GraphClust: alignment-free structural clustering of local RNA secondary structures

Steffen Heyne†, Fabrizio Costa†, Dominic Rose and Rolf Backofen†
Bioinformatics Group, Department of Computer Science, University of Freiburg, Georges-Köhler-Allee 106, D-79110 Freiburg, Germany

ABSTRACT

Motivation: Clustering according to sequence-structure similarity has now become a generally accepted scheme for ncRNA annotation. Its application to complete genomic sequences as well as whole transcriptional states is therefore desirable but hindered by extremely high computational costs.

Results: We present a novel linear-time, alignment-free method for comparing and clustering RNAs according to sequence and structure. The approach scales to datasets of hundreds of thousands of sequences. The quality of the retrieved clusters has been benchmarked against known ncRNA datasets and is comparable to state-of-the-art sequence-structure methods although achieving speedups of several orders of magnitude. A selection of applications aiming at the detection of novel structural ncRNAs are presented. Exemplarily, we predicted local structural elements specific to lincRNAs likely functionally associating involved transcripts to vital processes of the human nervous system. In total, we predicted 349 local structural RNA elements.

Availability: The GraphClust pipeline is available on request. Contact: backofen@informatik.uni-freiburg.de

Supplementary information: Supplementary data are available at Bioinformatics online.

1 INTRODUCTION

In recent years, tiling arrays and high-throughput sequencing have strikingly driven the discovery of novel transcripts, of which an outstanding amount is not translated into proteins (ENCODe Project Consortium, 2009). These non-protein coding RNAs (ncRNAs) have moved to a key research in molecular biology, fundamentally challenging established paradigms and rapidly transforming our perception of genome complexity. There is mounting evidence that the eukaryotic genome is pervasively transcribed (Clark et al. 2007, 2008; Washietl et al. 2007, 2008; Brosnan and Woinoski, 2004).

Recent advances in the computational de-novo prediction of ncRNAs enabling genome-wide analysis have uncovered a wealth of signals for potentially novel ncRNAs in genomic sequences from basically all kingdoms of life. Examples include metazoa with thousands of predicted ncRNA signals in human, fish or insects (Rose et al. 2007, 2008; Washietl et al. 2008). Consequently, the prediction, comparison and (functional) annotation of ncRNAs are major tasks of current RNA research. Annotation of ncRNAs, however, is still not part of generic annotation pipelines of genome or next-generation sequencing data and precise functional annotations of the majority of predictions remain elusive.

One of the major reasons for the multitude of unannotated ncRNAs is that in contrast to protein-coding genes, ncRNAs belong to a diverse array of classes with vastly different structures, functions, and evolutionary patterns (Bonfünneweber Consortium 2007). Albeit their heterogeneity, ncRNAs can be divided into RNA families according to inherent functional, structural, or compositional similarities. Today, the Rfam 10.1 database already lists 1973 different RNA families (Faulcher et al. 2009). In contrast to RNA families, an RNA class groups together ncRNAs whose members have no discernible homology at the sequence level, but still share common structural and functional properties. Prominent examples are the two distinct classes of snoRNAs (box H/ACA and box C/D) and micro RNAs (miRNAs).

Since an RNA-class consists of RNA with similar structure and function, clustering according to sequence–structure similarity has now become a generally accepted scheme for ncRNAs annotation. The quality and complexity of the clusters are however, largely determined by the pairwise sequence comparison method. The most generic methods, as introduced with LocARNA (Will et al. 2013) and FoldAligner (Tommison et al. 2009), use derivatives of the full Sankoff algorithm (Sankoff, 1985) of simultaneous alignment and folding. They suffer, however, from a very high computational complexity (at least $O(n^3)$) in time and are thus limited to relative small sequence sets. As poignantly stated by Karolchik et al. (2004): ‘Even using substantially more sophisticated techniques, genome-scale ncRNA analyses often consume tens to hundreds of computer years. These high computational costs are one reason why ncRNA gene finding is still in its infancy.’ For this reason, many approaches use different heuristics to achieve a reasonable trade-off between time and quality. Oversimplifying and without completeness, given the variety of approaches present in literature, one can distinguish two main classes of clustering approaches. The first class uses simplifications in the representation of structures. Ritchie et al. (2007) and Khaladkar et al. (2009) use comparison tools that consider single-predicted structures such as RNAforester (Tochmann et al. 2004) or MARNA (Siebert and Backofen, 2005). These comparison approaches heavily depend on the correctness of the structure, although computational prediction of secondary structures are known to be error prone. Other approaches such as Rate et al. (2007) use simplified structural models. To some extend, the work by Parker et al. (2011) can also be listed under this class as they use an approximate measure between two structural RNAs: SCFG models for clustering hits found by EvoFold (Pedersen et al. 2010).

© The Author(s) 2012. Published by Oxford University Press.
This is an Open Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License (http://creativecommons.org/licenses/by-nc/3.0), which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.
The second class uses sequence information as prior knowledge to speed up the computation. In the simplest case, sequences are first clustered by sequence-alignment (Rinn et al. 2007; Shi et al. 2009). These alignments are then used to predict conserved consensus structures using approaches such as RNAfold (Bernhart et al. 2008) or Petros (Keckowski et al. 2008). A similar overall scheme is e.g. applied by Weinberg et al. (2011), who use CMFinder (Yao et al. 2006) to determine a consensus motif from a cluster of unaligned sequences. Yet, another set of tools uses sequential encoding of structural information (Kner et al. 2009). Finally, one can use the information from an ensemble of sequence-alignments to speed up the computation as done by Saito et al. (2011), an approach that is commonly seen in sequence-structure alignment. The major problem, however, is that the sequence of an ncRNA evolves much faster than its structure, which implies that in most of the cases, no homology on the sequence level is detectable. Indeed, it has been shown that family assignments of structured RNAs obtained from sequence alignments at pairwise sequence identities below 60% are often wrong (Karder et al. 2009).

