# A general paradigm for fast, adaptive clustering of biological sequences

Knut Reinert, Markus Bauer, Andreas Döring, Gunnar W. Klau, Aaron L. Halpern

reinert,mbauer,doering,gunnar@mi.fu-berlin.de
ahalpern@venterinstitute.org

**Abstract:** There are numerous methods that compute clusterings of biological sequences based on pairwise distances. This necessitates the computation of $O(n^2)$ sequence comparisons. Users usually want to apply the most sensitive distance measure which normally is the most expensive in terms of runtime. This poses a problem if the number of sequences is large or the computation of the measure is slow.

In this paper we present a general heuristic to speed up distance based clustering methods considerably while compromising little on the accuracy of the results. The speedup comes from using fast comparison methods to perform an initial 'top-down' split into relatively homogeneous clusters, while the slower measures are used for smaller groups. Then profiles are computed for the final groups and the resulting profiles are used in a bottom-up phase to compute the final clustering.

The algorithm is general in the sense that any sequence comparison method can be employed (e.g. for DNA, RNA or amino acids). We test our algorithm using a prototypical implementation for agglomerative RNA clustering and show its effectiveness.

## 1 Introduction

Clustering biological sequences like DNA, RNA or proteins is a daily task in bioinformatics research (see for example [ZLZ05, LUDT05, YLL99, SLL02]). The main challenge lies in the ever growing number of sequences to be compared which makes classic phylogenetic tree construction too costly. Instead, researchers are clustering sequences based on measures of similarity which might be purely sequence based [DSO78, HH92], or even incorporate structural features in the case of RNA or proteins. Lazareva-Ulitsky and colleagues [LUDT05] showed that functional similarity scores and evolutionary distance are indeed highly correlated and hence groupings based on the results of hierarchical clustering techniques often sufficiently approximate the evolutionary relations. Almost all clustering methods have in common that they require as input a matrix containing the pairwise distances (or similarities) of the sequences to be clustered. Although hierarchical clustering is fast, this can pose a problem if there are many sequences to cluster and the method for pairwise comparison is relatively expensive. For example in the case of RNA comparison the computation of the single sequence structure similarity is for most programs in the range of several seconds to several minutes. If the comparison took one second, then the pairwise comparison of $10,000$ sequences would take about 1.6 CPU years. Clearly this is not feasible and one has to resort to faster methods (e.g. [LJG01]). This, however, has the unwanted effect that the faster method is less accurate which might lead to incorrect clustering results.

In this paper we describe a general procedure to address this problem and to speed up the computation of any method based on pairwise distances considerably. We exemplify our idea by implementing a method to quickly cluster a set of RNA sequences using agglomerative clustering (average linkage) and an accurate, yet costly distance measure. We compare the results to the clustering obtained without our speedup.

Informally our method is illustrated in Fig. 1. Our goal is to minimize the number of comparisons using the expensive method. To achieve this, we first heuristically divide the input top-down into smaller, meaningful groups using a fast comparison method. Success at this point requires some "easy" splits in the data. Then we compute profiles of all sequences in these groups and all pairwise distances between the profiles. We finally apply the clustering algorithm of our choice to this data. The general idea of dividing data into smaller groups is not new [LJG01], however the systematic approach we take is.

We use a hierarchy of methods to compute similarities between sequences. In this hierarchy the methods become subsequently slower (from the first method on level 1 to the most sensitive method at the final level) while they also become more accurate. As illustrated in Fig. 1 we divide the initial input set into groups using the fastest method. Depending on their size, those groups are then further refined by using slower, but more accurate methods, until the groups are small enough for applying the final method.

,

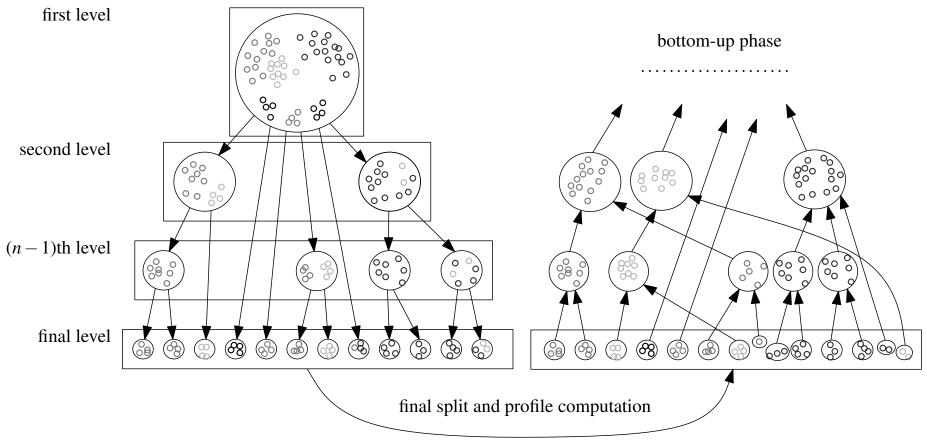


Figure 1: Illustration of our approach. Each colored circle represents a sequence and their spatial proximity should resemble their distance. The colors should indicate the "true" cluster they belong to.

