A Divide-and-Conquer Clustering Approach based on Optimum-Path Forest

Adán Echemendía Montero and Alexandre Xavier Falcão
Laboratory of Image Data Science,
Institute of Computing, University of Campinas,
{aemontero7,afalcao}@lids.ic.unicamp.br

***Abstract*—Data clustering is one of the main challenges when solving Data Science problems. Despite its progress over almost one century of research, clustering algorithms still fail in identifying groups naturally related to the semantics of the problem. Moreover, the technological advances add crucial challenges with a considerable data increase, which are not handled by most techniques. We address these issues by proposing a divide-and-conquer approach to a clustering technique, which is unique in finding one group per dome of the probability density function of the data — the Optimum-Path Forest (OPF) clustering algorithm. Our approach can use all samples, or at least many samples, in the unsupervised learning process without affecting the grouping performance and, therefore, being less likely to lose relevant grouping information. We show that it can obtain satisfactory results when segmenting natural images into superpixels.**

I. Introduction

Data Clustering is the problem of finding meaningful groups of similar samples according to a distance function and a mathematical representation. It is a well-known problem with large numbers of contributions and applications [1]. The samples are very often drawn from a set of possible categories (classes) related to the problem. In this case, the clustering algorithm should be able to learn a grouping model that can minimize the number of samples from distinct classes in the same cluster, while keeping the number of groups as small as possible (since the trivial solution is to consider each sample as a distinct group). Given that the design of such a grouping model must be done with no category information about the samples, the problem is referred to as unsupervised learning. The model should also be able to propagate group labels to new samples with a minimum mixture of classes per group.

The advances in software and hardware technology have provided large data collections from a wide variety of scenarios. A considerable number of clustering techniques have tried to handle large datasets [2]. Some techniques assign cluster labels based on the labels of the nearest neighbors in the feature space [3]. Approaches based on data summarization first reduce a large dataset into a relatively smaller one and then partition the reduced data [4], [5]. The samples of the original dataset receive the cluster labels that their corresponding representatives obtained in the partitioning phase. Techniques based on distributed computing split a clustering algorithm into a number of procedures that can be executed independently by a set of machines [6], [7]. Google's Map-Reduce framework [8] provides an effective method to analyze large amounts of data, especially when computing linear functions over the samples of data streams. Other approaches, in contrast to most clustering algorithms, only allow a single pass over the data stream [9], [10]. Techniques like CURE [11] perform a clustering over a reduced sample set of a large dataset and the result is transferred to the original data. CLARA [12] and CLARANS [13] are two classic large-scale clustering algorithms based on k-Medoids that rely on a sampling process to reduce the search space of the data.

Our focus here is on clustering algorithms based on the Optimum-Path Forest (OPF) framework [14], [15]. This framework interprets a training set as a graph, whose nodes are the samples and arcs are defined by an adjacency relation between samples. It exploits the "strength of connectedness" between samples in the feature space, as defined by a path-value function, for clustering and classification. In OPF, a grouping model or a pattern classifier is an optimum-path forest computed on the input graph. The group/class label assignment to new samples is performed by finding the root of the forest which would offer an optimum path to the new sample, as though that sample were part of the training set, and assigning the label of that root.

The OPF-clustering method [15] can be very effective when using all data to construct the forest, but it becomes prohibitive in memory requirement and processing time as samples and features per sample increase in number. Given that the datasets have grown large very rapidly, it is crucial to maintaining the OPF-clustering method viable with minimum loss in effectiveness. For large datasets, the authors in [15], [16] suggest the use of a small training set with randomly selected samples, which makes very efficient the construction of a node-weighted graph used by the method and the label propagation process to new samples. We name this technique OPF-Large-Data. It has already succeeded with training sets of about 400 voxels when classifying gray matter, white matter, and cerebral spinal fluid in magnetic resonance images of the brain with about 1.5 million voxels. However, the result of the grouping may be compromised whenever relevant information is lost in the construction of the training set by random sampling.

As the main contribution of this paper, we propose a two-level divide-and-conquer OPF-clustering approach suitable for large datasets. The idea is to divide the dataset into parts, use the OPF-clustering algorithm (or OPF-Large-Data) to group

samples in each part, and then combine the clustering results from each part. For validation, we compare this approach with OPF-Large-Data, k-Means, and other state-of-the-art methods for superpixel segmentation. The remainder of this paper is organized as follows. Section II describes the theoretical background on the OPF-clustering technique. Section III details the proposed approach. Section IV presents the experiments and discusses the obtained results. Finally, Section V states conclusion and provides directions for future work.

II. Optimum-Path Forest Clustering

In OPF clustering [15], the training samples are the nodes $s \in N$ of a graph (N, A_k), whose arcs $(s,t) \in A_k$ connect each sample with its k-nearest neighbors in the feature space (i.e., a k-nn graph). A probability density value is estimated at each sample s by

$$\rho(s) = \frac{1}{\sqrt{2\pi\sigma^2|A_k(s)|}} \sum_{\forall t \in A_k(s)} \exp\left(\frac{-d^2(s,t)}{2\sigma^2}\right), \quad (1)$$

where $A_k(s)$ is the set of k-nearest neighbors of s, $d(s,t)$ is a distance between s and t in the feature space, and $\sigma = \max_{\forall (s,t) \in A_k}\{\frac{d(s,t)}{3}\}$ guarantees that all samples in $A_k(s)$ are used for density computation. Let S be a set with one node per maximum of the probability density function ρ. In order to define one cluster per dome of ρ, a connectivity function f must assign a value to each path in the graph

$$f(\langle t \rangle) = \begin{cases} \rho(t) & \text{if } t \in S, \\ \rho(t) - \delta & \text{otherwise,} \end{cases} \quad (2)$$

where $\delta = min_{\forall (s,t) \in A_k | \rho(t) \neq \rho(s)}\{|\rho(t) - \rho(s)|\}$, when $\langle t \rangle$ is a trivial path (one single node), and

$$f(\langle \pi_s \cdot \langle s,t \rangle \rangle) = min\left\{f(\pi_s), \rho(t)\right\},$$

