Exact Algorithms and Experiments for Hierarchical Tree Clustering

Jiong Guo* Sepp Hartung† Christian Komusiewicz‡ Rolf Niedermeier
Johannes Uhlmann§

Universität des Saarlandes, Campus E 1.4, D-66123 Saarbrücken, Germany
jguo@mmci.uni-saarland.de

Institut für Informatik, Friedrich-Schiller-Universität Jena,
Ernst-Abbe-Platz 2, D-07743 Jena, Germany
{sepp.hartung,c.komus,rolf.niedermeier,johannes.uhlmann}@uni-jena.de

Abstract

We perform new theoretical as well as first-time experimental studies for the NP-hard problem to find a closest ultrametric for given dissimilarity data on pairs. This is a central problem in the area of hierarchical clustering, where so far only polynomial-time approximation algorithms were known. In contrast, we develop efficient preprocessing algorithms (known as kernelization in parameterized algorithmics) with provable performance guarantees and a simple search tree algorithm. These are used to find optimal solutions. Our experiments with synthetic and biological data show the effectiveness of our algorithms and demonstrate that an approximation algorithm due to Ailon and Charikar [FOCS 2005] often gives (almost) optimal solutions.

1. Introduction

Hierarchical representations of data play an important role in biology, the social sciences, and statistics [9, 10, 2, 6]. The basic idea behind hierarchical clustering is to obtain a recursive partitioning of the input data in a tree-like fashion such that the leaves one-to-one represent the single items and all inner points represent clusters of various granularity degrees. Hierarchical clusterings do not require a prior specification of the number of clusters and they allow to understand the data at many levels of fine-grainedness (the root of the tree representing the whole data set). We contribute new theoretical and experimental results for a well-studied NP-hard problem in this context, which is called M-Hierarchical Tree Clustering. The essential point of our work is that we can efficiently find provably *optimal* (not only approximate) solutions in cases of practical interest.

Hierarchical Tree Clustering. Let X be the input set of elements to be clustered. The dissimilarity of the elements is expressed by a positive-definite symmetric function $D : X \times X \to \{0, \ldots, M+1\}$, briefly called *distance function*. Herein, the constant $M \in \mathbb{N}$ specifies the depth of the clustering tree to be computed. We focus on the case to find a *closest ultrametric* that fits the given data.

Definition 1. *A distance function $D : X \times X \to \{0, \ldots, M+1\}$ is called* ultrametric *if for all $i, j, l \in X$ the following condition holds:*

$$D(i, j) \leq \max\{D(i, l), D(j, l)\}.$$

The central M-Hierarchical Tree Clustering problem[1] can be formulated as follows:

Input: A set X of elements, a distance function $D : X \times X \to \{0, \ldots, M+1\}$, and $k \geq 0$.
Question: Is there a distance function $D' : X \times X \to \{0, \ldots, M+1\}$ such that D' is an ultrametric and $\|D - D'\|_1 \leq k$?

Herein, $\|D - D'\|_1 := \sum_{\{i,j\} \subseteq X} |D'(i, j) - D(i, j)|$. In other words, given any distance function D, the goal is to modify D as little as possible to obtain an ultrametric D'. An ultrametric one-to-one corresponds to a rooted depth-$(M+1)$ tree where all leaves have distance exactly $M+1$ to the root and they are bijectively labeled with the elements of X [2]. This problem is closely related to the reconstruction of phylogenetic trees [7, 2]. 1-Hierarchical Tree Clustering is the same as the Correlation Clustering problem on complete graphs [4, 3], also known as Cluster Editing [8, 5].

Related Work. M-Hierarchical Tree Clustering is NP-complete [10] and APX-hard [1], excluding any hope for polynomial-time approximation schemes. Ailon and Charikar [2] presented a randomized polynomial-time combinatorial algorithm for M-Hierarchical Tree Clustering that achieves an approximation ratio of $M+2$. Moreover, there is a deterministic algorithm achieving the same approximation guarantee [13]. Numerous papers deal with 1-Hierarchical Tree Clustering and its approximability [3, 13] or fixed-parameter tractability [8, 5]. In particular, there have been encouraging experimental studies

*Partially supported by the DFG, research project DARE, GU 1023/1, and the DFG cluster of excellence "Multimodal Computing and Interaction".

†Partially supported by the DFG, research project DARE, GU 1023/1.

‡Supported by a PhD fellowship of the Carl-Zeiss-Stiftung.

