KAM Series:

DISCRETE MATHEMATICS AND COMBINATORICS
OPERATIONS RESEARCH
MATHEMATICAL LINGUISTICS

No.
90 - 189

Connected admissible hierarchical clustering

M. Křivánek

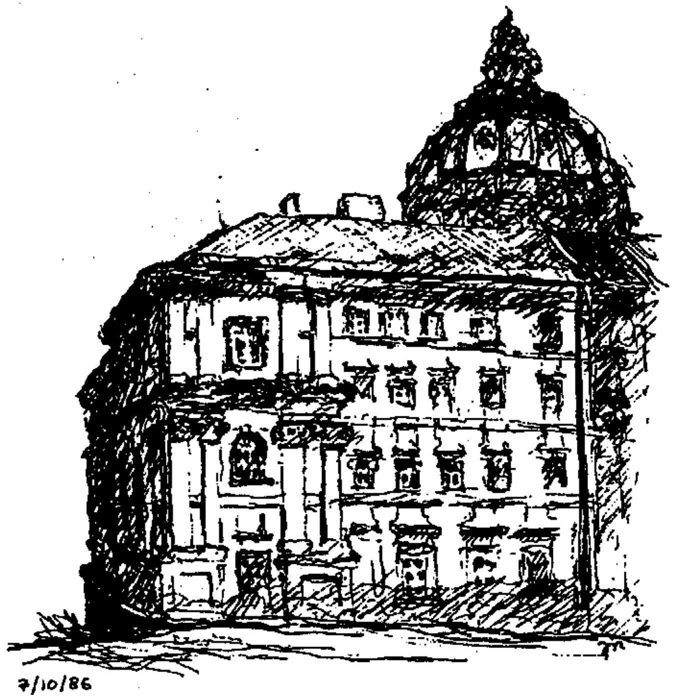
December 1990

Department of
Applied Mathematics

Faculty of
Mathematics and Physics

Charles University

(KAM MFF UK)
Malostranské nám. 25
118 00 Praha 1
Czechoslovakia



CONNECTED ADMISSIBLE HIERARCHICAL CLUSTERING*

MIRKO KŘIVÁNEK

Department of Computer Science
Charles University
Malostranské nám. 25
118 00 Praha 1, Czechoslovakia

Summary.

Using the concept of connected admissibility we investigate the algorithmical and geometrical aspects of generalized single linkage hierarchical clustering procedures. The main emphasis is concentrated on efficient algorithms in the plane within the framework of both agglomerative and divisive schemes of hierarchical clustering paradigm.

I. Introduction and background

Much research effort during the two past decades has been directed towards the development of hierarchical clustering procedures [2,9,10]. Significant steps were made in systematization and comparisons of numerous existing methods. In this paper we discuss hierarchical clustering algorithms from the point of view of their so-called connected admissibility [7] and time efficiency.

Let us review and formalize the goal of hierarchical clustering. Through out this paper let X be a set of n points in some metric space which are to be clustered. Further, let their mutual dissimilarity be given by a corresponding metric σ. The aim of hierarchical clustering is to produce a *hierarchy* $H \in \mathcal{P}(\mathcal{P}(X))$ of subsets of X , called *clusters*, such that

(i) $X \in H$ and $(\forall x \in X)\{x\} \in H$,
(ii) $(\forall h, h' \in H)\ h \cap h' \in \{\emptyset, h, h'\}$,
(iii) $\emptyset \notin H$.

The hierarchy is called *binary* if it contains exactly $2n-1$ clusters. Let us remark that with little loss of generality we may assume that the hierarchies in question are supposed to be binary. Of course, various optimality criteria can be placed on an output hierarchy H in accordance with vague demand of *closeness* and *separability* which is usually read as follows :
" $\forall h, h' \in H,\ h \cap h' = \emptyset$, the distances in clusters h and h' are smaller than distances between clusters h and h'."