(the minimum value along the path), when a path π_s with terminus s is extended by an arc $(s,t) \in A_k$. Given that, the OPF-clustering algorithm maximizes a connectivity map by $V(t) = \max_{\forall \pi_t \in \Pi}\{f(\pi_t)\}$, where Π is the set of all paths. This process obtains a partition of (N, A_k) into an optimum-path forest P (predecessor map) rooted at S, a label map L, and a root map R. The set S is found during the algorithm. Each optimum-path tree in P is a cluster rooted at one maximum of ρ, the root value $R(t) \in S$ assigns the cluster representative (prototype) to each node $t \in N$, and its cluster label is assigned to $L(t)$. Different choices of $k \in [1, k_{\max}]$ lead to more (lower values) or less (higher values) clusters. The best value of k is then computed by executing the algorithm for each value in $[1, k_{\max}]$ and choosing the value of k that minimizes a normalized graph cut measure

$$C(k) = \sum_{i=1}^{c} \frac{W_i'}{W_i + W_i'}, \quad (3)$$

$$W_i = \sum_{\forall (s,t) \in A_k | L(s)=L(t)=i} \frac{1}{d(s,t)},$$

$$W_i' = \sum_{\forall (s,t) \in A_k | L(s)=i, L(t)\neq i} \frac{1}{d(s,t)},$$

for c clusters.

A. Extension to large datasets (OPF-Large-Data)

The competition procedure in the OPF-clustering technique takes $O(k|N| + |N|log|N|)$ operations when a binary heap is used. The estimation of the best k requires its computation several times. This method becomes unfeasible for a large number of samples as in a 2D/3D image with thousands/millions of pixels/voxels. OPF-Large-Data [15] deals with this problem by random sampling the set N into a considerably reduced training set $N' \subset N$. This makes the training process feasible, but the classification of new samples still requires the identification of their k-nearest neighbors. In [16], the authors solve this last problem by changing the training algorithm to output a list O of the nodes $s \in N'$ in their non-increasing order of optimum path values and an adjacency radius $w(s)$ (e.g., the maximum distance between s and its k-nearest nodes in N') to classify new samples. By giving higher priority to nodes with higher path values (i.e., by following the order of nodes in O), this variant classifies new samples t by identifying the node $s \in N'$, such that

$$s = argmax_{\forall s' \in O, d(s',t) \leq w(s')}\{V(s')\}, \quad (4)$$

and propagating to t the same cluster label $L(s)$.

III. Proposed Approach

The following algorithm describes our technique. We call it OPF-Blocks-2 in reference to the fact that it has two clustering levels. At the first level, the large number of samples in N is divided into b disjoint blocks (Line 1), so that each segment of the data (block) consists of approximately N/b samples. This number (N/b) must be reasonable and sufficient to extract useful clustering information from N. This data division can be random or based on some application-specific strategy. Then, the OPF-clustering algorithm is used to group the samples in each block (Line 4). This phase is easy to parallelize because the blocks are clustered separately. Let us assume that block i produces c_i clusters, for $i = 1, 2, \ldots, b$, or what is the same, block i can be summarized by c_i prototypes. Subsequently, all the $\sum_{i=1}^{b} c_i$ prototypes are taken as samples of a new dataset M (Line 5) to be grouped in the second level also by the OPF-clustering algorithm (Line 7). It is expected that this last result will reveal the natural number c of groups in the original dataset. Our assumption is that the samples in M summarize the original data, therefore, a clustering over these samples will represent a good approximation of the underlying partition of the dataset N. Finally, the group labels acquired in M are transferred to the original dataset N as follows. Line 9 copies the group labels of the samples in M to the same samples in N — i.e., the roots of optimum-path trees in the first level — and then to the samples of their optimum-path trees (Lines 13 and 14). This algorithm returns the label map L, the root map R, and the predecessor map P (optimum-path forest) defined in N.

This technique can be easily extended to a higher number of levels than two, but we found two levels enough for the

Input: Large dataset N, adjacency relation A_k, probability density function ρ, and number of blocks b.
Output: Label map L, predecessor map P, and root map R.

Divide N into b disjoint sets N_1,...,N_b.
Create empty set M.
for $i = 1$ **to** b **do**
  $(L_i, P_i, R_i) \leftarrow$ Execute the OPF-clustering algorithm in (N_i, A_k, ρ).
  Add the representative samples of N_i to M.
end for
$(L_m, P_m, R_m) \leftarrow$ Execute the OPF-clustering algorithm in (M, A_k, ρ).
for $s \in M$ **do**
  Set $L(s) \leftarrow L_m(s)$, $R(s) \leftarrow R_m(s)$, and $P(s) \leftarrow P_m(s)$.
end for
for $i = 1$ **to** b **do**
  for $s \in N_i \setminus M$ **do**
    Set $u \leftarrow R_i(s)$.
    Set $L(s) \leftarrow L(u)$, $R(s) \leftarrow R(u)$, $P(s) \leftarrow P_i(s)$.
  end for
end for

datasets used in this work. When the number of samples in a dataset is considered very large (for instance, the pixels of an image, where $|N| >> 200,000$), we prefer to partition the data into a smaller number of blocks and use the variant OPF-Large-Data to cluster each block rather than partition them into a larger number of blocks. By that, our algorithm can use the OPF-clustering technique with considerably larger training sets without degrading its execution time.

