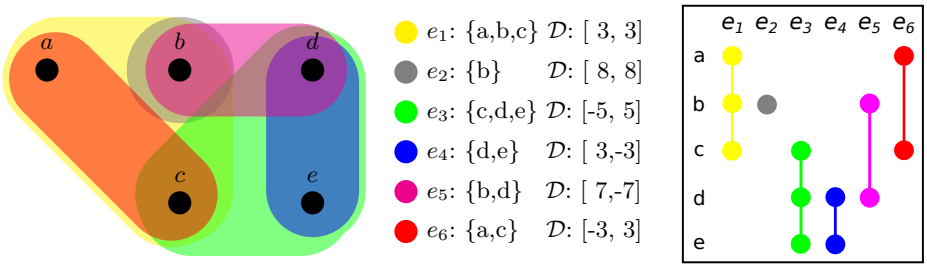

Fig. 1. Graphical representations of an hypergraph with data associated to its edges.

Let H be an hypergraph. Two distinct edges are *neighbors* if they share a vertex, that is, $N(e_i, e_j) \leftrightarrow ((e_i \cap e_j) \neq \emptyset)$, for any $e_i, e_j \in E$. The *set of neighbors* of an edge e is defined as $N(e) = \{u \in E \,|\, (e \cap u \neq \emptyset) \wedge (e \neq u)\}$. For instance, in the example depicted in Figure 1, edges e_1 and e_3 are neighbors, since both include node c, but e_1 and e_4 are not.

We define a *path* π as an ordered sequence of distinct edges of H, $\pi = \langle e_0, e_1, \ldots, e_\ell \rangle$ such that any two consecutive edges are neighbors. For instance, considering the hypergraph in Figure 1, $\langle e_1, e_2, e_5 \rangle$ is a valid path, where $\langle e_1, e_4, e_6 \rangle$ is not. If $\ell = 0$, π is a *trivial path* $\pi = \langle e_0 \rangle$. If there is a path between any two edges of H, the hypergraph is *connected*.

We define the distance between edges e_i and e_j as $d(e_i, e_j)$, where d is a distance metric between the feature vectors, which can be any distance metric suitable for the considered problem. This distance is analogous to the gradient information in the traditional watershed [18], representing the height to be surmounted by the water in the relief map of the data.

A *descending path* is a path π in which the distance between consecutive edges do not increase, $d(e_{i-1}, e_i) \geq d(e_i, e_{i+1})$, for any $i \in [1, \ell - 1]$. Intuitively, a *steepest descent path* is a descending path where the distance values decrease the most at each step. Therefore, each consecutive edge corresponds to smallest possible distance from the previous edge, $d(e_i, e_{i+1}) = \min\{d(e_i, u), u \in N(e_i)\}$, for any $i \in [0, \ell - 1]$. To simplify the notation, we define a function to represent this minimal distance as $d^{\ominus}(e) = \min\{d(e, u), u \in N(e)\}$, therefore, in a path of steepest descent, $d(e_i, e_{i+1}) = d^{\ominus}(e_i)$, for any $i \in [0, \ell - 1]$.

Definition 1. ***Watershed clustering.*** *Let $H = (V, E, \mathcal{D})$ be an hypergraph and $\Pi = \{\pi_0, \pi_1, \ldots, \pi_{|E|}\}$ be a collection of steepest descent paths, such that every edge of H is the first edge of one path, that is, $\pi_i = \langle e_i, \ldots \rangle$, for any $i \in [0, |E|]$. Then a watershed clustering of the edges of H is a function $\Psi : E \to \mathbb{N}$ that attributes labels to the edges according to the last edge of the path, that is $\Psi(e_i) = \Psi(e_j) \leftrightarrow \exists \pi_i, \pi_j \in \Pi, e_z \in E \,|\, (\pi_i = \langle e_i, \ldots, e_z \rangle) \wedge (\pi_j = \langle e_j, \ldots, e_z \rangle)$.*

3.2 Relationship to watershed cuts

While definition 1 characterizes watershed clustering on hypergraphs, it does not provide a way of obtaining one. To this end, we directly leverage the watershed cuts algorithm, applied on a graph generated from the considered hypergraph.