In this article, we propose an efficient approach for clustering very large sets of RNA sequences according to sequence and structure information. The size of current datasets exclude the use of alignment-based techniques. Hence, to achieve the required efficiency, we propose here an alignment-free hashing technique over a novel encoding for RNA structures. Although there are alignment-free methods for comparison of RNA with respect to sequence and structure (Tan et al. 2008; Liu and Wang 2006), we are not aware of any alignment-free method capable to perform RNA sequence-structure comparisons on hundred of thousands of sequences. In a second phase, we increase the quality of the resulting clusters by employing alignment methods to filter away inconsistent elements. More specifically, we extend a fast graph kernel technique that we have recently developed (Costa and Grave 2010) for chemoinformatics applications and we adapt it to detect similarities between RNA secondary structures. The key novelty that we introduce lies in an explicit representation of the associated sequence-structure information which we encode as sparse vectors. This allows us to use efficient locality sensitive hashing methods to accurately retrieve dense data regions with a complexity that is linear in the number of sequences \(N\), rather than quadratic as it would be with both alignment methods or standard kernels that work with implicit representations.

We have integrated the approach in a ready-to-use pipeline for large-scale clustering of putative ncRNA. The method has been evaluated on known ncRNA classes and compared against existing approaches, that is, the complete LocARNA-pipeline and RNAclust (Keckowski et al. 2008). We show that our clustering is of high quality yielding reliable clusters. Due to the algorithmic improvements presented in this contribution, we achieve a striking performance speedup (from years to days for serial computation and even hours when parallelized) outperforming any of the existing approaches. We applied our method to six heterogeneous large-scale datasets containing >220,000 sequence fragments. First, we analyzed computationally derived predictions of short ncRNAs lacking reliable class assignments. Next, we searched for local structural elements specific to experimentally validated lincRNAs. We observed enriched GO-terms for lincRNAs containing predicted local motifs likely functionally ‘linc’-ing these transcripts to vital processes of the human nervous system. In general, both application scenarios aim at the detection of novel structural non-coding RNAs. In total, we predicted 349 local structural RNA classes.

2 APPROACH

We propose to retrieve candidate clusters using an efficient hashing technique over a novel RNA structure representation. To enhance the quality of the results, we filter away inconsistent elements using alignment-based techniques. To increase the recall, we induce covariance models for each cluster and scan the dataset to collect missed sequences. More in details, we propose to sample a small number of probable, but sufficiently different, structures for each RNA sequence. We then encode each structure as a labeled graph preserving all information about the nucleotide type and the bond type (i.e. binding or backbone). In this way, a sequence’s structure is represented as a graph with several disconnected components. We could now compute the similarity between the representative graphs using a graph kernel. However, to avoid a quadratic number of comparisons, we first extract an explicit vector representation for each graph, and then build an inverse index on a compressed representation obtained via hashing techniques. This allows us to retrieve in constant time the nearest neighbors sequences for any given query structure. We evaluate each neighborhood and select as candidate clusters those that contain very similar elements. Each (small) candidate cluster is then refined using alignment techniques. Finally, a covariance model is induced for each cluster. The dataset is scanned using these models and before reiterating the procedure, the recognized missing sequences are associated to the corresponding cluster and removed from the set.

2.1 Vector-based representation of RNA structures using explicit graph kernel

Graph encoding for structures. Minimum free energy-based secondary structure prediction has been shown to be error prone. For this reason, we want to use a representation of the entire ensemble of low-energy conformations. Resorting to a complete enumeration of near-optimal structures would yield a tremendously large number of structures. Instead, we opt for abstract shape analysis methods introduced in Fizgirich et al. (2008). Here, one analyzes the complete folding space using the notion of partition function (McCaskill 1990) but classifies the structures a priori into abstract classes called shapes.

We note, however, that there is an issue regarding the true sequence boundaries: while benchmark test for sequence-structure alignments are usually given as full-length transcripts, this setting is quite unlikely in practical application scenarios, where one often has either a partial transcript (e.g. from RNA-seq data), or just transcripts with wrong boundaries. The fact that secondary structure prediction may change when additional context is considered adds considerable noise to the task of functional families identification. To deal with this issue, we consider several subsequences obtained from the original sequence, as overlapping windows of different sizes (parameterized by a set of values \(W\)) and at different starting locations (parameterized by the overlap value \(O\)). Note that the window size influences the locality of the structural features that the method is aware of. In the experiments, we chose two window sizes, one of 30 nt and the other of 150 nt to capture both local hairpins
and larger multiloop structures. Finally, for each subsequent in each window, we consider the set of most representative structures obtained by RNASHAPE and encode them as disconnected components. The vertex set for each of the graph components is derived from the sequence nucleotides, whereas the edge set encodes both the nucleotide sequence adjacency information and the pairwise binding status. In addition, an extra set of vertices (and corresponding edges) is introduced to better match biological knowledge on important RNA motifs, namely, for each stacking base pairs quadruplet, an additional vertex is added (and linked to each of the four nucleotides involved; in this way, the notion of neighborhood subgraph (see next section) coincides with that of a sequence of stacking base pairs (Fig 1).

