technische universität
dortmund

SAHN Clustering in Arbitrary Metric Spaces Using Heuristic Nearest Neighbor Search

Nils Kriege
Petra Mutzel
Till Schäfer

Algorithm Engineering Report
TR13-1-003
August 2013
ISSN 1864-4503

Faculty of Computer Science
Algorithm Engineering (Ls11)
44221 Dortmund / Germany
http://ls11-www.cs.uni-dortmund.de/

SAHN Clustering in Arbitrary Metric Spaces Using Heuristic Nearest Neighbor Search

Nils Kriege, Petra Mutzel, and Till Schäfer

Dept. of Computer Science, Technische Universität Dortmund, Germany
{nils.kriege,petra.mutzel,till.schaefer}@cs.tu-dortmund.de

Abstract. Sequential agglomerative hierarchical non-overlapping (SAHN) clustering techniques [10] belong to the classical clustering methods that are applied heavily in many application domains, e.g., in cheminformatics [4]. Asymptotically optimal SAHN clustering algorithms are known for arbitrary dissimilarity measures, but their quadratic time and space complexity even in the best case still limits the applicability to small data sets.
We present a new pivot based heuristic SAHN clustering algorithm exploiting the properties of arbitrary metric distance measures in order to obtain a best case running time of $\mathcal{O}(n \log n)$ for the input size n. Our approach requires only linear space and supports median and centroid linkage. It is especially suitable for expensive distance measures, as it needs only a linear number of exact distance computations. In extensive experimental evaluations on real-world and synthetic datasets, we compare our approach to exact state-of-the-art SAHN algorithms in terms of quality and running time. The evaluations show a subquadratic running time in practice and a very low memory footprint.

Keywords: SAHN clustering, nearest neighbor heuristic, data mining

1 Introduction

Clustering is a generic term for methods to identify homogeneous subsets, so-called *clusters*, in a set of objects. As unsupervised classification method it is a key technique in exploratory data analysis and widely applied in many fields like drug discovery, storage and retrieval, network analysis and pattern recognition [4,10]. A wealth of different clustering algorithms have emerged with varying definition of homogeneity. Typically this definition is based on a symmetric dissimilarity measure for pairs of objects to be clustered.

A special class of clustering algorithms are hierarchical methods, which provide additional information on the relationship between clusters and can reveal nested cluster structures. A prominent example are sequential agglomerative hierarchical non-overlapping clustering techniques (SAHN) [10]. These approaches start with singleton clusters and iteratively merge two clusters with minimal dissimilarity until only one cluster remains. The inter-cluster dissimilarity is determined by a *linkage* strategy and based on the dissimilarity of the objects contained in the clusters. The single, complete, average, median, centroid, and Ward linkage methods are well-studied and widely used [9]. A unique advantage of hierarchical methods is that the result can naturally be visualized as a *dendrogram*, a rooted binary tree where each node is linked to a merge operation with a certain dissimilarity. Cutting the dendrogram horizontally at a specific height leads to a set of subtrees where each root is associated with a subcluster. Thus, the result of a SAHN clustering allows for iterative refinement of clusters making these methods especially suitable for an interactive exploration process, even for very large data sets [3].

We motivate further requirements for our clustering algorithm by a concrete example arising in cheminformatics, although similar constraints apply in other application areas: (1) Data sets in cheminformatics are often large containing tens of thousands of molecules. (2) A hierarchical method is needed since the whole similarity structure of the data is important. Furthermore, SAHN clustering methods are well-known and studied in cheminformatics [4] and users may be accustomed to dendrogram representations. (3) Support for arbitrary metric distance measures is required, since chemical compounds are complex structures, which are typically represented as graphs or bit vectors, so-called *fingerprints*. (4) Distance measures between these objects may be expensive, e.g., based on the maximum common subgraph of two molecular graphs. Thus, we desire a low dependence on the computational complexity of the distance measure.