We create a weighted-edge graph G, using information from the hypergraph H. Since we aim to cluster the edges of the hypergraph, each edge is represented by a node of G, and edges are placed representing the corresponding neighbors, the edge weights are given by the distance between the edges (feature vectors), using an arbitrary distance metric. An example of this procedure for the hypergraph depicted in Figure 1 is illustrated in Figure 2, using the cosine between the feature vector as the distance. In the resulting clustered hypergraph, each edge belongs to exactly one cluster, but each node can be contained by edges of several clusters, since it can be contained by edges on different clusters. Indeed, in the result illustrated on the last panel of Figure 2 all nodes belong to edges in two distinct clusters. This information is not as easily obtained on the graph constructed from the edges, since the nodes are not explicitly represented.

A node with edges in two or more clusters can be seen as the equivalent of the watershed lines in the classic image processing framework, acting as a barrier between them. For data analysis this aspect can be interesting, because it elicits patterns of behavior.

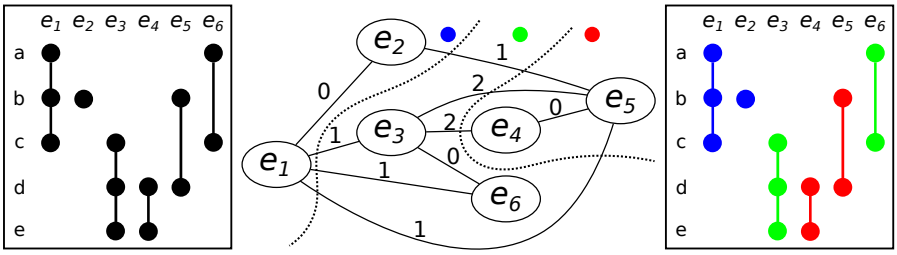

Fig. 2. Hypergraph from Figure 1, the corresponding weighted edge graph with the result of watershed cuts, and the clustered hypergraph. The distance used for the edge weights is the cosine between the vectors.

Property 1. A watershed cut of the nodes of the graph G is a watershed clustering of the edges of the hypergraph H.

This property is self evident, because there is a bijection between the edges of the hypergraph H and the nodes of the graph G, the neighborhood relationships are preserved, as well as the distances.

3.3 Algorithm for watershed on hypergraphs

While we used the close relationship between watershed on hypergraphs and watershed cuts in graphs to provide an easy way to compute the clustering, to explicitly construct another structure is inefficient, particularly when considering large amounts of data. To avoid this reconstruction, we adapt the original watershed-cuts algorithm [8] to hypergraphs.

Algorithm 1 Watershed on hypergraphs

```
function watershed(H)
    for all e ∈ E do Ψ(e) ← NO_LABEL
    nb_labs ← 0
    for all e ∈ E such that Ψ(e) = NO_LABEL do
        [L, lab] ← stream(H, Ψ, e)
        if lab=-1 then
            nb_labs ← nb_labs + 1
            for all y ∈ L do Ψ(y) ← nb_labs
        else
            for all y ∈ L do Ψ(y) ← lab
    return Ψ
```

The two functions introduced in Algorithms 1 and 2 are identical to the original watershed cuts algorithm [8], edges are considered instead of nodes and the neighborhood relation is changed; we refer to the original work for an in depth analysis of the algorithm and its clustering performance.