At the final level the groups are possibly split again using the most sensitive method and then for the $c$ resulting groups of this split we compute a profile. This requires the computation of all pairwise distances for each group. Those distances are finally used as the input of any bottom-up clustering method (e.g. agglomerative clustering). The main rationale behind using a hierarchy of methods is that many data sets will exhibit characteristics that allow them to be divided into large divergent groups by almost any method, while more sensitive methods may be required to resolve structure within the smaller groups. This in turn allows us to use very fast, less sensitive methods first, and more sensitive but more costly methods later. Obviously our method will give best results (both regarding accuracy and speed), if we divide the input into reasonable sets with few mistakes (as illustrated in Fig. 1). However, our method will always use the most sensitive method at the final level, so mistakes as illustrated in Fig. 1 can be corrected before the bottom-up phase.

The method outlined raises a number of questions. For example, what should the size of the final groups be? How do we incorporate the intrinsic properties of the pairwise distances into our top-down phase? What speed-up can be achieved? What are the trade offs of using our methods versus computing a clustering based on an all-against-all comparison?

We will address those questions in Sect. 2. After this, we describe the effectiveness of our method by implementing a method to cluster RNA sequences using structure-sequence alignment algorithms and demonstrate how much time we can gain and how sensitive our method is.

## 2 Methods

We are looking for a partition of $n$ sequences $\{S_1, S_2, \ldots, S_n\} =: M$ into about $c$ clusters of size $u \approx \lceil n/c \rceil$. In the following we argue with normalized distance functions (between 0 and 1). We assume that normalized similarity functions can be easily converted into a distance function, since we do not make any assumptions about the functions. In particular they do not have to form a metric.

Our top-down method is generic and supports strategies inspired by average, single, or complete linkage clustering. Given a distance function $d$ that computes the distance between two sequences and two sets $A$ and $B$ of sequences (with $|A \cup B| > 1$), we define:

$$\begin{aligned} d_+(A,B) &:= \sum_{a \in A, b \in B, a \neq b} d(A,B) \\ d_{\min}(A,B) &:= \min_{a \in A, b \in B, a \neq b} d(A,B) \\ d_{\max}(A,B) &:= \max_{a \in A, b \in B, a \neq b} d(A,B) \end{aligned}$$

In the case that $|A \cup B| = 1$, the distance is naturally defined as 0. We define the average distance of two nonempty, disjoint sets $A$ and $B$ as $\overline{d_+}(A,B) := \frac{d_+(A,B)}{|A| \cdot |B|}$. We also define $\overline{d_+}(A,A) := \frac{2 \cdot d_+(A,A)}{|A| \cdot (|A|-1)}$ for sets with $|A| > 1$ and as 0 for $|A| = 1$. We also define $\overline{d_{\min}}(A,B) := d_{\min}(A,B)$, and $\overline{d_{\max}}(A,B) := d_{\max}(A,B)$.

Note that after joining two sets $A$ and $B$, we can compute the distances $d_{\mathrm{op}}(A \cup B, C)$ to another set $C$ simply by $d_{\mathrm{op}}(A \cup B, C) = \mathrm{op}\ (d_{\mathrm{op}}(A,C), d_{\mathrm{op}}(B,C))$ and $d_{\mathrm{op}}(A \cup B, A \cup B) = \mathrm{op}\ (d_{\mathrm{op}}(A,A), d_{\mathrm{op}}(B,B), d_{\mathrm{op}}(A,B))$

Our top-down method is a variation of a standard agglomerative clustering algorithm: $\overline{d_+}$ is used for 'average linkage', $\overline{d_{\min}}$ for 'single linkage', and $\overline{d_{\max}}$ for 'complete linkage' clustering.

## 2.1 The choice of $c$

One of the first questions arising when implementing our method is how large the final groups should be. We can compute a good target size as follows: If we have $c$ final groups, each group contains on average $n/c$ many sequences. For each group we have to compute $\binom{n/c}{2}$ comparisons using our expensive final method. Given this, one would like to make the groups not too big. However, if the groups are small, then $c$ is larger and we also have to compute $\binom{c}{2}$ comparisons between group profiles. Those are two conflicting goals. The total number of comparisons is:

$$v(c) := \frac{1}{2} \cdot (\frac{n^2}{c} + c^2) \approx \binom{n/c}{2} \cdot c + \binom{c}{2}$$

If we take the first derivative $v'(c) = c - \frac{n^2}{2c^2}$ we can easily see that the function $v$ has a minimum at $c_{\text{min}} = \sqrt[3]{\frac{n^2}{2}}$ and hence a good final group size would be approximately $\sqrt[3]{2n}$. If we consider the speedup we observe that $v(1)/v(c_{\text{min}})$ is in $\Theta(n^{2/3})$. For example for $n = 2000$ we could reduce the number of expensive comparisons from about 2 million to about 24000 comparisons, or (assuming 1 second for each comparison) from 23 days to less than seven hours.

## 2.2 The division threshold $t$

During the top-down phase we divide each group at a given level into subgroups using all pairwise distances of the members of this group computed with the respective method of this level. The question is of course, how we define the subgroups for the next level? Assume the maximal input size that can be handled by the method of the next level is $u$. As simple strategy would be to compute a standard agglomerative clustering and identify the $u$ nearest elements (one group), then the next $u$ elements (another group) and so on. This has two disadvantages. First, we always pass to the next method the largest set it can handle, and second the grouping does not use already computed distance information which for example could clearly exhibit many clusters of size smaller than $u$. Hence we decided to implement a strategy that examines the distance data derived at the initial level.