Very often closeness and separability is measured by means of some function W assigning weights to clusters. An example is as follows :

$$W(h) = \max\{\sigma(x,y)\ ;\ x, y \in h\}.$$

The operator 'max' can be formally replaced by e.g.'+', 'min', 'median', 'k-quantile' or any other "order statistics".

Moreover, we expect W to be *monotone*, i.e

$$(\ \forall h, h' \in H)\ h \subset h' \ => \ W(h) \leq W(h').$$

A hierarchy H is said to be *feasible* for a given weight function W defined on its clusters if W is monotone.

*The paper was presented at DIANA III conference, Bechyně (Czechoslovakia), June 1990.

Since we want to construct hierarchies of maximal closeness and separability we define the computational problem **HIC** of hierarchical clustering by the aid of feasibility :

INSTANCE : A set X such that $\text{card}(X) = n$;

OUTPUT : A binary hierarchy H on X such that it is feasible for a given weight function W and $\sum \{W(h), h \in H\}$ is minimal.

Speaking about computational complexity of the problem **HIC** we must distinguish between the sizes of the input data. The input instance is viewed as a complete graph $K(X)$ on X with edge weights given by σ. Thus the size of the input is $O(n^2)$. However, an alternative approach considers the input instance to be of size $O(n)$. This is in the case when the distance between two objects (points) can be computed from its coordinates in constant time. The latter case is discussed in Section IV. for Euclidean distance in the plane. Now, let us return to the weight function W. Mostly we shall use the midrange form of W :

"for every $h \in H$ such that $\exists\ h', h''$, $h = h \cup h''$ put
$W(h) = \min\{\sigma(x,y)\ ;\ x \in h' \text{ and } y \in h''\} + \max\{\sigma(x,y)\ ;\ x \in h' \text{ and } y \in h''\}$."

We have

THEOREM 1. *The problem* **HIC** *of hierarchical clustering is* NP*-hard under the midrange form of the weight function* W.

Proof. We prove the theorem for a metric σ such that $\text{Range}(\sigma) = \{0,1,2\}$. The NP-hardness follows by the polynomial transformation from the problem Δ [8] :

INSTANCE : A graph G on $3k$ vertices without induced complete subgraphs on 4 vertices;

QUESTION : Is there a vertex partition of G into k classes such that each class induces a triangle in G ?

Now, given an instance of Δ we consider the instance of **HIC** defined by

$$\sigma(x,y) = \begin{cases} 1, & \text{if } \{x,y\} \text{ is an edge of } G \\ 2, & \text{if } \{x,y\} \text{ is not an edge of } G. \end{cases}$$

It is easily verified that there exists a hierarchy H on vertex set of G such that $\sum W(h) = 4k + 3(k-1)$ if and only if there exists a vertex partition of G into triangles. □

Let us conclude the section by two remarks. First, note that in entirely similar manner we can show the intractability for the other before-mentioned formal operators of the weight function W. However, the only exceptional case is 'min' operator where the optimal hierarchy is produced by single linkage strategy in polynomial time, cf. next section.

At the second place, since the hierarchy on X is in one-to-one correspondence with some ultrametric on X the computational problem of hierarchical clustering is mostly formalized as the problem of the best fit approximation of given dissimilarity measure on X by an ultrametric on X. The underlying computational problems are, however, also intractable [11].

