A New Quartet Tree Heuristic for Hierarchical Clustering

Rudi Cilibrasi and Paul M.B. Vitányi

***Abstract*— We present a new quartet tree heuristic for hierarchical clustering from weighted quartet topologies, and a standard manner to derive those from a given distance matrix. We do not assume that there is a true ternary tree that generated the quartet topologies or distances which we wish to recover as closely as possible. Our aim is to just model the input data as faithfully as possible by the quartet tree. Our method is capable of handling up to 60–80 objects in a matter of hours, while no existing quartet heuristic can directly compute a quartet tree of more than about 20–30 objects without running for years. The method is implemented and available as public software.**

I. Introduction

We present a method of hierarchical clustering based on a novel fast randomized hill-climbing heuristic of a new quartet tree optimization criterion. Given a matrix of the pairwise distances between the objects, we score how well the resulting tree represents the information in the distance matrix on a scale of 0 to 1. Then, as proof of principle, we run the program on three data sets, where we know what the final answer should be: (i) reconstruct a tree from a distance matrix obtained from a randomly generated tree; (ii) reconstruct a tree from files containing artificial similarities; and (iii) reconstruct a tree from natural files of heterogenous data of vastly different types. We give an example in whole-genome phylogeny using the whole mitochondrial DNA of the species concerned. We compare the hierarchical clustering of our method with a more standard method of two-dimensional clustering (to show that our dendrogram method of depicting the clusters is more informative). The new method was developed as an auxiliary tool for [10], [11], since the available quartet tree methods were too slow and could only handle too small data sets for our requirements. Our new quartet tree heuristic runs orders of magnitudes faster than any other quartet tree methods (that directly reconstructs a single tree from (weighted) quartet topologies), and gives consistently good results in practice. Practitioners in the field of phylogeny have drawn our attention to the fact that it merits more careful description and separate publication to reach the appropriate audience.

Relation with Previous Work: The Minimum Quartet Tree Cost (MQTC) problem below for which we give a new computational heuristic is essentially the Quartet Puzzling problem, [36], where all (or a subset) of the quartet topologies are provided with a probability value, and the goal is to find a ternary tree that maximizes the summed probabilities of the embedded quartet topologies. Previous methods include a Maximum Likelihood incremental construction and consensus [36], geometric algorithm and dynamic programming [3], and linear programming [39]. These methods, other methods, as well as methods related to the MQT problem, cannot handle more than 30 objects [39], [29], [31], [4] directly, even while using farms of desktops. In contrast, our method has been shown to easily handle 60–80 objects directly. To handle more objects one needs to construct a supertree from the constituent quartet trees for subsets of the original data sets, [33], as in [29], [31].

Materials and Methods: The experiments reported are taken from [10], [11] where many more can be found. The data samples we used were obtained from standard data bases accessible on the world-wide web, generated by ourselves, or obtained from research groups in the field of investigation. We supply the details with each experiment. The clustering heuristic generates a tree with a certain confidence, called standardized benifit score or $S(T)$ value in the sequel. Generating trees from the same distance matrix many times resulted in the same tree in case of high $S(T)$ value, or a similar tree in case of moderately high $S(T)$ value, for all distance matrices we used, even though the heuristic is randomized. That is, there is only one way to be right, but increasingly many ways to be increasingly wrong which can all be realized by different runs of the randomized algorithm. The quality of the results depends on how well the hierarchical tree represents the information in the matrix. That quality is measured by the $S(T)$ value, and is given with each experiment. In general, the $S(T)$ value deteriorates for large sets. The reason is that with increasing size of a natural data set the projection of the information in the distance matrix into a ternary tree gets inecessarily increasingly distorted.

Figures: We use two styles to display the hierarchical clusters. In the case of genomics of Eutherian orders, it is convenient to follow the dendrograms that are customary in that area (suggesting temporal evolution) for easy comparison with the literature. In the other experiments (even the genomic SARS experiment) it is more informative to display an unrooted ternary tree (or binary tree if we think about incoming and outgoing edges) with explicit internal nodes. This facilitates identification of clusters in terms of subtrees rooted at internal nodes or contiguous sets of subtrees rooted at branches of internal nodes.

Rudi Cilibrasi is with the Centre for Mathematics and Computer Science (CWI). Address: CWI, Kruislaan 413, 1098 SJ Amsterdam, The Netherlands. Email: Rudi.Cilibrasi@cwi.nl. Part of his work was supported by the Netherlands BSIK/BRICKS project, and by NWO project 612.55.002. Paul Vitányi is with the Centre for Mathematics and Computer Science (CWI), and the University of Amsterdam. Address: CWI, Kruislaan 413, 1098 SJ Amsterdam, The Netherlands. Email: Paul.Vitanyi@cwi.nl. He was supported in part by the EU project RESQ, IST–2001–37559, the NoE QUIPROCONE IST–1999–29064, the ESF QiT Programmme, and the EU NoE PASCAL, the Netherlands BSIK/BRICKS project.

Dagstuhl Seminar Proceedings 06061
Theory of Evolutionary Algorithms
http://drops.dagstuhl.de/opus/volltexte/2006/598

II. Hierarchical Clustering

Given a set of objects as points in a space provided with a (not necessarily metric) distance measure, the associated *distance matrix* has as entries the pairwise distances between the objects. Regardless of the original space and distance measure, it is always possible to configure n objects is n-dimensional Euclidean space in such a way that the associated distances are identical to the original ones, resulting in an identical distance matrix. This distance matrix contains the pairwise distance relations according to the chosen measure in raw form. But in this format that information is not easily usable, since for $n > 3$ our cognitive capabilities rapidly fail. Just as the distance matrix is a reduced form of information representing the original data set, we now need to reduce the information even further in order to achieve a cognitively acceptable format like data clusters. To extract a hierarchy of clusters from the distance matrix, we determine a dendrogram (ternary tree) that agrees with the distance matrix according to a cost measure. This allows us to extract more information from the data than just flat clustering (determining disjoint clusters in dimensional representation).

Clusters are groups of objects that are similar according to our metric. There are various ways to cluster. Our aim is to analyze data sets for which the number of clusters is not known a priori, and the data are not labeled. As stated in [14], conceptually simple, hierarchical clustering is among the best known unsupervised methods in this setting, and the most natural way is to represent the relations in the form of a dendrogram, which is customarily a directed binary tree or undirected ternary tree. With increasing number of data items, the projection of the distance matrix information into the tree representation format gets increasingly distorted. A similar situation arises in using alignment cost in genomic comparisons. Experience shows that in both cases the hierarchical clustering methods seem to work best for small sets of data, up to 25 items, and to deteriorate for larger sets, say 40 items or more. A standard solution to hierarchically cluster larger sets of data is to first cluster nonhierarchically, by say multidimensional scaling of k-means, available in standard packages, for instance *Matlab*, and then apply hierarchical clustering on the emerging clusters.

III. The Quartet Method

Given a set N of n objects, we consider every set of four elements from our set of n elements; there are $\binom{n}{4}$ such sets. From each set $\{u, v, w, x\}$ we construct a tree of arity 3, which implies that the tree consists of two subtrees of two leaves each. Let us call such a tree a *quartet topology*. The set of $3\binom{n}{4}$ quartet topologies induced by N is denoted by Q. We denote a partition $\{u, v\}, \{w, x\}$ of $\{u, v, w, x\}$ by $uv|wx$. There are three possibilities to partition $\{u, v, w, x\}$ into two subsets of two elements each: (i) $uv|wx$, (ii) $uw|vx$, and (iii) $ux|vw$. In terms of the tree topologies: a vertical bar divides the two pairs of leaf nodes into two disjoint subtrees (Figure 1).

Definition 3.1: For the moment we consider the class $\mathcal{T}$ of undirected trees of arity 3 with $n \geq 4$ leaves, labeled with the elements of N.

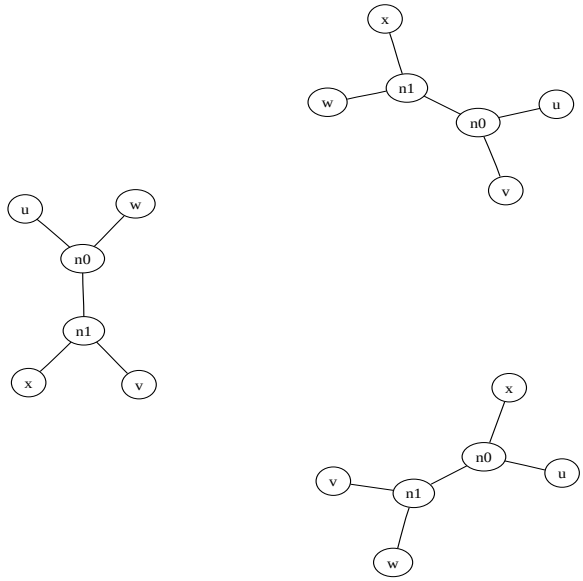

Fig. 1. The three possible quartet topologies for the set of leaf labels *u,v,w,x*

Such trees have n leaves and $n - 2$ internal nodes. For any given tree T from this class, and any set of four leaf labels $u, v, w, x \in N$, we say T is $consistent$ with $uv|wx$ if and only if the path from u to v does not cross the path from w to x. It is easy to see that precisely one of the three possible quartet topologies for any set of 4 labels is consistent for a given tree from the above class, and therefore a tree from $\mathcal{T}$ contains precisely $\binom{n}{4}$ different quartet topolgies. We may think of a

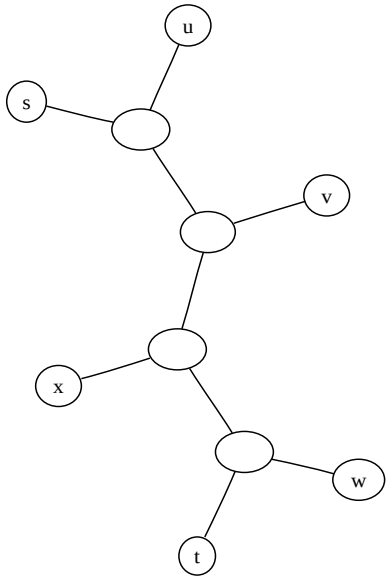

Fig. 2. An example tree consistent with quartet topology $uv|wx$

large tree having many smaller quartet topologies embedded within its structure. Commonly the goal in the quartet method is to find (or approximate as closely as possible) the tree that embeds the maximal number of consistent (possibly weighted) quartet topologies from a given set $P \subseteq Q$ of quartet topologies [16] (Figure 2). A *weight function* $W : P \rightarrow \mathcal{R}$,

with $\mathcal{R}$ the set of real numbers determines the weights. The unweighted case is when $W(uv|wx) = 1$ for all $uv|wx \in P$.