A. Improving the estimation of the k-nn graph

The parameter k represents the observation scale of the data in the feature space. The authors in [15] find the best value of k by exhaustive search within $[1, k_{\max}]$. As a drawback, for each candidate k, the technique must compute ρ for all nodes, execute the competition process, and evaluate the normalized cut (see Equation 3) produced by the group labels on the graph. Another problem with this technique is that it cannot partition the data into a predetermined number of clusters without playing with the parameter $k_{\max}$. In [17], the authors try to solve the issue by performing the clustering at different levels of abstractions (scales) to get as close as possible to the desired number of clusters. In our case, we are only interested in reducing the computational time for the search of k within $[1, k_{\max}]$. Therefore, our idea is to start the search at $k = k_{\max}$ and stop it whenever the first local minimum of the normalized cut function is found. It turns out that this heuristic produces good results with earlier search termination. Of course, the higher is $k_{\max}$ the more likely is that the exhaustive and heuristic searches will choose different values of k, being the chosen value in the former lower than the value selected by the latter. However, one can always reduce the upper limit $k_{\max}$ in order to make them equivalent again.

B. Algorithm for image segmentation

When a dataset consists of image pixels, the expected clustering result is a partition of the image into regions, called superpixels, such that the image objects can be represented by the union of their superpixels. We have addressed this problem by divide-and-conquer OPF clustering as follows. We first divide the image into a grid of blocks in order to take advantage of their spatial information. One question is how to choose the training samples when clustering each block by OPF-Large-Data. The number of blocks and their corresponding training set sizes are also important issues. Many blocks or larger training sets seems to allow for better results because of the usage of more training samples; however, these decisions slow down the run-time of the technique and may be unnecessary. Therefore, it is usually enough to work with few blocks and reduced training sets per block, as obtained by grid sampling in each block. Afterward, it is possible to merge adjacent clusters from neighboring blocks by means of a post-processing, a variant of the algorithm named OPF-Blocks-1, or to continue the process by clustering the roots from each block in a second level and then propagating the cluster labels through them to all image pixels (OPF-Blocks-2). The merging procedure in OPF-Blocks-1 consists of computing a color histogram for the superpixels and join adjacent pairs whose the Bhattacharyya coefficient [18] between them is close to 1. While OPF-Blocks-1 is limited to merge adjacent clusters from neighboring blocks, OPF-Blocks-2 can merge clusters from blocks in any part of the image. To produce the final segmentation, we need to apply a relabelling on the clustering result of both methods because the obtained clusters are not restricted to be compacted while the superpixels are. In fact, this is not the only post-processing that we recommend. We also propose to apply a shape smoothing (diffusion filter) to the resulting superpixels [19] and an area filter to remove noise (small sets of pixels). Figure 1 depicts all the pipeline of OPF-Blocks-2 for image segmentation.

IV. Experiments and Results

We compare the two clustering techniques derived from our proposal, OPF-Blocks-1 and OPF-Blocks-2, against two state-of-the-art superpixel generation methods (SLIC [20][1] and Quick-Shift [21][2]), one method based on the watershed transform [22][3], and two generic clustering algorithms (OPF-Large-Data[3] and k-Means[3]. To evaluate the methods across different application domains, we select databases involving natural, biological, and medical images. The first database corresponds to 50 natural and colorful images from the GrabCut database [23]. The second database is formed by 36 color images of different parasites where some are connected to impurities. The goal in these images is to isolate the pixels that represent the parasites. However, impurities may overlap the parasites and/or present similar sizes, shapes, colors, and

[1] http://ivrl.epfl.ch/supplementary_material/RK_SLICSuperpixels/
[2] http://www.vlfeat.org/download.html
[3] Our own implementation.