Algorithm 2 Auxiliary function to identify streams

```
function stream(H, Ψ, e)
    L ← {e}
    L' ← {e}
    while ∃y ∈ L' do
        L' ← L' \ y
        breadth_first ← True
        while breadth_first ∧ (∃z ∈ N(y) | (z ∉ L) ∧ (d(y, z) = d^⊖(y))) do
            if Ψ(z) ≠ NO_LABEL then
                return [L, Ψ(z)]
            else
                if d^⊖(z) < d^⊖(y) then
                    L ← L ∪ {z}
                    L' ← {z}
                    breadth_first ← False
                else
                    L ← L ∪ {z}
                    L' ← L' ∪ {z}
    return [L, −1]
```

Implementation considerations. While the algorithm is identical to watershed cuts, there are implementation details that are more relevant when considering hypergraphs. For instance, the weights of each edge of the weighted graph are usually precomputed; the algorithm can simply access these values, without any increase in the computational time. Since our algorithm does not explicitly construct the graph, the distances are calculated on-demand, as the algorithm explores the hypergraph. Moreover, the adopted metric can be computationally expensive, so repeated computations of the distance between two edges should be avoided. This can be easily accomplished using memoization, which can use less memory than the explicit computation of the distance between all edges.

If deterministic results are desired, the method needs to have a stable sorting method, where ties in the distance values are broken in the same way. In any case, all possible results are valid clusters, satisfying Definition 1.

4 Illustration of the method

To illustrate the behavior of our method on real data, we considered information from the The Movie Database (`www.themoviedb.org`), using a breadth-first search, alternating movies and actors, starting on *Lord of the Rings: The fellowship of the ring*. Each movie is represented as an edge and the involved actors as nodes. By construction, this hypergraph is connected. The feature vectors are derived from they keywords and genres associated to each movie; the distance metric used was the cosine between the two feature vectors. Actors are represented as nodes and movies as edges, including the whole cast.

Therefore, both the characteristics of the movies and the relationships between movies and actors are expressed in the structure, allowing the differentiation between identical movies with different actors. For instance, it might not be desirable to aggregate Peter Jackson's Lord of the Rings trilogy with Ralph Bakshi's 1978 movie, despite the fact that the feature vectors of both should be remarkably similar. This distinction is a direct result of including the topology into the clustering process, and this effect can be avoided by using a clustering method that ignores the topology.

Further, the objective of this section is only to illustrate the behavior of the method, demonstrating that it can be used for network data exploration. Since our method is effectively a translation of watershed cuts into hypergraphs, we refer the reader to the works by Cousty et al. [8,7] for a performance comparison against other segmentation methods.

In this example, we considered a dataset with 100 movies and 1,487 actors. Our implementation of the method was done in Python and is freely available at `https://github.com/015988/hypershed`. The processing time was approximately 0.1 second on a regular i7 computer. For reference, the processing time for a dataset with 5,000 movies and 39,029 actors took 50 seconds. Our clustering result is illustrated in Figure 3. However, due to size constraints, we depict only the 87 actors related to two or more clusters.

Between the 17 resulting clusters, the blue cluster on the leftmost part of the figure groups movies from the Tolkien universe, including, however, two movies classified as documentaries: *The watchmaker's apprentice* and *Slacker uprising*, which can be considered as significantly different when compared to the other movies in the cluster. While similar between themselves, these two movies did not form a separate cluster because they are not neighbors, there is no overlap in the casting. However, both are neighbors of the movies in the Lord of the Rings trilogy, in the former, John Rhys-Davies is the narrator, while in the latter Viggo Mortensen is part of the cast. Similarly, *Dracula*, starred by Christopher Lee, is also included in this cluster.

Similarly, the movies from the *Pirates of the Caribbean* franchise were grouped together in the brown cluster, as well as two movies from the *Indiana Jones* and *James Bond* franchises, in the two gray clusters. Most of the movies from the *X-Men* universe were grouped in the light green cluster, with two movies, *The Wolverine* and *X-Men: Apocalypse* separated into another cluster, most likely because the feature vectors of these two movies are very similar, creating a new minimum. Moreover, the feature vectors of some of the movies are very small, some movies in this dataset contain only one genre and no keywords, compromising the representability of the distances between the movies.