Definition 3.2: The (weighted) *Maximum Quartet Consistency (MQC)* is defined as follows:

GIVEN: N, P, and W.

QUESTION: Find $T_0 = \max_T \sum\{W(uv|wx) : uv|wx \in P$ and $uv|wx$ is consistent with $T\}$.

IV. Quartet Puzzling

The rationale for the MQC optimization problem is the assumption that there is exists a tree T_0 as desired in the class $\mathcal{T}$ under consideration, and our only problem is to find it. This assumption reflects the genesis of the method in the phylogeny community. Under the assumption that biological species developed by evolution in time, and N is a subset of the now existing species, there is a phylogeny (tree in $\mathcal{T}$) that represents that evolution. The set of quartet topologies consistent with this tree, has one quartet topology per quartet which is the true one. The quartet topologies in P are the ones which we assume to be among the true quartet topologies, and weights are used to express our relative certainty about this assumption concerning the individual quartet topologies in P.

However, the data may be corrupted so that this assumption is no longer true. In the general case of hierarchical clustering we do not even have a priori knowledge that certain quartet topologies are objectively true and must be embedded. Rather, we are in the position that we can somehow assign a relative importance to the different quartet topologies. Our task is then to balance the importance of embedding different quartet topologies against one another, leading to a tree that represents the concerns as well as possible. The most widely used heuristic of this type is *Quartet Puzzling*, specifically oriented to biological phylogeny, [36], where the weights are taken from a posterior maximum likelihood distribution. Below we review the relevant theory and give formal details where they are lacking or folklore in the literature.

Consider a slight generalization of Quartet Puzzling, where we start from a cost-assignment to the quartet topologies; the method by which we assign costs to the $3\binom{n}{4}$ quartet topologies is for now immaterial to our problem. Given a set N of n objects, let Q be the set of quartet topologies, and let $C : Q \to \mathcal{R}$ be a *cost function* assigning a real valued cost $C_{uv|wx}$ to each quartet $uv|wx \in Q$.

Definition 4.1: The *cost* C_T of a tree T with a set N of leaves (external nodes of degree 1) is defined by $C_T = \sum_{\{u,v,w,x\}\subseteq N}\{C_{uv|wx} : T$ is consistent with $uv|wx\}$—the sum of the costs of all its consistent quartet topologies.

Definition 4.2: Given N and C, the *Minimum Quartet Tree Cost (MQTC)* is $\min_T\{C_T : T$ is a tree with the set N labeling its leaves$\}$.

We normalize the problem of finding the MQTC as follows: Consider the list of all possible quartet topologies for all four-tuples of labels under consideration. For each group of three possible quartet topologies for a given set of four labels u, v, w, x, calculate a best (minimal) cost $m(u, v, w, x) = \min\{C_{uv|wx}, C_{uw|vx}, C_{ux|vw}\}$, and a worst (maximal) cost $M(u, v, w, x) = \max\{C_{uv|wx}, C_{uw|vx}, C_{ux|vw}\}$. Summing all best quartet toplogies yields the best (minimal) cost $m = \sum_{\{u,v,w,x\}\subseteq N} m(u, v, w, x)$. Conversely, summing all worst quartet toplogies yields the worst (maximal) cost $M = \sum_{\{u,v,w,x\}\subseteq N} M(u, v, w, x)$. For some distance matrices, these minimal and maximal values can not be attained by actual trees; however, the score C_T of every tree T will lie between these two values. In order to be able to compare the scores of quartet trees for different numbers of objects in a uniform way, we now rescale the score linearly such that the worst score maps to 0, and the best score maps to 1:

Definition 4.3: The *normalized tree benefit score* $S(T)$ is defined by $S(T) = (M - C_T)/(M - m)$.

Our goal is to find a full tree with a maximum value of $S(T)$, which is to say, the lowest total cost. Now we can rephrase the MQTC problem in such a way that solutions of instances of different sizes can be uniformly compared in terms of relative quality:

Definition 4.4: Definition of the *MQTC problem*:

GIVEN: N and C.

QUESTION: Find a tree T_0 with $S(T_0) = \max\{S(T) : T$ is a tree with the set N labeling its leaves$\}$.

A. Computational Hardness

The hardness of Quartet Puzzling is informally mentioned in the literature [39], [29], [31], but we provide explicit proofs. To express the notion of computational difficulty one uses the notion of "nondeterministic polynomial time (NP)". If a problem concerning n objects is NP-hard this means that the best known algorithm for this (and a wide class of significant problems) requires computation time exponential in n. That is, it is infeasible in practice. The *MQC decision problem* is the following: Given a set N of n objects, let T be a tree of which the n leaves are labeled by the objects, and let Q be the set of quartet topologies and Q_T be the set of quartet topologies embedded in T. Given a set of quartet topologies $P \subseteq Q$, and an integer k, the problem is to decide whether there is a binary tree T such that $P \bigcap Q_T > k$. In [35] it is shown that the MQC decision problem is NP-hard. We have formulated the NP-hardness of the so-called *incomplete* MQC decision problem, the less general *complete MQC decision problem* requires P to contain precisely one quartet topology per quartet out of N, and is proven to be NP-hard as well in [4].

Theorem 4.5: The MQTC decision problem is NP-hard.

Proof: By reduction from the MQC decision problem. For every MQC decision problem one can define a corresponding MQTC decision problem that has the same solution: give the quartet topologies in P cost 0 and the ones in $Q - P$ cost 1. Consider the MQTC decision problem: is there a tree T with the set N labeling its leaves such that $C_T < \binom{n}{4} - k$? An alternative equivalent formulation is: is there a tree T with the set N labeling its leaves such that

$$S(T) > \frac{M - \binom{n}{4} + k}{M - m}?$$

Note that every tree T with the set N labeling its leaves has precisely one out of the three quartet topologies of every of the

$\binom{n}{4}$ quartets embedded in it. Therefore, the cost $C_T = \binom{n}{4} - |P \bigcap Q_T|$. If the answer to the above question is affirmative, then the number of quartet topologies in P that are embedded in the tree exceeds k; if it is not then there is no tree such that the number of quartet topologies in P embedded in it exceeds k. This way the MQC decision problem can be reduced to the MQTC decision problem, which shows also the latter to be NP-hard. ■

Is it possible that the best $S(T)$ value is always one, that is, there always exists a tree that embeds all quartets at minimum cost quartet topologies? Consider the case $n = |N| = 4$. Since there is only one quartet, we can set T_0 equal to the minimum cost quartet topology, and have $S(T_0) = 1$. A priori we cannot exclude the possibility that for every N and C there always is a tree T_0 with $S(T_0) = 1$. In that case, the MQTC Problem reduces to finding that T_0. However, the situation turns out to be more complex. Note first that the set of quartet topologies uniquely determines a tree in $\mathcal{T}$, [6].

Lemma 4.6: Let T, T' be different labeled trees in $\mathcal{T}$ and let $Q_T, Q_{T'}$ be the sets of embedded quartet topologies, respectively. Then, $Q_T \neq Q_{T'}$.

A *complete set* of quartet topologies on N is a set containing precisely one quartet topology per quartet. There are $3^{\binom{n}{4}}$ such combinations, but only $2^{\binom{n}{2}}$ labeled undirected graphs on n nodes (and therefore $|\mathcal{T}| \leq 2^{\binom{n}{2}}$). Hence, not every complete set of quartet topologies corresponds to a tree in $\mathcal{T}$. This already suggests that we can weight the quartet topologies in such a way that the full combination of all quartet topologies at minimal costs does not correspond to a tree in $\mathcal{T}$, and hence $S(T_0) < 1$ for $T_0 \in \mathcal{T}$ realizing the MQTC optimum. For an explicit example of this, we use that a complete set corresponding to a tree in $\mathcal{T}$ must satisfy certain transitivity properties, [12], [13]:

Lemma 4.7: Let T be a tree in the considered class with leaves N, Q the set of quartet topologies and $Q_0 \subseteq Q$. Then Q_0 uniquely determines T if

(i) Q_0 contains precisely one quartet topology for every quartet, and

(ii) For all $\{a, b, c, d, e\} \subseteq N$, if $ab|bc, ab|de \in Q$ then $ab|ce \in Q$, as well as if $ab|cd, bc|de \in Q$ then $ab|de \in Q$.

Theorem 4.8: There are N (with $n = |N| = 5$) and a cost function C such that, for every $T \in \mathcal{T}$, $S(T)$ does not exceed $4/5$.

Proof: Consider $N = \{u, v, w, x, y\}$ and $C(uv|wx) = 1 - \epsilon(\epsilon > 0), C(uw|xv) = C(ux|vw) = 0$, $C(xy|uv) = C(wy|uv) = C(uy|wx) = C(vy|wx) = 0$, and $C(ab|cd) = 1$ for all remaining quartet topologies $ab|cd \in Q$. We see that $M = 5 - \epsilon$, $m = 0$. The tree $T_0 = (y, ((u, v), (w, x)))$ has cost $C_{T_0} = 1 - \epsilon$, since it embeds quartet topologies $uw|xv, xy|uv, wy|uv, uy|wx, vy|wx$. We show that T_0 achieves the MQTC optimum. *Case 1:* If a tree $T \neq T_0$ embeds $uv|wx$, then it must by Item (i) of Lemma 4.7 also embed a quartet topology containing y that has cost 1.

