Minimum Spanning Tree Based Clustering Algorithms

Oleksandr Grygorash, Yan Zhou, Zach Jorgensen
School of Computer and Information Sciences
University of South Alabama, Mobile, AL 36688 USA
ogrygorash@usouthal.edu, zhou@cis.usouthal.edu, zdjorgen@usouthal.edu

Abstract

The minimum spanning tree clustering algorithm is known to be capable of detecting clusters with irregular boundaries. In this paper, we propose two minimum spanning tree based clustering algorithms. The first algorithm produces a k-partition of a set of points for any given k. The algorithm constructs a minimum spanning tree of the point set and removes edges that satisfy a predefined criterion. The process is repeated until k clusters are produced. The second algorithm partitions a point set into a group of clusters by maximizing the overall standard deviation reduction, without a given k value. We present our experimental results comparing our proposed algorithms to k-means and EM. We also apply our algorithms to image color clustering and compare our algorithms to the standard minimum spanning tree clustering algorithm.

1. Introduction

A spanning tree is an acyclic subgraph of a graph G, which contains all the vertices from G. The minimum spanning tree (MST) of a weighted graph is the minimum-weight spanning tree of that graph. With the classical MST algorithms [18, 13, 15], the cost of constructing a minimum spanning tree is $O(m \log n)$, where m is the number of edges in the graph, n is the number of vertices. More efficient algorithms for constructing MSTs have also been extensively researched [12, 7, 8]. These algorithms promise close to linear time complexity under different assumptions. A Euclidean minimum spanning tree (EMST) is a spanning tree of a set of n points in a metric space ($\mathbb{E}^n$), where the length of an edge is the Euclidean distance between a pair of points in the point set.

The MST clustering algorithm is known to be capable of detecting clusters with irregular boundaries [24]. Unlike traditional clustering algorithms, the MST clustering algorithm does not assume a spherical shaped clustering structure of the underlying data. The EMST clustering algorithm [17, 24] uses the Euclidean minimum spanning tree of a graph to produce the structure of point clusters in the n-dimensional Euclidean space. Clusters are detected to achieve some measure of optimality, such as minimum intracluster distance or maximum intercluster distance [1]. The (E)MST clustering algorithm has been widely used in practice. An example application of the algorithm is image color clustering in web image analysis. Web images are usually supplied with shaded or multicolored complex backgrounds, often found in photographs, maps, engineering drawings and commercial advertisements [20]. Analyzing web images is a challenging task due to their low spatial resolution and the large number of colors in the images [19]. The purpose of color clustering in web image analysis is to reduce thousands of colors to a representative few that clearly differentiate objects of interest in an image.

Once the MST is built for a given input, there are two different ways to produce a group of clusters. If the number of clusters k is given in advance, the simplest way to obtain k clusters is to sort the edges of the minimum spanning tree in descending order of their weights, and remove the edges with the first $k-1$ heaviest weights [1, 22]. We call this approach the *standard EMST clustering algorithm* or **SEMST** in the rest of the paper. The second approach does not require a preset cluster number. Edges, that satisfy a predefined inconsistency measure, are removed from the tree. We use the inconsistency measure suggested by Zahn in [24], and therefore we call the clustering algorithm *Zahn's EMST clustering algorithm* or **ZEMST**.

In this paper, we propose two EMST based clustering algorithms to address the issues—undesired clustering structures and an unnecessarily large number of clusters, commonly faced by the SEMST and the ZEMST algorithm respectively. Our first algorithm assumes the number of clusters is given. The algorithm constructs a EMST of a point set and removes the *inconsistent edges* that satisfy an inconsistency measure. The process is repeated to create a hierarchy of clusters until k clusters are obtained. The second algorithm partitions the point set into a group of clusters by maximizing the overall standard deviation reduction. The final number of clusters is determined by finding the local minimum of the standard deviation reduction function.

The rest of the paper is organized as follows. In Section 2 we review some existing work on graph-based clustering algorithms and their applications. We present two EMST based clustering algorithms in Section 3. Section 4 presents the experimental results comparing our proposed clustering algorithms to k-means, EM, SEMST and ZEMST. Section 5 concludes our work and discusses future directions.

2. Related work

Clustering algorithms based on minimum and maximum spanning trees have been extensively studied. In the mid 80's, Avis [2] found an $O(n^2 \log^2 n)$ algorithm for the min-max diameter 2-clustering problem. Asano, Bhattacharya, Keil, and Yao [1] later gave an optimal $O(n \log n)$ algorithm using maximum spanning trees for minimizing the maximum diameter of a bipartition. The problem becomes NP-complete when the number of partitions is beyond two [11]. Asano, Bhattacharya, Keil, and Yao also considered the clustering problems in which the goal is to maximize the minimum intercluster distance. They gave an $O(n \log n)$ algorithm for computing a k-partition of a point set by removing the $k-1$ longest edges from the minimum spanning tree constructed from that point set [1].