§Supported by the DFG, research project PABI, NI 369/7.

[1] Our algorithms also deal with the practically more relevant optimization problem where one wants to minimize the "perturbation value" k.

based on fixed-parameter algorithms [12, 5]. In the area of phylogeny reconstruction, M-Hierarchical Tree Clustering is known as "Fitting Ultrametrics under the ℓ_1 norm".

Our Results. On the theoretical side, we provide polynomial-time preprocessing algorithms with provable performance guarantees (in the field of parameterized complexity analysis [11] known as "kernelization"). More precisely, we develop efficient data reduction rules that provably transform an original input instance of M-Hierarchical Tree Clustering into an equivalent instance consisting of only $O(k^2)$ elements or $O(M \cdot k)$ elements, respectively. Moreover, a straightforward exact algorithm based on a size-$O(3^k)$ search tree is presented.

On the practical side, we contribute implementations and experiments for our new data reduction rules (combined with the search tree strategy) and the known approximation algorithm: First, with our exact algorithms, we can solve a large fraction of non-trivial problem instances. Second, we observe that Ailon and Charikar's algorithm [2] often yields optimal results.

Basic Notation. Throughout this work let $n := |X|$. A *conflict* is a triple $\{i, j, l\}$ of elements from the data set X that does not fulfill the condition of Definition 1. A pair $\{i, j\}$ is the *max-distance pair* of a conflict $\{i, j, l\}$ if $D(i, j) > \max\{D(i, l), D(j, l)\}$. For $Y \subseteq X$ the restriction of D to Y is denoted by $D[Y]$ and is called the distance function induced by Y. For some of our data reduction rules we use notation from graph theory. We only consider undirected, simple graphs $G = (V, E)$, where V is the vertex set and $E \subseteq \{\{u, v\} \mid u, v \in V\}$. The (open) neighborhood $N_G(v)$ of a vertex v is the set of vertices that are adjacent to v, and the closed neighborhood is defined as $N_G[v] := N_G(v) \cup \{v\}$. For a vertex set $S \subseteq V$, let $N_G(S) := \bigcup_{v \in S} N_G(v) \setminus S$ and $N_G[S] := S \cup N_G(S)$. With $N_G^2(S) := N_G(N_G(S)) \setminus N_G[S]$ we denote the second neighborhood of a vertex set S.

Fixed-Parameter Tractability and Kernelization. Parameterized algorithmics is a two-dimensional framework for studying the computational complexity of problems [11]. One dimension is the input size n (as in classical complexity theory), and the other one is the *parameter* k (usually a positive integer). A problem is called *fixed-parameter tractable* if it can be solved in $f(k) \cdot n^{O(1)}$ time, where f is a computable function only depending on k.

A core tool in the development of fixed-parameter algorithms is *problem kernelization*, which can be viewed as polynomial-time preprocessing. Here, the goal is to transform a given problem instance x with parameter k by applying so-called *data reduction rules* into a new instance x' with parameter k' such that the size of x' is upper-bounded by some function only depending on k, the instance (x, k) is a yes-instance if and only if (x', k') is a yes-instance, and $k' \leq k$. The reduced instance, which must be computable in polynomial time, is called a *problem kernel*; the whole process is called *kernelization*.

Several details and proofs are deferred to a full version of this article.

2. A Simple Search Tree Strategy

The following search tree strategy solves M-Hierarchical Tree Clustering: As long as D is not an ultrametric, search for a conflict, and branch into the three cases (either decrease the max-distance pair or increase one of the other two distances) to resolve this conflict by changing the pairwise distances. Solve each branch recursively for $k - 1$.

Proposition 1. *M-Hierarchical Tree Clustering can be solved in $O(3^k \cdot n^3)$ time.*

In the next section, we will show that by developing polynomial-time executable data reduction rules one can achieve that the above search tree no longer has to operate on sets X of size n but only needs to deal with sets X of size $O(k^2)$ or $O(M \cdot k)$, respectively.

3. Preprocessing and Data Reduction

In this section, we present our main theoretical results, two kernelization algorithms for M-Hierarchical Tree Clustering. Both algorithms partition the input instance into small subinstances, and handle these subinstances independently. This partition is based on the following lemma.