Case 2: If a tree $T \neq T_0$ embeds $uw|xv$ and $xy|uv$, then it must by Item (ii) of the Lemma 4.7 also embed $uw|xy$, and hence have cost $C_T \geq 1$. Similarly, all other remaining cases of embedding a combination of a quartet topology not containing y of 0 cost with a quartet topology containing y of 0 cost in T, imply an embedded quartet topology of cost 1 in T. ■

Altogether, the MQTC optimization problem is infeasible in practice, and natural data can have an optimal $S(T) < 1$. In [4] a polynomial time approximation scheme for complete MQC is exhibited, a theoretical approximation scheme allowing the approximation of the optimal solution up to arbitrary precision, with running time polynomial in the inverse of that precision. We say "theoretical" since that algorithm would run in something like n^{19}. For incomplete MQC it is shown that even such a theoretical algorithm does not exist, unless P=NP. Hence, computation of the MQTC optimum, and even its approximation with given precision, requires superpolynomial time unless P=NP. Therefore, any practical approach to obtain or approximate the MQTC optimum requires heuristics. The most widely used method is Tree Puzzle [36] using consensus; there are many other methods like the geometric algorithim and dynamic programming in [3], linear programming in [39], and quartet cleaning methods in [5]. In all these cases (and other previous methods not mentioned) result in far too computationally intensive calculations; they run many months or years on moderate-sized problems of 30 objects. The method presented in this paper is a simple, feasible, heuristic based on randomization and hill-climbing that has been shown to handle routinely up up to 60–80 objects in a couple of hours [11], [38].

V. New Heuristic

Our algorithm is essentially randomized hill-climbing, where undirected trees evolve in a random walk driven by a prescribed fitness function. We are given a set N of n objects and a wighting function W.

Definition 5.1: We define a *simple mutation* on a labeled undirected ternary tree as one of three possible transformations:

1) A *leaf swap*, which consists of randomly choosing two leaf nodes and swapping them.
2) A *subtree swap*, which consists of randomly choosing two internal nodes and swapping the subtrees rooted at those nodes.
3) A *subtree transfer*, whereby a randomly chosen subtree (possibly a leaf) is detached and reattached in another place, maintaining arity invariants.

Each of these simple mutations keeps the number of leaf nodes and internal nodes in the tree invariant; only the structure and placements change.

Definition 5.2: A *k-mutation* is a sequence of k simple mutations. Thus, a simple mutation is a 1-mutation.

A. Algorithm

Step 1: First, a random tree $T \in \mathcal{T}$ with $2n - 2$ nodes is created, consisting of n leaf nodes (with 1 connecting edge) labeled with the names of the data items, and $n-2$ non-leaf or *internal* nodes labeled with the lowercase letter "n" followed by a unique integer identifier. Each internal node has exactly three connecting edges.

Step 2: For this tree T, we calculate the total cost of all embedded quartet topologies, compute $S(T)$.

Comment: A tree is consistent with precisely $\frac{1}{3}$ of all quartet topologies, one for every quartet. A random tree is likely to be consistent with about $\frac{1}{3}$ of the best quartet topologies—but because of dependencies this figure is not precise.

Step 3: The *currently best known tree* variable T_0 is set to T: $T_0 \leftarrow T$.

Comment: This T_0 is used as the basis for further searching.

Step 4: Pick a number k with probability $p(k) = c/(k(\log k)^2)$ where $1/c = \sum_{k=1}^{\infty} 1/(k(\log k)^2)$.

Comment: This number k is the number of simple mutations that we will perform in the next k-permutation. The probability distribution $p(k)$ is easily generated by running a random fair bit generator and set k to the length of the first self-delimiting sequence generated. That is, if $x = x_1 \ldots x_k \in \{0,1\}^k$ ($|x| = k \geq 1$), then $\bar{x} = 1^{k-1}0x$, $x' = \overline{|x|}x$, and $x'' = \overline{|x'|}x'$. Thus, the length $|x''| = k + \log k + 2\log\log k$. The probability of generating x'' corresponding to a given x of length k by fair coin flips is $2^{-|x''|} = 2^{-k-\log k-2\log\log k} = 2^{-k}/(k(\log k)^2)$. The probability of generating x'' corresponding to *some* x of length k is 2^k times as large, that is, $1/(k(\log k)^2)$.

Step 5: Compose a k-mutation by, for each such simple mutation, choosing one of the three types listed above with equal probability. For each of these simple mutations, we uniformly at random select leaves or internal nodes, as appropriate.

Comment: Notice that trees which are close to the original tree (in terms of number of simple mutation steps in between) are examined often, while trees that are far away from the original tree will eventually be examined, but not very frequently.

Step 6: In order to search for a better tree, we simply apply the k-mutation constructed in **Step 5** on T_0 to obtain T', and then calculate $S(T')$. If $S(T') \geq S(T_0)$, then replace the current candidate in T_0 by T (as the new best tree): $T_0 \leftarrow T$.

Step 7: If $S(T_0) = 1$ or a **termination condition** to be discussed below holds, then output the tree in T_0 as the best tree and halt. Otherwise, go to **Step 4**.

Remark 5.3: We have chosen $p(k)$ to be a "fat-tail" distribution, with the fattest tail possible, to concentrate maximal probability also on the larger values of k. That way, the likelihood of getting trapped in local minima is minimized. In contrast, if one would choose an exponential scheme, like $q(k) = ce^{-k}$, then the larger values of k would arise so scarcely that practically speaking the distinction between being absolutely trapped in a local optimum, and the very low escape probability, would be insignificant. Considering positive-valued probability mass functions $q : \mathcal{N} \to (0,1]$, with $\mathcal{N}$ the natural numbers, as we do here, we note that such a function (i) $\lim_{k\to\infty} q(k) = 0$, and (ii) $\sum_{k=1}^{\infty} q(k) = 1$. Thus, every function of the natural numbers that has stricly positive values and converges can be normalized to such a probability mass function. For smooth analytic functions that can be expressed a series of fractional powers and logarithms, the borderline between converging and diverging is as follows: $\sum 1/k, \sum 1/(k\log k)$, $\sum 1/(k\log k\log\log k)$ and so on diverge, while $\sum 1/k^2, \sum 1/(k(\log k)^2)$,$\sum 1/(k\log k(\log\log k)^2)$ and so on converge. Therefore, the maximal fat tail of a "smooth" function $f(x)$ with $\sum f(x) < \infty$ arises for functions at the edge of the convergence family. The distribution $p(k) = c/(k(\log k)^2)$ is as close to the edge as is reasonable, and because the used coding $x \to x''$ is a prefix code we have $\sum 1/(k(\log k)^2) \leq 1$ by the Kraft Inequality (see for ecample [28]) and therefore $c \geq 1$. Let us see what this means for our algorithm using the choosen distribution $p(k)$. For $N = 64$, say, we can change any tree in $\mathcal{T}$ to any other tree in $\mathcal{T}$ with a 64-mutation. The probability of such a complex mutation occurring is quite large with such a fat tail: $1/(64 \cdot 6^2) = 1/2304$, that is, more than 40 times in 100,000 generations. If we can get out of a local minimum with already a 32-mutation, then this occurs with probability at least $1/800$, so 125 times, and with a 16-mutation with probability at least $1/196$, so 510 times. ♢

B. Performance

The main problem with hill-climbing algorithms is that they can get stuck in a local optimum. However, by randomly selecting a sequence of simple mutations, longer sequences with decreasing probability, we essentially run a Metropolis Monte Carlo algorithm [30], reminiscent of simulated annealing [19] at random temperatures. Since there is a nonzero probability for every tree in $\mathcal{T}$ being transformed into every other tree in $\mathcal{T}$, there is zero probability that we get trapped forever in a local optimum that is not a global optimum. That is, trivially:

Lemma 5.4: (i) The algorithm approximates the MQTC optimal solution monotonically in each run.

(ii) The algorithm without termination condition solves the MQTC optimization problem eventually with probability 1 (but we do not in general know when the optimum has been reached in a particular run).

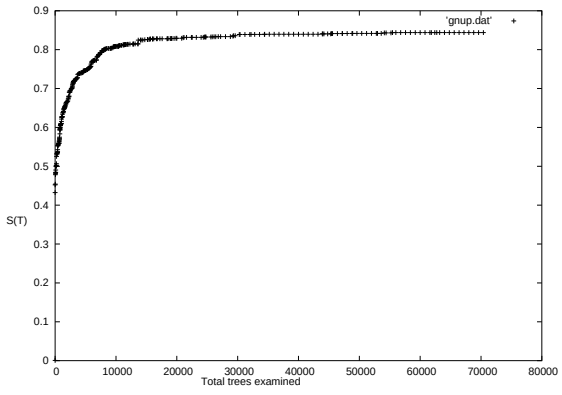

Fig. 3. Progress of a 60-item data set experiment over time

The main question therefore is the convergence speed of the algorithm on natural data, and a termination criterion to terminate the algorithm when we have an acceptable approximation. From the impossibility result in [4] we know that there is no polynomial approximation scheme for MQTC optimization, and whether our scheme is expected polynomial time seems to require proving that the involved Metropolis chain is rapidly mixing [37], a notoriously hard and generally unsolved problem. In practice, in our experiments there is

unanymous evidence that for the natural data and the weighting function we have used, convergence is always fast. We have to determine the cost of $\binom{n}{4}$ quartets to determine the $S(T)$ value. Hence the algorithm runs in time at least that much. In experiments we found that for the same data set different runs consistently showed the same behavior, for example Figure 3 for a 60-object computation. There the $S(T)$ value leveled off at about 70,000 examined trees, and the termination condition was 'no improvement in 5,000 trees'. Different random runs of the algorithm always gave the same behavior, returning a tree with the same $S(T)$ value, albeit a different tree in most cases with here $S(T) \approx 0.865$, a relatively low value. That is, since there are many ways to find a tree of optimal $S(T)$ value, and apparently the algorithm never got trapped in a lower local optimum. For problems with hight $S(T)$ value, as we see later, the algorithm consistently returned the same tree. This situation is perhaps similar to the behavior of the Simplex method in linear programming, that can be shown to run in exponential time on a badly choosen problem instance, but in practice on natural problems consistently runs in linear time.