II. Agglomerative and divisive approximation paradigm

Since the general optimization computational problem of hierarchical clustering is NP-hard there is no hope for an exact polynomial algorithm. However, the bicriterion goal of closeness and separability suggests two natural approximation schemes. If we prefer the closeness criterion, we start in the construction of a hierarchy with one-element clusters and at each step of the algorithm the closest pair of clusters is agglomerated. This strategy is called *agglomerative.* On the other hand, the process of the construction can be done in the opposite way by means of separability. Starting with the whole set X at each step one cluster is divided into two parts. This strategy is called *divisive.* Especially hierarchical agglomerative clustering algorithms are well understood [3,9]. Two most popular and in some sense extremal cases of agglomerative algorithms are called *single* and *complete* linkages. In general, precisely the notion of the 'closest pair' makes the difference between agglomerative algorithms. In single linkage two clusters with the minimum distance are agglomerated while in complete linkage two clusters with the minimum maximum distance are agglomerated. Note that single and complete linkages can be viewed as approximation procedures to the problem **HIC** specified by the 'min' and 'max' operator in the formal definition of the cluster weight function W. In algorithmic respect, the recursion formula of Lance and Williams [14] is useful. Day and Edelsbrunner [3] developed $O(n^2 \log n)$ implementation making use of the data structure called *heap* [1]. Since generally the size of input instance is of order $O(n^2)$ there is still a gap of factor $\log n$ between lower and upper time complexities bounds. However, the lower bound is achievable from the point of view of amortized complexity [15,19]. The amortized complexity measure 'averages the running times of operations in a sequence over the sequence.' More formally, a sequence of operations $\{op\}$ is said to have *amortized time bounds* $\{b\}$ if $\sum t \leq \sum b$, where t is the actual time needed by op. Intuitively, if op uses less time than its allotated b units, then the leftover time may be held in reserve to be used by latter operations.

THEOREM 2. *The hierarchical agglomerative clustering scheme can be implemented in $O(n^2)$ amortized time.*

Proof. We use the data structure known as (a, b)-*tree* [16]. This is a kind of balanced search trees where each father has at least a and at most b sons, recommedable values are $a = 2, b = 4$. It is known that the sequence of m insertions and deletions (in arbitrary order) takes $O(m)$ amortized time [16]. Let us consider the following implementation of the agglomerative paradigm:

```
1. {Setting-up the (a,b)-tree}
Perform $O(n^2)$ insertions of all distances (dissimilarities) between $n$ given objects.
2. do $n-1$ times
    2.1 identify 'closest pair' $C, C'$ of clusters { a query for the minimal element in the $(a, b)$-
        tree now consumes $O(\log n)$ time, c.f. [16]}
    2.2 replace $C$ and $C'$ by $C \cup C'$ and update dissimilarities between this new cluster and the
        remaining clusters { it requires one deletion of the minimal element, $O(n)$ deletions
        of distances from $C$ and $C'$ and $O(n)$ insertions of new distances from $C \cup C'$ to the
        other clusters}.
```

Examining steps 1 and 2 we derive the overall $O(n^2)$ amortized time complexity from $O(n^2)$ deletions and insertions and $O(n)$ identifications of minimal element of (a, b)-tree.□

Let us turn to the hierarchical divisive clustering paradigm. To our knowledge, there is no unified way, similar to the agglomerative counterpart, for the desription of divisive clustering procedures. The main reason is that the division of clusters for maximum separability is tightly connected to 'max cut' problems in graphs that are intractable [8]. The only known eficient dual part to the agglomerative algorithm is the divisive single linkage algorithm. This observation stems in the fact that single linkage is closely related to the construction of the minimum spanning tree of $K(X)$, c.f. the well-known greedy algorithm for the construction of minimum spanning trees [15,19] which is in fact an agglomerative procedure. Let us denote mst(A) a minimum spanning tree of the complete graph defined on the set of vertices A, $A \subseteq X$. In fact the divisive single linkage algorithm is based on the following property.

LEMMA 1. *Let e be an edge of mst(X) and A, B two sets of vertices of connectivity components of mst(X)-{e} . Then mst(A)$\subseteq$ mst(X) $\supseteq$ mst(B).* □

THEOREM 3. *The divisive single linkage algorithm can be implemented in $O(n^2)$ time.*

Proof. At first we construct mst(X) in $O(n^2)$ time [16]. Further, we continue as follows:

1. Find maximum edge e (with maximum dissimilarity) of mst(X)
   {$O(n)$ time suffices }
2. Create two new clusters by deleting e in mst(X)
   { traverse the tree and report two sets of vertices of two connectivity components of mst(X) $-$ {e}, it takes $O(n)$time}
3. Repeat recursively Steps 1. and 2. for each cluster A having at least two vertices (using A instead of X).