Lemma 1. *Let D be a distance function over a set X. If there is a subset $X' \subseteq X$ such that for each conflict C either $C \subseteq X'$ or $C \subseteq (X \setminus X')$, then there is a closest ultrametric D' to D such that for each $i \in X'$ and $j \in X \setminus X'$, $D(i, j) = D'(i, j)$.*

An $O(k^2)$-Element Problem Kernel. Our first and simpler kernelization algorithm uses two data reduction rules which handle two extremal cases concerning the elements: The first rule corrects the distance between two elements which together appear in many conflicts, while the second rule safely removes elements which are not in conflicts.

Reduction Rule 1. *If there is a pair $\{i, j\} \subseteq X$ which is the max-distance pair (or not the max-distance pair) in at least $k + 1$ conflicts, then decrease (or increase) the distance $D(i, j)$ and decrease the parameter k by one.*

Reduction Rule 2. *Remove all elements which are not part of any conflict.*

Lemma 2. *Rule 2 is correct.*

Proof. Let D be a distance function over a set X, and let $x \in X$ be an element which is not part of any conflict. We show the correctness of Rule 2 by proving the following.

> *Claim.* (X, D) has an ultrametric D' with $\|D - D'\|_1 \leq k$ iff $(X \setminus \{x\}, D[X \setminus \{x\}])$ has an ultrametric D'' with $\|D[X \setminus \{x\}] - D''\|_1 \leq k$.

Only the "$\Leftarrow$"-direction is non-trivial. The proof is organized as follows. First, we show that $X \setminus \{x\}$ can be partitioned into $M + 1$ subsets $X_1, \ldots, X_{M+1}$ such that the maximum distance within each X_r is at most r. Then, we show that for each conflict $\{i, j, l\}$ we have $\{i, j, l\} \subseteq X_r$ for some r. Using these facts and Lemma 1, we then show

that there is a closest ultrametric D' to $D[X \setminus \{x\}]$ that only changes distances within the X_r and that "reinserting" x into D' results in an ultrametric D'' within distance k to D.

First, we show that there is a partition of $X \setminus \{x\}$ into $M+1$ subsets $X_1, \ldots, X_{M+1}$ such that the maximum distance within each X_r is at most r. For $1 \leq r \leq M+1$, let $X_r := \{y \in X \mid D(x,y) = r\}$. Clearly, this yields a partition of $X \setminus \{x\}$. Furthermore, the distance between two elements $i, j \in X_r$ is at most r, because otherwise there would be a conflict $\{i, j, x\}$ since $D(i,x) = D(j,x) = r$ and $D(i,j) > r$. This, however, contradicts that x is not part of any conflict.

Next, we show that for each conflict $\{i, j, l\}$ all elements belong to the same X_r. Suppose towards a contradiction that this is not the case. Without loss of generality, assume that $D(i,x) = r > D(j,x)$ and $D(i,x) \geq D(l,x)$. Since $\{i, j, x\}$ is not a conflict, we have $D(i,j) = r$.

We distinguish two cases for $D(l,x)$:

Case 1: $D(l,x) = r$**.** Since $\{j, l, x\}$ is not a conflict, we also have $D(j,l) = r$. Then, since $\{i, j, l\}$ is a conflict, we have $D(i,l) > r$. Since $D(i,x) = D(l,x) = r$ the triple $\{i, l, x\}$ is a conflict, a contradiction.

Case 2: $D(l,x) < r$**.** Analogous to Case 1.

Consequently, there are no conflicts that contain elements from different X_r's.

Finally, let D' be an ultrametric for $D[X \setminus \{x\}]$. Since there are no conflicts that contain elements from different X_r's and by Lemma 1, we can assume that D' only modifies distances within the X_r and not between different X_r's. From D' we can obtain a distance function D'' over X as follows:

$$D''(i,j) := \begin{cases} D'(i,j) & i \neq x \wedge j \neq x, \\ D(i,j) & \text{otherwise.} \end{cases}$$

That is, D'' is obtained from D' by reinserting x and the original distances from x to $X \setminus \{x\}$. Clearly, $\|D - D''\|_1 \leq k$. It thus remains to show that D'' is an ultrametric. Since D' is an ultrametric, it suffices to show that there are no conflicts containing x.