Zahn [24] proposes to construct an MST of a point set and delete inconsistent edges — the edges, whose weights are significantly larger than the average weight of the nearby edges in the tree. Zahn's inconsistency measure is defined as follows. Let e denote an edge in the MST of the point set, and v_1 and v_2 be the end nodes of e, w be the weight of e. A *depth d neighborhood N* of an end node v of an edge e is defined as a set of all edges that belong to all the paths of length d originating from the node v, excluding the paths that include the edge e. Let N_1 and N_2 be the depth d neighborhoods of the end nodes v_1 and v_2. Let $\overline{w}_{N_1}$ be the average weight of edges in N_1 and σ_{N_1} be its standard deviation. Similarly, let $\overline{w}_{N_2}$ be the average weight of edges in N_2 and σ_{N_2} be its standard deviation. The inconsistency measure requires one of the following three conditions hold:

1. $w > \overline{w}_{N_1} + c \times \sigma_{N_1}$ or $w > \overline{w}_{N_2} + c \times \sigma_{N_2}$
2. $w > \max(\overline{w}_{N_1} + c \times \sigma_{N_1},\ \overline{w}_{N_2} + c \times \sigma_{N_2})$
3. $\dfrac{w}{\max(c \times \overline{w}_{N_1},\ c \times \overline{w}_{N_2})} > f,$

where c and f are preset constants. All the edges of a tree that satisfy the inconsistency measure are considered *inconsistent* and are removed from the tree. This results in a set of disjoint subtrees each representing a separate cluster. Note that the resulting cluster structure is affected by the depth d of the neighborhood N, and the constants c and f.

Eldershaw and Hegland [6] re-examine the limitations of many clustering algorithms that assume the underlying clusters of a data set are spherical. They provide a broader definition of a cluster based on the rule of transitivity: if two points p_1 and p_2 are close to the same point p_0, p_1 and p_2 are the members of the same cluster in which they are indirectly related through p_0, despite the fact that p_1 and p_2 are not constrained by distance measure. They present a clustering algorithm by constructing a graph using Delaunay triangulation, and removing the edges between neighbors that are longer than a cut-off point. Next, they apply a graph partitioning algorithm to find the isolated connected components in the graph, and each discovered component is treated as a cluster. Similar to Zahn's MST clustering algorithm, this algorithm divides a point set into a certain number of clusters at once by removing all edges in the graph that are longer than a threshold. Unlike in Zahn's method, they choose a cut-off point which corresponds to the "global" minimum of a function that measures how well the consistent edges and the inconsistent edges in the graph are separated.

More recently, Päivinen [16] proposed a scale-free minimum spanning tree (SFMST) clustering algorithm which constructs a scale free network and outputs clusters containing highly connected vertices and those connected to them.

The MST clustering algorithm has been widely used in practice. Xu (Ying), Olman and Xu (Dong) [22] use an MST to represent multidimensional gene expression data. They point out that an MST-based clustering algorithm does not assume that data points are grouped around centers or separated by a regular geometric curve. Thus the shape of a cluster boundary has little impact on the performance of the algorithm. They describe three objective functions and the corresponding clustering algorithms for computing a k-partition of the spanning tree for any predefined $k > 0$. The first algorithm simply removes the $k-1$ longest edges so that the total weight of the k subtrees is minimized. The second objective function is defined to minimize the total distance between the center and each data point in a cluster. The algorithm first removes $k-1$ edges from the tree, which creates a k-partition. Next, it repeatedly merges a pair of adjacent partitions and finds its optimal 2-clustering solution. They observe that the algorithm quickly converges to a local minimum. The third objective function is defined to minimize the total distance between the "representative" of a cluster and each point in the cluster. The representatives are selected so that the objective function is optimized. This algorithm runs in exponential time in the worst case.

Xu and Uberbacher [23] partition a gray-level image into connected homogeneous regions by constructing an MST from the image. The tree partitioning algorithm minimizes the sum of the variations of the gray-levels of all subtrees, and the gray-levels of two adjacent subtrees are required to be significantly different. Each subtree contains several gray-levels and represents a homogeneous region in the image. Other applications of the MST clustering algorithm in the area of image processing can be found in [21, 9].

Lopresti and Zhou [14] suggest an RGB color clustering method by constructing a Euclidean minimum spanning tree. Each distinct color in a given image is considered as a point in the three dimensional RGB color space. Thus each color is a node in the EMST. The weight of an edge is the Euclidean distance between two color nodes in the tree. They compute the average distance of the edges in the EMST once it is built. Subsequently the edges that are "longer" than the average weight by a predetermined amount are removed from the tree, leaving a set of disjoint subtrees. Colors in each subtree are the members of a color cluster. They point out that the EMST based color clustering algorithm may fail when dealing with textures and when there are a large number of colors in an image.

3. Our EMST based clustering algorithms

By and large, the inherent cluster structure of a point set in a metric space is closely related to how objects or concepts are embedded in the point set. In practice, the approximate number of embedded objects can sometimes be acquired with the help of domain experts. Other times, this information is hidden and unavailable to the clustering algorithm. In this section, we present two EMST-based clustering algorithms, one assumes a given cluster number and we call it a *hierarchical EMST clustering algorithm* or **HEMST**, the other does not and we call it a *maximum standard deviation reduction clustering algorithm* or **MSDR**.