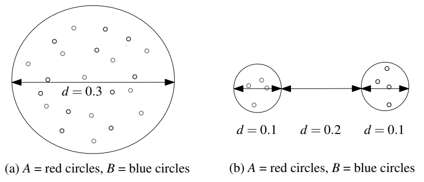
To define the subgroups we use for $\text{op} \in \{+, \min, \max\}$, the *linkage quotient* for two disjoint, nonempty sets $A$ and $B$ of sequences that is defined as:

$$T_{\text{op}}(A, B) = \lambda_{\text{sep}}(A, B) \cdot \lambda_{\text{size}}(A, B) \ , \tag{1}$$

with

$$\begin{aligned}\lambda_{\text{sep}}(A,B) &= \frac{\overline{d_{\text{op}}}(A,B)}{\max\{d_{\text{pm}}(A),\overline{d_{\text{op}}}(A,A)\}+\max\{d_{\text{pm}}(B),\overline{d_{\text{op}}}(B,B)\}} \quad \text{and} \\ \lambda_{\text{size}}(A,B) &= \frac{4\cdot|A|\cdot|B|}{(|A|+|B|)^2}\ .\end{aligned}$$

where $d_{\text{pm}}(A)$ is the smallest positive distance of elements in $A$ to any other element in the input set. The division by $\overline{d_{\text{op}}(B,B)}$ should accommodate for the "density" of a group. In the case of singletons we approximate this by introducing $d_{\text{pm}}(A)$.



$d = 0.3$
$d = 0.1$
$d = 0.2$
$d = 0.1$
(a) $A$ = red circles, $B$ = blue circles
(b) $A$ = red circles, $B$ = blue circles

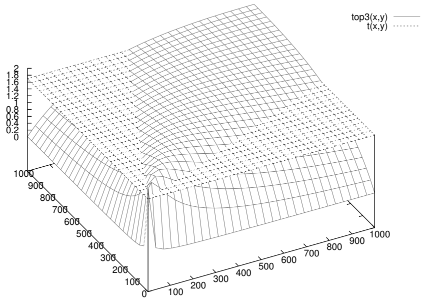


Figure 2: Left figure: In situation (a) we want to join the groups $A$ and $B$, whereas in (b) we might want to keep them apart. The value $d$ denotes the approximate distance of each pair of sequences. Right figure: 3D plot of linkage quotient with maximal value 2 for different group sizes $t = 1.7$. The axes show the number of elements in $A$ and $B$. All combinations that lie above the plane would not be joined by the procedure SPLITSET.

To define the groups for the next level we start with singletons for each element of a group and greedily join sets. The smaller the linkage quotient $T_{\text{op}}(A,B)$ for two sets $A$ and $B$, the more likely they are joined together. The idea behind the definition of linkage quotient is that the separation factor $\lambda_{\text{sep}}(A,B)$ becomes larger, the better $A$ and $B$ are separated compared to the "density" they already have (This motivates our definition of "distance" for singletons. If the distance was 0 or small we would never join them). For example $\lambda_{\text{sep}}(A,B)$ would be near 0.5 if we sampled randomly from sequences with similar distances (see Fig. 2 (a)). Figure 2 (b) shows a situation where the separation factor $\lambda_{\text{sep}}(A,B)$ is around 1. Generally it can become very large if very dense clusters are far apart. The size factor $\lambda_{\text{size}}(A,B)$ is normalized to 1 and is larger if the two sets are similar in size reaching its maximum for $|A| = |B|$. The minimum value approaches $4/(|A| + |B|)$.

Hence we will preferably join badly separated sets of unequal size for subsequent separation with a more sensitive method, whereas equally sized, well separated sets will be joined much later, if at all. More specifically, the algorithm joins two disjoint sets $A$ and $B$ if $T_{\text{op}}(A,B) \leq t$ for a suitable *linkage threshold* $t$. Figure 2 shows the $T_{\text{op}}$ function for varying group sizes and with $\overline{d_{\text{op}}}(A,B) = 0.2, \overline{d_{\text{op}}}(A,A) = \overline{d_{\text{op}}}(B,B) = 0.05$ together with a cut-off plane at $t = 1.7$.

How can we define a suitable cutoff for the linkage quotient? We argue as follows: From the discussion in the previous section we know that we would like to obtain $c_{\text{min}}$ groups each of size $\approx n/c_{\text{min}}$. Also, if we assume for the sake of argument that the final $c_{\text{min}}$ groups are the most homogeneous with respect to our final method, then the $c_{\text{min}} \cdot \binom{n/c_{\text{min}}}{2}$ smallest distances should occur in the pairwise comparisons of elements in the respective groups. All larger distances should occur between elements of different groups. Our threshold should definitely encourage the joining of groups that have small inter and small intra distance whereas it should refrain from joining groups with large distances in between the groups. We can choose a threshold $t_{\text{cut}}$ that achieves this as follows. We sort all pairwise distances using our initial method. Then we take the average of the $c_{\text{min}} \cdot \binom{n/c_{\text{min}}}{2}$ smallest distances and denote it by $d_{\text{intra}}$ as well as the average of the remaining, large distances and denote it by $d_{\text{inter}}$. Then we define $t_{\text{cut}} = d_{\text{inter}}/(2 \cdot d_{\text{intra}})$.