The Steps 1. and 2. are repeated $n-1$ times and the proof is concluded. □

Single linkage clustering algorithms seem to have a dominant position in hierarchical clustering [21]. Fisher and van Ness [7] have introduced the notion of connected admissibility by means of single linkage. We shall further use a slightly modified definition of connected admissibility. We shall say that two minimum spanning trees intersect if there is an intersection of their edges, not necessarily at endpoints. Note that this definition considers the topology of such trees in a given metric space and is especially vivid in the plane.

A hierarchy H is said to be *connected admissible* if

$$(\forall h, h' \in H, h \cap h' = \emptyset) \text{ mst}(h) \cap mst(h') = \emptyset.$$

Indeed, it is desirable to produce separable hierarchies. In Euclidean spaces the natural separability is measured by the disjointness of convex hulls. However, it seems that the latter criterion is sometimes too strong and it imposes an artificial clustering structure on input data and does not reflect the reality. Thus a weakened version of separability by means of connected admissibility will be discussed.

Obviously it holds:

THEOREM 4. *The single linkage clustering algorithm is connected admissible.* □

On the other hand, it was shown by a counter example [7] that complete linkage is not connected admissible. Our further aim is to develop efficient connected admissible hierarchical clustering algorithms which generalize single linkage, output feasible hierarchies and which lie somewhere in between single and complete linkages.

III. Midrange, median, mean and k-single linkages

In this section we introduce a family of generalized single linkage algorithms for hierarchical clustering. They are based on the construction of the minimum spanning tree of $K(X)$, but its edges are being reweighted in some special way. The implementation can be done in both agglomerative and divisive manner. The main idea is justified by the following result.

LEMMA 2. *Let $e_1, ..., e_{n-1}$ be weights of edges of mst(X). Let us define*
$f_i = \max\{e\ ;\ e = \{x, y\},\ x$ *and* y *belong to different components of mst(X)* $- \{e_i\}\}$.
Then the sequence $(e_i, f_i), i = 1, ..., n-1$, is invariant under any choice of minimum spanning tree on X.

Proof. It is a well-known fact that the sequence $e_1, \ldots, e_{n-1}$ is invariant under the choice of mst(X). It follows from the fact that any edge e of the minimum spanning tree mst(X) can be exchanged by an edge e' with the same weight sharing the same cycle with e in mst(X)$\cup$ {e'}.

Now, the preceding observation is easily extendable to the sequence $(e_i, f_i), i = 1, ..., n-1$, since the maximum edges f of 'cuts' corresponding to edges e and e' are the same. □

It should be noted that, for the same reason, the assertion of Lemma 2 is true for another cut order statistics along the edges of minimum spanning tree. It enables us to speak about special single linkage algorithms. The *midrange* linkage is given by the midrange (i.e. minimal + maximal edge) of cuts, the *median* linkage by the median edge of cuts, the *mean* linkage by the arithmetic mean of cuts and the *k-single* linkage by the k-th minimal (or maximal) edge of corresponding cuts. In all cases, cuts are related to the minimum spanning tree and due to Lemma 2 they are independent on the choice of this tree.

Now, we describe two strategies. We distinguish between *global* and *local* strategies. The global strategy assumes reweighting of edges of the minimum spanning tree of the whole set X at once and then performing the ordinary single linkage. Local divisive and agglomerative strategies at each step consider common cuts in 'smaller' clusters which may vary after each division or agglomeration. It is worthwile noting that the implementation of a local strategy in agglomerative or divisive way may lead to different output hierarchies. Recall that we deal only with approximations to computationally intractable problems.