Graph kernel. To deal with entities represented as graphs, graph kernels of several types have been proposed. Graph kernels compute a similarity measure between graphs in terms of a dot product function. We start from the recently introduced \cite{costa2010fast} fast kernel called neighborhood subgraph pairwise distance kernel (NESPDK), as this kernel is suitable for large datasets of sparse graphs with discrete vertex and edge labels. However, here we choose to materialize and explicitly list all the features. This will turn out to be key in successive phases, where, for efficiency reasons, we need to build an index over these features.

The NESPDK is an instance of a decomposition kernel \cite{haussler1999sphere}, i.e. a composite kernel that operates over all possible "parts" defined by a given relation. In this case, the parts are pairs of special subgraphs, called 'neighborhood' subgraphs. More formally, for a given graph $G = (V, E)$, and an integer $r \geq 0$, let $N_r(G)$ denote the neighborhood subgraph, i.e. the subgraph of $G$ rooted in $v$ induced by the set of vertices at distance $d$ (The distance between two vertices $v, u$ is the number of edges of the shortest path between $v$ and $u$, not $\geq r$. The neighborhood pair relation $R_{r,d}$, is defined to hold when the distance between the roots of two neighborhood subgraphs of radius $r$ is exactly equal to $d$. The authors define a kernel $\kappa_{r,d}$ as the decomposition kernel on the relation $R_{r,d}$, that is,

$$\kappa_{r,d}(G, G') = \sum_{A,B \in R_{r,d}(G)} \delta(A) \delta(B)$$

where $R_{r,d}$ is the inverse of the relation $R_{r,d}$ and indicates all the possible pairs of neighborhood subgraphs of radius $r$, whose root vertices are at distance $d$ in the given graph $G$. $I$ denotes the indicator function, and $\delta$ is the isomorphism between rooted graphs.

For efficiency reasons, the authors consider the zero extension of $K$ obtained by imposing an upper bound on the radius and the distance parameter: $K_{r,d}(G, G') = \sum_{d \leq r} \|K_{d}(G)\|$, i.e. they limit the sum of the $\kappa_{r,d}$ kernels for all increasing values of the radius (distance) parameter up to a maximum given value $r^*$ (and corresponding edges) is introduced to better match biological knowledge on important RNA motifs, namely, for each stacking base pairs quadruplet, an additional vertex is added (and linked to each of the four nucleotides involved; in this way, the notion of neighborhood subgraph (see next section) coincides with that of a sequence of stacking base pairs (Fig 1).

The novel idea here is to materialize the implicit feature encoding which is key to obtain sublinear efficiency in the clustering phase. Differently from \cite{costa2010fast}, we here make use of the integer code for the invarian graph encoding as a feature indicator. In this way, we can interpret the integer associated to each feature (i.e. each pair or neighborhood subgraphs of radius $r$ whose roots are at distance $d$) as the feature key and the (normalized) count of occurrences as its value. This allows us to obtain an explicit feature encoding for a given graph $G$ as a sparse vector in $\mathbb{R}^m$ (with a very high dimension $m$). The feasibility of the approach lies in the fact that the encoding does not produce an exponential number of features, as it would happen with most graph kernels that enumerate all possible general subgraphs. Instead, NESPDK limits the number of features to $O(r^*d^2\|V(G)\|^2)$ pairs of neighborhood subgraphs, i.e. one feature for each pair of vertices times each possible combination of values for the radius.
and the distance. Note that typically \( r^* \in [0, 5] \) and \( d^* \in [0, 10] \) and hence the multiplicative factor \( (\approx 50) \) is reasonable. Moreover, for sparse graphs, the number of vertices that are reachable within fixed small distance is typically small (depending on the average degree) so that the dependency on the vertex set size can be more tightly approximated by \( O(N \log G) \). As a result each graph is mapped into a sparse vector that lives in a very high-dimensional feature space but that has a number of non-zero features which is linear in the number of the graph's vertices. As the computation of the encoding for each neighbor subgraph can be cached, the practical complexity for the overall encoding of a graph is linear in its vertex set size with small hidden multiplicative coefficient, making it one of the fastest graph kernels available (Costa and Grave, 2010).

2.2 Efficient candidate cluster determination using locality sensitive hashing

As datasets size increases (i.e. \( \times 10^3 \)), algorithms that directly make use of pairwise distance or similarity information become infeasible as they inevitably exhibit a quadratic complexity. The key idea then is to formulate the clustering problem in terms of approximate nearest neighbors queries which can be answered efficiently (sublinearly). That is, given a set of \( n \) instances \( P = \{p_1, \ldots, p_n\} \) in a metric space \( d \) with a distance function \( d \), a neighborhood query is a procedure that returns the instance in \( P \) closest to a query instance \( q \in X \). The nearest neighbor search problem is formulated as a dataset preprocessing that allows nearest neighbors queries to be answered efficiently. For high-dimensional cases, an efficient solution was proposed by Indyk and Motwani (1998). The key idea is to rely on the requirements, ask for \( \epsilon \)-approximate nearest neighbors queries, and use locality-sensitive hashing techniques. The \( \epsilon \)-approximate nearest neighbor query returns an instance \( p \) for a given query \( q \) such that \( \forall p \in P, d(p, q) \leq (1 + \epsilon) d(p', q) \). A locality sensitive hash function is a hash function such that the probability of collision is higher for objects that are close to each other than for those that are far apart. As locality sensitive hash function, we choose the min-hash function (Broder, 1997) as it approximates the natural similarity notion defined by the Jaccard index. However, these techniques require instances to be represented as sparse binary vectors rather than sparse real vectors. Therefore we binarize all instances from \( \mathbb{R}^m \rightarrow [0, 1]^m \) setting to 1 all non-null components.