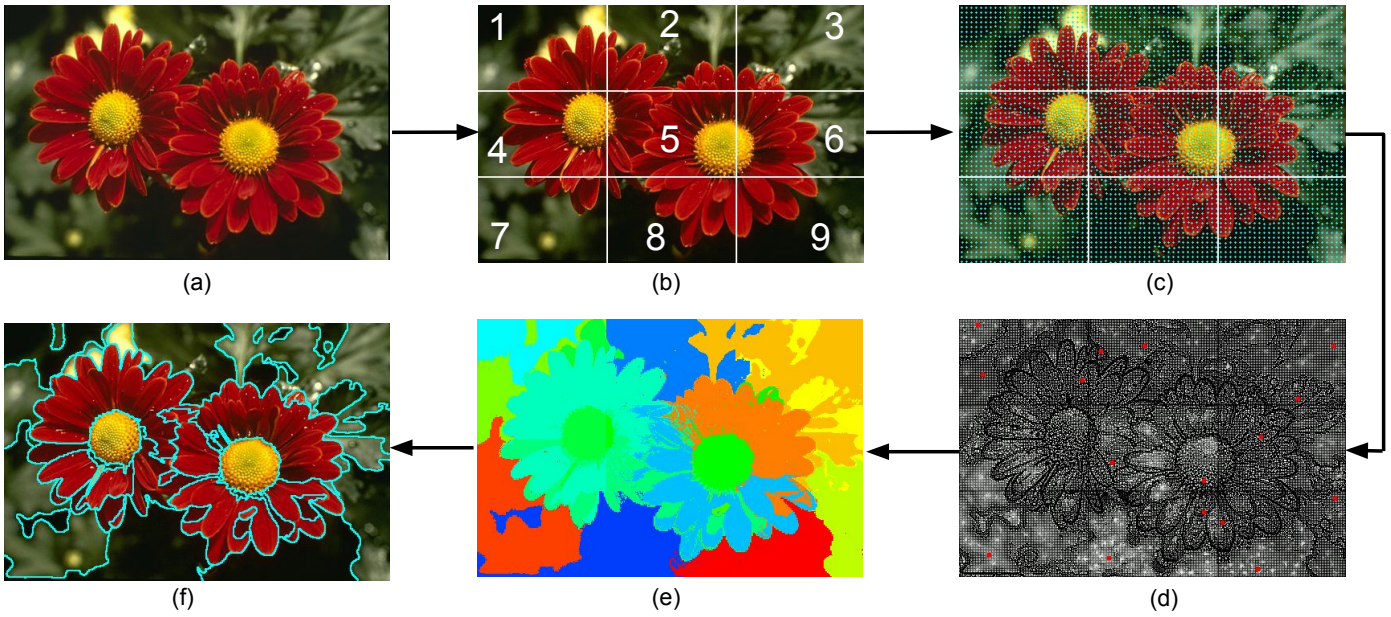


Fig. 1. Pipeline of OPF-Blocks-2 for image segmentation. (a) Input image. (b) The image is divided into 9 blocks. (c) There is selected a training set (blue points) by grid sampling in each block. (d) The PDF values of the samples are estimated separately in each block. Brighter values indicate samples with higher values of the PDF. Afterward, each block is clustered with OPF-Large-Data and the prototypes of the groups are promoted to the second level where they are clustered with the OPF algorithm. The roots of the final forest are indicated with red points. (e) The group labels obtained in the second level are propagated to all samples of the image. Each color indicates a different group. (f) Resulting superpixels after relabelling and filtering operations.

textures. The third database contains 29 images obtained from slices of 10 thoracic computed tomography (CT) studies. The object of interest is the liver in each slice.

To choose the descriptor for the images in the tested databases, we help ourselves with the t-Distributed Stochastic Neighbor Embedding (t-SNE) data visualization technique [24]. We define the descriptor for the samples of colorful images with both, the color and spatial information of the pixel. For the gray-scale images, corresponding to the CT slices, we determine a feature vector formed by the brightness value and the spatial information of the pixel, in addition to the brightness data of the pixels in the 8-neighborhood. Both descriptors allow separating the object samples from the background samples in the corresponding t-SNE projections.

We employ two widely used boundary adherence measures for evaluating the quality of superpixels. The first one is the boundary recall (BR) which measures the fraction of the ground-truth boundaries overlapping the segmentation boundaries in an image within a certain tolerance distance d of pixels. We use $d = 2$ for our experiments. The second used metric is the under-segmentation error (UE) [25] which does not penalize over-segmentation and indicates how well the superpixels adhere to the object boundaries. In addition, we also analyze the ability of the methods to retrieve the object of interest with the resulting partition. Assuming that all the samples from a cluster have the same label of their representative[4], we assign the correct label to each representative and propagate these labels to the remaining samples of the corresponding groups. In this way, we compute the Dice similarity coefficient (DSC or F_1 score) after applying some post-processing to the images resulted by the label propagation operation. We apply morphological open and close operations to reduce noise, and we only remain with the largest object component because we know that the evaluated images have a single object of interest. In the shown experiments, OPF-Large-Data is always executed with 1500 training samples when clustering each image or block in the case of the divide-and-conquer extensions. We choose this number because it usually allows getting a good result without compromising the efficiency of the technique, but in fact, it could be used any other number (e.g., 2000, 3000). The same happens when deciding the number of blocks to divide the images in the divide-and-conquer extensions. For this work, OPF-Blocks-1 and OPF-Blocks-2 divide the images into 4, 9, or 16 blocks at the first level, so in turn, they are using a training set with a size between 4 * 1500 and 16 * 1500 samples.

All experiments are executed on a server with a processor Intel Core i7-3770K CPU @ 3.50GHz x 8 and a memory RAM of 32GB. We randomly divide each database into two sets: a training set and a test set. The images in the training set are used to tune up the hyper-parameters of the compared methods. The best hyper-parameters are found by grid search. All the results shown below correspond to the execution of the methods exclusively on the images of the test set. We average the metrics obtained in these images to create the comparative graphics. The databases are divided as follows.

[4]In OPF-based methods, the representatives are the roots of the optimum-path trees; in k-Mean and SLIC, the representatives are the closer samples to the centers of the clusters.

- **GrabCut database: 15** images for training and **35** images for testing.
- **Parasites/Impurites database: 12** images for training and **24** images for testing.
- **Liver database: 11** images for training and **18** images for testing.

Figures 2, 3, and 4 compare the segmentation results of the methods in the images of the GrabCut database. It can be seen that OPF-Blocks-2 outperforms the others techniques, according to the boundary recall and the under-segmentation error. OPF-Blocks-1 has the second best performance, only surpassed by OPF-Large-Data up to 50 superpixels. OPF-Large-Data overcomes SLIC, k-Means, Quick-Shift, and Watershed in boundary recall, but it is surpassed by these methods according to the under-segmentation error from 150 superpixels on. Quick-shift has the worst performance up to 80 superpixels but from there, its result improves. According to the Dice metric, after true label propagation by each cluster representative, OPF-Large-Data has the best results up to 50 groups, while the divide-and-conquer extensions exceed all others techniques from 60 groups on.