A major drawback of hierarchical clustering algorithms is their high time and space complexity. The best exact algorithms known for arbitrary dissimilarity measure have a worst-case running time of $\mathcal{O}(n^2)$ [6] and are optimal since the general problem requires time $\Omega(n^2)$ [11]. Exact approaches are typically based on a symmetric distance matrix, which leads to quadratic memory requirements and a quadratic number of distance computations. However, quadratic time and space complexity is prohibitive when applied to large datasets in practice.

Related Work. Several exact algorithms with quadratic worst-case running time are known, some of which are limited to specific linkage methods, e.g., the NN-Chain algorithm [9], the single linkage minimum spanning tree algorithm [14] and methods based on dynamic closest pairs [6]. Some SAHN algorithms (e.g., NNChain) can avoid the quadratic distance matrix when using representatives, e.g., centroids, for cluster representation. However, this approach is limited to vector space and leads to an increased amount of exact distance computations.

Several methods to speedup clustering have been proposed. Data summarization is a common accelerating technique. An easy approach is to draw a random sample and cluster it instead of the whole dataset. However, using random sampling leads to distortions in the clustering results. The kind of distortion is influenced by the used linkage method and because of this, many sophisticated summarization techniques are only suitable for special linkages. For example Patra et al. [13] proposed to use an accelerated leaders algorithm to draw a better sampling for average linkage. Another example is the Data Bubble summarization technique [19,2] which was originally developed for OPTICS clustering [1], but is also suitable for single linkage SAHN clustering.

Further acceleration is possible when using heuristic methods. Koga et al. [7] proposed Locality Sensitive Hashing (LSH) for a single linkage like algorithm. Its time complexity is reduced to $\mathcal{O}(nB)$, where B is practically a constant factor. Although the runtime is very promising, it relies on vector data and is limited to single linkage, which is rarely used in cheminformatics.

Using the properties of metric distance function is a common approach to accelerate different clustering techniques. Pivot based approaches have been proposed to reduce the number of exact distance computations for hierarchical clustering [12] and to speedup k-means [5]. To accelerate OPTICS a pivot based approach for heuristic k-close neighbor rankings was proposed by Zhou and Sander [17,18]. They also introduced a pivot tree data structure that enhances the effectiveness of the pivots for close neighbor rankings. SAHN clustering algorithms often rely on nearest neighbor (NN) queries (e.g., NNChain, Generic Clustering [11], Conga Line data structure [6]), which can be accelerated for metric distance functions [16]. However, the reduction of the NN search complexity does not necessarily reduce the asymptotic runtime of the clustering algorithms (see Sect. 3 for more details).

Our contribution. We propose a new SAHN clustering algorithm for centroid and median linkage that benefits from sublinear NN queries and combine it with a pivot based indexing structure to obtain subquadratic running time in practice. The theoretical time complexity of our algorithm for clustering n objects is $\mathcal{O}(n^2 \log(n))$ in the worst case and $\mathcal{O}(n \log(n))$ in the best case. Our approach is broadly applicable since it is not limited to the Euclidean vector space and many dissimilarity measures actually are a metric. Moreover, the new method requires only linear space and a linear number of distance computations and therefore allows to cluster large datasets even when distance computations are expensive. Our extensive experimental evaluation on a real-world dataset from cheminformatics and on two synthetic datasets shows that the new method yields high-quality results comparable to exact algorithms, in particular when the datasets indeed contain a nested cluster structure.

2 Preliminaries

A *clustering* of a set of objects $\mathcal{X} = \{x_1, \dots, x_n\}$ is a partition $\mathcal{C} = \{C_1, \dots, C_k\}$ of $\mathcal{X}$. A hierarchical clustering of n objects yields n distinct clusterings obtained from cutting the associated dendrogram at different heights. We refer to a clustering which results from such a cut and containing i clusters as the clustering at level $i \in \{1, \dots, n\}$. SAHN clustering is performed based on a distance function $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^{\geq 0}$ between the objects and an inter-cluster distance $D : \mathcal{P}(\mathcal{X}) \times \mathcal{P}(\mathcal{X}) \to \mathbb{R}^{\geq 0}$ which is also called *linkage*.