Note that if a tree is ever found such that $S(T) = 1$, then we can stop because we can be certain that this tree is optimal, as no tree could have a lower cost. In fact, this perfect tree result is achieved in our artificial tree reconstruction experiment (Section V-E) reliably in a few minutes. For real-world data, $S(T)$ reaches a maximum somewhat less than 1, presumably reflecting distortion of the information in the distance matrix data by the best possible tree representation, as noted above, or indicating getting stuck in a local optimum or a search space too large to find the global optimum. On many typical problems of up to 40 objects this tree-search gives a tree with $S(T) \geq 0.9$ within half an hour. For large numbers of objects, tree scoring itself can be slow (as this takes order n^4 computation steps), and the space of trees is also large, so the algorithm may slow down substantially. For larger experiments, we used a C++/Ruby implementation with MPI (Message Passing Interface, a common standard used on massively parallel computers) on a cluster of workstations in parallel to find trees more rapidly. We can consider the graph mapping the achieved $S(T)$ score as a function of the number of trees examined. Progress occurs typically in a sigmoidal fashion towards a maximal value ≤ 1, Figure 3.

C. Termination Condition

The *termination condition* is of two types and which type is used determines the number of objects we can handle.

Simple termination condition: We simply run the algorithm until it seems no better trees are being found in a reasonable amount of time. Here we typically terminate if no improvement in $S(T)$ value is achieved within 100,000 examined trees. This criterion is simple enough to enable us to hierarchically cluster data sets up to 80 objects in a few hours. This is way above the 15–20 objects in the previous incremental methods (see Introduction).

Agreement termination condition: In this more sophisticated method we select a number $0 \leq r \leq 4$ of runs, and we run four invocations of the algorithm in parallel. Each time an $S(T)$ value in run $i = 1, \ldots, r$ is increased in this process it is compared with the $S(T)$ values in all the other runs. If they are all equal, then the candidate trees of the runs are compared. This can be done by simply comparing the ordered lists of embedded quartet topologies, in some standard order, since the set of embedded quartet topologies uniquely determines the quartet tree by [6]. If the r candidate trees are identical, then terminate with this quartet tree as output, otherwise continue the algorithm.

This termination condition takes (for the same number of steps per run) about r times as long as the simple termination condition. But the termination condition is much more rigorous. Since all the runs are randomized, it seems very unlikely that with natural data all of them get stuck in the same local optimum with the same quartet tree instance. There is only one tree with $S(T) = 1$ (if that is possible for the data), and random trees (the majority of all possible quartet trees) have $S(T) \approx 1/3$ (above). This gives evidence that the number of quartet trees with large $S(T)$ values is much smaller than the number of trees with small $S(T)$ values. It is furthermore evident that the precise relation depends on the data set involved, and hence cannot be expressed by a general formula without further assumptions on the data. However, we can safely state that small data sets, of say ≤ 15 objects, that in our experience often lead to $S(T)$ values close to 1 have very few quartet trees realizing the optimal $S(T)$ value. On the other hand, large sets of say 60 objects, that in our experiments invariably lead to $S(T)$ values below 0.9, have many quartet trees satisfying these (possibly optimal) $S(T)$ values. This suggests that the agreement termination method each run will get stuck in a different quartet tree of the same $S(T)$ value, so termination with the same tree is not possible. Experiments show that with agreement termination we can handle sets of up to 35 objects.

oth methods improve all existing quartet methods in at least two ways:

(i) Existing quartet methods are all incremental: first two quartets or merged to a tree based on some criterion, then a next quartet is added, and so on. While the coice of the first two quartets optimizes the used criterion, this is not so for the next incremental steps. Since the optimal tree is optimal according to a global cost, this may mean that no subtree is optimalaccording to that cost. Thus, a wrong initial choice in an incremental method cannot be undone at a later stage. In contrast, in our approach the total tree is modified to optimize the global cost, and hence monotonically approximates the global optimum.

(ii) In none of the previous methods quartet trees of more than 20 objects can be directly handled, while our method with the agreement termination handles up to 35 objects, and with the simple termination up to at least 80 objects.

D. Tree Building Statistics

We look at the case $k = 2$: The algorithm starts with two randomly initialized trees. It tries to improve each one randomly and finishes when they match. Thus, every run

produces an output tree, a maximum score associated with this tree, and has examined some total number of trees, T, before it finished. We may imagine that T comes from a distribution, getting one sample per run of the tree reconstruction program for a given distance matrix. Figure 4 shows a graph displaying a histogram of T over several hundred runs. The x-axis represents a number of trees examined in a single run of the program. This axis is binned to 100-wide histogram bars. The maximum number is about 7700 trees examined. The graph

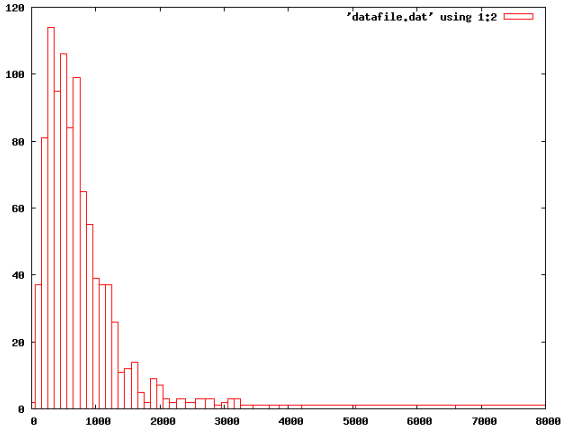

Fig. 4. Histogram of run-time number of trees examined before termination.

suggests a Poisson distribution. About $2/3$rd of the trees take less than 1000 trials.

Another interesting distribution is the mutation stepsize. Recall that the mutation length is drawn from a fat-tail distribution. But if we restrict our attention to just the mutations that improve the $S(T)$ value, then we may examine these statistics to look for evidence of a modification to this distribution due to, for example, the presence of very many isolated areas that have only long-distance ways to escape. Thankfully, Figure 5 shows the histogram of successful mutation lengths (that is,

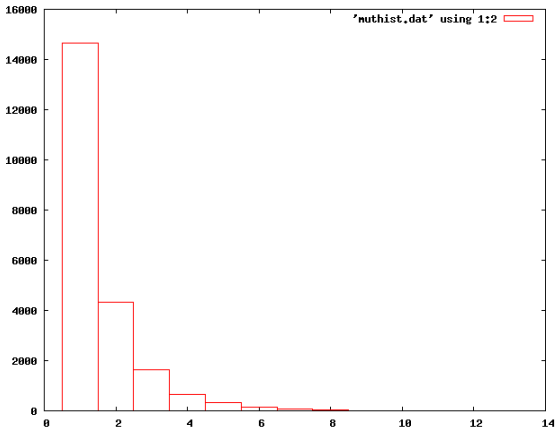

Fig. 5. Histogram of number of k-mutations per run.

number of simple mutations composing a single kept complex mutation) shows that this is not the case; Here the x-axis is the number of mutation steps and the y-axis is the number of times that step size occured. This gives good empirical evidence that in this case, at least, we have a relatively easy search space, without large gaps.

E. Controlled Experiments

With natural data sets, say music data, one may have the preconception (or prejudice) that music by Bach should be clustered together, music by Chopin should be clustered together, and so should music by rock stars. However, the preprocessed music files of a piece by Bach and a piece by Chopin, or the Beatles, may resemble one another more than two different pieces by Bach—by accident or indeed by design and copying. Thus, natural data sets may have ambiguous, conflicting, or counterintuitive outcomes. In other words, the experiments on natural data sets have the drawback of not having an objective clear "correct" answer that can function as a benchmark for assessing our experimental outcomes, but only intuitive or traditional preconceptions. We discuss three experiments that show that our program indeed does what it is supposed to do—at least in artificial situations where we know in advance what the correct answer is.

VI. Quartet Topology Costs based on Distance Matrix

Given a distance matrix, with entries giving the pairwise distances between the objects, we want to hierarchically cluster them by representing the objects as leaves of a ternary tree representing the distances in the matrix as faithfully as possible. It is important that we do not assume that there is a true tree; rather, we want to model the data as well as possible. The cost of a quartet topology is defined as the sum of the distances between each pair of neighbors; that is, $C_{uv|wx} = d(u,v) + d(w,x)$. This seems most natural given a distance matrix.

A. Distance Measure Used

Recall that the problem of clustering data consists of two parts: (i) extracting a distance matrix from the data, and (ii) constructing a tree from the distance matrix using our novel quartet-based heuristic. To check the new quartet tree method in action we use a new compression-based distance, called NCD . This metric distance was co-developed by us in [25], [26], [27], as a normalized version of the "information metric" of [28], [1]. Roughly speaking, two objects are deemed close if we can significantly "compress" one given the information in the other, the idea being that if two pieces are more similar, then we can more succinctly describe one given the other. The mathematics used is based on Kolmogorov complexity theory [28]. In [27] we defined a new class of (possibly non-metric) distances, taking values in $[0,1]$ and appropriate for measuring effective similarity relations between sequences, say one type of similarity per distance, and *vice versa*. It was shown that an appropriately "normalized" information metric minorizes every distance in the class. It discovers all effective similarities in the sense that if two objects are close according to some effective similarity, then they are also close according

to the normalized information distance. Put differently, the normalized information distance represents similarity according to the dominating shared feature between the two objects being compared. In comparisons of more than two objects, different pairs may have different dominating features. The normalized information distance is a metric and takes values in $[0,1]$; hence it may be called *"the" similarity metric*. To apply this ideal precise mathematical theory in real life, we have to replace the use of the noncomputable Kolmogorov complexity by an approximation using a standard real-world compressor, resulting in the NCD , see [11]. This has been used in the first completely automatic construction of the phylogeny tree based on whole mitochondrial genomes, [25], [26], [27], a completely automatic construction of a language tree for over 50 Euro-Asian languages [27], detects plagiarism in student programming assignments [8], gives phylogeny of chain letters [2], and clusters music [10], Analyzing network traffic and worms using compression [38], and many more topics [11]. The method turns out to be robust under change of the underlying compressor-types: statistical (PPMZ), Lempel-Ziv based dictionary (gzip), block based (bzip2), or special purpose (Gencompress).

B. CompLearn Toolkit

Oblivious to the problem area concerned, simply using the distances according to the NCD above, the method described in this paper fully automatically classifies the objects concerned. The method has been released in the public domain as open-source software: The CompLearn Toolkit [9] is a suite of simple utilities that one can use to apply compression techniques to the process of discovering and learning patterns in completely different domains, and hierarchically cluster them using the new quartet method described in this paper. In fact, this method is so general that it requires no background knowledge about any particular subject area. There are no domain-specific parameters to set, and only a handful of general settings.