3.1. Our HEMST clustering algorithm

Given a point set S in $\mathbb{E}^n$ and the desired number of clusters k, the hierarchical method starts by constructing an MST from the points in S. The weight of an edge in the tree is the Euclidean distance between the two end points. Next, the average weight $\overline{w}$ of the edges in the entire EMST and its standard deviation σ are computed; any edge with a weight $w > \overline{w} + \sigma$ is removed from the tree. This leads to a set of disjoint subtrees $S_T = \{T_1, T_2, \ldots\}$. Each of the subtrees T_i is treated as a cluster, which has a centroid c_i. If the number of the subtrees $|S_T| < k$, $k - |S_T|$ additional longest edges are removed from the entire edge set of S_T to produce k disjoint subtrees. If $|S_T| > k$, a representative point is identified for each subtree. The representative point r_i for a cluster $T_i \in S_T$ is defined as the point $p \in T_i$ that is closest to the centroid c_i of T_i. In other words, $d(p, c_i) = \min_{p_j \in T_i} d(p_j, c_i)$. Once all the representative points are found, each point in a particular subtree is replaced with the representative point of the subtree, thus reducing the number of points in S to $|S_T|$. A EMST is constructed from the representative points of the clusters, and the same tree partitioning process is repeated. When $|S_T| = k$, the clustering process is considered complete, having produced the required k clusters.

Instead of removing the $k-1$ longest edges all at once to create a k-partition, our algorithm first partitions the point set into a number of more compact clusters. Subsequently, a new partitioning process is repeated on the EMST constructed from a much smaller set of representative points. Each representative point is close to the centroid of the subset created in the previous round. The algorithm eventually outputs k representative points $r_1, \ldots, r_k$ for the final required k clusters. Each point in the given point set is grouped according to its membership in a particular subtree. A point p is assigned to a cluster i if $p \in T_i$. The detailed pseudocode is given in Table 1.

Table 1. The pseudocode of HEMST.

```
Algorithm: HEMST(k)
Initialize $n_c \leftarrow 1$ //number of clusters
Let $S$ be the point set
Let $e$ be an edge in the EMST constructed from $S$
Let $w_e$ be the weight of $e$
Let $\sigma$ be the standard deviation of the edge weights
Let $S_T = \emptyset$ be the set of disjoint subtrees of the EMST

Repeat
  Construct an EMST from $S$
  Compute the average weight $\overline{w}$ of all the edges
  Compute the standard deviation $\sigma$ of the edges
  For each $e \in EMST$
    If $w_e > \overline{w} + \sigma$
    Remove $e$ from EMST
    $n_c \leftarrow n_c + 1$
    $S_T = S_T \cup \{T'\}$ //$T'$ is the new disjoint subtree
  /$\star$ If the number of clusters $n_c$ is less than $k$,
  remove $n_c - k$ longest edges so that $n_c = k$ $\star$/
  If $n_c < k$
    While $n_c \neq k$
      Remove the current longest edge
      $n_c \leftarrow n_c + 1$
      $S_T = S_T \cup \{T'\}$ //$T'$ is the new disjoint subtree
    Return $k$ clusters
  /$\star$ If the number of clusters $n_c$ is greater than $k$ $\star$/
  If $n_c > k$
    Compute the centroid $c_i$ of each $T_i \in S_T$
    Find the representative $r_i \in T_i$ closest to $c_i$
    $S = \cup_{T_i \in S_T}\{r_i\}$
until $n_c = k$
Return $k$ clusters
```

Figure 1 [24] illustrates a typical example of the cases in which simply removing the $k-1$ longest edges does not necessarily output the desired cluster structure. This is also a good example where the nearest neighbor or the single linkage [10] method does not work well. Our HEMST algorithm, on the other hand, will find the appropriate representative points for the desired clusters. Figure 2 shows the possible distribution of the representative points after a few rounds. Eventually, if given $k = 2$, our algorithm will detect the two-cluster structure.

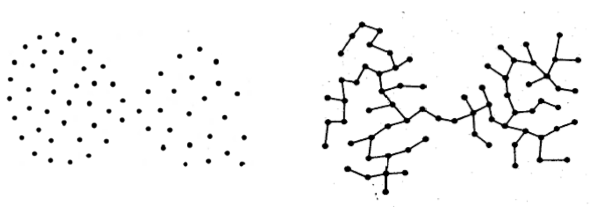


Figure 1. Clusters connected through a point.

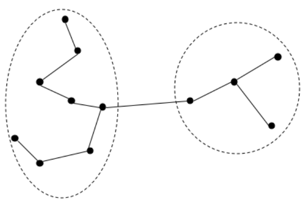


Figure 2. The representative points of the two clusters connected through a point.

3.2. Our MSDR clustering algorithm

Our HEMST algorithm assumes that the desired number of clusters is given in advance. In practice, determining the number of clusters is often coupled with discovering the cluster structure. In this section, we present another EMST-based clustering algorithm—the *maximum standard deviation reduction clustering algorithm*, or **MSDR** which does not require a predefined cluster number.

Our MSDR algorithm first constructs a EMST from the given point set S. Next, it computes the standard deviation of the edges in the EMST, and removes an edge to obtain a set of two disjoint subtrees such that the overall standard deviation reduction is maximized. This edge removing process is repeated to create more disjoint subtrees until the overall standard deviation reduction is within a threshold. The desired number of clusters is obtained by finding the local minimum of the standard deviation reduction function.