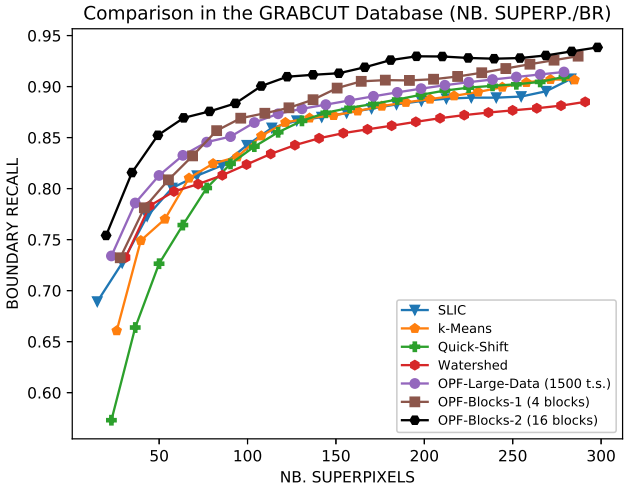


Fig. 2. Comparison between the segmentation results of the methods, according to the boundary recall, in the GrabCut database.

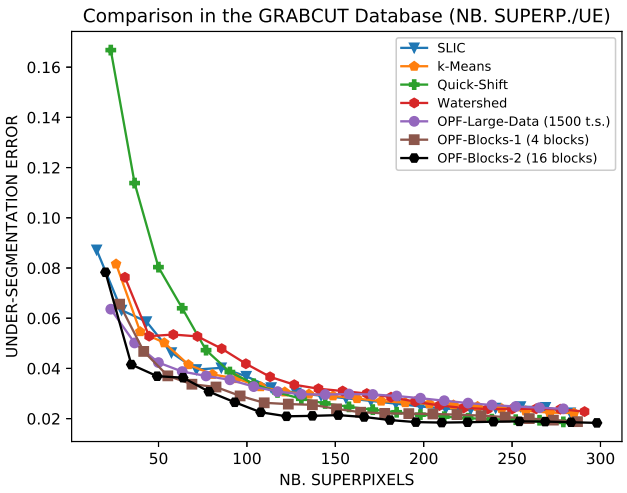


Fig. 3. Comparison between the segmentation results of the methods, according to the under-segmentation error, in the GrabCut database.

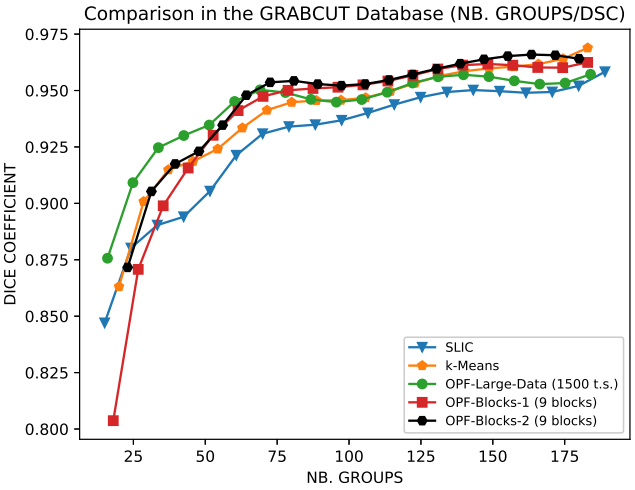


Fig. 4. Comparison between the segmentation results of the methods after true label propagation from the cluster prototypes, according to the Dice coefficient, in the GrabCut database.

Considering the run-time efficiency, Watershed and SLIC are superior to the others. The two methods never reach a second, regardless of the number of superpixels generated. OPF-Large-Data and the divide-and-conquer extensions have a similar performance, taking approximately 1.5 seconds to cluster each image into any number of superpixels. The performance of k-Means deteriorates as the number of superpixels increases, reaching almost 10 seconds to produce about 300 superpixels per image. Instead, the performance of Quick-Shift improves as the number of superpixels augments. It takes 7 seconds to cluster an image into 300 superpixels. These results are equivalent in the three databases evaluated.

Figures 5, 6, and 7 reveal comparisons between the segmentation results of the tested methods in the Parasites/Impurities database. It can be observed that OPF-Blocks-2 has the best performance both in boundary recall and the under-segmentation error. k-Means and OPF-Blocks-1 also obtain very good results, being the first capable of overcoming OPF-Blocks-2 in boundary recall from 180 superpixels on. OPF-Large-Data and SLIC have similar behavior in both measures. Watershed is the technique with the worst performance and only beats Quick-Shift up to 60 superpixels. OPF-Large-Data obtains the worst results in the Dice metric. OPF-Blocks-2 and k-Means get the best results, however, the second exceeds the first one up to 60 groups.

The graphics from Figures 8, 9, and 10 compare the effectiveness of the methods when segmenting the images of the Liver database. In these experiments, we discard all the

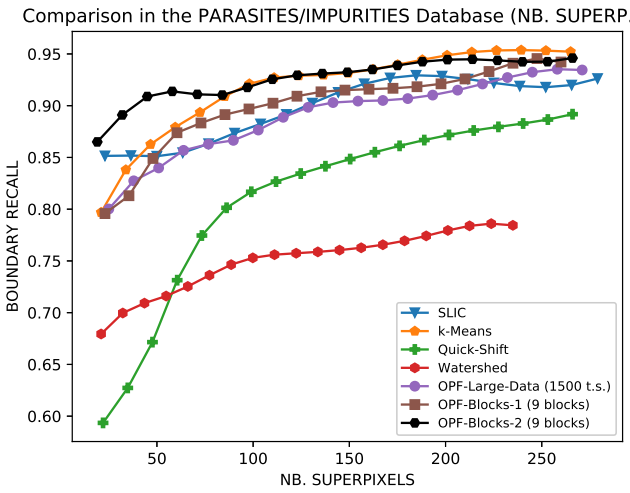


Fig. 5. Comparison between the segmentation results of the methods, according to the boundary recall, in the Parasites/Impurities database.