The plot on the bottom left of Figure 3 illustrates a T-SNE [15] projection of the feature vectors associated with the movies. This method consider each feature vector as a point in a high dimensional space and aims to find a two dimensional embedding of the vectors that preserves the distances between the points in the original high dimensional space. The color of the circles correspond to the clusters of the top part of the figure. This plot ignores the topological

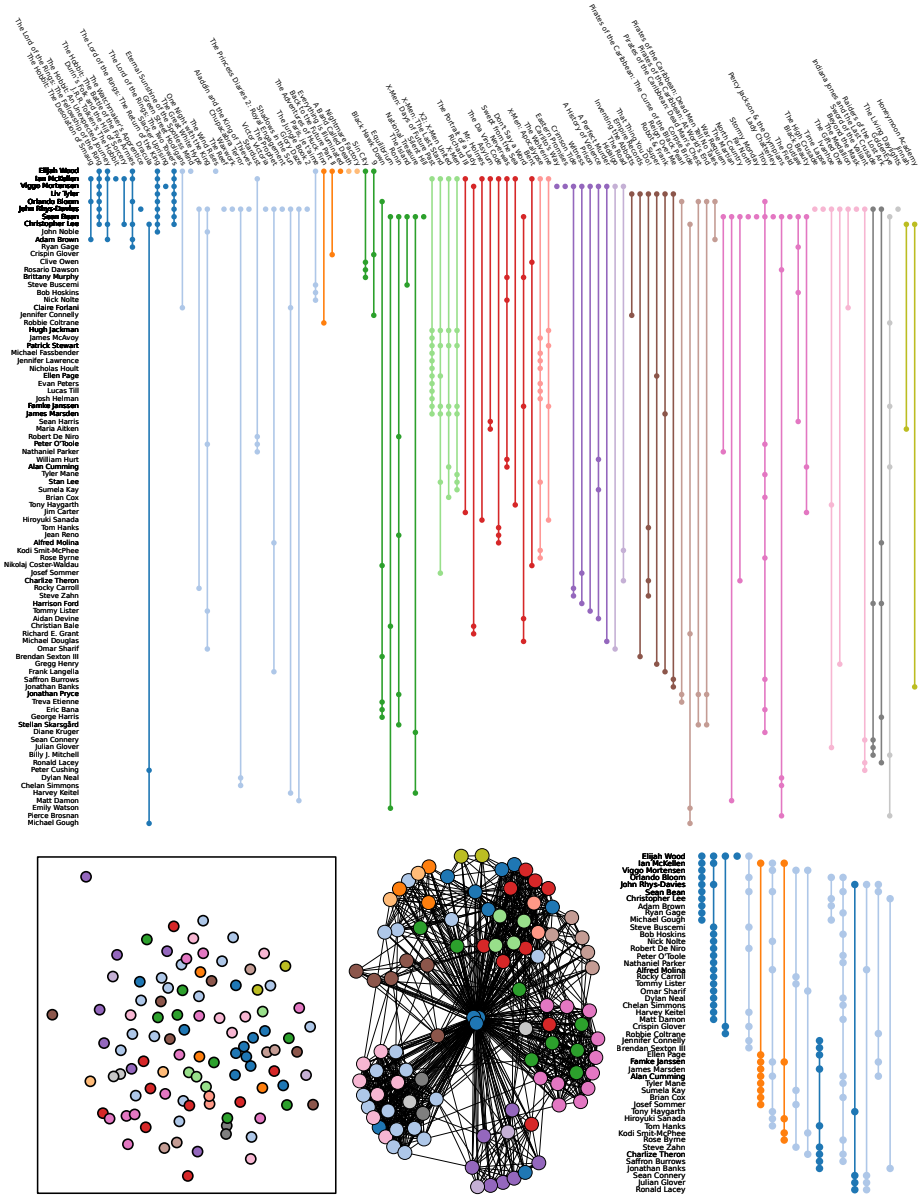

Fig. 3. Clustering result for TMDB dataset with 100 movies/edges. Top: Initial clustering. Bottom left: T-SNE projection of the feature vectors of the movies from its natural high dimensional space into $\mathbb{R}^2$, with the colors corresponding to the clusters. Bottom middle: Force layout on the equivalent graph. Bottom right: Clustering of the initial clusters.

connections induced by the actors, illustrating that, while there is some correspondence between the watershed clustering and the projection of the points, there is no clear separation of the clusters when only the feature vectors are considered. Therefore, the topology of the hypergraph is tremendously significant to the watershed results, as expected.