Given a point set S, our algorithm groups the data points in such a way that each pair of points in the group are either directly or indirectly close to each other in the metric space. Two points are directly close to each other if the distance between them is small. They are indirectly close to each other if they are far apart, but there exists a point in the same group, to which both points are close. This objective allows us to detect clusters that have more complex geometric shapes than spherical clusters. For a given point set and the corresponding minimum spanning tree, we partition the MST into a set of disjoint subtrees $S_K = \{T_1, T_2, \ldots, T_K\}$ such that the following objective function is satisfied:

$$\begin{cases} S_K = \mathrm{argmax}(\sigma(T_0) - \sigma(S_K)) \\ |\Delta\sigma(S_K) - \Delta\sigma(S'_K)| < |\epsilon \cdot (\Delta\sigma(S_K) + 1)| \end{cases}$$

where T_0 denotes the original EMST and S_K denotes the final partition $S_K = \{T_1, T_2, \ldots, T_K\}$ of T_0 that results in maximum overall standard deviation reduction, $\sigma(T_0)$ denotes the standard deviation of the edges in T_0, $\sigma(S_K)$ denotes the weighted average of the standard deviation of the edges in the disjoint trees $T_{j=1,\ldots,K} \in S_K$:

$$\sigma(S_K) = \frac{\sum_{\forall T_j \in S_K} |T_j| \cdot \sigma(T_j)}{\sum_{\forall T_j \in S_K} |T_j|} \quad (1)$$

$\Delta\sigma(S_K)$ denotes the maximum standard deviation reduction that leads to the partition $S_K = \{T_1, T_2, \ldots, T_K\}$, $\Delta\sigma(S'_K)$ denotes the maximum standard deviation reduction leading to the immediate precedent of S_K, i.e. $S'_K = \{T_1, \ldots, T_{K-1}\}$. ϵ is a small positive value that determines when the iterative edge removing process stops. The desired number of clusters is determined by applying polynomial regression on $\Delta\sigma(S_K)_i$ produced in each iteration i of the edge removing process. The number of clusters, as a critical point of the regression function, with a positive second derivative is chosen as the final number of clusters. Points in each subtree $T_j \in S_K$ are members of a cluster. The pseudocode of our MSDR algorithm is given in Table 2.

Note that Bansal, Blum and Chawla [3] have recently introduced correlation clustering which also does not require the desired number of clusters. The clustering problem they consider is a complete graph in which each edge is labeled qualitatively as "+" (similar) or "-" (dissimilar). The objective is to find a clustering that maximizes the number of agreements or minimizes the number of disagreements with the edge labels. They proved NP-hardness and provided approximation algorithms which have been further improved by others [5, 4]. Unlike correlation clustering, our MSDR algorithm aims to maximize the overall standard deviation reduction measured quantitatively from a given point set in the Euclidean space.

Table 2. The pseudocode of MSDR.

```
Algorithm: MSDR()
Let $S$ be the point set
Let $T_0$ be the EMST constructed from $S$
Let $S_K$ be the set of disjoint subtrees of $T_0$
Let $e$ be an edge in $S_K$
Let $\sigma(S_K)$ be the overall StdDev of all edges in $S_K$
Let $\sigma(T_j)$ be the StdDev of edges in subtree $T_j \in S_K$
Let $\Delta\sigma(S_K)[i] = 0$ be the maximum StdDev reduction
    after the removal of an edge $e$ at each iteration $i$
Let $\epsilon = 0.0001$

$S_K = \{T_0\}$
$\sigma(S_K) = \sigma(T_0)$
$i = 0$
Repeat
    $i \leftarrow i + 1$
    $temp = \sigma(S_K)$
    /⋆ Choose an edge that leads to max StdDev reduction
    once it is removed from $S_K$ ⋆/
    For each $e \in S_K$
        Assume $e$ is removed from $S_K$ thus $S_K = \bigcup_{j=1}^{i+1} T_j$
        $\sigma(S_K) = \frac{\sum_{\forall T_j \in S_K} |T_j| \cdot \sigma(T_j)}{\sum_{\forall T_j \in S_K} |T_j|}$
        /⋆ Compute StdDev reduction ⋆/
        If $\Delta\sigma(S_K)[i] < \sigma(S_K) - temp$
            $\Delta\sigma(S_K)[i] = \sigma(S_K) - temp$
    Remove $e$ from $S_K$ that corresponds to $\Delta\sigma(S_K)[i]$
    $\sigma(S_K) = temp - \Delta\sigma(S_K)[i]$
until $|\Delta\sigma(S_K)[i] - \Delta\sigma(S_K)[i-1]| <$
        $|\epsilon \cdot (\Delta\sigma(S_K)[i] + 1)|$

$f(j) = PolyRegression(\bigcup_{j=1}^{i} \Delta\sigma(S_K)[j])$
/⋆ No. of clusters corresponds to the 1st local minimum ⋆/
$K = \min(j \in [1, i])$ that satisfies $f'(j) = 0$ & $f''(j) > 0$
Return $S_K = \{T_1, \ldots, T_K\}$
```

4. Experimental results

We set up two experiments. In the first experiment, we selected three clustering problems and compared the two proposed algorithms to k-means and EM respectively. In our second experiment, we compared our proposed algorithms to the standard EMST based algorithms—SEMST and ZEMST in image color clustering.

4.1. HEMST vs. k-Means & MSDR vs. EM

We selected three relatively difficult clustering problems in this experiment. The first problem is presented in Figure 3, in which two clusters are desired. Each cluster is formed as a curving irregular shaped line. The second problem shown in Figure 4 contains two clusters, with one inside the other. The third problem shown in Figure 5 contains two clusters, each with a non-homogeneous density.