Let \( x, z \in \{0, 1\}^m \) be two instances; the Jaccard similarity between two instances is defined as \( s(x, z) = |(1 \cap z)/(1 \cup z)| \), that is, the ratio of the number of features that the instances have in common over the overall number of features. We build a min-hash function starting from a set of random hash functions \( f_j : \mathbb{N} \rightarrow \mathbb{N} \). The min-hash function derived from \( f_j \) is defined as \( h_j(x) = \min_{i \in \mathbb{N}} f_j(x)_i \). The min-hash returns the first feature indicator under a random permutation of the features order. A rather surprising (and useful) fact is that a min-hash collision is an unbiased estimator of the Jaccard similarity:

\[
P(h(x) = h(z)) = \frac{|\{i \in I \mid x_i = z_i\}|}{|I|} = s(x, z),
\]

i.e. the probability to select as the minimum feature indicator a non-null feature that belongs to both \( x \) and \( z \) is exactly the fraction of features that \( x \) and \( z \) have in common over the total number of non-null features of \( x \) and \( z \). To decrease the (high) variance of this estimate, one can take \( N \) independent min-hash functions and compute the number \( n \) of times that \( h_i(x) = h_i(z) \). The estimated value \( n/N \) is the average of \( N \) different 0-1 random variables, which evaluates to one when \( h_i(x) = h_i(z) \) and zero in all other cases. The average of these unbiased estimators of \( s(x, z) \) is also an unbiased estimator, with an expected error bounded by \( O(1/\sqrt{N}) \), or, equivalently, for any constant \( \gamma > 0 \), we can compute a constant \( N = O(1/\gamma^2) \) such that the expected error of the estimate is at most \( \gamma \). For example, with 400 hash functions, the estimate of \( s(x, z) \) would have an expected error \( < 0.05 \).

We collect the results of the entire set of min-hash functions in an instance sketch as the tuple \( \langle h_1(x), \ldots, h_N(x) \rangle \). To obtain an efficient neighbor search procedure, we build an inverse index that returns all instances with the same min-hash value in \( O(1) \). More precisely, given the \( i \)-th hash function and a value \( h = h_i(x) \), we collect the set of instances \( Z_i(h) = \{ z \in P : h_i(z) = h \} \). The approximate neighborhood \( Z \) of an instance \( x \) is then induced from the multiset \( Z = \bigcup_{i=1}^N Z_i(h) \). Note that when \( \gamma \) (or equivalently \( N \)) is fixed, the complexity to build a single signature is constant and therefore the complexity for building the index is linear in the size of the dataset. To improve the quality of the returned neighbors, we consider only the most frequent elements in \( Z \) and sort them according to their similarity to \( x \). The \( k \)-neighborhood \( N_k(x) \) is finally the set of the \( k \)-closest elements. If the size of \( Z \) is small and independent of the dataset size \( |P| \), these steps can be performed in constant time.

Given the efficient \( \epsilon \)-approximate nearest neighbor search procedure offered by the min-hash technique, we can define candidate clusters using the notion of data density. Intuitively, we prefer as a candidate cluster a set of closely related instances. The idea is therefore to rank each instance \( x \) according to the density \( D_h \) of its \( k \)-neighborhood \( N_h(x) \), defined as the average pairwise similarity between \( x \) and all elements in its \( k \)-neighborhood \( D_h(x) = 1/k \sum_{z \in N_h(x)} K_r(x, z) \). The candidate clusters are finally obtained from the densest neighborhoods.

2.3 GraphClust pipeline

Details of our clustering pipeline are given in the following (Fig. 2). We distinguish nine phases: initially, near-duplicates are filtered away (1), suboptimal structures for each sequence are computed in parallel (2); the sparse encoding is extracted for each structure (3); the encoding is then used to build the feature index for fast similarity searches; the top dense sets are returned as candidate clusters (4) and subsequently refined using structural alignment procedures (5); the remaining instances are used as high quality seeds to build coverage models (6) with which additional instances are retrieved to further populate the clusters (7); before reiterating starting from Step 4, the clustered elements are eliminated from the working set (8). Finally, redundant clusters are merged and all instances receive a unique cluster assignment (9). To illustrate, we provide below the clustering pipeline.

Phase 1: Preprocessing (sequential). The GraphClust pipeline is able to cluster RNA sequences which originate from different sources such as RNA-seq or computational methods like RNA22 (Washietl et al. 2008) or EvoFold (Pedersen et al. 2009).

Therefore, a solid preprocessing of the input sequences is essential. We mask repeats to avoid clusters made of genomic repeats. Next, we split long sequences into smaller fragments to enable the
Fig. 2. Complete clustering pipeline diagram. Phases that are executed in parallel are represented in stacked boxes. (1) filter near-duplicates, (2) compute suboptimal structures, (3) compute sparse vector encoding, (4) compute global feature index and return top dense sets, (5) refine clusters with structural alignment procedure, (6) build covariance model with remaining high quality instances, (7) populate each cluster with retrieved instances, (8) remove clustered instances and iterate from Step 4 and (9) merge redundant clusters.

Note that the neighborhood queries, the density estimates, and the returned neighborhood are computed on the complete dataset, not on the sample.