In D' the distance between two elements $i \in X_r$ and $j \in X_r$ is at most r since this is the maximum distance in $D[X_r]$ and this maximum distance will clearly not be increased by a closest ultrametric. Hence, there can be no conflicts in D'' containing x and two vertices from the same X_r. There also can be no conflicts containing x, an element $i \in X_r$, and some other element $j \in X_{r'}$ because the distances between these elements have not been changed from D to D'', and these elements were not in conflict in D. Hence, D'' is an ultrametric. □

Theorem 1. *M-Hierarchical Tree Clustering admits a problem kernel with $k \cdot (k+2)$ elements. The running time for the kernelization is $O(M \cdot n^3)$.*

Proof. Let $I = (X, D, k)$ be an instance that is reduced with respect to Rules 1 and 2. Assume that I is a yes-instance, that is, there exists an ultrametric D' on X with distance at most k to D. We show that $|X| \leq k \cdot (k+2)$. For the analysis of the kernel size we partition the elements of X into two subsets A and B, where $A := \{i \in X \mid \exists j \in X : D'(i,j) \neq D(i,j)\}$ and $B := X \setminus A$. The elements in A are called *affected* and the elements in B are called *unaffected*. Note that $|A| \leq 2k$ since D' has distance at most k to D. Hence, it remains to show that $|B| \leq k^2$. Let $S := \{\{i,j\} \subseteq X \mid D'(i,j) \neq D(i,j)\}$ denote the set of pairs whose distances have been modified, and for each $\{i,j\} \in S$ let $B_{\{i,j\}}$ denote the elements of B that are in some conflict with $\{i,j\}$. Since the input instance is reduced with respect to Rule 2 we have $B = \bigcup_{\{i,j\} \in S} B_{\{i,j\}}$. Furthermore, since the input instance is reduced with respect to Rule 1, we have $|B_{\{i,j\}}| \leq k$ for all $\{i,j\} \in S$. The size bound $|B| \leq k^2$ then immediately follows from $|S| \leq k$.

The running time can be seen as follows. First, we calculate for each pair of elements the number of conflicts in which it is the max-distance pair and the number of conflicts in which it is not the max-distance pair in $O(n^3)$ time. Then we check whether Rule 1 can be applied. If this is the case, we update the number of conflicts for all pairs that contain at least one of the elements whose distance has been modified in $O(n)$ time. This is repeated as long as a pair to which the rule can be applied has been found, at most $O(M \cdot n^2)$ times. Hence, the overall running time of exhaustively applying Rule 1 is $O(M \cdot n^3)$. Afterwards, we exhaustively apply Rule 2 in $O(n^3)$ time overall. □

Using the standard technique of interleaving search trees with kernelization [11], one can improve the worst-case running time of the search tree algorithm from Section 2. As our experiments show (see Section 4), there is also a speed-up in practice.

Corollary 1. *M-Hierarchical Tree Clustering can be solved in $O(3^k + M \cdot n^3)$ time.*

An $O(M \cdot k)$-Element Problem Kernel. Our second kernelization algorithm extends the basic idea of an $O(k)$-element problem kernel for Cluster Editing [8]. Consider a distance function D on $X := \{1, \ldots, n\}$. For an integer t with $1 \leq t \leq M$, the *t-threshold graph* G_t is defined as (X, E_t) with $\{i,j\} \in E_t$ if and only if $D(i,j) \leq t$. If D is an ultrametric, then, for all $1 \leq t \leq M$, the corresponding graph G_t is a disjoint union of *cliques*. We call each of these cliques a *t-cluster*. Recall that a *clique* is a set of pairwisely adjacent vertices. A clique K is a *critical clique* if all its vertices have an identical closed neighborhood and K is maximal under this property.

The kernelization algorithm employs Rule 2 and one further data reduction rule. This new rule works on the t-threshold graphs G_t, beginning with $t = M$ until $t = 1$. In each G_t, this rule applies a procedure which deals with large critical cliques in G_t:

Procedure: *Critical-Clique*.
Input: A set $X' \subseteq \{1, \ldots, n\}$ and an integer t.
1. Construct G_t for X'.
2. **While** G_t contains a non-isolated critical clique K with

- $|K| \geq t \cdot |N_{G_t}(K)|$ and
- $|K| \geq |N_{G_t}(K)| + t \cdot |N^2_{G_t}(K)|$, **do**

3. **For all** $x \in N_{G_t}[K]$ and $y \in X' \setminus (N_{G_t}[K])$, set $D(x,y) := t+1$, and for all $x, y \in N_{G_t}(K)$ with $D(x,y) = t+1$, set $D(x,y) := t$.
4. Update G_t.
5. Decrease the parameter k correspondingly, that is, by the distance between the original and new instances.
6. **If** $k < 0$ **then** return "no".
7. **End while**