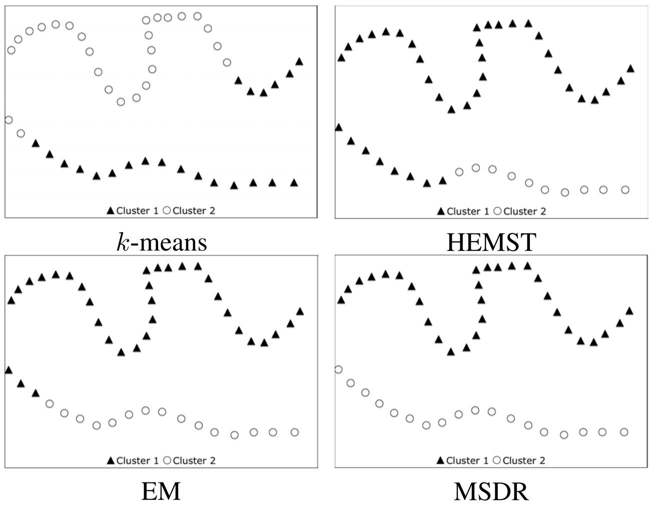


Figure 3. Two clusters formed by two lines.

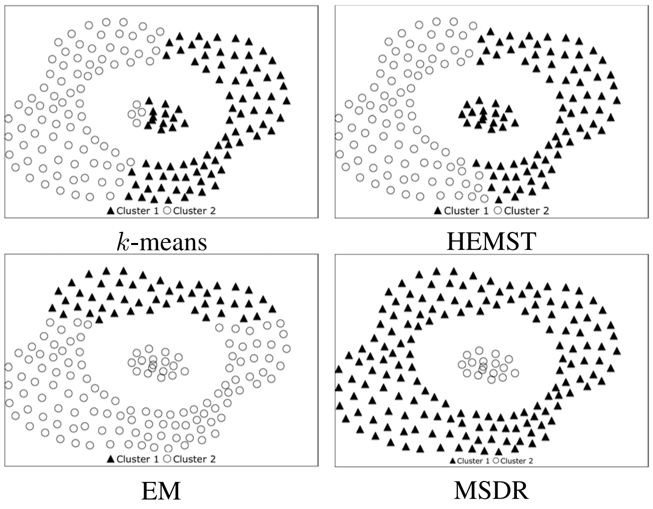


Figure 4. Two clusters—one inside the other.

Both HEMST and k-means require a preset cluster number. The EM algorithm determines the number of clusters through cross validation. Our MSDR algorithm selects the number of clusters with the largest second derivative. As we can see in Figure 3, k-means breaks both clusters and mixes them up into two undesired groups. HEMST and EM fragment one of the clusters, and only MSDR successfully identifies the clusters as desired. Similarly, in Figure 4, k-means fragments both the surrounding and the central cluster, while HEMST and EM manage to identify the

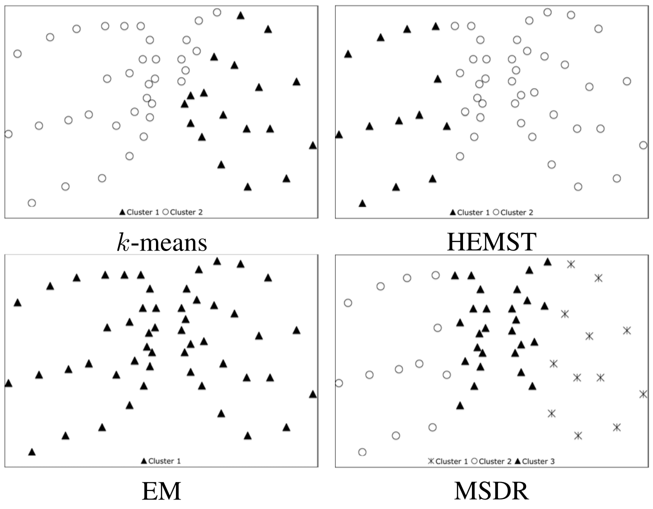


Figure 5. Clusters with non-homogeneous densities.

central cluster, although the surrounding cluster is separated into two. Again, only MSDR successfully outputs the correct clusters. In Figure 5, both k-means and HEMST tend to group points in a high density region into one cluster. EM only outputs one cluster, while MSDR outputs three clusters, grouping two high density regions into one cluster and identifying low density regions as two clusters on opposite sides. As can be observed, our MSDR algorithm is most successful in identifying the desired cluster structures. HEMST is slightly better than the k-means algorithm.

4.2. Image color clustering with SEMST & ZEMST

As mentioned in earlier sections, given k, SEMST produces clusters by removing the $k-1$ longest edges. ZEMST removes inconsistent edges without a preset k value. Typical issues each algorithm commonly faces include:

1. For a given k, simply removing the $k-1$ heaviest edges is not sufficient to obtain the desired cluster structure.
2. For an unknown k, removing inconsistent edges often produces an unnecessarily large number of clusters for a given input, or leads to undesired partitions. More problematically, the performance of the algorithm heavily relies on a set of constants that must be determined by the users.

These problems are illustrated in Figure 6. On the left is the original JPEG image containing 5328 different colors, even though to human eyes, there are only five different colors including the background color. The middle image is a result of color clustering using SEMST with $k = 5$, which removes the four heaviest edges to create five clusters. However, to human eyes there are only three colors left in the image. On the right, the color clusters are created by ZEMST. Even though the image looks very close to the original, the number of clusters is 51 which is far greater than the ideal five clusters.