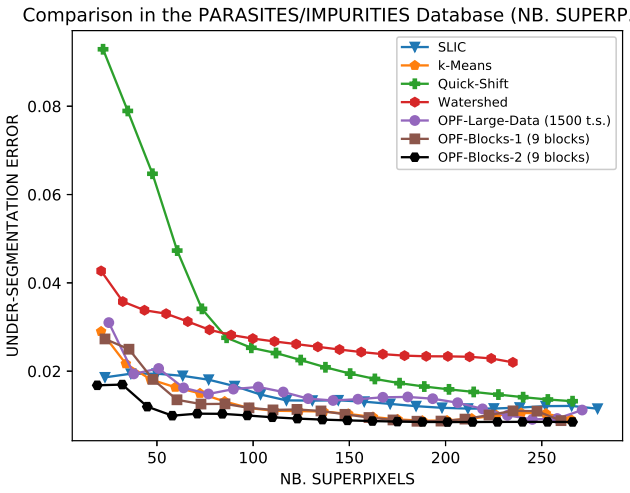


Fig. 6. Comparison between the segmentation results of the methods, according to the under-segmentation error, in the Parasites/Impurities database.

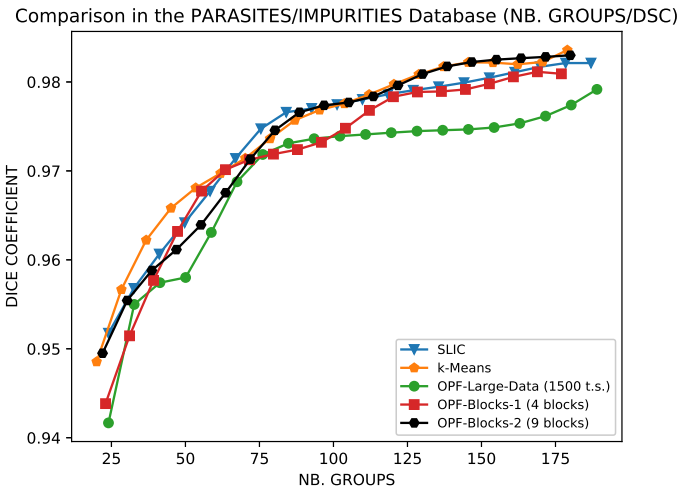


Fig. 7. Comparison between the segmentation results of the methods after true label propagation from the cluster prototypes, according to the Dice coefficient, in the Parasites/Impurities database.

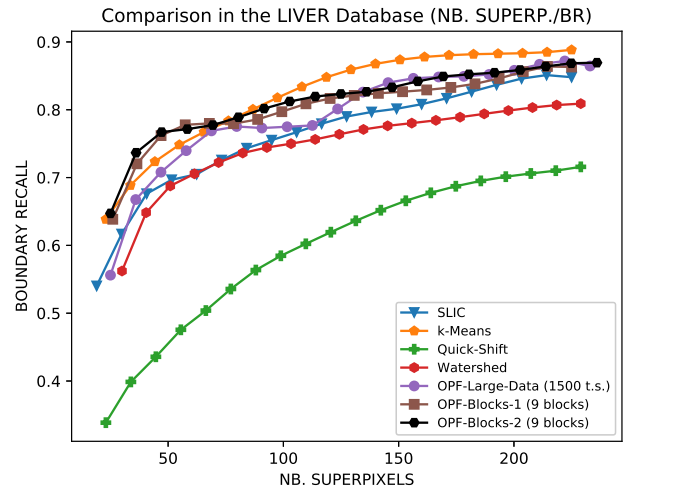


Fig. 8. Comparison between the segmentation results of the methods, according to the boundary recall, in the Liver database.

dark pixels because we know they are part of the background. OPF-Blocks-1, OPF-Blocks-2, and Watershed exceed all other techniques in under-segmentation error; while OPF-Large-Data, SLIC, and k-Means get similar results according to this metric. The results corresponding to the boundary recall metric are different. Watershed has the second-worst performance while the divide-and-conquer extensions surpass SLIC and OPF, but are exceeded by k-Means from 100 superpixels on. According to the Dice metric, OPF-Blocks-2 has the best performance for any number of groups. The other methods have a similar behavior from 75 groups on.

Some segmentation results are exhibited in Figures 11, 12, and 13. The yellow arrows indicate leaking between object and background. In addition, some true label propagation results by cluster representatives are shown in Figures 14 and 15.

V. Conclusion

In order to address the large dataset problem, we presented an extension of the OPF-clustering technique that exploits a divide-and-conquer model with two levels. We also proposed a local search to find the adjacency parameter k within the interval $[1, k_{\max}]$ which does not affect effectiveness. Our approach, OPF-Blocks-2, was evaluated in the image segmentation scenario and outperformed the baselines in most experiments. We also assessed OPF-Blocks-1, a method that only makes use of the first level and merges the generated superpixels between blocks. Despite its excellent results, it is not competitive with OPF-Blocks-2.

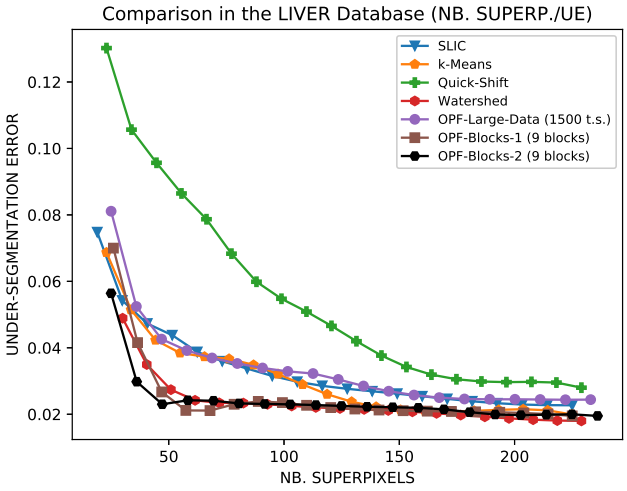


Fig. 9. Comparison between the segmentation results of the methods, according to the under-segmentation error, in the Liver database.