Phase 5: Cluster refinement (parallel). Candidate clusters reaching-phase 5 contain sequences (≥15) that are deemed similar under the NSFPDR similarity measure. In order to incorporate domain-specific knowledge (consider for example compensatory mutations) and create a high quality model of the cluster, we perform a sequence-structure alignment of the candidate set with the tool LOCARNA [Will et al., 2009]. A cluster tree is created by applying an average-linkage algorithm (UPGMA) to the pairwise distance matrix induced by the LOCARNA alignment score. All subtrees with at least three leaves are then ranked by their quality, which is defined as the average pairwise alignment scores of its leaves. Note that the quality is not necessarily inversely proportional to the subtree size, that is, large subtrees can be ranked higher than smaller ones Sadreyev and Grishin, 2003]. Only the top ranked subtree, which contains the sequences (≥3–7) that exhibit the best quality, is retained.

Phase 6: Cluster model (parallel). The selected top ranked RNA candidates are realigned with the tool Infernal [Nawrocki et al., 2009], which allows the computation of an alignment reliability score. The reliability signal can be used to identify a trusted alignment region and estimate the borders of the common local motif. A covariance model (CM) is finally created by applying the Infernal [Nawrocki et al., 2009] tool on the identified subsequences.

Phase 7: Model scanning (parallel). Each cluster induces a CM model which is used to search the full dataset for residual potential cluster members. The sequence hits that are considered significant on the basis of their bit-score (≥20) or E-value are added to the final cluster. Note that, as every time we perform the search on the entire dataset, a sequence can be assigned to multiple clusters. This ambiguity is allowed in this phase as, until all clusters are available, there is not enough information to decide unambiguously for the best cluster assignment.

Phase 8: Iteration and removal (sequential). All cluster members found in the previous phase are removed from the dataset and a new iteration starts from Phase 4. The termination condition is given either by a predetermined max number of iterations, a time limit or when the remaining dataset is exhausted. Note that Phases 5 to
7 allow us to go beyond the graph kernel similarity. The instances identified in Phase 4 come from the neighborhood of elements in high density regions. These clusters have a spherical shape in the kernel feature space. By first filtering, the cluster via alignment procedures and then expanding it using the covariance model, we remove this bias and obtain non-spherical clusters. This is one of the main justifications for the iterative nature of the pipeline as the iterative removal of the non-spherical clusters alters the density distribution in the kernel feature space and allows novel clusters to emerge.

Phase 9: Post-processing (sequential). Redundant clusters are merged and instances that belong to multiple clusters are assigned unambiguously. For every pair of clusters, we compute the relative overlap (i.e., the fraction of instances that occur in both clusters) and merge them if the overlap exceeds 50%. Cluster members are finally ranked by their CM bitscore.

### 3 METHODS

#### 3.1 Datasets

**Benchmarking.** We have tested and measured the performance of our method by clustering known ncRNA classes obtained from two different sources. (i) We clustered a set of 503 families obtained from the Rfam database [Finn et al. (2010)](https://rfam.xfam.org), which has been previously been successfully used to benchmark LocARNA-based structural clustering [Will et al. (2005)](https://doi.org/10.1073/pnas.1413186101). The original set consists of all Rfam seed sequences (v7.1), filtered for 80% sequence similarity and lengths <400 nt. Application of BlastClust to remove trivial sequence clusters leaves 3900 sequences. The majority of families (252/503), however, has <3 members. Only 124 families, comprising ~80% (3118) of all sequences, have >5 members. (ii) We collected a comprehensive set of 49 bacterial small ncRNA families (941 sequences) from the NCBI Genome Database. Non-coding RNAs present in at least 10 species were considered. Removing sequences exceeding 400 nt in length and similar sequences using BlastClust leaves 363 bacterial ncRNAs. These were randomly embedded in 50 nt genomic context sequences to burden the classification problem. The set contains 6 families with <5 members and 37 families with >5 members. Sequences of both benchmark sets were not split. In addition, to recover the structural classes presented in Parker et al. (2011), we clustered 725 EvoFold hits that form 220 EvoFast families in Parker et al. (2011).

**Application.** We applied different sets of predicted ncRNAs: we clustered 37 381 human EvoFold hits, 16 377 RNAz ncRNA candidates of the fruit fly Drosophila melanogaster [Rose et al. (2008)](https://www.ncbi.nlm.nih.gov/pubmed/18810921), and 11 536 ncRNA candidates of the teleost Takifugu rubripes [Rose et al. (2008)]. EvoFold predictions are generally short and were not split before clustering. For both RNAseq reads, sequences were fragmented into stretches of 150 nt (minimum length 50 nt). Removing nearly similar sequences using BlastClust left 17 765 fragments for fruit fly and 11 287 fragments for teleosts. Moreover, we searched for novel local structural motifs in long ncRNAs. (i) We clustered the collection of 8195 human long ncRNAs described in Cech et al. (2011). Splitting (150 nt windows, minimum length 50 nt) and BlastClust filtering resulted in 534 418 fragments. (ii) We clustered the set of 1313 zebrafish long ncRNAs expressed in zebrafish embryos recently reported by Brain et al. (2013). Preprocessing yielded 5377 fragments, ready for clustering.