THEOREM 5. *The local aglomerative midrange, median, mean and k-single linkage algorithms can be implemented in $O(n^2)$ amortized time.*

Proof. First we construct the minimum spaning tree mst(X). It takes $O(n^2)$ time [16]. The agglomeration process is controlled by the structure of mst(X) and in fact it is a slight modification of the general agglomeration paradigm from Section II. In this case the underlying data structures are even simpler. We maintain a dissimilarity (distance) matrix of $K(X)$ which enables the direct access to the edges of mst(X). Indeed, in each repetition of the main loop we search for minimum reweigted edge of mst(X) and then we perform an agglomeration and update of the new between-cluster dissimilarities. In this way a new distance matrix is created. This takes $O(n)$ time for all linkages mentioned in the assumptions of the theorem. Note that the reweighting process is done at the same time and that the step of updating of all new between cluster dissimilarities is not superfluous . New dissimilarities will be used later for the correct computation of new weights of appropriate edges of mst(X). Thus, we get $O(n^2)$ time complexity. □

Now, we turn our attention to divisive generalized single linkages. Clearly, we proceed as follows :

1. Construct mst (X), reweight edges by suitable cut statistics and find the maximum reweighted edge e of mst(X)
2. Divide the corresponding cluster according to e
3. Repeat recursively Steps 1 and 2 in all clusters.

This algorithmic scheme is in fact a global divisive strategy. On the other hand, in a local divisive strategy we must perform reweighting of remaining edges of mst(X) in Step 2. after each division.

The most important part of the divisive single linkage algorithms is the computation of an appropriate characteristic of cuts belonging to edges of mst(X). The straightforward implementation is in $O(n^4)$ time since a cut may contain $O(n^2)$ edges and we maintain $O(n^2)$ cuts during the division process. Using a sophisticated data structure for dynamic trees [20] we can do it even better.

THEOREM 6. *Given mst(X), we can compute cuts belonging to edges of mst(X) in $O(n^2 \log n)$ time and in $O(n^2)$ amortized time.*

Proof. Assume that we are given mst(X) rooted at some inner vertex. We traverse the tree mst(X) simultaneously from leaves. The cuts belonging to its edges are representated by the *search tree* data structure , see [20] for technical details. Let us describe the common step of traversing. Let p be a parental node and we are to maintain the cut along the edge joining p to its father f. At this point we have at our disposal k trees representing cuts for edges (s,p) where s is a son of p. In order to create new tree for the edge (p,f) we perform deletions of edges joining vertices of filial trees of p and $k-1$ joins of updated filial trees. The joins and deletions take $O(\log n)$ worst case time and $O(1)$ amortized time. Since the sum of degrees of all nodes of mst(X) is $O(n)$ we perform $O(n)$ joins. Further, each edge of $K(X)$ is processed at most twice and thus in summary $O(n^2)$ deletions are applied. Clearly, the initialization consumes at most $O(n^2)$ time. An $O(n^2 \log n)$ worst case time complexity bound follows. The amortized complexity bound relies on the (a,b)-tree representation,c.f.[16] and Theorem 2. □

Note that we cannot associate all $n-1$ cuts with corresponding tree data structures in preprocessing step since maintainance of all 'cut-trees' may require cubic space and time. However, in the case of midrange and k-single linkages we can design asymptotically optimal algorithms. Note that the latter linkages seem to have practical importance in approximations of the **HIC** problem [12]. In the rest of the paper, they will be discussed in greater detail.

THEOREM 7. *Given mst(X), we can reweight its edges by the midrange or the k-th smallest (greatest) edge of the corresponding cuts in $\Theta(n^2)$ amortized time.*

Proof. Let us choose a root of mst(X). Each vertex is associated with a data structure (e.g.(a,b)-tree) storing all distances to other vertices. We traverse the tree from leaves to the root. After visiting the parental node simultaneously from all descendants the contraction of this node is performed. Thus we obtain a new leaf and we have to update the associated data structure. Now, the 'contracting' distance to the remaining vertices is measured by the maximum or the k-th smallest (greatest) distance from all old descendants (in fact the previously recognized sons). The same argument as in the case of Theorem 2 finishes the amortized complexity estimate. The recursive implementation of this idea is suggested. □

Although the minimum spanning tree need not be computed again after each division of clusters the correponding cut may vary drastically. This way in the case of the local divisive linkage strategy we perform $n-1$ times new reweighting in the global divisive linkage fashion. Thus we have.