C. Testing The Quartet-Based Tree Construction

We first test whether the quartet-based tree construction heuristic is trustworthy: We generated a ternary tree T with 18 leaves, using the pseudo-random number generator "rand" of the Ruby programming language, and derived a metric from it by defining the distance between two nodes as follows: Given the length of the path from a to b, in an integer number of edges, as $L(a,b)$, let

$$d(a,b) = \frac{L(a,b)+1}{18},$$

except when $a = b$, in which case $d(a,b) = 0$. It is easy to verify that this simple formula always gives a number between 0 and 1, and is monotonic with path length. Given only the 18×18 matrix of these normalized distances, our quartet method exactly reconstructed the original tree T represented in Figure 6, with $S(T) = 1$.

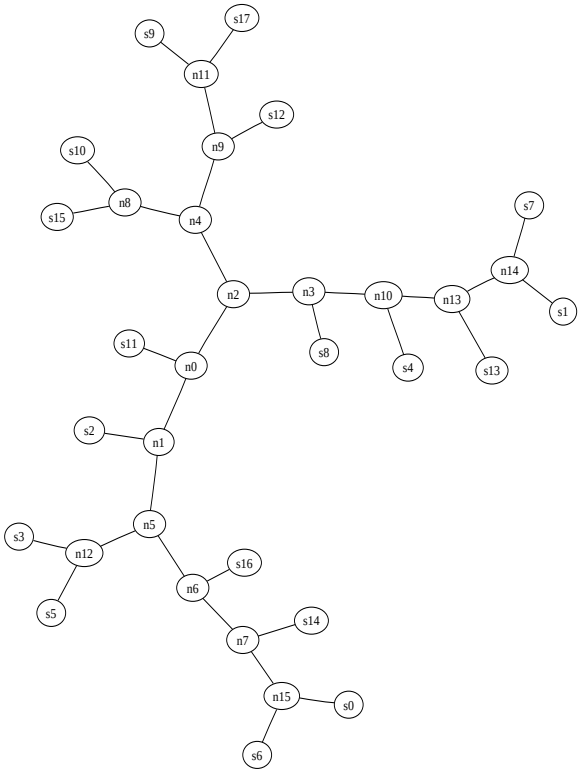

Fig. 6. The randomly generated tree that our algorithm reconstructed. $S(T) = 1$.

VII. Testing on artificial data

Given that the tree reconstruction method is accurate on clean consistent data, we tried whether the full procedure works in an acceptable manner when we know what the outcome should be like. We used the "rand" pseudo-random number generator from the C programming language standard library under Linux. We randomly generated 11 separate 1-kilobyte blocks of data where each byte was equally probable and called these *tags*. Each tag was associated with a different lowercase letter of the alphabet. Next, we generated 22 files of 80 kilobyte each, by starting with a block of purely random bytes and applying one, two, three, or four different tags on it. Applying a tag consists of ten repetitions of picking a random location in the 80-kilobyte file, and overwriting that location with the globally consistent tag that is indicated. So, for instance, to create the file referred to in the diagram by "a," we start with 80 kilobytes of random data, then pick ten places to copy over this random data with the arbitrary 1-kilobyte sequence identified as tag *a*. Similarly, to create file "ab," we start with 80 kilobytes of random data, then pick ten places to put copies of tag *a*, then pick ten more places to put copies of tag *b* (perhaps overwriting some of the *a* tags). Because we never use more than four different tags, and therefore never place more than 40 copies of tags, we can expect that at least half of the data in each file is random and uncorrelated with the rest of the files. The rest of the file is correlated with other files that also contain tags in common; the more tags in common, the more related the files are. The compressor used to compute the NCD matrix was bzip2. The resulting tree is

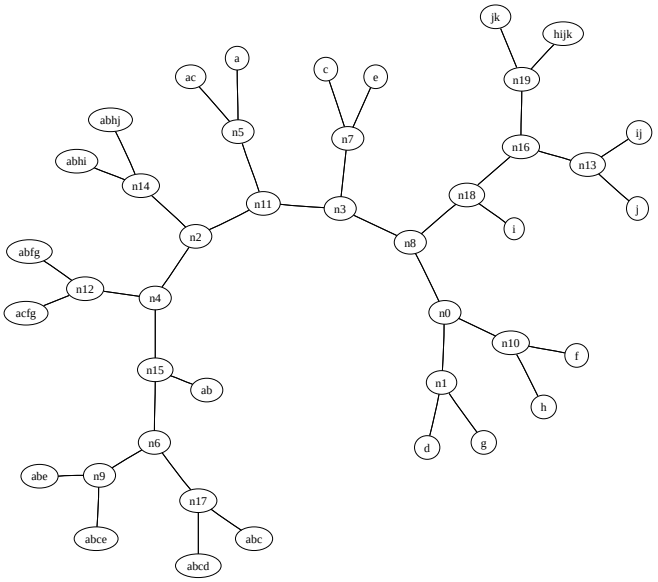

Fig. 7. Classification of artificial files with repeated 1-kilobyte tags. Not all possiblities are included; for example, file "b" is missing. $S(T) = 0.905$.

given in Figure 7; it can be seen that the clustering has occured exactly as we would expect. The $S(T)$ score is 0.905.

VIII. Testing On Heterogenous Natural Data

We test gross classification of files based on heterogenous data of markedly different file types: (i) Four mitochondrial gene sequences, from a black bear, polar bear, fox, and rat obtained from the GenBank Database on the world-wide web; (ii) Four excerpts from the novel *The Zeppelin's Passenger* by E. Phillips Oppenheim, obtained from the Project Gutenberg Edition on the World-Wide web; (iii) Four MIDI files without further processing; two from Jimi Hendrix and two movements from Debussy's Suite Bergamasque, downloaded from various repositories on the world-wide web; (iv) Two Linux x86 ELF executables (the *cp* and *rm* commands), copied directly from the RedHat 9.0 Linux distribution; and (v) Two compiled Java class files, generated by ourselves. The compressor used to compute the NCD matrix was bzip2. As expected, the program correctly classifies each of the different types of files together with like near like. The result is reported in Figure 8 with $S(T)$ equal to the very high confidence value 0.984. This experiment shows the power and universality of the method: no features of any specific domain of application are used. We believe that there is no other method known that can cluster data that is so heterogenous this reliably. This is borne out by the massive experiments with the method in [17].

IX. Testing on Natural Data

Like most hierarchical clustering methods for natural data, the quartet tree method has been developed in the biological setting to determine phylogeny trees from genomic data. In that setting, the data are (parts of) genomes of currently existing species, and the purpose is to reconstruct the evolutionary tree that led to those species. Thus, the species are labels of

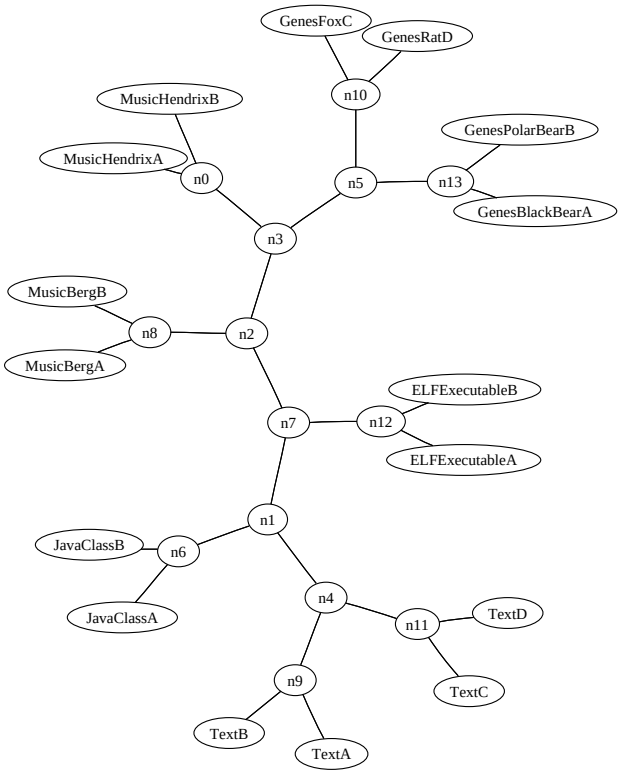

Fig. 8. Classification of different file types. Tree agrees exceptionally well with NCD distance matrix: $S(T) = 0.984$.

the leaves, and the tree is traditionaly binary branching with each branching representing a split in lineages. The internal nodes and the root of the tree correspond with extinct species (possibly a still existing species in a leaf directly connected to the internal node). The case is roughly similar for the language tree reconstruction mentioned in the Introduction. The root of the tree is commonly determined by adding an object that is known to be less related to all other objects than the original objects are with respect to each other. Where the unrelated object joins the tree is where we put the root. In these settings, the direction from the root to the leaves represents an evolution in time, and the assumption is that there is a true tree we have to discover. However, we can also use the method for hierarchical clustering, resulting an unrooted ternary tree, and the assumption is not that there is a true tree we must discover. To the contrary, there is no true tree, but all we want is to model the similarity relations between the objects as well as possible, given the distance matrix. The interpretation is that objects in a given subtree are pairwise closer (more similar) to each other than any of those objects is with respect to any object in a disjoint subtree.