Moreover, the plot on the bottom middle of Figure 3 illustrates an unweighted graph which topology is equivalent to the neighborhood relationships between the edges of the hypergraph, as defined in Section 3.2. The graph is depicted using a force layout [11], with the colors also corresponding to the clusters in the top part of Figure 3. This layout aims to group heavily connected nodes, and clearly depicts several different groupings, interconnected by the blue nodes in the middle. These blue nodes correspond to the *The Lord of the Rings* movies, including the seed movie used to generate the dataset, which explains this topology. As expected, the clustering result is more similar, due to the influence of the induced topology, but not quite equivalent to the strongly connected nodes of the graph, because the feature vectors are also considered.

By representing each cluster as an edge, the clustering can be recursively applied, as illustrated on the bottom right of Figure 3, where each edge represents a cluster of the top visualization, in the same order. The feature vector of these new edges is defined as the average of the feature vectors of its composing edges. The colors represent the three new clusters. The orange cluster in this clustering corresponds to the groups containing movies of the *X-Men* franchise, the blue group corresponds to, in general, fantasy movies and the light blue cluster to action movies and dramas. Interestingly, Sir Ian McKellen is the only actor in this small dataset to have movies on all three clusters, illustrating his well known versatility. Similarly, the nodes that are not depicted in these figures, nodes whose edges belong to a single cluster, could be used to identify "niche" actors, considering a more comprehensive dataset.

5 Discussion and Conclusions

In this work, we presented a novel way to represent and cluster network data, using hypergraphs to represent relationships between portions of the data.

While data clustering with and without the topology may seem similar on an abstract level, these are two very different problems, and methods that consider the topology cannot be directly compared to the classic methods that consider only the data points. Similarly, neither can be directly compared to network clustering methods that do not consider data associated with the elements. The same argument applies to most clustering score methods as well. The included results aim only to illustrate the use of watershed cuts on hypergraphs as a tool for visual analytics on network data.

We adopted hypergraphs because they are a natural extension of graphs, increasing the applicability of the method. Moreover, edges and clusters are more conceptually similar, both can be interpreted as sets of nodes, leading to naturally hierarchical structure, when the method is recurrently applied. We did

not explore this option beyond what is illustrated in Figure 3 because the visual analytics interface needed to properly represent these results is challenging and beyond the scope of this work; a potentially interesting future work.

Our work leverages the advantages of the watershed algorithm, but it also includes its disadvantages as well, including over segmentation, as illustrated by the two separate clusters containing *X-Men* movies in Figure 3. Further, not all edges on the same cluster are necessarily similar, the watershed can group edges in a chain of similarity, where two sequential edges are similar, but edges far apart in the chain are not, which can be counterintuitive for data sciences. However, we believe that this effect would be less pronounced on massive, more connected, datasets, with plenty of data points to properly compose each cluster. In this context, the number of clusters would be massive as well, and an our hierarchical approach could present a viable alternative for the visual exploration of such data.

While our proposed method can be considered, quite correctly, as a simple reinterpretation of watershed cuts into a new digital structure, these subtle semantic differences are novel and unexplored in the literature, leading to crucially different results. Further, they allow the application of this method for data sciences, illustrating that the knowledge from image processing can be transported to this context, which was the inspiration for this work.

Acknowledgments

Grants 2016/04391-2, 2014/12815-1, 2015/14426-5, 2013/21779-6, 2013/14089-3, 2011/22749-8 São Paulo Research Foundation (FAPESP). The views expressed are those of the authors and do not reflect the official policy or position of the São Paulo Research Foundation.