THEOREM 8. *The local divisive midrange and k-single linkages can be implemented in $O(n^3)$time.* □

Finally, we shall outline how the cubic upper bound can be beaten in the case of the local midrange linkage. The idea consist in refining the proof of Theorem 7. Inspecting that proof we observe that computational bottleneck is the initialization which takes $\Theta(n^2)$ time. Suppose we are given the rooted minimum spanning tree mst(X) such that each vertex v is associated with a heap storing distances to the other vertices, i.e. the root contains the information about the farthest neighbor of v. Further, we associate each vertex v of mst(X) with the set of leaves of the subtree rooted at v. Let all heap roots be decreasingly sorted in a queue Q. This preprocessing takes $O(n^2)$ time. Now, we show that the reweighting of mst(X) by maximum cut edge can be done in linear time. We are successively taking maximal items $(x, f(x))$, where $f(x)$ is farthest neighbor of x, from Q and reweight all edges of the path $xf(x)$ in mst(X) by the distance $\sigma(x, f(x))$. During the traversal of this path we perform contraction of all vertices of this path in mst(X). Repeating this actions by taking further elements from Q we reweight contracting trees and thereby $\mathrm{mst}(X)$ appropriately. Now, the midrange linkage is performed recursively as follows. Having reweighted minimum spanning tree we select maximal edge with respect to the sum of 'old' and 'new' weights, remove it and consider vertices of remaining two parts as new clusters C_1 and C_2. Let n_1 and n_2, $n_1 \leq n_2$, denote the corresponding cardinalities of C_1 and C_2. In time $O(n_1^2)$ we reweight the $\mathrm{mst}(X)$ restricted to C_1, c.f. the preprocessing step. Then we modify the structure associated with $\mathrm{mst}(X)$ restricted to C_2 as follows. In each heap associted with vertices from C_2 we delete n_1 items related to distances to C_1, and prepare sorted queue Q of heaps roots for further recursive step. It takes $O(n_1 n_2 \log n_2)$ time. Then we reweight $\mathrm{mst}(X)$ restricted to C_2 in $O(n_2)$ time. Let us estimate the overall time complexity. Sorting is called $n-1$ times and takes $O(n_2 \log n)$ time. Similarly reweighting consumes in total $O(n_2)$ time. It remains to deal with the recursion over C_1 and C_2. Since $n_1 \leq n_2$ and $n_1 + n_2 \leq n$, the most unfavorable case is when $n_1 = n_2$. Hence the time is upperbounded by $O(\sum_{i=1}^{n-1} \frac{1}{2^i} n^2 \log n) = O(n^2 \log n)$. We complete the discussion by the following.

THEOREM 9. *The midrange linkage consumes $O(n^2 \log n)$ time.* □

We conclude this section by a simple assertion.

THEOREM 10. *The global and local divisive single linkages are connected admissible and output feasible hierarchies.* □

IV. Midrange and k-single linkages in the plane

This section is devoted to the discussion of an efficient implementation of the midrange and k-single linkages in the plane. Now, the set X is supposed to be embedded in the plane and the mutual dissimilarities are given by the Euclidean distance. Recall that the size of input instance is just n. The single linkage algorithm at first agglomerates the nearest pair of objects. Since the NEAREST PAIR problem requires $\Omega(n \log n)$ time [18] we have the same lower bound for the computational problem of the implementation of the single linkage algorithm. On the other hand the minimum spanning tree in the plane is computable in $\Theta(n \log n)$ time [18] and therefore we have.

THEOREM 11. *The agglomerative and divisive single linkage algorithms can run in optimal $\Theta(n \log n)$ time.* □

However, on the other hand no $o(n^2)$ implementation of the complete linkage strategy is known [5]. Recent results from the computational geometry [18] are ready to use for design of $O(n \log n)$ algorithms in the framework of a generalized single linkage paradigm. This is indicated in the next section. The main result of this section is the following theorem.