Reduction Rule 3. *Recursively apply the Critical-Clique procedure to the t-threshold graphs G_t from $t = M$ to $t = 1$ by calling the following procedure with parameters X and M.*

Procedure: *RR3*
Input: *A set $X' \subseteq \{1, \ldots, n\}$ and an integer t.*
Global variables: *A distance function D on $X = \{1, \ldots, n\}$ and an integer k.*
1. Critical-Clique(X', t);
2. **For** *each isolated clique K in G_t that does not induce an ultrametric* **do** *RR3($K, t-1$).*

In the following, we show that the *Critical-Clique* procedure is correct, that is, an instance (X, D, k) has a solution if and only if the instance (X, D', k') resulting by one application of *Critical-Clique* has a solution. Then, the correctness of Rule 3 follows from the observation that every call of *RR3* is on a subset K and an integer t such that K is an isolated clique in G_{t+1}. Then, K can be solved independently from $X \setminus K$: Since the elements in K have distance at least $t+1$ to all elements in $X \setminus K$, there is no conflict that contains vertices from both K and $X \setminus K$. By Lemma 1, we thus do not need to change the distance between an element in K and an element in $X \setminus K$.

For the correctness of *Critical-Clique*, we consider only the case $t = M$; for other values of t the proof works similarly. The following lemma is essential for our proof.

Lemma 3. *Let K be a critical clique in G_M with*

- $|K| \geq M \cdot |N_{G_M}(K)|$ *and*
- $|K| \geq |N_{G_M}(K)| + M \cdot |N^2_{G_M}(K)|$.

Then, there exists a closest ultrametric U for D such that $N_{G_M}[K]$ is an M-cluster in U.

Lemma 4. *The Critical-Clique procedure is correct.*

Proof. Let D denote a distance function and let K denote a critical clique of G_M fulfilling the while-condition (line 2) in the *Critical-Clique* procedure. Furthermore, let D' denote the distance function that results by executing lines 3-5 of *Critical-Clique* on K and let $d := \|D - D'\|_1$. To show the correctness of *Critical-Clique* it suffices to show that (X, D, k) is a yes-instance if and only if $(X, D', k-d)$ is a yes-instance.

"$\Rightarrow$": If (X, D, k) is a yes-instance, then, by Lemma 3, there exists an ultrametric U of distance at most k to D such that $N_{G_M}[K]$ is an M-cluster of U. Hence, it must hold that $U(i,j) = M+1$ for all $i \in N_{G_M}[K]$ and $j \in X \setminus N_{G_M}[K]$ and $U(i,j) \leq M$ for all $i, j \in N_{G_M}[K]$. Hence, the changes performed by *Critical-Clique* are necessary to obtain U.

"$\Leftarrow$": After the application of *Critical-Clique* $N_{G_M}[K]$ is an isolated clique in the M-threshold graph for D'. That is, $D'(i,j) = M+1$ all $i \in N_{G_M}[K]$ and $j \in X \setminus N_{G_M}[K]$ and $D'(i,j) \leq M$ for all $i, j \in N_{G_M}[K]$, which implies that there is no conflict with vertices in $N_{G_M}[K]$ and $X \setminus N_{G_M}[K]$. Let U denote an ultrametric with minimum distance to D'. By Lemma 1, we can assume that $U(i,j) = M+1$ for all $i \in N_{G_M}[K]$ and $j \in X \setminus N_{G_M}[K]$ and $U(i,j) \leq M$ for all $i, j \in N_{G_M}[K]$. Hence, the distance of U to D is at most k. □

Theorem 2. *M-Hierarchical Tree Clustering admits a problem kernel with $2k \cdot (M+2)$ elements. The running time for the kernelization is $O(M \cdot n^3)$.*