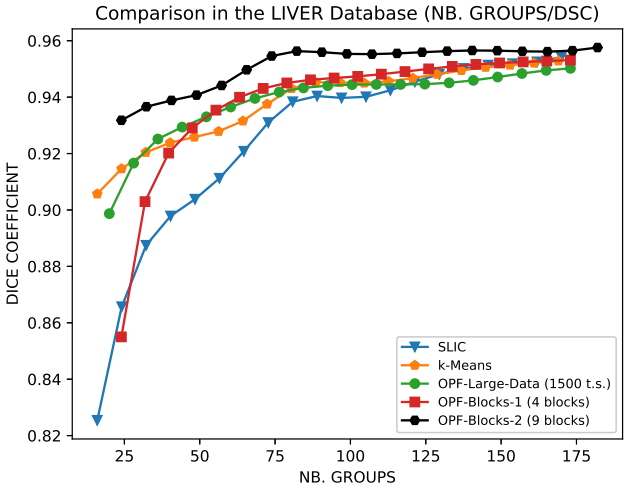


Fig. 10. Comparison of the segmentation results of the methods after true label propagation from the cluster prototypes, according to the Dice coefficient, in the Liver database.

As future work, the upper limit $k_{\max}$ per block, the block sizes, and their spatial locations, should take into account the data entropy inside the blocks of the first level, in order to preserve all natural clusters when grouping samples in the second level. Higher entropy may require lower values of $k_{\max}$ and reduced block sizes. The technique can also be explored in several contexts, such as active learning, saliency detection, and brain tissue segmentation from magnetic resonance images [16]. Extensions to more than two levels might also be relevant in some applications.

Acknowledgment

The authors thank CAPES, FAPESP (2014/12236-1) and CNPq (302970/2014-2).

References

[1] C. C. Aggarwal and C. K. Reddy, *Data clustering: algorithms and applications*. CRC press, 2013.

[2] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern recognition letters*, vol. 31, no. 8, pp. 651–666, 2010.

[3] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration." *VISAPP (1)*, vol. 2, no. 331-340, p. 2, 2009.

[4] T. Zhang, R. Ramakrishnan, and M. Livny, "Birch: an efficient data clustering method for very large databases," in *ACM Sigmod Record*, vol. 25, no. 2. ACM, 1996, pp. 103–114.

[5] G. Karypis and V. Kumar, "Multilevelk-way partitioning scheme for irregular graphs," *Journal of Parallel and Distributed computing*, vol. 48, no. 1, pp. 96–129, 1998.

[6] W. Zhao, H. Ma, and Q. He, "Parallel k-means clustering based on mapreduce," in *IEEE International Conference on Cloud Computing*. Springer, 2009, pp. 674–679.

[7] G. Karypis and V. Kumar, "Parallel multilevel series k-way partitioning scheme for irregular graphs," *Siam Review*, vol. 41, no. 2, pp. 278–300, 1999.

[8] J. Dean and S. Ghemawat, "Mapreduce: a flexible data processing tool," *Communications of the ACM*, vol. 53, no. 1, pp. 72–77, 2010.

[9] P. S. Bradley, U. M. Fayyad, C. Reina *et al.*, "Scaling clustering algorithms to large databases." in *KDD*, 1998, pp. 9–15.

[10] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for clustering evolving data streams," in *Proceedings of the 29th international conference on Very large data bases-Volume 29*. VLDB Endowment, 2003, pp. 81–92.

[11] S. Guha, R. Rastogi, and K. Shim, "Cure: an efficient clustering algorithm for large databases," in *ACM Sigmod Record*, vol. 27, no. 2. ACM, 1998, pp. 73–84.

[12] L. Kaufman and P. J. Rousseeuw, *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 1990, vol. 344.

[13] R. T. Ng and J. Han, "Clarans: A method for clustering objects for spatial data mining," *IEEE transactions on knowledge and data engineering*, vol. 14, no. 5, pp. 1003–1016, 2002.

[14] J. P. Papa, A. X. Falcão, and C. T. Suzuki, "Supervised pattern classification based on optimum-path forest," *International Journal of Imaging Systems and Technology*, vol. 19, no. 2, pp. 120–131, 2009.

[15] L. M. Rocha, F. A. Cappabianco, and A. X. Falcão, "Data clustering as an optimum-path forest problem with applications in image analysis," *International Journal of Imaging Systems and Technology*, vol. 19, no. 2, pp. 50–68, 2009.

[16] F. A. Cappabianco, A. X. Falcão, C. L. Yasuda, and J. K. Udupa, "Brain tissue mr-image segmentation via optimum-path forest clustering," *Computer Vision and Image Understanding*, vol. 116, no. 10, pp. 1047–1059, 2012.

[17] L. Afonso, A. Vidal, M. Kuroda, A. X. Falcão, and J. P. Papa, "Learning to classify seismic images with deep optimum-path forest," in *Graphics, Patterns and Images (SIBGRAPI), 2016 29th SIBGRAPI Conference on*. IEEE, 2016, pp. 401–407.

[18] T. Kailath, "The divergence and bhattacharyya distance measures in signal selection," *IEEE transactions on communication technology*, vol. 15, no. 1, pp. 52–60, 1967.