THEOREM 12. *The global divisive midrange and k-single linkages consume $O(n \log n)$ time.*

Proof. We shall use structures known in computational geometry as *Voronoi diagram* $V(X)$, *farthest point Voronoi diagram* $FV(X)$ and *order-k Voronoi diagram* $V_k(X)$ on X [18]. Voronoi diagram $V(X)$ is a partition of the plane into n convex polygonal regions such that each region contains exactly one point p from X. Moreover it also contains all plane points that are closer to p than to other remaining points from X. On the other hand the regions of $FV(X)$ associated with points from X contain exactly those points from the plane that have common farthest neighbor. Since $V(X) \equiv V_1(X)$ and $FV(X) \equiv V_{n-1}(X)$ we can define Voronoi regions $VR_k(S)$ for $S \subset X, |S| = k$, more formally as follows:

$$VR_k(S) = \{y \; ; \; (\forall x \in X)(\forall z \in X - S) \; \sigma(x,y) < \sigma(z,y)\}.$$

It is known that order-k Voronoi diagrams can be constructed in optimal $O(n \log n)$ time each fixed k [18].

We shall discuss 2-single linkage first. Let us given mst(X). We again traverse the tree simultaneously from leaves and determine corresponding cuts along edges of mst(X). It is a well-known fact that mst(X) contains exactly those edges whose endpoints lie in Voronoi regions that share common edge. It should be noted that the smallest proximities are realized by 'neighboring' points in the context of neighboring Voronoi regions. Consequently cuts can be reduced to cuts containing only edges given by labels of edges of Voronoi regions. The reduction is essential since each cut now contains at most $O(n)$ edges. This observation justifies the following algorithm. Let us process an edge (p, f) of mst(X), where f is the father of p. The corresponding cut will be represented by a search tree of [20]. Having at our disposal filial trees of edges (s, p), s being a son of p, we join these trees and delete all items with duplicity. Note that each duplicity item corresponds to some common edge of neighboring Voronoi regions of descendants of p and it has to be deleted. Then we insert labels (x, f) of edges of Voronoi region containing f such that x lies in the subtree rooted in f. Each operation consumes $O(\log n)$ time. This is repeated at most $n-1$ times and the $O(n \log n)$ time complexity bound follows.

Let us remark that the k-single linkage, $k > 2$, can be treated similarly using labels of edges of Voronoi regions of $V_k(X)$.

Now, let us turn our attention to the midrange linkage. By the aid of $FV(X)$ we can determine in $O(n \log n)$ time for each point $p \in X$ its farthest neighbor $f(p)$ and the set $f^{-1}(p)$ of points from X for which p is their farthest neighbor. Note that $f^{-1}(p)$ is nonempty if and only if p is lying on convex hull $CH(X)$ of X. Now, we observe that the maximum edge (= *distance*) of any cut along edges of mst(X) is given by a pair $(x, f(x)), f(x) \in CH(X)$. Therefore the algorithm of midrange linkage again traverses the mst(X) simultaneously from leaves and successively updates 'parental' edges (p, f), f is a father of p. For this sake the maximum filial distance $(x, f(x))$ or (y, z), where x, y are vertices of current subtree T of mst(X) rooted at p, $f(x), z \in \text{mst}(X) - T$ and $y \in f^{-1}(z)$, is taken into account. Using e.g. search trees as a data structure for maintaining necessary information we get at most $O(\log n)$ time per update and we conclude the proof. □

Since the local strategy in fact consists in $n-1$ times applications of reweighting in global sense we have the following corollary.

THEOREM 13. *The local divisive midrange and k-single linkages consume $O(n^2 \log n)$ time.* □

V. Concluding remarks

There exist several other ways how to generalize single linkage clustering in order to obtain other feasible and connected admissible hierarchies. At first we need not use necessarily the minimum spanning tree of the instance graph $K(X)$. For this sake any other spanning tree framework can be used and, of course, the notion of adjusted connected admissibility. Also other optimization criteria can be put on particular spanning trees. However, in many cases we replace the polynomial solvability of the minimum spanning tree construction by NP-hardness, cf. [17]. In the plane, the structure of so-called *Delaunay triangulation* [18] seems to be very useful. It serves as a very succint and 'faithful' representation of $K(X)$, cf.[4]. In the context of $O(n \log n)$ complexity requirement we recommend to assign the following measure of 'neighborhood influence' to edges of Delaunay triangulation of X :

(i) perimeter of Delaunay triangle containing the given edge
(ii) number of points inside of Gabriel circle [15]
(iii) number of points inside of the relative neighborhood lune of X [19] with respect to a given edge

(iv) perimeter of the smallest triangle containing the given edge (the area of the smallest ellipse with foci at endpoints of a given edge such that it contains no third point of X) [13].