A. Identifying The SARS Virus

As an application of our methods we clustered the SARS virus after its sequenced genome was made publicly available, in relation to potential similar virii. The 15 virus genomes were downloaded from The Universal Virus Database of the International Committee on Taxonomy of Viruses, available on the world-wide web. The SARS virus was downloaded

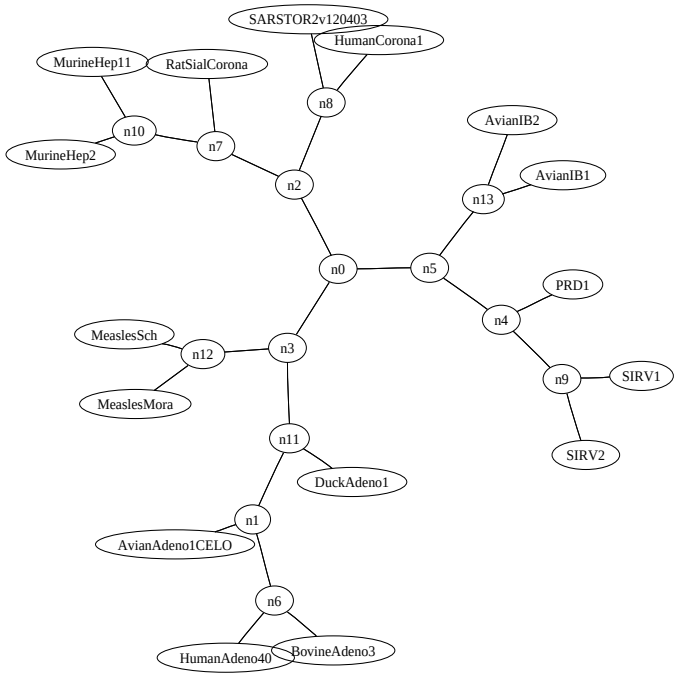

Fig. 9. SARS virus among other virii. Legend: AvianAdeno1CELO.inp: Fowl adenovirus 1; AvianIB1.inp: Avian infectious bronchitis virus (strain Beaudette US); AvianIB2.inp: Avian infectious bronchitis virus (strain Beaudette CK); BovineAdeno3.inp: Bovine adenovirus 3; DuckAdeno1.inp: Duck adenovirus 1; HumanAdeno40.inp: Human adenovirus type 40; HumanCorona1.inp: Human coronavirus 229E; MeaslesMora.inp: Measles virus strain Moraten; MeaslesSch.inp: Measles virus strain Schwarz; MurineHep11.inp: Murine hepatitis virus strain ML-11; MurineHep2.inp: Murine hepatitis virus strain 2; PRD1.inp: Enterobacteria phage PRD1; RatSialCorona.inp: Rat sialodacryoadenitis coronavirus; SARS.inp: SARS TOR2v120403; SIRV1.inp: Sulfolobus virus SIRV-1; SIRV2.inp: Sulfolobus virus SIRV-2. $S(T) = 0.988$.

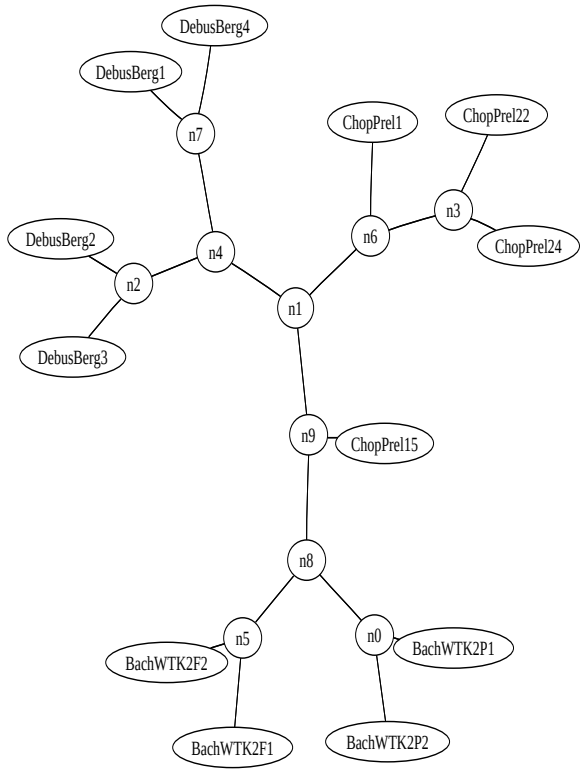

Fig. 10. Output for the 12-piece set. Legend: J.S. Bach [Wohltemperierte Klavier II: Preludes and Fugues 1,2— BachWTK2{F,P}{1,2}]; Chopin [Préludes op. 28: 1, 15, 22, 24 —ChopPrel{1,15,22,24}]; Debussy [Suite Bergamasque, 4 movements—DebusBerg{1,2,3,4}]. $S(T) = 0.968$.

from Canada's Michael Smith Genome Sciences Centre which had the first public SARS Coronovirus draft whole genome assembly available for download (SARS TOR2 draft genome assembly 120403). The NCD distance matrix was computed using the compressor bzip2. The relations in Figure 9 are very similar to the definitive tree based on medical-macrobio-genomics analysis, appearing later in the New England Journal of Medicine, [22]. We depicted the figure in the ternary tree style, rather than the genomics-dendrogram style, since the former is more precise for visual inspection of proximity relations.

B. Music

The amount of digitized music available on the internet has grown dramatically in recent years, both in the public domain and on commercial sites. Napster and its clones are prime examples. Websites offering musical content in some form or other (MP3, MIDI, ...) need a way to organize their wealth of material; they need to somehow classify their files according to musical genres and subgenres, putting similar pieces together. The purpose of such organization is to enable users to navigate to pieces of music they already know and like, but also to give them advice and recommendations ("If you like this, you might also like..."). Currently, such organization is mostly done manually by humans, but some recent research has been looking into the possibilities of automating music classification. In [10] we cluster music using the CompLearn Toolkit (NCD plus new quartet tree method). One example is a small set of classical piano sonatas, consisting of the 4 movements from Debussy's "Suite Bergamasque," 4 movements of book 2 of Bach's "Wohltemperierte Klavier," and 4 preludes from Chopin's "Opus 28." As one can see in Figure 10, our program does a pretty good job at clustering these pieces. The $S(T)$ score is also high: 0.968. The 4 Debussy movements form one cluster, as do the 4 Bach pieces. The only imperfection in the tree, judged by what one would intuitively expect, is that Chopin's Prélude no. 15 lies a bit closer to Bach than to the other 3 Chopin pieces. This Prélude no 15, in fact, consistently forms an odd-one-out in our other experiments as well. This is an example of pure data mining, since there is some musical truth to this, as no. 15 is perceived as by far the most eccentric among the 24 Préludes of Chopin's opus 28.

C. Mammalian Evolution

In recent years, as the complete genomes of various species become available, it has become possible to do whole genome phylogeny (this overcomes the problem that using different targeted parts of the genome, or proteins, may give different trees [32]). Traditional phylogenetic methods on individual genes depended on multiple alignment of the related proteins and on the model of evolution of individual amino acids.

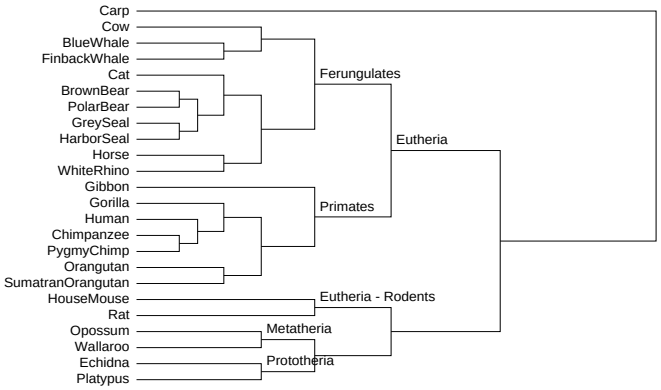

Fig. 11. The evolutionary tree built from complete mammalian mtDNA sequences of 24 species, using the NCD matrix of Figure 13. We have redrawn the tree from our output to agree better with the customary phylogeny tree format. The tree agrees exceptionally well with the NCD distance matrix: $S(T) = 0.996$.

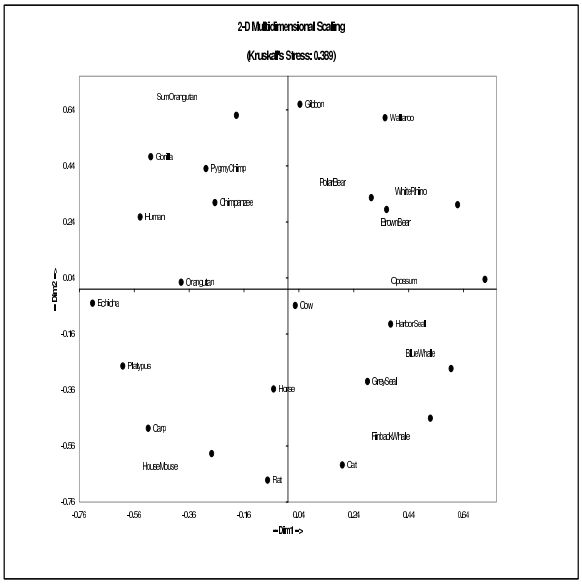

Fig. 12. Multidimensional clustering of same NCD matrix (Figure 13) as used for Figure 11. Kruskal's stress-1 = 0.389.

Neither of these is practically applicable to the genome level. In absence of such models, a method which can compute the shared information between two sequences is useful because biological sequences encode information, and the occurrence of evolutionary events (such as insertions, deletions, point mutations, rearrangements, and inversions) separating two sequences sharing a common ancestor will result in the loss of their shared information. Our method (in the form of the CompLearn Toolkit) is a fully automated software tool based on such a distance to compare two genomes. In evolutionary biology the timing and origin of the major extant placental clades (groups of organisms that have evolved from a common ancestor) continues to fuel debate and research. Here, we provide evidence by whole mitochondrial genome phylogeny for competing hypotheses in two main questions: the grouping of the Eutherian orders, and the Therian hypothesis versus the Marsupionta hypothesis.

Eutherian Orders: We demonstrate (already in [27]) that a whole mitochondrial genome phylogeny of the Eutherians (placental mammals) can be reconstructed automatically from *unaligned* complete mitochondrial genomes by use of an early form of our compression method, using standard software packages. As more genomic material has become available, the debate in biology has intensified concerning which two of the three main groups of placental mammals are more closely related: Primates, Ferungulates, and Rodents. In [7], the maximum likelihood method of phylogeny tree reconstruction gave evidence for the (Ferungulates, (Primates, Rodents)) grouping for half of the proteins in the mitochondial genomes investigated, and (Rodents, (Ferungulates, Primates)) for the other halves of the mt genomes. In that experiment they aligned 12 concatenated mitochondrial proteins, taken from 20 species: rat (*Rattus norvegicus*), house mouse (*Mus musculus*), grey seal (*Halichoerus grypus*), harbor seal (*Phoca vitulina*), cat (*Felis catus*), white rhino (*Ceratotherium simum*), horse (*Equus caballus*), finback whale (*Balaenoptera physalus*), blue whale (*Balaenoptera musculus*), cow (*Bos taurus*), gibbon (*Hylobates lar*), gorilla (*Gorilla gorilla*), human (*Homo sapiens*), chimpanzee (*Pan troglodytes*), pygmy chimpanzee (*Pan paniscus*), orangutan (*Pongo pygmaeus*), Sumatran orangutan (*Pongo pygmaeus abelii*), using opossum (*Didelphis virginiana*), wallaroo (*Macropus robustus*), and the platypus (*Ornithorhynchus anatinus*) as outgroup.