Let $P \subset \mathcal{X}$ be a set of *pivots*. The triangle inequality in combination with the symmetric property fulfilled by metric distance functions yields a lower and upper bounds for the distances between any two objects based on the exact distances between the objects and the pivots:

$$\forall x_i, x_j \in \mathcal{X}, p \in P : |d(x_i, p) - d(x_j, p)| \leq d(x_i, x_j) \leq d(x_i, p) + d(x_j, p) \quad (1)$$

In the following we are using a tilde as notion for heuristic functions (e.g. $\mathrm{NN}(x)$ is the exact NN of x while $\tilde{\mathrm{NN}}(x)$ represents the heuristic NN).

3 A Heuristic SAHN Clustering Algorithm

We present a Heuristic SAHN (HSAHN) algorithm that is based on a variation of the generic clustering algorithm from [11] and utilizes an index structure for NN search. To efficiently determine heuristic NNs we adopt the approach of [18,17] based on best frontier search combined with a pivot tree and generalize it to support the specific requirements for use with SAHN clustering.

3.1 Generic Clustering

The original generic clustering algorithm has a complexity of $\Omega(n\ (\log(n) + k + m))$ and $\mathcal{O}(n^2\ (\log(n) + k))$ for geometric linkages in vector space and $\Omega(n^2)$ for arbitrary linkages and distance measures. The value k is the complexity of the NN search and m the complexity of the merge process. Although other SAHN clustering algorithms have a quadratic upper bound, the practical runtime of the generic clustering competes with the other algorithms [11] and is orientated towards the lower bound. We have chosen the generic clustering algorithm, because our modified version of the algorithm (Alg. 1) achieves $\Omega(n\ (\log(n) + k + m))$ for arbitrary linkage and distance measures and therefore the lower bound is directly influenced by the complexity of the NN search. This is also the case for the NNChain algorithm, but it requires the reducibility property, which is not guaranteed for heuristic NN searches. Note that HSAHN requires metric distance function and is therefore limited to median and centroid linkage, but this is due to the NN search and not a limitation of the clustering algorithm.

Additionally to the runtime modifications, we modified the generic clustering algorithm, that it minimizes distance distortions for our NN search. In our case the heuristic NN search directly implies a non-symmetric, heuristic distance $\tilde{D}$. Since we use the lower bound of (1) as a heuristic for the real distance, we know:

$$D(x, \tilde{\mathrm{NN}}(x)) \geq \max\{\tilde{D}(x, \tilde{\mathrm{NN}}(x)), \tilde{D}_{\mathrm{rev}} = \tilde{D}(\tilde{\mathrm{NN}}(x), x)\}$$