4. Experimental Results

Implementation Details. We briefly describe some notable differences between the theoretical algorithms from Sections 2 and 3 and their actual implementation.[2]
Main algorithm loop: We call the search tree algorithm (see Section 2) with increasing k, starting with $k = 1$ and aborting when an (optimal) solution has been found.
Data reduction rules: We implemented all of the presented data reduction rules. However, in preliminary experiments, Rule 3 showed to be relatively slow and was thus deactivated.[3]
Interleaving: In the search tree we interleave branching with the application of the data reduction rules, that is, after a suitable number of branching steps the data reduction rules are invoked. In the experiments described below we performed data reduction in every second step, since this value yielded the largest speed-up.
Modification flags: We use flags to mark distances that may not be decreased (or increased) anymore. There are three reasons for setting such a mark: the distance has been already increased (or decreased); decreasing (or increasing) it leads to a solution with distance more than k; decreasing (or increasing) it leads to a conflict that cannot be repaired without violating previous flags.
Choice of conflicts for branching: We choose the conflict to branch on in the following order of preference: First, we choose conflicts where either both non-max-distance pairs cannot be increased or the max-distance pair cannot be decreased and one non-max-distance pair cannot be increased. In this case, no actual branching takes place since only one option to destroy the conflict remains. Second, if no such conflicts exist we choose conflicts where the max-distance pair cannot be decreased or one of the non-max-distance pairs cannot be increased. If these conflicts are also not present, we choose the smallest conflict with respect to a predetermined lexicographic order. This often creates a conflict of the first two types.

We also implemented the randomized $(M+2)$-factor approximation algorithm by Ailon and Charikar [2]. In our

[2] The Java program is free software and available from http://theinf1.informatik.uni-jena.de/tree-cluster

[3] For some larger instances, however, the rule is very effective because it reduces the search tree size by up to 33%.

experiments, we repeated the algorithm 1000 times and compared the best ultrametric that was found during these trials with the exact solution found by our algorithm.

Experiments were run on an AMD Athlon 64 3700+ machine with 2.2 GHz, 1 M L2 cache, and 3 GB main memory running under the Debian GNU/Linux 5.0 operating system with Java version 1.6.0_12.

Synthetic Data. We generate random instances to chart the border of tractability with respect to different values of n and k. We perform two studies, considering varying k for fixed values of n and considering varying n for fixed values of k. In the experiments either $M = 2$ or $M = 4$.

For each value of n we generate five ultrametrics and perturb each of these instances, increasing step by step the number of perturbations k. For each pair of n and k we generate five distance functions. We thus create 25 instances for each pair of n and k. Next, we describe in detail how we generate and disturb the ultrametrics.

Generation of Ultrametrics. We generate the instances by creating a random ultrametric tree of depth $M + 1$. We start at the root and randomly draw the number of its children under uniform distribution from $\{2, \ldots, \lceil \ln n \rceil\}$. Then, the elements are randomly (again under uniform distribution) assigned to the subtrees rooted at these newly created nodes. For each child we recursively create ultrametric trees of depth M. The only difference for a node at a lower level is that we randomly draw the number of its children under uniform distribution from $\{1, \ldots, \lceil \ln n \rceil\}$. That is, in contrast to the root node, we allow that an inner node has only one child.

Perturbation of Generated Ultrametrics. We randomly choose a pair $\{i, j\}$ of elements under uniform distribution and change the distance value $D(i, j)$. This step is repeated until k distance values have been changed. For each chosen pair, we randomly decide whether $D(i, j)$ will be increased or decreased (each with probability 1/2). We do not increase $D(i, j)$ if it has been previously decreased or if $D(i, j) = M + 1$; we do not decrease $D(i, j)$ if it has been previously increased or if $D(i, j) = 1$. Note that with this approach a generated instance may have a solutions that has distance $< k$ to the input distance function.

Experiments with fixed n. First, we study the effect of varying k for fixed values of n. As to be expected by the theoretical analysis, the running time increases for increasing k. Figure 1 shows the running times of the instances that could be solved within 5 minutes. The combinatorial explosion that is common to exponential-time algorithms such as fixed-parameter algorithms sets in at $k \approx n$. This is due to the fact that most instances with $k < n$ could be solved without branching, just by applying the data reduction rules. Regression analysis shows that the running time is best described by exponential functions of the type α^k with $\alpha \leq 1.4$. This is due to the data reduction: switching it off leads to running times with $\alpha \approx 2.4$.

Experiments with fixed k. We study the effect of different input sizes n for $k = 50$ and $k = 100$, with $n \geq 10$. The results are shown in Figure 2. Roughly speaking, the instances are difficult to solve when $k > n$. Again, this

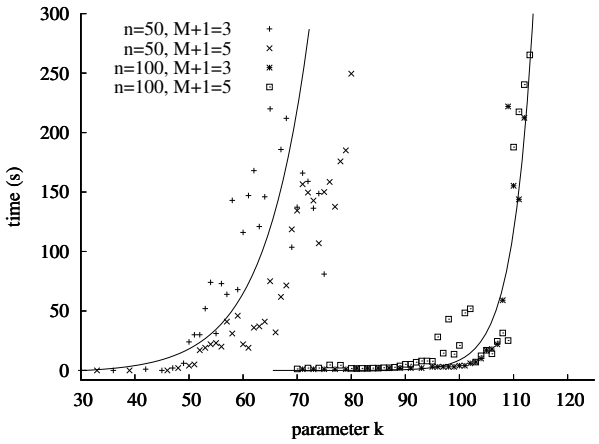

Figure 1: Running times for fixed n and varying k.