We illustrate that this is a reasonable choice using the following small examples of 12 sequences.

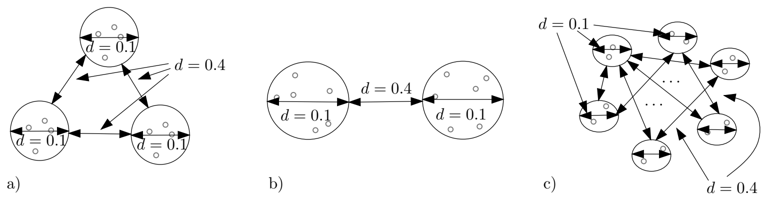


Figure 3: Assume that $d_{\text{intra}}$ is always $0.1$ and $d_{\text{inter}}$ is always $0.4$.

For $12$ sequences the optimal group size is $\lceil 12/4.16 \rceil = 3$ and $c_{\text{min}} = 4$. Hence we take the average of the $4 \cdot \binom{3}{2} = 12$ distances as $d_{\text{intra}}$ and the average of the remaining distances as $d_{\text{inter}}$. For the three cases depicted in Fig. 3 this yields: a) $t_{\text{cut}} = d_{\text{inter}}/(2 \cdot d_{\text{intra}}) = 0.366/0.2 = 1.83$, b) $t_{\text{cut}} = 0.3/0.2 = 1.5$, and c) $t_{\text{cut}} = 0.4/0.25 = 1.6$.

It is easy to check that our algorithm will perform as expected in all three cases. It will join singletons with small distances (since the linkage coefficient is $\frac{0.1}{0.1+0.1} \cdot \frac{4 \cdot 1 \cdot 1}{2^2} = 0.5$, which is smaller than the threshold) and will not join singletons with large distance since the linkage coefficient is $\frac{0.4}{0.1+0.1} \cdot \frac{4 \cdot 1 \cdot 1}{2^2} = 2$. Also, it will not join the low distance two element groups in c) since the linkage coefficient is $\frac{0.4}{0.1+0.1} \cdot \frac{4 \cdot 2 \cdot 2}{4^2} = 2$.

## 2.3 The top-down-phase in detail

We are now ready to describe the top-down procedure in pseudo code. Recall that we are looking for a partition of $n$ sequences $\{S_1, S_2, \ldots, S_n\} =: M$ into about $c$ clusters of size $u \approx \lceil n/c \rceil$.

Assume $S$ is a subset of $M$ that contains candidates for being grouped together (initially the top-down phase starts with $M$ itself.) The constant $u$ is the maximal group size for a given level (provided by the user) and and $Res$ is the resulting set of groups that is passed to the next level of comparisons. The variable $t$ denotes the linkage threshold and $D$ the matrix of all pairwise distances. Then the following pseudo code in Fig. 4 describes how subgroups are generated.

In this pseudo code the function UPDATEDISTANCES chooses the appropriate method $A_i$ and

```
SPLITSET($D, S, u, t$)
    $PS \leftarrow \{\{S_i\} | S_i \in S\}$;
    $Res \leftarrow \emptyset$;
    while $| PS | > 1$
        do
            Select $X, Y \in PS$ with $\overline{d_{op}}(X, Y)$ minimal; $|X| \geq |Y|$;
            if ($|X| + |Y| \leq u$) and ($T_{\text{op}}(X, Y) \leq t$)
              then join $X$ and $Y$;
                    UPDATEDISTANCES($D, X, Y, S$);
                    $PS$.remove($X$);
                    $PS$.remove($Y$);
                    $PS$.insert($X \cup Y$);
              else  extract $X$
                    $PS$.remove($X$);
                    $Res$.insert($X$);
    $Res \leftarrow Res \cup PS$;
    return $Res$;
```

Figure 4: Pseudo code for procedure SPLITSET.

updates only the distances involving $X$ and $Y$.

Assume we are given our preferred and most sensitive method $A_{\text{final}}$ for which we require that it can compute a profile (needed for the later bottom-up phase). In addition we are given a set of methods to compute distances between two sequences where $A_{\text{initial}} = A_1$ is the fastest and $A_{\text{final}} = A_n$ is the slowest. Further we are given $u_i$ for all $1 \leq i \leq n$ with $u_1 = n + 1 > u_2 > \ldots > u_{\text{final}}$.

Then the pseudo code in Fig. 5 describes the complete subdivision phase.

In the above code the procedure COMPUTEDISTANCES($D, S, A_i$) only recomputes the pairwise distances of elements in $S$ using method $A_i$. Note that the final method is never used in the while loop, but only in the last call of SPLITSET used for the profile computation. The returned profiles can now be used in the bottom-up phase which can employ any sort of clustering that requires a distance matrix.

## 3 Results

To demonstrate our method we implemented a program for computing an average linkage clustering of RNA sequences based on a pairwise structural distance measure.