References

1. Aggarwal, C. C., and Reddy, C. K. *Data Clustering: Algorithms and Applications*. Chapman and Hall/CRC, 2013.
2. Bertrand, G. On topological watersheds. *Journal of Mathematical Imaging and Vision 22*, 2-3 (2005), 217–230.
3. Bretto, A., and Gillibert, L. *Hypergraph-Based Image Representation*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005, pp. 1–11.
4. Bulò, S. R., and Pelillo, M. A game-theoretic approach to hypergraph clustering. In *Advances in Neural Information Processing Systems 22*, Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, Eds. Curran Associates, Inc., 2009, pp. 1571–1579.
5. Couprie, C., Grady, L. J., Najman, L., and Talbot, H. Power watershed: A unifying graph-based optimization framework. *IEEE Trans. Pattern Anal. Mach. Intell. 33* (2011), 1384–1399.
6. Couprie, M., Najman, L., and Bertrand, G. Quasi-linear algorithms for the topological watershed. *Journal of Mathematical Imaging and Vision 22* (2005), 231–249.

7. Cousty, J., Bertrand, G., Couprie, M., and Najman, L. Collapses and watersheds in pseudomanifolds of arbitrary dimension. *Journal of mathematical imaging and vision 50*, 3 (2014), 261–285.
8. Cousty, J., Bertrand, G., Najman, L., and Couprie, M. Watershed cuts: Minimum spanning forests and the drop of water principle. *IEEE Transactions on Pattern Analysis and Machine Intelligence 31*, 8 (2009), 1362–1374.
9. Ducournau, A., and Bretto, A. Random walks in directed hypergraphs and application to semi-supervised image segmentation. *Computer Vision and Image Understanding 120* (2014), 91 – 102.
10. Fortunato, S. Community detection in graphs. *Physics reports 486*, 3 (2010), 75–174.
11. Fruchterman, T. M. J., and Reingold, E. M. Graph drawing by force-directed placement. *Software: Practice and Experience 21*, 11 (nov 1991), 1129–1164.
12. Jain, A. K. Data clustering: 50 years beyond k-means. *Pattern recognition letters 31*, 8 (2010), 651–666.
13. Leordeanu, M., and Sminchisescu, C. Efficient hypergraph clustering. In *AISTATS* (2012).
14. Lotfifar, F., and Johnson, M. A serial multilevel hypergraph partitioning algorithm. *arXiv preprint arXiv:1601.01336* (2016).
15. Maaten, L. v. d., and Hinton, G. Visualizing data using t-sne. *Journal of Machine Learning Research 9*, Nov (2008), 2579–2605.
16. Meyer, F. Watersheds on weighted graphs. *Pattern Recognition Letters 47* (2014), 72–79.
17. Meyer, F., and Beucher, S. Morphological segmentation. *Journal of Visual Communication and Image Representation 1*, 1 (1990), 21 – 46.
18. Najman, L., and Talbot, H., Eds. *Mathematical Morphology*. Wiley-Blackwell, 2013.
19. Passat, N., Ronse, C., Baruthio, J., Armspach, J.-P., and Foucher, J. Watershed and multimodal data for brain vessel segmentation: Application to the superior sagittal sinus. *Image and Vision Computing 25*, 4 (2007), 512–521.
20. Roerdink, J. B. T. M., and Meijster, A. The watershed transform: Definitions, algorithms and parallelization strategies. *Fundam. Inform. 41* (2000), 187–228.
21. Schaeffer, S. E. Graph clustering. *Computer science review 1*, 1 (2007), 27–64.
22. Villarreal, S. E. G., and Schaeffer, S. E. Local bilateral clustering for identifying research topics and groups from bibliographical data. *Knowledge and Information Systems 48*, 1 (2016), 179–199.
23. Vincent, L., and Soille, P. Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *IEEE Trans. Pattern Anal. Mach. Intell. 13* (1991), 583–598.
24. Zhou, D., Huang, J., and Schölkopf, B. Learning with hypergraphs: Clustering, classification, and embedding. In *NIPS* (2006).
25. Zhou, Y., Cheng, H., and Yu, J. X. Graph clustering based on structural/attribute similarities. *PVLDB 2* (2009), 718–729.