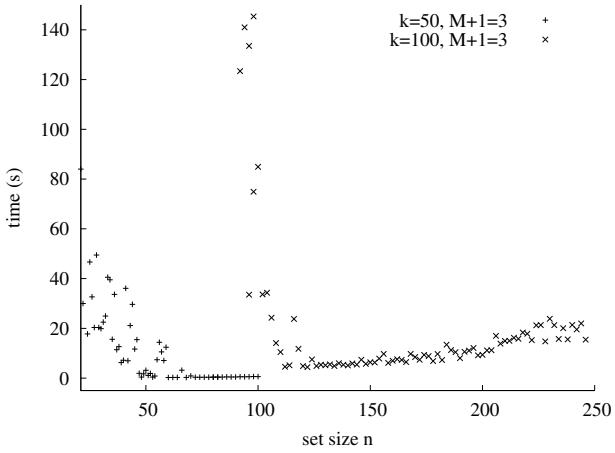

Figure 2: Running times for fixed k and varying n.

behavior can be attributed to the data reduction rules that are very effective for $k < n$.

Protein Similarity Data. We perform experiments on protein similarity data which have been previously used in experimental evaluation of fixed-parameter algorithms for Cluster Editing [12]. The data set contains 3964 files with pairwise similarity data of sets of proteins. The number of proteins n for each file ranges from 3 to 8836. We consider a subset of these files, where $n \leq 60$, covering about 90% of the files.

From each file, we create four discrete distance matrices for $M = 2$ as follows. We set the distance of the c% of the pairs with lowest similarity to 3, where c is a predetermined constant. From the remaining pairs the c% of the pairs with lowest similarity are set to 2, and all others to 1. In our experiments, we set c to 75, 66, 50, and 33. This approach is motivated by the following considerations. In a distance function represented by a balanced ultrametric tree of depth $M + 1$ at least half of all distances are $M + 1$ and with increasing degree of the root of the clustering tree

Table 1: Summary of our experiments for the protein similarity data. The second column contains the number of instances within the respective range. The next four columns provide the percentage of instances that can be solved within 2, 10, 60, and 600 seconds by our search tree algorithm. Further, k_m^{ST} denotes the maximum, and k_{avg}^{ST} the average distance to a closest ultrametric. For the approximation algorithm, k_{avg}^{AP} denotes the maximum distance and $\%_{ex}$ is the percentage of instances which were solved optimal. Finally, d denotes the maximum difference between the distances found by the two algorithms.

		$c = 75$									$c = 66$								
range	#	2s	10s	60s	600s	k_m^{ST}	d	k_{avg}^{ST}	k_{avg}^{AP}	$\%_{ex}$	2s	10s	60s	600s	k_m^{ST}	d	k_{avg}^{ST}	k_{avg}^{AP}	$\%_{ex}$
$n \leq 20$	2806	100	100	100	100	23	3	2.4	2.4	98	99	100	100	100	35	5	2.8	2.8	96
$20 < n \leq 40$	486	63	72	79	89	69	6	25.7	26.2	78	57	67	74	82	74	7	27.1	27.5	76
$40 < n \leq 60$	298	4.7	8.1	13	22	84	6	48.3	48.8	76	2	4	8	16	76	4	53.6	53.9	80
$n \leq 60$	3590	87	89	90	92	84	6	6.4	6.5	95	86	88	89	91	76	7	6.52	6.62	94
		$c = 50$									$c = 33$								
range	#	2s	10s	60s	600s	k_m^{ST}	d	k_{avg}^{ST}	k_{avg}^{AP}	$\%_{ex}$	2s	10s	60s	600s	k_m^{ST}	d	k_{avg}^{ST}	k_{avg}^{AP}	$\%_{ex}$
$n \leq 20$	2806	97	98	99	100	65	10	6.8	7	88	94	96	98	99	73	17	9.4	9.8	83
$20 < n \leq 40$	486	18	25	38	50	82	18	46	47	59	7.4	13	20	31	82	22	53	55	65
$40 < n \leq 60$	298	0	1.3	1.7	3.4	85	0	62	62	100	0	0	0	0	/	/	/	/	/
$n \leq 60$	3590	78	81	83	85	85	18	10.1	10.4	86	74	77	79	82	82	22	11.6	12.1	82

the number of pairs with distance of $M + 1$ increases. If we assume that the ultrametric tree is more or less balanced we thus expect a large portion of pairwise distances to have maximum value making the choices of $c = 75$ and $c = 66$ the most realistic.