Using this program we demonstrate how much we can decrease the run time in reality by comparing the runtime of our method to a normal agglomerative clustering procedure for which we compute all pairwise distances. Secondly we show that the definition of a linkage quotient thresh-

```
PARTITION($M$)
    $n \leftarrow |M|$;
    $c_{\text{min}} \leftarrow \sqrt[3]{n^2/2}$;
    $u_{\text{final}} = b \cdot \lceil n/c_{\text{min}} \rceil$;        // $b$ between 1.0 and 2.0
    COMPUTEDISTANCES($D, M, A_{\text{initial}}$);
    $t_{\text{cut}} \leftarrow \alpha \cdot$ COMPUTETHRESHOLD($D, n, c_{\text{min}}$);
    $Res \leftarrow$ SPLITSET($D, M, u_2, t_{\text{cut}}$);
    while $\exists S \in Res$ with $|S| > u_{\text{final}}$
        do
            choose $i$ with $u_i \leftarrow \min u_j$ with $|S| < u_j$;
            COMPUTEDISTANCES($D, S, A_i$);
            $R \leftarrow$ SPLITSET($D, S, u_{i+1}, t_{\text{cut}}$);
            $Res = Res - S$;
            $Res = Res \cup R$;
    $Profiles \leftarrow \emptyset$;
    for all $S \in Res$
        do
            COMPUTEDISTANCES($D, S, A_{\text{final}}$);
            $R \leftarrow$ SPLITSET($D, S, u_{\text{final}}, t_{\text{cut}}$);
            $Profiles \leftarrow Profiles \cup$ profiles of all $S \in R$ computed with $A_{\text{final}}$;
    return $Profiles$;
```

Figure 5: Pseudo code for main procedure.

old effectively prevents small, well-defined clusters from being merged only because the final group size is not reached. Finally we show that our prototypical RNA clustering procedure produces high quality results. Our tests are intended to show the effectiveness of the generic method for speeding up clustering, rather than the merits of the specific methods for RNA comparison that are employed. However we give exemplary evidence of its performance.

The program was implemented using the SeqAn library for biological sequence analysis [DWRR, DWRR07]. We start by describing the methods we use for our implementation and point out again that our hierarchical method will work for any set of distance functions.

**Methods used for RNA comparison** We use the following methods to cluster RNA sequences. As the final, and most costly, method we choose LaRA [BKR07]. LaRA produces high-quality multiple sequence-structure RNA alignments. It transforms the original problem into an integer linear programming formulation and finds provably optimal or near-optimal solutions using Lagrangian relaxation. LaRA is part of the LiSA library for structural alignments [KBMP07].

To be applied within the clustering framework described in this paper, our RNA alignment method provides two functionalities: First, given a number of sequences $S$, it returns a consensus representation of a multiple structural alignment of $S$. This occurs at the final level, where LaRA computes multiple structural alignment for each of the small groups. Currently, we compute the multiple alignment by first computing all pairwise sequence-structure alignments of $S$ and then passing this information to the program T-COFFEE [NHH00], which computes a consistency-based multiple alignment. Given such a multiple alignment we compute a consensus base pair probability matrix based on the structural information of the sequences in $S$.

Second, as the final method is used in the bottom-up phase, we must be able to align two of these consensus representations and return a distance that reflects the similarity of the two representations. This distance is used by the clustering method in the bottom-up phase. LaRA employs a profile-structure alignment approach for this task.

As our initial method we implemented a fast $k$-mer based distance as described in [Edg04]. More specifically, if $s_1$ and $s_2$ are two sequences of length $n_1$ and $n_2$ resp., then $\mathrm{dist\_kmer}(s_1, s_2) = 1 - \sum_q \min(c_1(q), c_2(q))/(\min(n_1, n_2) - k + 1)$, where the sum ranges over all possible $k$-mers $q$ and $c_1(q)$ and $c_2(q)$ is the number of occurrences of $q$ in the respective sequences. In our experiments we used $k = 4$.

**Runtime evaluation** We used subsets of 39 RNAs stemming from 4 different families from the Rfam database [GJMM$^+$05]: the families are RF00034 (RprA), RF00198 (SL1), RF00243 (traJ), and RF00434 (BTE) which all have an average sequence length between 100-110 nucleotides (we list the accession numbers in Appendix A.1). The subsets contain 7, 94, 6, and 17 sequences, resp.

We then randomly generated subsets of size $3, 6, 9, 12, 16, 20, 25, 30, 35$. For this test we only used our fast initial $k$-mer method and as the final method LaRA, so the results do not get diluted by intermediate methods. Figure 6 clearly illustrates that our method achieves a large speedup.

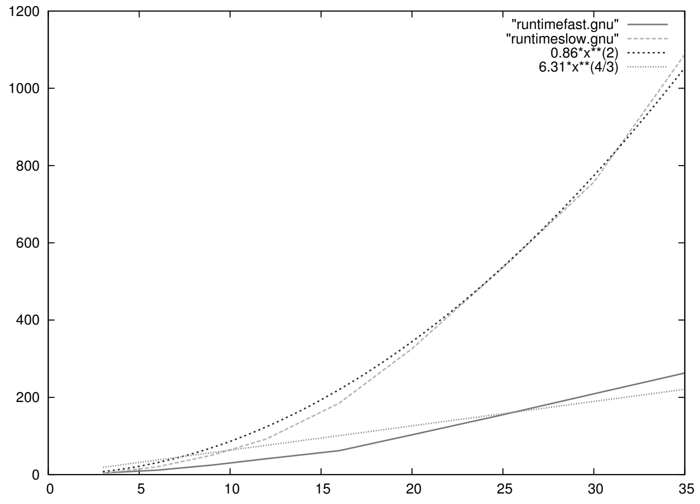


Figure 6: Run time comparison of the hierarchical method compared to the time needed for all pairwise comparisons. Data shown together with a fitted polynomial. The $x$-axis shows the number of sequences, the $y$-axis the run time in seconds.