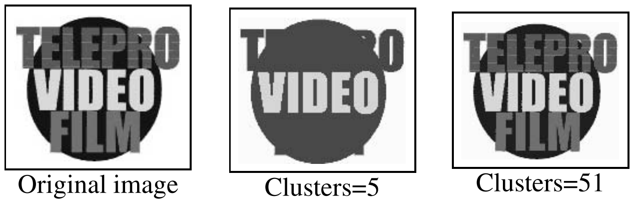


Figure 6. Problems with color clustering.

Next we present the results of our second experiment, comparing our proposed algorithms to HEMST and ZEMST on a collection of 35 GIF and JPEG images, many of which contain a large number of colors.

4.2.1 Color clustering with a given k value

All distinct colors in a given image are used to construct a minimum spanning tree as an aid to color clustering. Each distinct color represents a node in the tree, and the edges are represented by the Euclidean distances between the RGB values of two nodes.

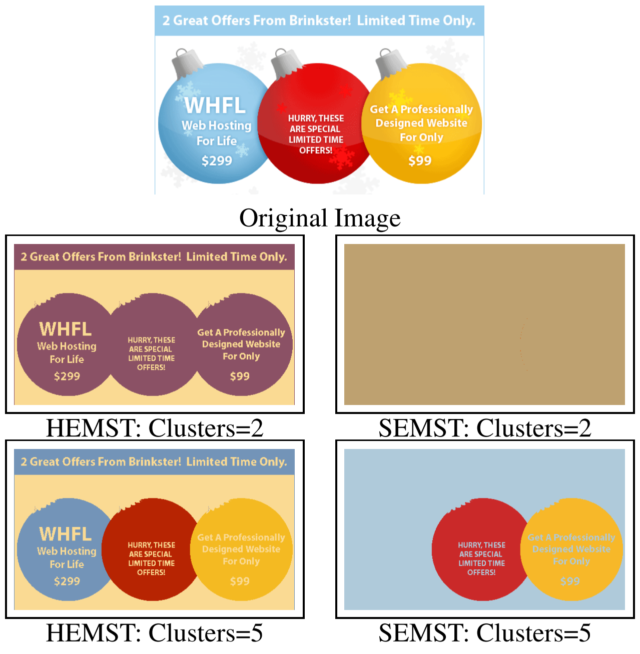


Figure 7. Image brinsker.gif before/after color clustering using HEMST and SEMST.

Figure 7 shows the results of our HEMST and the

SEMST algorithm on a GIF image containing 128 distinct colors. On the left are the color clustering results of our HEMST algorithm, on the right are the results of SEMST. When $k = 2$ the output of our algorithm has managed to catch most of the objects in the original image, while the output of the SEMST algorithm was not able to detect any object in the image. When $k = 5$, our algorithm catches even more details of the original image.

Figure 8 and 9 show two other image examples and the clustering results of the two algorithms. The first image contains 69 distinct colors. With our technique, when $k = 2$ it is sufficient to identify all the objects including the text in the image. The SEMST algorithm was not able to output anything significant. The second image contains 189 distinct colors. With only two colors ($k = 2$), our algorithm allows us to see clearly all the street names and the location of the destination. The SEMST algorithm managed to output a red star which indicates the destination. When $k = 8$ (8 representative colors), the output of our algorithm is almost identical to the original one, while the SEMST algorithm still could not output the street names and other detailed information on the map.

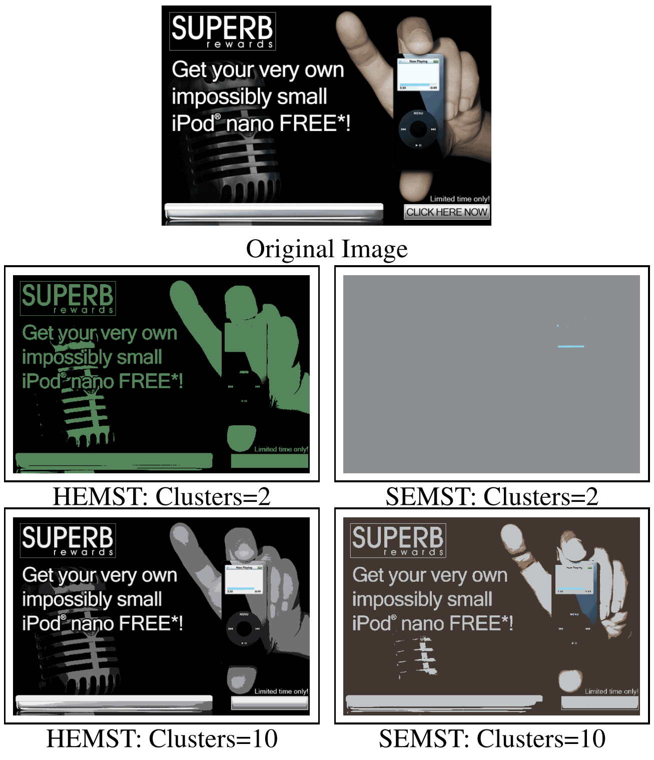


Figure 8. Image ipodnano.gif before/after color clustering using HEMST and SEMST.