We summarize our experimental findings (see Table 1) on these data as follows. First, our algorithm solves instances with $n \leq 20$ in few seconds. Second, for $n \leq 40$, many instances can be solved within 10 minutes. Third, using our exact algorithms, we can show that the approximation algorithm often yields almost optimal solutions. Finally, for decreasing values of c and increasing instance sizes the solution sizes and, hence, the running times increase. Interestingly, the approximation quality decreases simultaneously. Altogether, we conclude that both our new exact and the known approximation algorithm are useful for a significant range of practical instances.

5. Conclusion

Our polynomial-time executable data reduction rules shrink the original instance to a provably smaller, equivalent one. They can be used in combination with every solving strategy for M-Hierarchical Tree Clustering. For instance, we started to explore the efficacy of combining our data reduction with Ailon and Charikar's approximation algorithm [2]. In case $k < n$ almost all instances could be solved exactly by data reduction within a running time that is competitive with the approximation algorithm by Ailon and Charikar [2]. Obviously, although having proven usefulness by solving biological real-word data, the sizes of the instances we can typically solve exactly are admittedly relatively small (up to around 50 vertices). In case of larger instances, one approach could be to e.g. use the approximation algorithm to create small independent subinstances, where our algorithms apply. Finally, our algorithms also serve for "benchmarking" heuristic algorithms indicating the quality of their solutions. For instance our experiments indicate that the solution quality of the approximation algorithm [2] gets worse with growing input sizes.

References

[1] Agarwala, R.; Bafna, V.; Farach, M.; Narayanan, B.; Paterson, M.; and Thorup, M. 1999. On the approximability of numerical taxonomy (fitting distances by tree matrices). *SIAM J. Comput.* 28(3):1073–1085. on p. 1.

[2] Ailon, N., and Charikar, M. 2005. Fitting tree metrics: Hierarchical clustering and phylogeny. In *Proc. 46th FOCS*, 73–82. IEEE Computer Society. on pp. 1, 2, 4, and 6.

[3] Ailon, N.; Charikar, M.; and Newman, A. 2008. Aggregating inconsistent information: Ranking and clustering. *J. ACM* 55(5). on p. 1.

[4] Bansal, N.; Blum, A.; and Chawla, S. 2004. Correlation clustering. *Machine Learning* 56(1–3):89–113. on p. 1.

[5] Böcker, S.; Briesemeister, S.; and Klau, G. W. 2009. Exact algorithms for cluster editing: Evaluation and experiments. *Algorithmica*. To appear. on pp. 1 and 2.

[6] Dasgupta, S., and Long, P. M. 2005. Performance guarantees for hierarchical clustering. *J. Comput. Syst. Sci.* 70(4):555–569. on p. 1.

[7] Farach, M.; Kannan, S.; and Warnow, T. 1995. A robust model for finding optimal evolutionary trees. *Algorithmica* 13:155–179. on p. 1.

[8] Guo, J. 2009. A more effective linear kernelization for Cluster Editing. *Theor. Comput. Sci.* 410(8-10):718–726. on pp. 1 and 3.

[9] Hartigan, J. 1985. Statistical theory in clustering. *J. Classifi.* 2(1):63–76. on p. 1.

[10] Křivánek, M., and Morávek, J. 1986. NP-hard problems in hierarchical-tree clustering. *Acta Informatica* 23(3):311–323. on p. 1.

[11] Niedermeier, R. 2006. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press. on pp. 2 and 3.

[12] Rahmann, S.; Wittkop, T.; Baumbach, J.; Martin, M.; Truß, A.; and Böcker, S. 2007. Exact and heuristic algorithms for weighted cluster editing. In *Proc. 6th CSB*, 391–401. Imperial College Press. on pp. 2 and 5.

[13] van Zuylen, A., and Williamson, D. P. 2009. Deterministic pivoting algorithms for constrained ranking and clustering problems. *Mathematics of Operations Research* 34:594–620. on p. 1.