### 3.2 Evaluation

**GraphClust** works iteratively and hence, we can measure the clustering quality at the end of each round. We use the F measure and the adjusted Rand index [Hubert and Arabie (1985)](https://doi.org/10.1007/BF02293812) as quality measures. The F measure is the harmonic mean of precision and recall and measures the quality of a single cluster, defined as $F = 2 \cdot \frac{TP}{TP + FP + FN}$. The adjusted Rand index is defined as Prec = $TP/(TP + FP)$, where TP is the number of correct cluster members and FP is the number of wrong cluster members. Recall is defined as the fraction of correctly clustered members over the full family size. To each cluster, we assign the family with the majority of members. We report the average F measure over all clusters at the end of each round. The Rand index compares two clustering hypotheses taking into account all possible pairs of instances. We measure the similarity measure at the end of each iteration on the fraction of times when the two clustering hypothesis agree that the elements in each pair belong to the same or to different clusters. We use the adjusted Rand index, which uses a hypergeometric model to control for chance effects, so that the value range is $[0,1]$, with random partitioning scoring 0 and perfect agreement scoring 1. Note that we measure only the quality of clustered RNA sequences, that is, we skip elements which are not part of any cluster, which is reasonable when dealing with large datasets.

We calculate the quality measures on the basis of different clustering hypotheses, called partition types. The initial clustering resulting from all candidate clusters is called SOFT clustering where a specific RNA candidate could belong to >1 cluster. From this SOFT clustering, we generate two partitions BEST and MERGED. In addition, we consider for evaluation purposes a theoretical ORACLE partition. Partition BEST assigns an RNA candidate to the model with the best score (without any merging). MERGED uses the described merging strategy (see Section 4.4). Phase 9 of overlapping clusters and applies BEST afterward. This clustering hypothesis is used as results for all application scenarios. The ORACLE partition assumes a supervised or perfect merging strategy and shows the maximum theoretical performance. In this case, all initial clusters (SOFT clustering) with the same majority true class were merged, using BEST as final partitioning strategy. The overall running time is an important measure of our pipeline. Here, we measure the time of each phase and provide a total time after each iteration and an average time per predicted cluster.

Furthermore, details on applied parameters are given in the Supplementary Material.

**Comparison to other methods.** Alignment-based RNA clustering methods which take into account structural properties need to calculate a pairwise distance matrix first. This information can be used to get a clustering hypothesis using different methods, for example, by creating a guide tree. We compare our clustering to a LocARNA-based clustering. The idea of the comparison is to show that our clustering approach achieves similar and high clustering qualities but with the discussed benefits especially in complexity and therefore runtime. We use RNAmotif to partition the LocARNA cluster tree into an optimal number of clusters and evaluate them with the given quality measures. A cluster is reported as optimal cluster if the sum-of-squared error for two clusters is not significantly smaller than expected by chance. The significance level of RNAmotif can be controlled by $k$ and authors give a range $0.8 < k < 1.2$ for Rfam sequences. The error of a cluster is determined via the free energy of its consensus structure and the minimum free energies of its individual sequences. Clearly, this procedure gives a full clustering whereas our pipeline only clusters a subset. Therefore, we eliminate all clusters with <3 members from the RNAmotif partition. We also measure the LocARNA runtime as aggregated serial time. We use LOCARNA without any speedup heuristics to stress the inherent complexity issue of existing structure-based clustering methods. Using speedup heuristics would give a much lower overall LOCARNA runtime, but not in the order of several magnitudes.
4 RESULTS

4.1 Evaluation of the GraphClust pipeline

Rfam benchmark: We run our pipeline for 15 iterations, retrieving 10 candidate clusters at each iteration. Table 1 gives an overview of the result for the clustering of 3901 Rfam sequences. See Supplementary Information Tables S1 and S2 in the Supplementary Material for more details. After 15 iterations, we identified 130 clusters (MERGED partition). The high F measure (0.834) and Rand index (0.984) indicate a correct clustering. The result reflects the fact that only 124 of 502 families have ≥5 members. Prior to the merging phase, we identified 148 clusters (SOFT clustering) with a quality of F = 0.796 (R = 0.483). This clearly indicates that the overall cluster quality can be significantly improved employing a merging strategy. Increasing the number of iterations does not produce additional meaningful clusters, resulting in a slightly decreased overall quality. We report the aggregated runtime for all serial and parallel phases. Running the entire pipeline took ~370 days (129 626 s) when viewed as serial process. The parallelized version, however, took ~3h. Note that the clustering step in Phase 4 took only between 1 and 8 min (serial time), which was the bottleneck in previous approaches. See also Figure S1 for more details. To compare our results to state-of-the-art benchmarking results, we applied RNASoup to the cluster tree obtained from LocARNA alignment scores. We chose the partition with F = 0.8 which gave a quality of F = 0.588 (R = 0.586) for 160 predicted clusters containing 3569 sequences. We considered only cluster with at least three members for a fair and effective heuristics also for a similar quality (see Supplementary Table S5).

We run our pipeline for 15 iterations, retrieving 10 candidate clusters at each iteration. Table 1 gives an overview of the results for the clustering of 363 small ncRNAs (see also Supplementary Tables S3 and S4). After 15 iterations, we identified 43 clusters (MERGED partition) from 38 unique families. The overall clustering quality is high with F = 0.858 and R = 0.866. The additional ranking functions do not disturb the quality. LocARNA applied to this set results in a quality of F = 0.729 and R = 0.88 using RNASoup with k = 0.4 (other k have lower qualities). The serial runtime of the pipeline is ~6.8h. LocARNA applied to the same dataset takes ~7 days.