We compared the two algorithms on 35 different images. Due to space limit, we do not show all the results. It is apparent that given a cluster number k, our HEMST algorithm

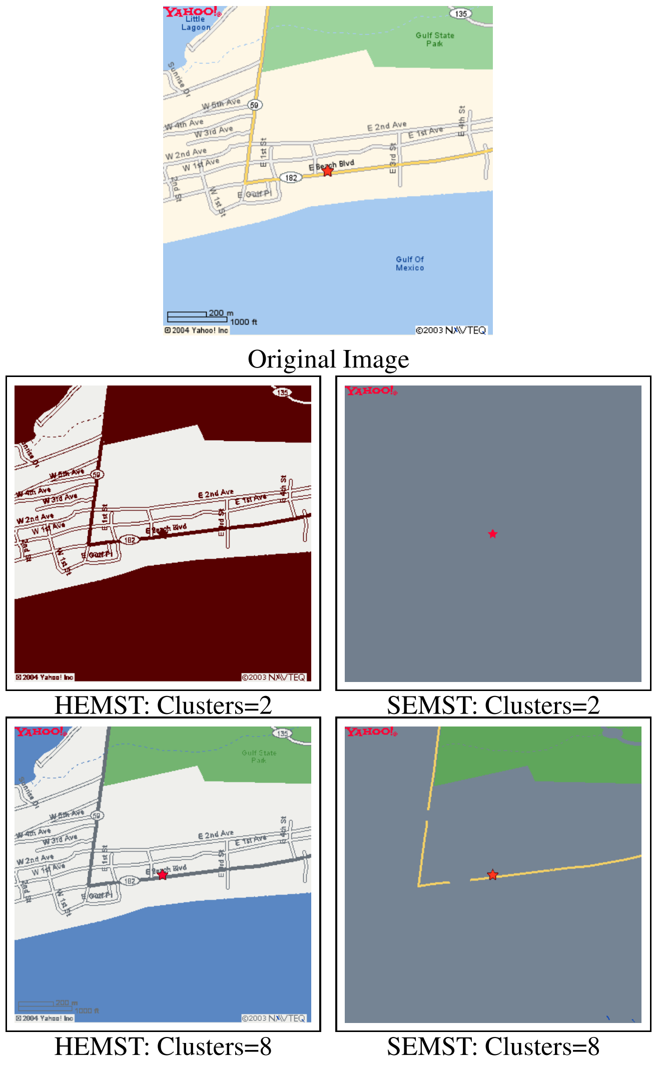
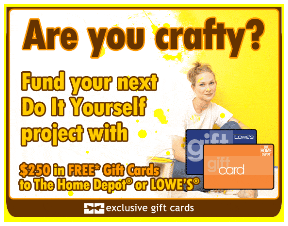


Figure 9. Image map.gif before/after color clustering using HEMST and SEMST.

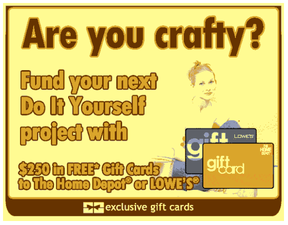
is much more effective than the SEMST algorithm that simply removes the $k - 1$ longest edges from the tree. Our algorithm is able to pick a very few most representative colors from a large number of colors in the original image, and still manage to capture all the objects in the images.

4.2.2 Color clustering without a given k value

In this experiment, the desired number of clusters as well as the structure of color clusters are unknown to the algorithms. We compared our MSDR algorithm to Zahn's EMST clustering algorithm [24] which also does not require a given cluster number. For Zahn's algorithm, we applied three different criteria to detect inconsistent edges in the tree as discussed in earlier sections: consider nearby edges on one side of the edge, consider nearby edges on both sides, consider the ratio of the average weights of nearby edges on both sides. Due to the complex nature of images, our MSDR algorithm selects the cluster number corresponding to a local minimum of the regression function in this experiment. For Zahn's method, we have to choose the size of the neighborhood explored, the number of standard deviations in excess to the average weight, often differently for different images in order to obtain better results. For each of the first two criteria, we choose three different neighborhood sizes: search depth into the tree d = 2, 3, 4, and three standard deviation factors $c = \{0.25, 0.75, 1.0\}$. We did not use the factor numbers suggested by Zahn because the algorithm could not output anything interesting with those factors. For the last criterion, we use the same search depths, and three different ratios $f = \{1.0, 1.25, 1.5\}$. Figure 10 shows the original image of a gift card and the result of our MSDR clustering algorithm. There are 58 colors in the original image. Our algorithm output eight color clusters and was able to capture nearly all the objects in the image.



Original Image



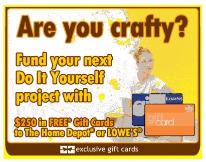
MSDR: Clusters=8

Figure 10. The original image giftcard.gif and the output of our MSDR algorithm.

Due to space limit, Figure 11 only shows the results of the ZEMST algorithm with the number of clusters closest possible to our MSDR output using the three different criteria with different parameter choices. As can be observed, the quality of the output relies heavily on the parameter choices. In fact, the output cluster number varies from 2 to 38 as the choice of the parameters varies. In practice, determining the parameter combinations could be rather challenging. In our experiment, we tested all parameter combinations on all 35 images. The general observation is that our MSDR algorithm is able to produce a much smaller number of colors while preserving objects embedded in all images, compared to the ZEMST algorithm.