It shows the original measurements and fitted polynomials. The usual method has to compute all pairwise comparisons so the pairwise comparisons fit neatly a quadratic curve (least square fit is $0.86 \cdot x^2$). Our method is considerably faster but does not completely reach the theoretical optimal speedup. The least square fit results in the polynomial $6.3 \cdot x^{4/3}$ but the real runtime seems to follow a polynomial with slightly higher degree (the discrepancy is due to the fact that the pairwise distance computation for profiles is indeed slower than the pairwise distance computation for single sequences and also from the not perfect cluster sizes).

**Effectiveness of linkage threshold and quality of clustering** In order to evaluate the effectiveness of the linkage threshold we constructed an input data set of 19 rather well separated RNA sequences: the data set is a subset of the 4 Rfam families that we used for our runtime test and contains 4 sequences from RF00034 (sequences $3, 4, 5, 6$), 9 from RF00198 (sequences $7, 8, 9, 10, 11, 12, 13, 14, 15$), 3 from RF00243 (sequences $16, 17, 18$), and 3 from RF00434 (sequences $0, 1, 2$). We then ran the method with a final maximal group size of 19. That means that the procedure SPLITSET will stop joining sets only if the linkage threshold is exceeded. After the distance computation with our initial $k$-mer method we inspected the groups computed by the first call of SPLITSET and found that the program forms the groups $\{0, 1, 2\}$, $\{3, 4, 5, 6\}$, $\{7, 8, 9, 10, 11, 12, 13, 14, 15\}$, and $\{16, 17, 18\}$ which is actually the correct clustering already. This shows that the linkage quotient indeed stops the formation of larger groups if the groups are separated sufficiently. The subsequent computation of the inter-group distances with the sensitive method resulted of course in the correct clustering.

As a second test we used 15 sequences from 3 families (again numbered {0,1,2,3,4}, {5,6,7,8,9}, {10,11,12,13,14}) from the Rfam database: 5 sequences from RF00504 (Glycine riboswitch), RF00005 (tRNA), and RF00031 (SECIS elements) showing an average pairwise sequence identity of 0.45, 0.33, and 0.35, respectively. We list the accession numbers in App. A.2. This test set was chosen because it exhibits very little sequence similarity and is hard to cluster using sequence similarity alone (Figure 9 in the appendix shows the clustering result obtained with T-COFFEE). Indeed, our $k$-mer method produces distances between the sequences in the range of $0.62 - 0.88$ and does not reveal clear clusters. On the other hand, our sensitive method is able to resolve the correct clusters in this difficult test set using the full distance matrix (see nodes $23, 24$, and $27$ in Fig. 7.)

How will our method work on this extreme example? Our method computes the following grouping before the final split with the most sensitive method: $\{0, 1, 2\}$,$\{3\}$,$\{4, 6\}$,$\{7, 8, 9\}$,$\{10, 13\}$, and $\{11, 12, 14\}$ and finally computes the clustering depicted in Fig. 8. As it can be seen we compute two clusters correctly, but do not succeed for the third. Nevertheless our fast method is much better than the result computed by T-COFFEE.

## 4 Conclusion

We presented in this paper a general adaptive scheme to speed up sequence based clustering algorithms in the best case from $O(n^2)$ to $O(n^{4/3})$. We demonstrated its effectiveness on a prototypic implementation of an RNA clustering tool.

As future work we will apply our method also to purely sequence based clustering of DNA and proteins and conduct a thorough evaluation of the resulting tools both in terms of speedup and accuracy (there are several recent methods for evaluating clusterings [Mei02, ZLZ05, SLL02, LUDT05, THG07] which we will use in this context). Also we plan to employ sparse matrix techniques in case that the number of sequences in the input is prohibitive for storing the entire distance matrix.

## References

[BKR07] M. Bauer, G. W. Klau, and K. Reinert. Accurate Multiple Sequence-Structure Alignment of RNA Sequences Using Combinatorial Optimization. *BMC Bioinformatics*, 2007. To appear.

[DSO78] M. O. Dayhoff, R. M. Schwartz, and B. C. Orcutt. A model of evolutionary change in proteins. In *Atlas of Protein Sequence and Structure*, volume 5, pages 345–352, 1978.

[DWRR] A. Döring, D. Weese, T. Rausch, and K. Reinert. SeqAn – the C++ library for sequence analysis. `http://www.seqan.de`.

[DWRR07] A. Döring, D. Weese, T. Rausch, and K. Reinert. SeqAn: An efficient, generic C++ library for sequence analysis. *BMC Bioinformatics, submitted*, 2007.

[Edg04] R.C. Edgar. Local homolgy recognition and distance measures in linear time using compressed amino acid alphabets. *Nucleic Acids Research*, 32(1):380–385, 2004.