Marsupionta and Theria: The extant monophyletic divisions of the class Mammalia are the Prototheria (monotremes: mammals that procreate using eggs), Metatheria (marsupials: mammals that procreate using pouches), and Eutheria (placental mammals: mammals that procreate using placentas). The sister relationships between these groups is viewed as the most fundamental question in mammalian evolution [18]. Phylogenetic comparison by either anatomy or mitochondrial genome has resulted in two conflicting hypotheses: the gene-isolation-supported *Marsupionta hypothesis*: ((Prototheria, Metatheria), Eutheria) versus the morphology-supported *Theria hypothesis*: (Prototheria, (Methateria, Eutheria)), the third possiblity apparently not being held seriously by anyone. There has been a lot of support for either hypothesis; recent support for the Theria hypothesis was given in [18] by analyzing a large nuclear gene (M6P/IG2R), viewed as important across the species concerned, and even more recent support for the Marsupionta hypothesis was given in [15] by phylogenetic analysis of another sequence from the nuclear gene (18S rRNA) and by the whole mitochondrial genome.

Experimental Evidence: To test the Eutherian orders simultaneously with the Marsupionta- versus Theria hypothesis, we added four animals to the above twenty: Australian echidna (*Tachyglossus aculeatus*), brown bear (*Ursus arctos*), polar bear (*Ursus maritimus*), using the common carp (*Cyprinus carpio*) as the outgroup. Interestingly, while there are many species of Eutheria and Metatheria, there are only three species of now living Prototheria known: platypus, and two types of echidna (or spiny anteater). So our sample of the Prototheria is large. The whole mitochondrial genomes of the total of 24 species we used were downloaded from the GenBank Database on the world-wide web. Each is around 17,000

bases. The NCD distance matrix was computed using the compressor PPMZ. The resulting phylogeny, with an almost maximal $S(T)$ score of 0.996 supports anew the currently accepted grouping (Rodents, (Primates, Ferungulates)) of the Eutherian orders, and additionally the Marsupionta hypothesis ((Prototheria, Metatheria), Eutheria), see Figure 11. Overall, our whole-mitochondrial NCD analysis supports the following hypothesis:

$$\overbrace{(\underbrace{(primates, ferungulates)}_{\text{Eutheria}}(rodents, (Metatheria, Prototheria)))}^{\text{Mammalia}},$$

which indicates that the rodents, and the branch leading to the Metatheria and Prototheria, split off early from the branch that led to the primates and ferungulates. Inspection of the distance matrix shows that the primates are very close together, as are the rodents, the Metatheria, and the Prototheria. These are tightly-knit groups with relatively close NCD 's. The ferungulates are a much looser group with generally distant NCD 's. The intergroup distances show that the Prototheria are furthest away from the other groups, followed by the Metatheria and the rodents. Also the fine-structure of the tree is consistent with biological wisdom.

X. Hierarchical versus Flat Clustering

This is a good place to contrast the informativeness of hierarchical clustering with multidimensional clustering using the same NCD matrix, exhibited in Figure 13. The entries give a good example of typical NCD values; we truncated the number of decimals from 15 to 3 significant digits to save space. Note that the majority of distances bunches in the range $[0.9, 1]$. This is due to the regularities the compressor can perceive. The diagonal elements give the self-distance, which, for PPMZ, is not actually 0, but is off from 0 only in the third decimal. In Figure 12 we clustered the 24 animals using the NCD matrix by multidimenional scaling as points in 2-dimensional Euclidean space. In this method, the NCD matrix of 24 animals can be viewed as a set of distances between points in n-dimensional Euclidean space ($n \leq 24$), which we want to project into a 2-dimensional Euclidean space, trying to distort the distances between the pairs as little as possible. This is akin to the problem of projecting the surface of the earth globe on a two-dimensional map with minimal distance distortion. The main feature is the choice of the measure of distortion to be minimized, [14]. Let the original set of distances be $d_1, \ldots, d_k$ and the projected distances be $d'_1, \ldots, d'_k$. In Figure 12 we used the distortion measure *Kruskall's stress-1*, [21], which minimizes $\sqrt{(\sum_{i \leq k}(d_i - d'_i)^2)/\sum_{i \leq k} d_i^2}$. Kruskall's stress-1 equal 0 means no distortion, and the worst value is at most 1 (unless you have a really bad projection). In the projection of the NCD matrix according to our quartet method one minimizes the more subtle distortion $S(T)$ measure, where 1 means perfect representation of the relative relations between every 4-tuple, and 0 means minimal representation. Therefore, we should compare distortion Kruskall stress-1 with $1 - S(T)$. Figure 11 has a very good $1 - S(T) = 0.04$ and Figure 12 has a poor Kruskal stress 0.389. Assuming that the comparison is significant for small values (close to perfect projection), we find that the multidimensional scaling of this experiment's NCD matrix is formally inferior to that of the quartet tree. This conclusion formally justifies the impression conveyed by the figures on visual inspection.

References

[1] C.H. Bennett, P. Gács, M. Li, P.M.B. Vitányi, and W. Zurek. Information Distance, *IEEE Transactions on Information Theory*, 44:4(1998), 1407–1423.

[2] C.H. Bennett, M. Li, B. Ma, Chain letters and evolutionary histories, *Scientifi c American*, June 2003, 76–81.

[3] A. Ben-Dor, B. Chor, D. Graur, R. Ophir, D. Pelleg, Constructing phylogenies from quartets: Elucidation of eutherian superordinal relationships, *J. Computational Biology*, 5:3(1998), 377–390.

[4] V. Berry, T. Jiang, P. Kearney, M. Li, T. Wareham, Quartet cleaning: improved algorithms and simulations. Algorithms–Proc. 7th European Symp. (ESA99), LNCS vol. 1643, Springer Verlag, Berlin, (1999), 313-324.

[5] D. Bryant, V. Berry, P. Kearney, M. Li, T. Jiang, T. Wareham and H. Zhang. A practical algorithm for recovering the best supported edges of an evolutionary tree. *Proc. 11th ACM-SIAM Symposium on Discrete Algorithms*, January 9–11, 2000, San Francisco, California, USA, 287–296, 2000.

[6] P. Buneman, The recovery of trees from measures of dissimilarity. Pp. 387–395 in: F. Hodson, D. Kenadall, P. Tautu (Eds.), Proc. of the Anlo-Romanian conference, The Royal Society of London and the Academy of the Socialist Republic of Romania, The University Press, Edinburgh, Scottland, UK.

[7] Y. Cao, A. Janke, P. J. Waddell, M. Westerman, O. Takenaka, S. Murata, N. Okada, S. Pääbo, M. Hasegawa, Conflict among individual mitochondrial proteins in resolving the phylogeny of Eutherian orders, *J. Mol. Evol.*, 47(1998), 307-322.

[8] X. Chen, B. Francia, M. Li, B. McKinnon, A. Seker, Shared information and program plagiarism detection, *IEEE Trans. Inform. Th.*, 50:7(2004), 1545–1551.

[9] R. Cilibrasi, The CompLearn Toolkit, 2003, http://complearn.sourceforge.net/ .

[10] R. Cilibrasi, P.M.B. Vitanyi, R. de Wolf, Algorithmic clustering of music based on string compression, *Computer Music J.*, 28:4(2004), 49-67.

[11] R. Cilibrasi, P.M.B. Vitanyi, Clustering by compression, *IEEE Trans. Information Theory*, 51:4(2005), 1523- 1545.

[12] H. Colonius, H.H. Schulze, Trees constructed from empirical relations, *Braunschweiger Berichte as dem Institut fuer Psychologie*, 1(1977).

[13] H. Colonius, H.-H. Schulze, Tree structures for proximity data. *British Journal of Mathematical and Statistical Psychology*, 34(1981), 167-180.

[14] R.O. Duda, P.E. Hart, D.G. Stork, *Pattern Classifi cation*, 2nd Edition, Wiley Interscience, 2001.

[15] A. Janke, O. Magnell, G. Wieczorek, M. Westerman, U. Arnason, Phylogenetic analysis of 18S rRNA and the mitochondrial genomes of wombat, Vombatus ursinus, and the spiny anteater, Tachyglossus acelaetus: increased support for the Marsupionta hypothesis, *J. Mol. Evol.*, 1:54(2002), 71–80.

[16] T. Jiang, P. Kearney, and M. Li. A Polynomial Time Approximation Scheme for Inferring Evolutionary Trees from Quartet Topologies and its Application. *SIAM J. Computing*, 30:6(2000), 1942–1961.

[17] E. Keogh, S. Lonardi, and C.A. Rtanamahatana, Toward parameter-free data mining, In: *Proc. 10th ACM SIGKDD Intn'l Conf. Knowledge Discovery and Data Mining*, Seattle, Washington, USA, August 22—25, 2004, 206–215.

[18] J.K. Killian, T.R. Buckley, N. Steward, B.L. Munday, R.L. Jirtle, Marsupials and Eutherians reunited: genetic evidence for the Theria hypothesis of mammalian evolution, *Mammalian Genome*, 12(2001), 513–517.

[19] S. Kirkpatrick, C.D. Gelatt and M.P. Vecchi, *Science* 220 (1983) 671-680.

[20] A. Kraskov, H. Stögbauer, R.G. Adrsejak, P. Grassberger, Hierarchical clustering based on mutual information, 2003, http://arxiv.org/abs/q-bio/0311039

[21] J.B. Kruskal, Nonmetric multidimensional scaling: a numerical method, *Psychometrika*, 29(1964), 115–129.