5. Conclusions and future work

We have demonstrated that our proposed EMST-based clustering algorithms are very effective when applied to various clustering problems. Our HEMST clustering algorithm assumes a given cluster number. The algorithm gradually



Zahn(1):Colors=27
c=1.00, d=2

Zahn(1):Colors=28
c=1.00, d=3

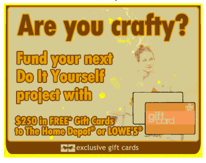
Zahn(1):Colors=27
c=1.00, d=4

Zahn(2):Colors=10
c=1.00, d=2

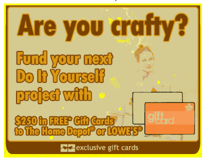
Zahn(2):Colors=11
c=1.00, d=3

Zahn(2)::Colors=11
c=1.00, d=4

Zahn(3):Colors=13
f=1.25, d=2



Zahn(3):Colors=12
f=1.25, d=3



Zahn(3):Colors=12
f=1.25, d=4

Figure 11. ZEMST using the 1st, 2nd, 3rd criteria with different parameter choices.

finds a set of k representative points that serve as an "attractor" to points nearby, and outputs the inherent cluster structure subsequently. We have shown that our algorithm works much more reliably than the simple SEMST clustering algorithm. Our MSDR algorithm automatically determines the desired number of clusters. The objective function is defined to maximize the overall standard deviation reduction. Our algorithm does not require the users to select and try various parameter combinations in order to get the desired output. In the future, we will explore and test our proposed clustering algorithms in various domains. We will further study the rich properties of the EMST-based clustering methods in solving different clustering problems.

References

[1] T. Asano, B. Bhattacharya, M. Keil, and F. Yao. Clustering algorithms based on minimum and maximum spanning trees. In *Proceedings of the 4th Annual Symposium on Computational Geometry*, pages 252–257, 1988.

[2] D. Avis. Diameter partitioning. *Discrete and Computational Geometry*, 1:265–276, 1986.

[3] N. Bansal, A. Blum, and S. Chawla. Correlation clustering. In *Proceedings of the 43rd FOCS*, pages 238–247, 2002.

[4] M. Charikar, V. Guruswami, and A. Wirth. Clustering with qualitative information. *Journal of Computer and System Sciences*, 71(3):360–383, 2005.

[5] E. Demaine and N. Immorlica. Correlation clustering with partial information. In *Proceedings of the 6th RANDOM-APPROX*, pages 1–13, 2003.

[6] C. Eldershaw and M. Hegland. Cluster analysis using triangulation. In B. Noye, M. Teubner, and A. Gill, editors, *Computational Techniques and Applications: CTAC97*, pages 201–208. World Scientific, 1997.

[7] M. Fredman and D. Willard. Trans-dichotomous algorithms for minimum spanning trees and shortest paths. In *Proceedings of the 31st Annual IEEE Symposium on Foundations of Computer Science*, pages 719–725, 1990.

[8] H. Gabow, T. Spencer, and R. Tarjan. Efficient algorithms for finding minimum spanning trees in undirected and directed graphs. *Combinatorica*, 6(2):109–122, 1986.

[9] R. Gonzalez and P. Wintz. *Digital Image Processing (2nd Edition)*. Addison-Wesley, Reading, MA, USA, 1987.

[10] J. Gower and G. Ross. Minimum spanning trees and single linkage cluster analysis. *Applied Statistics*, 18:54–64, 1969.

[11] D. Johnson. The np-completeness column: An ongoing guide. *Journal of Algorithms*, 3:182–195, 1982.

[12] D. Karger, P. Klein, and R. Tarjan. A randomized linear-time algorithm to find minimum spanning trees. *Journal of the ACM*, 42(2):321–328, 1995.

[13] J. Kruskal. On the shortest spanning subtree and the traveling salesman problem. In *Proceedings of the American Mathematical Society*, pages 48–50, 1956.

[14] D. Lopresti and J. Zhou. Locating and recognizing text in www images. *Information Retrieval*, 2:177–206, 2000.

[15] J. Nesetril, E. Milková, and H. Nesetrilová. Otakar boruvka on minimum spanning tree problem: Translation of both the 1926 papers, comments, history. *DMATH: Discrete Mathematics*, 233, 2001.

[16] N. Päivinen. Clustering with a minimum spanning tree of scale-free-like structure. *Pattern Recogn. Lett.*, 26(7):921–930, 2005.

[17] F. Preparata and M. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, New York, NY, USA, 1985.

[18] R. Prim. Shortest connection networks and some generalization. *Bell Systems Technical Journal*, 36:1389–1401, 1957.

[19] J. Sun, H. Yu, and Y. Katsuyama. Effective text extraction and recognition for www images. In *Proceedings of the 2003 ACM symposium on Document engineering*, pages 115–117, 2003.

[20] V. Wu, R. Manmatha, and E. Riseman. Finding text in images. In *Proceedings of the 2nd intl. conf. on Digital Libraries*, pages 1–10, 1997.

[21] Y. Xu, V. Olman, and E. Uberbacher. A segmentation algorithm for noisy images: design and evaluation. *Pattern Recognition Letters*, 19:1213–1224, 1998.

[22] Y. Xu, V. Olman, and D. Xu. Minimum spanning trees for gene expression data clustering. *Genome Informatics*, 12:24–33, 2001.

[23] Y. Xu and E. Uberbacher. 2d image segmentation using minimum spanning trees. *Image and Vision Computing*, 15(1):47–57, 1997.

[24] C. Zahn. Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Transactions on Computers*, C-20:68–86, 1971.