EvoFam families: Application of GraphClust (five iterations) to the 725 EvoFam-annotated sequences yielded 37 structural classes. We recovered 14 known families with F ≥ 0.5. In particular, even raising the threshold (F ≥ 0.7), we identified 5 out of 8 families with ≥10 members. Applying GraphClust on all 37 381 human EvoFam hits (20 iterations, see Table A for runtimes and Figure S3 for exemplary clusters) recovered the same amount (5/8) of EvoFam families (albeit different in type) having ≥10 members. Infernal annotates ~38% (14/37) of the GraphClust-derived EvoFam clusters as known ncRNA classes. For example, we identified the Histone 3’ UTR stem-loop motif and the let-7 miRNA family. The vast majority of clusters (11/14) were known miRNAs. For 10 of the 11 miRNA cluster, the E-value of the best Infernal hit was <10^-13 indicating a reliable class annotation. Interestingly, these GraphClust results can also be used to identify novel human miRNA candidates, see Supplementary Figure S4.

4.2 GraphClust predicts novel local structural motifs

Most of the known families, including tRNAs, snRNAs (U2, U5), and miRNAs, were recovered in the fruit fly RNAz screen (six annotable clusters). Throughout all de-novo discovery screens, miRNAs were most abundantly detected (four clusters in fugu- and two clusters in the fruit fly RNAz screen; two clusters in

Table 1. Overview

<table>
<thead>
<tr>
<th>Species</th>
<th>Type</th>
<th>Method</th>
<th>Input</th>
<th>Size (Mb)</th>
<th>Timea</th>
<th>Cluster</th>
<th>MIPS, SCI, SCI_a.5</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Benchmark</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bacteria</td>
<td>Small ncRNAs</td>
<td>Misc</td>
<td>363</td>
<td>0.06</td>
<td>6.8h</td>
<td>39</td>
<td>0.75</td>
<td>29</td>
</tr>
<tr>
<td>Human</td>
<td>Predicted RNA elements</td>
<td>EvoFam</td>
<td>699</td>
<td>0.03</td>
<td>0.3h</td>
<td>37</td>
<td>0.52</td>
<td>36</td>
</tr>
<tr>
<td>Misc</td>
<td>Small ncRNAs</td>
<td>Rfam</td>
<td>3900</td>
<td>0.51</td>
<td>36h</td>
<td>130</td>
<td>0.64</td>
<td>98</td>
</tr>
<tr>
<td><strong>De-novo discovery</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fugu</td>
<td>LincRNAs</td>
<td>RNA-seq</td>
<td>5877</td>
<td>0.09</td>
<td>10.3h</td>
<td>99</td>
<td>0.39</td>
<td>16</td>
</tr>
<tr>
<td>Fugu</td>
<td>Predicted RNA elements</td>
<td>RNAz</td>
<td>11 287</td>
<td>1.36</td>
<td>13.3h</td>
<td>97</td>
<td>0.39</td>
<td>22</td>
</tr>
<tr>
<td>Fruit fly</td>
<td>Predicted RNA elements</td>
<td>RNAz</td>
<td>17 765</td>
<td>2.15</td>
<td>20.4h</td>
<td>95</td>
<td>0.34</td>
<td>23</td>
</tr>
<tr>
<td>Human</td>
<td>LincRNAs</td>
<td>RNA-seq</td>
<td>31 418</td>
<td>5.40</td>
<td>3.6d</td>
<td>95</td>
<td>0.34</td>
<td>3</td>
</tr>
<tr>
<td>Human</td>
<td>Predicted RNA elements</td>
<td>EvoFam</td>
<td>37 258</td>
<td>1.37</td>
<td>5.7d</td>
<td>117</td>
<td>0.75</td>
<td>109</td>
</tr>
<tr>
<td>Human</td>
<td>3’UTRs</td>
<td>RefSeq</td>
<td>118 514</td>
<td>21.91</td>
<td>12.8d</td>
<td>106</td>
<td>0.34</td>
<td>13</td>
</tr>
<tr>
<td><strong>Σ</strong></td>
<td></td>
<td></td>
<td>227 081</td>
<td>32.88</td>
<td>25.7d</td>
<td>815</td>
<td>–</td>
<td>349</td>
</tr>
</tbody>
</table>

aPlease see text for different parameters influencing the run-times.


The table summarizes the datasets used in this study and gives an overview on GraphClust-detected clusters. For each screen, we provide the number of input instances and the sum of their lengths (size). We denote the required serial running time to process the input and list the number of obtained clusters. Next, we report the mean pairwise identity (MIPS) and the number of clusters that have a structure conservation index (SCI) above 0.5. These are prime candidates for structured ncRNA classes.
Table 1. Our obtained motifs are functionally linked to the nervous system, neuronal diseases and brain function (Chodroff et al. 2011; Dareshi et al. 2010).

Furthermore, we extracted 99 local motifs from the 1133 teleost lincRNAs recently reported by Pauli et al. (2011). Interestingly, these contain up to five times more novel structural classes than their human counterpart. This might be explained by the fact that teleost fish underwent an additional whole-genome duplication (Christoffels et al. 2010) which increases the likelihood to identify paralogous genes.

5 DISCUSSION

We introduced for the first time an ultra-fast pipeline for large-scale comparison and clustering of RNAs according to sequence and structure, which is key to the functional annotation of ncRNAs. Strikingly, its core steps are alignment-free. As clearly indicated by the result, the approach is linear in time and thus scales to sets of hundreds of thousands of sequences. The largest dataset we considered in this study consists of ~118 thousand sequences, and they can be clustered by the proposed pipeline on a single computer in ~12.8 days. Furthermore, we have parallelized 5/9 phases of the pipeline: this allows to reduce the run-time for clustering the 3901 Rfam seed sequences from 36 to ~3 h. When compared with the time required by an efficient pairwise sequence-structure alignment, namely LocARNa, we observe a ~250-fold speedup. It indeed took us 370 days to perform the clustering based on this state-of-the-art complete all-against-all sequence/structure comparison.