It is possible to detect some cases of $\tilde{D}_{\min} < \tilde{D}_{\min_{\mathrm{rev}}}$ for the minimal distance $\tilde{D}_{\min}$ without recalculating the distance over all pivots after a NN search. For symmetric distance measures the minimal pairwise distance implies a reciprocal NN pair. Although this assumption does not hold for the used heuristic NN search, it does hold with a high probability. If the minimal distance implies a reciprocal NN object pair, we can use the already computed reverse distance and improve the quality of our heuristic by reinserting the tuple $(x, \tilde{\mathrm{NN}}(x))$ in the priority queue with a distance of $\max\{\tilde{D}_{\min}, \tilde{D}_{\min_{\mathrm{rev}}}\}$ (lines $11 - 13$ of Alg. 1). Our benchmarks have proven that this approach is faster than recalculating the distance over all pivots and it does not harm the clustering quality significantly.

```
 1: function GENERICCLUSTERING($\mathcal{X}$)
 2:     $currentLevel \leftarrow$ singletonClusters($\mathcal{X}$)          ▷ clusters of the actual level
 3:     for all $C \in currentLevel$ do                                        ▷ initialization of $Q$
 4:         $Q$.insert($C, \tilde{\text{NN}}(C), \tilde{D}(C, \tilde{\text{NN}}(C))$)      ▷ $Q$ is sorted by $\tilde{D}(C, \tilde{\text{NN}}(C))$
                                                                       (value at the time of insertion)
 5:     while $currentLevel$.size() $> 1$ do                                          ▷ main loop
 6:         $(C_i, C_j) \leftarrow Q$.extractMin()
 7:         while !$currentLevel$.contains($C_i$) or !$currentLevel$.contains($C_j$) do
                                                      ▷ invalid entry → recalculation of NN
 8:             if $currentLevel$.contains($C_i$) then
 9:                 $Q$.insert($C_i, \tilde{\text{NN}}(C_i), \tilde{D}(C_i, \tilde{\text{NN}}(C_i))$)
10:             $(C_i, C_j) \leftarrow Q$.extractMin()
11:         if $\tilde{\text{NN}}(C_j) = C_i$ and $\tilde{D}(C_i, C_j) < \tilde{D}(C_j, \tilde{\text{NN}}(C_j))$ then
                                                     ▷ using already calculated $\tilde{\text{NN}}(C_j)$
12:             $Q$.insert($C_i, C_j, \tilde{D}(C_j, \tilde{\text{NN}}(C_j))$)
13:             continue
14:         $C_k \leftarrow$ mergeCluster($C_i, C_j$)              ▷ $(C_i, C_j)$ minimal reciprocal NN pair
15:         $currentLevel \leftarrow currentLevel \setminus \{C_i, C_j\} \cup C_k$
16:         $Q$.insert($C_k, \tilde{\text{NN}}(C_k), \tilde{D}(C_k, \tilde{\text{NN}}(C_k))$)
17:     return $currentLevel$.get(0)                        ▷ return root node of the dendrogram
```

Algorithm 1. Modified Generic Clustering Algorithm

3.2 Pivot Tree

As mentioned before, we are using the lower bound of (1) for heuristic distance approximations:

$$\tilde{D}(C_i, C_j) = \max_{p \in P} |D(C_i, p) - D(p, C_j)|$$

At this point it becomes clear that the inter-cluster distance D (i.e. linkage) needs to be metric, and therefore the HSAHN algorithm is limited to centroid and median linkage.

To increase the effectiveness of the pivots for close or NN queries Zhou and Sander proposed a pivot tree data structure [18]. The main idea behind this structure is, that the distance between close objects must be more precise than the distance between further objects to calculate the correct close or nearest neighbors. In our case we are searching for NNs with the following formula:

$$\tilde{\text{NN}}(C_i) = \text{argmin}_{C_j}\{\tilde{D}(C_i, C_j)\} \tag{2}$$

The original pivot tree is a static data structure. In contrast SAHN clustering merges clusters and therefore we extended the data structure to allow deletion and insertion of objects and clusters, respectively. Additionally we used a different strategy to calculate the heuristic distances within the pivot tree and a simplified notion.

As shown in Fig. 1 each node of the pivot tree is linked to a set of clusters X and a set of pivots $P \subseteq X$. The set of pivots is randomly chosen from X. One