	BlueWhale	BrownBear	Carp	Cat	Chimpanzee	Cow	Echidna	FinWhale	Gibbon	Gorilla	GreySeal	Horse	HarborSeal	HouseMouse	Human	Opossum	Orangutan	PolarBear	Platypus	PygmyChimp	Rat	SumOrang	Wallaroo	WhiteRhino
BlueWhale	0.005	0.906	0.943	0.897	0.925	0.883	0.936	0.616	0.928	0.931	0.901	0.898	0.896	0.926	0.920	0.936	0.928	0.929	0.907	0.930	0.927	0.929	0.925	0.902
BrownBear	0.906	0.002	0.943	0.887	0.935	0.906	0.944	0.915	0.939	0.940	0.875	0.872	0.910	0.934	0.930	0.936	0.938	0.937	0.269	0.940	0.935	0.936	0.923	0.915
Carp	0.943	0.943	0.006	0.946	0.954	0.947	0.955	0.952	0.951	0.957	0.949	0.950	0.952	0.956	0.946	0.956	0.953	0.954	0.945	0.960	0.950	0.953	0.942	0.960
Cat	0.897	0.887	0.946	0.003	0.926	0.897	0.942	0.905	0.928	0.931	0.870	0.872	0.885	0.919	0.922	0.933	0.932	0.931	0.885	0.929	0.920	0.934	0.919	0.897
Chimpanzee	0.925	0.935	0.954	0.926	0.006	0.926	0.948	0.926	0.849	0.731	0.925	0.922	0.921	0.943	0.667	0.943	0.841	0.946	0.931	0.441	0.933	0.835	0.934	0.930
Cow	0.883	0.906	0.947	0.897	0.926	0.006	0.936	0.885	0.931	0.927	0.890	0.888	0.893	0.925	0.920	0.931	0.930	0.929	0.905	0.931	0.921	0.930	0.923	0.899
Echidna	0.936	0.944	0.955	0.942	0.948	0.936	0.005	0.936	0.947	0.947	0.940	0.937	0.942	0.941	0.939	0.936	0.947	0.855	0.935	0.949	0.941	0.947	0.929	0.948
FinbackWhale	0.616	0.915	0.952	0.905	0.926	0.885	0.936	0.005	0.930	0.931	0.911	0.908	0.901	0.933	0.922	0.936	0.933	0.934	0.910	0.932	0.928	0.932	0.927	0.902
Gibbon	0.928	0.939	0.951	0.928	0.849	0.931	0.947	0.930	0.005	0.859	0.932	0.930	0.927	0.948	0.844	0.951	0.872	0.952	0.936	0.854	0.939	0.868	0.933	0.929
Gorilla	0.931	0.940	0.957	0.931	0.731	0.927	0.947	0.931	0.859	0.006	0.927	0.929	0.924	0.944	0.737	0.944	0.835	0.943	0.928	0.732	0.938	0.836	0.934	0.929
GreySeal	0.901	0.875	0.949	0.870	0.925	0.890	0.940	0.911	0.932	0.927	0.003	0.399	0.888	0.924	0.922	0.933	0.931	0.936	0.863	0.929	0.922	0.930	0.920	0.898
HarborSeal	0.898	0.872	0.950	0.872	0.922	0.888	0.937	0.908	0.930	0.929	0.399	0.004	0.888	0.922	0.922	0.933	0.932	0.937	0.860	0.930	0.922	0.928	0.919	0.900
Horse	0.896	0.910	0.952	0.885	0.921	0.893	0.942	0.901	0.927	0.924	0.888	0.888	0.003	0.928	0.913	0.937	0.923	0.936	0.903	0.923	0.912	0.924	0.924	0.848
HouseMouse	0.926	0.934	0.956	0.919	0.943	0.925	0.941	0.933	0.948	0.944	0.924	0.922	0.928	0.006	0.932	0.923	0.944	0.930	0.924	0.942	0.860	0.945	0.921	0.928
Human	0.920	0.930	0.946	0.922	0.667	0.920	0.939	0.922	0.844	0.737	0.922	0.922	0.913	0.932	0.005	0.949	0.834	0.949	0.931	0.681	0.938	0.826	0.934	0.929
Opossum	0.936	0.936	0.956	0.933	0.943	0.931	0.936	0.936	0.951	0.944	0.933	0.933	0.937	0.923	0.949	0.006	0.960	0.938	0.939	0.954	0.941	0.960	0.891	0.952
Orangutan	0.928	0.938	0.953	0.932	0.841	0.930	0.947	0.933	0.872	0.835	0.931	0.932	0.923	0.944	0.834	0.960	0.006	0.954	0.933	0.843	0.943	0.585	0.945	0.934
Platypus	0.929	0.937	0.954	0.931	0.946	0.929	0.855	0.934	0.952	0.943	0.936	0.937	0.936	0.930	0.949	0.938	0.954	0.003	0.932	0.948	0.937	0.949	0.920	0.948
PolarBear	0.907	0.269	0.945	0.885	0.931	0.905	0.935	0.910	0.936	0.928	0.863	0.860	0.903	0.924	0.931	0.939	0.933	0.932	0.002	0.942	0.940	0.936	0.927	0.917
PygmyChimp	0.930	0.940	0.960	0.929	0.441	0.931	0.949	0.932	0.854	0.732	0.929	0.930	0.923	0.942	0.681	0.954	0.843	0.948	0.942	0.007	0.935	0.838	0.931	0.929
Rat	0.927	0.935	0.950	0.920	0.933	0.921	0.941	0.928	0.939	0.938	0.922	0.922	0.912	0.860	0.938	0.941	0.943	0.937	0.940	0.935	0.006	0.939	0.922	0.922
SumOrangutan	0.929	0.936	0.953	0.934	0.835	0.930	0.947	0.932	0.868	0.836	0.930	0.928	0.924	0.945	0.826	0.960	0.585	0.949	0.936	0.838	0.939	0.007	0.942	0.937
Wallaroo	0.925	0.923	0.942	0.919	0.934	0.923	0.929	0.927	0.933	0.934	0.920	0.919	0.924	0.921	0.934	0.891	0.945	0.920	0.927	0.931	0.922	0.942	0.005	0.935
WhiteRhino	0.902	0.915	0.960	0.897	0.930	0.899	0.948	0.902	0.929	0.929	0.898	0.900	0.848	0.928	0.929	0.952	0.934	0.948	0.917	0.929	0.922	0.937	0.935	0.002

Fig. 13. Distance matrix of pairwise NCD . For display purpose, we have truncated the original entries from 15 decimals to 3 decimals precision.

[22] T.G. Ksiazek, et.al., A Novel Coronavirus Associated with Severe Acute Respiratory Syndrome, *New England J. Medicine*, Published at www.nejm.org April 10, 2003 (10.1056/NEJMoa030781).

[23] J.R. Koza, Hierarchical genetic algorithms operating on populations of computer programs, *Proc. 11th Intn'l Joint Conf. Artifi cial Intell. (IJCAI'89)*, Morgan-Kaufmann, 1989, 768–774.

[24] P.S. Laplace, *A philosophical essay on probabilities*, 1819. English translation, Dover, 1951.

[25] M. Li, J.H. Badger, X. Chen, S. Kwong, P. Kearney, and H. Zhang. An information-based sequence distance and its application to whole mitochondrial genome phylogeny, *Bioinformatics*, 17:2(2001), 149–154.

[26] M. Li and P.M.B. Vitányi. Algorithmic Complexity, pp. 376–382 in: *International Encyclopedia of the Social & Behavioral Sciences*, N.J. Smelser and P.B. Baltes, Eds., Pergamon, Oxford, 2001/2002.

[27] M. Li, X. Chen, X. Li, B. Ma, P.M.B. Vitányi. The similarity metric, *IEEE Trans. Inform. Th.*, 50:12(2004), 3250- 3264.

[28] M. Li and P.M.B. Vitányi. *An Introduction to Kolmogorov Complexity and its Applications*, Springer-Verlag, New York, 2nd Edition, 1997.

[29] T. Liu, J. Tang, B.M.E. Moret, Quartet methods for phylogeny reconstruction from gene orders. Dept. CS and Engin., Univ. South-Carolina, Columbia, SC, USA. Manuscript.

[30] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth. A.H. Teller and E. Teller, *J. Chem. Phys.* 21 (1953) 1087-1092.

[31] R. Piaggio-Talice, J. Gordon Burleigh, O. Eulenstein, Quartet supertrees. Chapter 4, pp. 173–191 in: O.R.P. Beninda-Edmonds (Ed.), *Phylogenetic Supertrees: Combining Information to Reveal the Tree of Life*. Computational Biology, volume 3 (Dress. A., series ed.), Kluwer Academic Publishers, 2004.

[32] A. Rokas, B.L. Williams, N. King, S.B. Carroll, Genome-scale approaches to resolving incongruence in molecular phylogenies, *Nature*, 425(2003), 798–804 (25 October 2003).

[33] U. Roshan, B.M.E. Moret, T. Warnow, T.L. Williams, Performance of supertree methods on various datasets decompositions, pp 301–328 in: O.R.P. Beninda-Edmonds (Ed.), *Phylogenetic Supertrees: Combining Information to Reveal the Tree of Life*. Computational Biology, volume 3 (Dress. A., series ed.), Kluwer Academic Publishers, 2004.

[34] N. Saitou, M. Nei, The neighbor-joining method: a new method for reconstructing phylogenetic trees, *Mol. Biol. Evol.*, 4(1987), 406–425.

[35] M. Steel, The complexity of reconstructiong trees form qualitative characters and subtrees, *Journal of Classifi cation*, 9(1992), 91–116.

[36] K. Strimmer, A. von Haeseler, Quartet puzzling: A quartet maximum-likelihood method for reconstructing tree topologies, *Mol. Biol. Evol.*, 13:7(1996), 964–969.

[37] P.M.B. Vitanyi, A discipline of evolutionary programming, *Theoret. Comp. Sci.*, 241:1-2 (2000), 3–23.

[38] S. Wehner, Analyzing network traffi c and worms using compression, http://arxiv.org/pdf/cs.CR/0504045

[39] J. Weyer-Menkoff, C. Devauchelle, A. Grossmann, S. Grünewald, Integer linear programming as a tool for constructing trees from quartet data. Preprint from the web submitted to Elsevier Science.