[19] N. Moya, "Interactive segmentation of multiple 3d objects in medical images by optimum graph cuts," Ph.D. dissertation, Masters thesis, Institute of Computing, University of Campinas (UNICAMP), 2015.

[20] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "Slic superpixels," Tech. Rep., 2010.

[21] A. Vedaldi and S. Soatto, "Quick shift and kernel methods for mode seeking," *Computer vision–ECCV 2008*, pp. 705–718, 2008.

[22] R. A. Lotufo, A. X. Falcão, and F. A. Zampirolli, "Ift-watershed from gray-scale marker," in *Computer Graphics and Image Processing, 2002. Proceedings. XV Brazilian Symposium on*. IEEE, 2002, pp. 146–152.

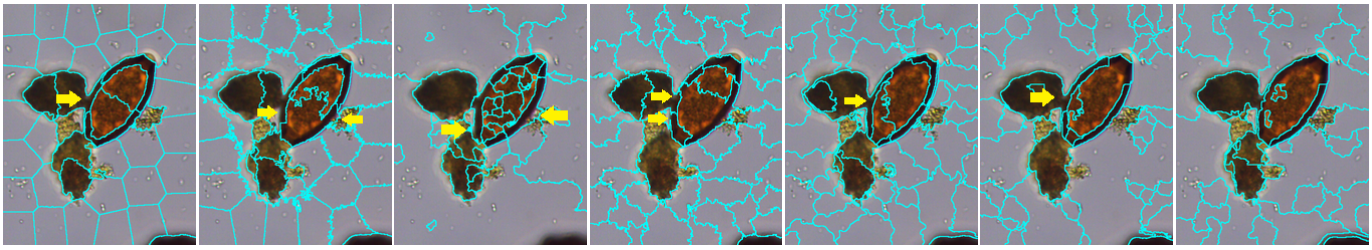
[23] C. Rother, V. Kolmogorov, and A. Blake, "Grabcut: Interactive foreground extraction using iterated graph cuts," in *ACM transactions on graphics (TOG)*, vol. 23, no. 3. ACM, 2004, pp. 309–314.

[24] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[25] P. Neubert and P. Protzel, "Superpixel benchmark and comparison," in *Proc. Forum Bildverarbeitung*, 2012, pp. 1–12.

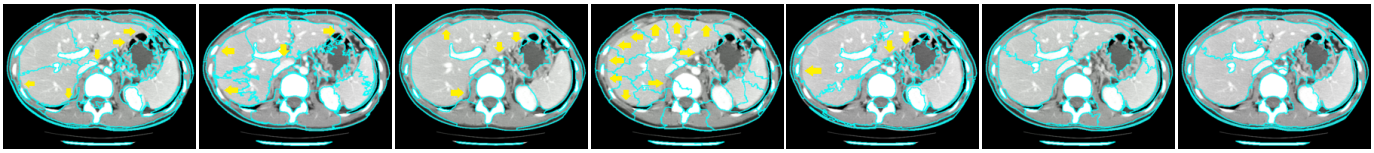
(a) k-Means (b) SLIC (c) Watershed (d) Quick-Shift (e) OPF-Large-Data (f) OPF-Blocks-1 (g) OPF-Blocks-2

Fig. 11. Segmentation results with approximately 100 superpixels using the compared methods in an image of the GrabCut database.



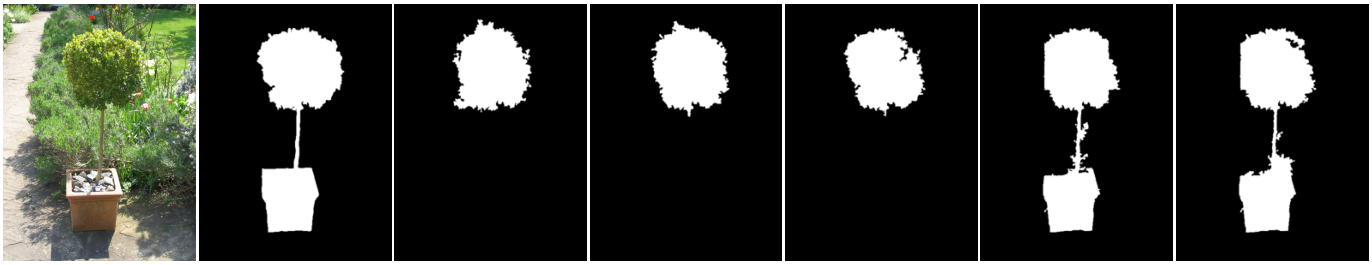
(a) k-Means (b) SLIC (c) Watershed (d) Quick-Shift (e) OPF-Large-Data (f) OPF-Blocks-1 (g) OPF-Blocks-2

Fig. 12. Segmentation results with approximately 50 superpixels using the compared methods in an image of the Parasites/Impurities database.



(a) k-Means (b) SLIC (c) Watershed (d) Quick-Shift (e) OPF-Large-Data (f) OPF-Blocks-1 (g) OPF-Blocks-2

Fig. 13. Segmentation results with approximately 30 superpixels using the compared methods in an image of the Liver database.



(a) Image (b) Ground-Truth (c) k-Means (d) SLIC (e) OPF-Large-Data (f) OPF-Blocks-1 (g) OPF-Blocks-2

Fig. 14. Resulting masks of the compared methods for about 100 groups, after true label propagation from the cluster prototypes, in an image of the GrabCut database.

(a) Image (b) Ground-Truth (c) k-Means (d) SLIC (e) OPF-Large-Data (f) OPF-Blocks-1 (g) OPF-Blocks-2

Fig. 15. Resulting mask of the compared methods for about 50 groups, after true label propagation from the cluster prototypes, in an image of the Liver database.