Eventually, as it is typical, we perform single linkage.

Finally, we should mention the paper [6] where efficient algorithms are designed for the dynamic maintenance of minimum spanning trees of planar graphs. Using this approach we can, for example, do experiments by changing weights of some edges of Delaunay triangulation (or other structures) and/or deleting some outlying vertices in $O(\log n)$ time per operation.

References

[1] AHO A.V.,HOPCROFT J.E.,ULLMAN J.D.:Data structures and algorithms. Addison Wesley, 1982.

[2] ANDERBERG M.R.:Cluster analysis for applications. Academic Press, 1973.

[3] DAY W.H.E.,EDELSBRUNNER H.:Efficient algorithms for agglomerative hierarchical clustering methods. Journal of Classification 1(1984), 7-24.

[4] DOBKIN D.D.,FRIEDMAN S.J.,SUPOWIT K.J.:Delaunay graphs are almost as good as complete graphs. Proc. IEEE FOCS(1987), 20-26.

[5] EDELSBRUNNER H.,GUIBAS L.J.,SHARIR M.:The upper envelope of piecewise linear functions. Discrete Comp.Geom. 4(1989), 285-309.

[6] EPPSTEIN D., ITALIANO G.F., TAMASSIA R., TARJAN R.E., WESTBROOK J., YOUNG M.: Maintenance of minimum spanning forest in a dynamic planar graph. Proc. ACM-SIAM Symp. on Discrete Algorithms, 1990, 1-11.

[7] FISHER L.,van NESS J.W.:Admissible clustering procedures. Biometrika 58(1971), 91-104.

[8] GAREY M.R.,JOHNSON D.S.:Computers and intractability : a guide to theory of NP-completeness. Freeman,1979.

[9] HARTIGAN J.:Clustering algorithms. Wiley, 1975.

[10] JAMBU M.:Classification automatique pour l'analyse des données. Dunod, 1978.

[11] KŘIVÁNEK M.: On the computational complexity of clustering. Proc. Data analysis and applications IV, Versailles,North Holland, 1986, 89-96.

[12] KŘIVÁNEK M.:The complexity of ultrametric partitions on graphs. Information proc.letters 27(1988), 265-270.

[13] KŘIVÁNEK M.:The use of graphs of elliptic influence in visual hierarchical clustering. Proc. MFCS'90, Springer LNCS 452, 1990, 392-398.

[14] LANCE G.N.,WILLIAMS W.T.:A general theory of classificatory sorting strategies. Comput.J. 9(1967), 373-380.

[15] MATULA D.W.,SOKAL R.R.:Properties of Gabriel graphs relevant to geographic variation research and the clustering of points in the plane. Geograph.analysis 3(1980), 205-222.

[16] MEHLHORN K.:Data structures and algorithms. Springer, 1984.

[17] PAPADIMITRIOU C.H.,YANNAKAKIS M.:The complexity of restricted spanning tree problems. JACM 29(1982), 285-309.

[18] PREPARATA F.,SHAMOS M.I.:Computational geometry - an introduction. Springer, 1985.

[19] SUPOWIT K.J.:The relative neighborhood graph, with an application to minimum spanning trees. JACM 30(1983), 428-448.

[20] TARJAN R.E.:Data structures and network algorithms. SIAM, Philadelphia, 1983.

[21] ZAHN C.T.:Graph-theoretical methods for detecting and describing gestalt clusters. IEEE Trans.on Computers, C-20(1971), 68-86.