We have integrated the alignment-free clustering approach in a pipeline that uses LocARNa and Infernal to improve the initial clusters found by a neighborhood search. The latter, now allows us for the first time to compile RNA classes of ncRNA and determine associated consenus structures for large-scale datasets without resorting to alignment-based clustering. This is important as it is known that sequence alignments often fail at pairwise sequence identities below ~60%. In addition, our pipeline exhibits an anytime characteristics, as we do not need to produce a complete hierarchical cluster tree, which is a computational bottleneck for large datasets. In contrast, we output as many best clusters as wanted by the user. The overall complexity of our pipeline is to a large extent determined by the number of reported clusters.

We have evaluated the approach on several benchmark sets consisting of Rfam seed alignments, EvoFast families and known bacterial ncRNA. Here, we achieve a high overall clustering quality, even if the known RNA signal is embedded in flanking context. To further elucidate the capacity of our approach, we have also clustered datasets where no clustering approach has been applied so far. By processing the complete dataset to generate its density landscape, our method in particular enables us to likely detect previously missed structural classes.

The screens of this pilot study only consisted of sequences from skeletal muscle, kidney and brain. Next, half of our clusters (49/95) contain transcripts for which the studies of Cabili et al. (2011) have already shown that most of their lincRNAs are expressed in testes. Nevertheless, we also obtained structural motifs from transcripts that are consistently and specifically expressed in either skeletal muscle, kidney and brain. Next, half of our clusters (47/95) contain transcripts for which the studies of Cabili et al. (2011) resulted in enriched GO-FAT (Gene Ontology subset with more specific terms in contrast to GO-SLIM) biological process terms (Huang et al. 2008) and hence are putatively functionally linked to their nearest protein-coding gene. We found 17 clusters that contain at least two different lincRNAs with enriched GO terms. Of these, ~53% (9/17) of structural lincRNA motifs are associated with exactly the same GO term. The actual number of structural motifs with a specific biological function is likely higher, as different GO terms can still convincingly refer to similar biological processes. For example, we obtained clusters whose elements on a broader scale by clustering all RefSeq 3'UTRs.

Table 2. Results for Rfam and small ncRNA benchmark set

Results for each iteration $i$ on the MERGED partition. Clustering quality is given as $F$ measure and Rand index. The total number of clustered sequences is indicated with $#Seq$. The total number of clusters after merging is given by $#C$. $Time_i$ denotes the total time for iteration $i$, $Time_{ALL}$ is the total serial time up to iteration $i$.

<table>
<thead>
<tr>
<th>#Seq</th>
<th>$F$</th>
<th>Rand</th>
<th>Phase 4</th>
<th>Time $i$ (s)</th>
<th>Time$_{ALL}$ (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>271</td>
<td>0.882</td>
<td>0.888</td>
<td>458</td>
<td>14 995</td>
<td>23 309</td>
</tr>
<tr>
<td>629</td>
<td>0.834</td>
<td>0.932</td>
<td>416</td>
<td>19 962</td>
<td>43 272</td>
</tr>
<tr>
<td>1076</td>
<td>0.868</td>
<td>0.956</td>
<td>334</td>
<td>15 108</td>
<td>58 380</td>
</tr>
<tr>
<td>2181</td>
<td>0.877</td>
<td>0.985</td>
<td>154</td>
<td>19 962</td>
<td>104 940</td>
</tr>
</tbody>
</table>

Furthermore, we have also clustered 99 local motifs from the 1133 teleost lincRNAs recently reported by Pauli et al. (2011). Interestingly, these contain up to five times more novel structural classes than their human counterpart. This might be explained by the fact that teleost fish underwent an additional whole-genome duplication (Christoffels et al. 2010) which increases the likelihood to identify paralogous genes.
of the found clusters consist of paralogs, structures from repeat-associated RNAs mobile elements, i.e. transposon-derived ncRNAs and maybe also pseudogenes. Even under this setting, we can show that we find many structured classes, when we use the commonly accepted SCI to determine structuredness of a cluster. This can easily be improved by using our approach, as it is for instance done in the EvoFam approach, where a 41-way multiple alignment is used.

Since the lincRNA dataset contains GO annotation, we have used this information for further evaluation. Although the GO enrichment analysis is limited by the low number of transcripts that are associated with GO terms (overall, GO terms are only available for 12% (1044/8195) of the Cahil et al. (2011) lincRNAs), we found nevertheless that the GO terms for the majority of clusters (53% but likely more) are consistent with a support clustering approach.

One of our ideas to soon improve our pipeline from an algorithmic point of view is to, instead of relying to fixed parameters, base the parameter optimization onto machine learning techniques, i.e. a support vector machines select the optimal parameters. This, together with future applications of graph kernel-based alignment-free clustering approaches will likely result in the detection of additional functionally relevant structured ncRNA classes.

**Funding**: German Federal Ministry of Education and Research (BMBF) [01III2912 to R.B. and his lab]; German Research Foundation [BA 2168/3-1 and BA 2168/4-1 to R.B. and his lab].

**Conflict of Interest**: none declared.

**REFERENCES**


Rickert, W. et al. (2007) RNA stem-loops: to be or not to be cleared by RNAse III. *RNA*, 13, 457–462.