[GJMM$^+$05] S. Griffiths-Jones, S. Moxon, M. Marshall, A. Khanna, S. R. Eddy, and A. Bateman. Rfam: Annotating Non-Coding RNAs in Complete Genomes. *Nucl. Ac. Res.*, 33:D121–D141, 2005.

[HH92] S. Henikoff and J. G. Henikoff. Amino Acid Substitution Matrices from Protein Blocks. *Proceedings of the National Academy of Sciences*, 89:10915–10919, 1992.

[KBMP07] G. W. Klau, M. Bauer, P. May, and L. Petzold. LiSA: a library for structural alignment algorithms, 2007. `http://www.planet-lisa.net`.

[LJG01] W. Li, L. Jaroszewski, and A. Godzik. Clustering of highly homologous sequences to reduce the size of large protein databases. *Bioinformatics*, 17(3):282–283, 2001.

[LUDT05] B. Lazareva-Ulitsky, K. Diemer, and P.D. Thomas. On the quality of tree-based protein classification. *Bioinformatics*, 21(9):1876–1890, 2005.

[Mei02] M. Meila. Comparing Clusterings. University of Washington, Statistics, Technical Report, 2002.

[NHH00] C. Notredame, D. G. Higgins, and J. Heringa. T-Coffee: A novel method for fast and accurate multiple sequence alignment. *Journal of Molecular Biology*, 2000.

[SLL02] O. Sasson, N. Linial, and M. Linial. The metric space of proteins – comparative study of clustering algorithms. *Bioinformatics*, 18:S14–S21, 2002.

[THG07] E. Torarinsson, J.H. Havgaard, and J. Gorodkin. Multiple structural alignment and clustering of RNA sequences. *Bioinformatics*, 23(8):926–932, 2007.

[YLL99] G. Yona, N. Linial, and M. Linial. ProtoMap: automatic classification of protein sequences, a hierarchy of protein families and local maps of protein space. *Proteins*, 37:360–378, 1999.

[ZLZ05] D. Zhou, J. Li, and H. Zha. A new Mallows Distance Based MEtric For Computing Clusterings. In *Proceedings of 22nd International Conference on Machine Learning*, pages 1028–1035, 2005.

# Appendix

## A Accession Numbers

### A.1 Runtime Evaluation

- RF00034 (RprA):
U000962.1768396-1768503, AE0169841.44786-44893, AE0087591.13089-12982, AL6272711.108510-108617, BX5718671.288515-288628, BX9508511.2106268-2106157, AJ4141521.107933-108041

- RF00198 (SL1):
U908311.1-100, U908321.1-99, U316461.55-159, U316351.55-159, U316371.55-157, AJ2790331.195-299, AJ3115271.55-157, AF2971191.1277-1381, AF2821791.149-252,

- RF00243 (traJ):
M209411.109-216, AF5506791.2929-3039,X558961.933-1040, AF2376971.390-500,AE0064711.61487-61592,AF3895291.1005-1112

- RF00434 (BTE):
AB0760431.1872-1973, AB0760421.1877-1973, AB0760461.1867-1967, D110281.4805-4903, D110321.4816-4915,AB0760471.1872-1973, AJ0074921.1972-2071, X076531.4819-4918, AJ0074911.1974-2073, AF2351671.4816-4915, D857831.4816-4915, X800501.6-105, L240491.5262-5363,AB0760481.1871-1967, AF4413931.5292-5389, AF2187982.4817-4938, AY2207391.4842-4940

### A.2 Clustering Evaluation

- RF00504 (Glycine riboswitch):
AP001516.1 246305-246211, AE016822.1 1714518-1714420, AP002996.2 13233-13334, AP004599.1 137133-137028, AE011751.1 9538-9457

- RF00005 (tRNA):
AL021918.1 94597-94678, M86495.1 1026-1092, S64977.1 800-870, X54408.1 1-73, X66438.1 1-72

- RF00031 (SECIS elements):
L12743.1 694-758, AF241527.2 359-424, X13710.1 946-1008, AB032826.1 1401-1464, AL049837.4 130674-130738

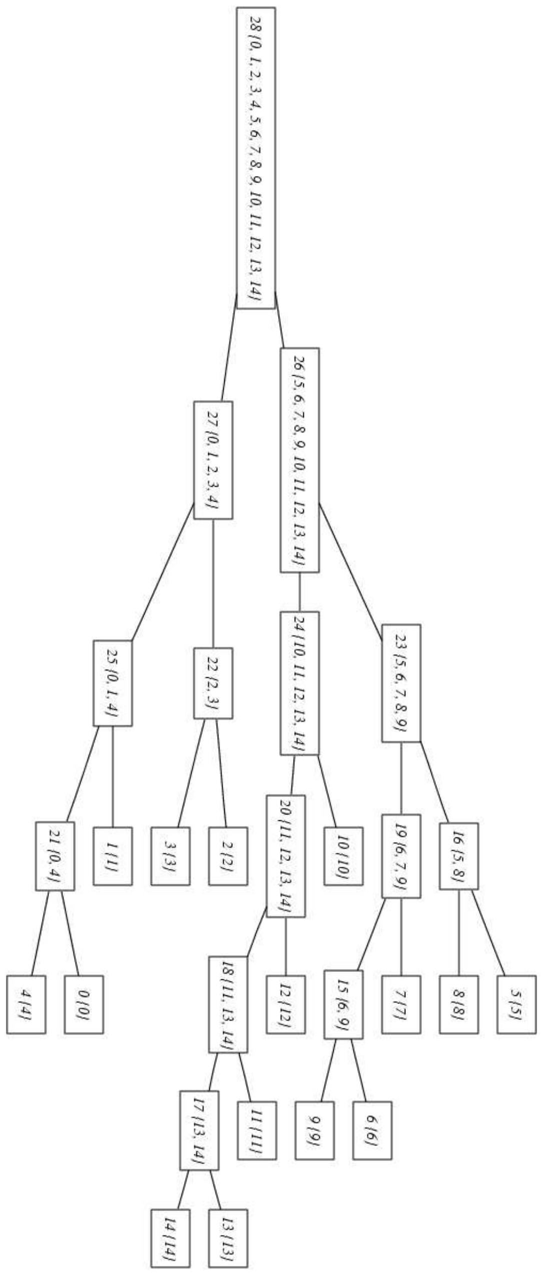


Figure 7: Result of average linkage clustering using the full distance matrix computed with the most sensitive method

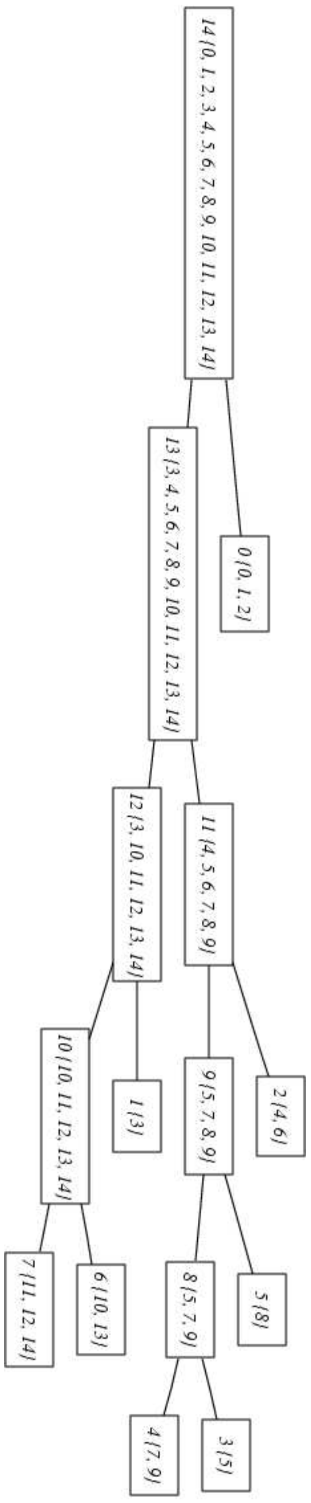


Figure 8: Result of average linkage clustering using the hierarchical method.

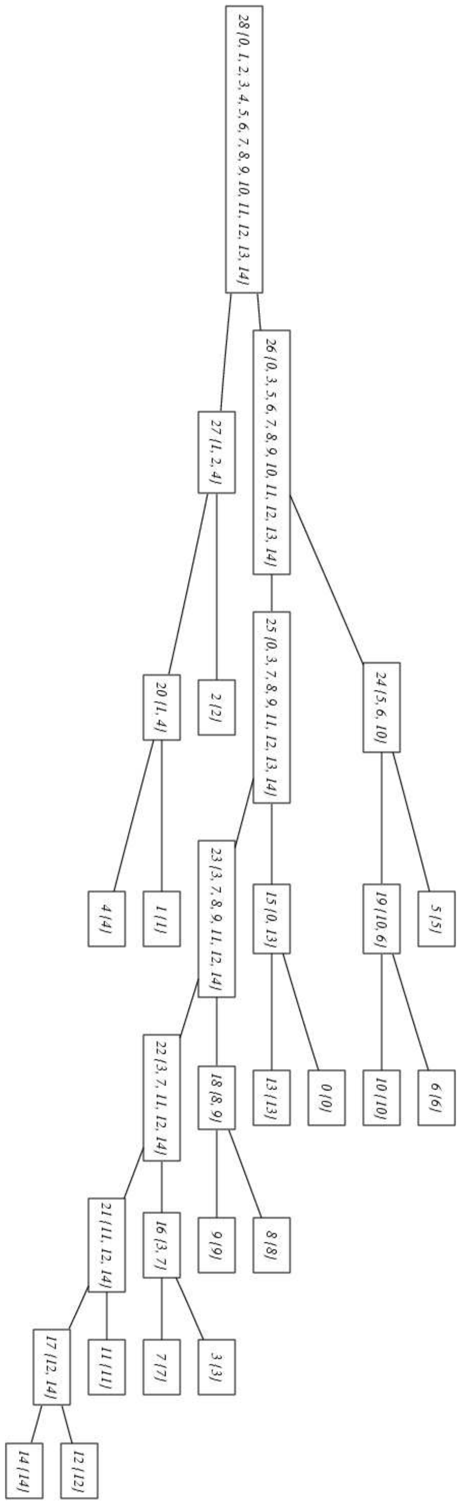


Figure 9: Result of sequence based clustering using TCOFFEE.