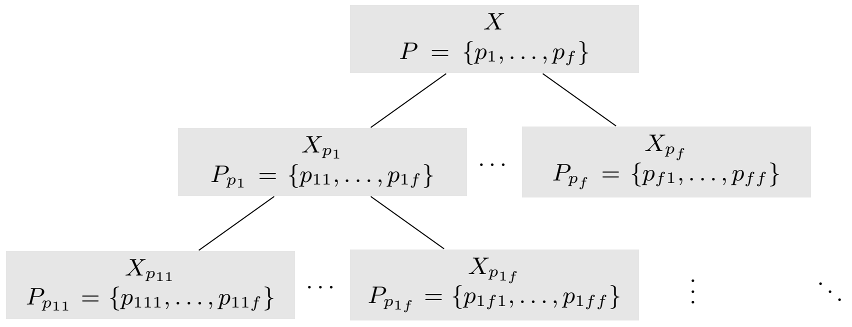

X
$P = \{p_1, \ldots, p_f\}$
X_{p_1}
$P_{p_1} = \{p_{11}, \ldots, p_{1f}\}$
X_{p_f}
$P_{p_f} = \{p_{f1}, \ldots, p_{ff}\}$
$X_{p_{11}}$
$P_{p_{11}} = \{p_{111}, \ldots, p_{11f}\}$
$X_{p_{1f}}$
$P_{p_{1f}} = \{p_{1f1}, \ldots, p_{1ff}\}$

Fig. 1. Pivot Tree

child node is created for each pivot. The root nodes set X contains all clusters, while the child nodes set X_{pi} contains all nodes from X which are closest to p_i. Therefore all clusters in X_{pi} are relatively close to each other. The calculation of the heuristic distance in combination with the pivot tree can be over the common pivots $P_{i\cup j}$:

$$P_{i\cup j} = \{p \in P_k \mid C_i, C_j \in X_k\}$$
$$\tilde{D}(C_i, C_j) = \max_{p \in P_{i\cup j}} |D(p, C_i) - D(p, C_j)|$$

It is computationally cheap to compute $P_{i\cup j}$ since we now that each cluster is present only on the direct path from a leaf to the root node. To find the leaf node in constant time we store this relationship in a hashing data structure during the construction process. The relevant nodes for $P_{i\cup j}$ are all ancestors of the lowest common ancestor (LCA) of the leaf nodes for C_i and C_j and the LCA itself.

The construction process starts with the root node followed by a series of split operations until a maximum number of leaf nodes are reached. Each split operation creates the child nodes for the node with the maximum size (i.e. cardinality of X). At construction time the data structure contains all singleton clusters.

The deletion of a cluster C from the pivot tree is simply the removal of C from all X_i. Inserting a merged cluster $C_{i\cup j}$ into the data structure is a bit more complicated since we cannot compute the exact distances to all pivots efficiently. It can be done efficiently with the Lance Williams Update Formula [8] for all ancestors of the LCA of C_i and C_j, because we know the distance of both clusters to the nodes pivots and we can use $\tilde{D}$ as distance between the merge clusters. This approach has the drawback that the depth of the pivot tree will decrease over a series of merge operations. However, this will happen relatively late in the merge process because close clusters will be merged first during SAHN clustering. Also the locality property of clusters in X will not be violated.

3.3 Best Frontier Search

The best frontier search was already suggested to accelerate the OPTICS clustering algorithm in [17,18]. We will briefly describe the main idea below.

When ordering all clusters by their distance to a single pivot p the heuristic distances $\tilde{D}(C_i, C_j)$ to a fixed cluster C_i are monotonically increasing when C_i and C_j are further away in the ordering. Hence, it is sufficient to consider the neighbors of C_i (in the ordering) to find C_i's NN with respect to a single pivot.

Typically more than one pivot is needed to achieve sufficient quality and the minimal maximum lower bound is what we are searching for. The best frontier search solves this problem by calculating a sorted list L_p of clusters for each pivot p. To find the position of a cluster in L_p in amortized constant time, the list elements are inserted into a hash table. Furthermore the best frontier search uses a priority queue which contains entries of these lists that form the *frontier*. It is sorted by the lower bound with respect to the single pivot to which the entry belongs.

When searching a NN of C the queue initially contains all clusters next to C in a list L_p for some $p \in P$. Then the clusters with the lowest bounds are successively retrieved from the queue and it is counted how often a certain cluster is retrieved. After the retrieval of each cluster C_x the frontier is *pushed* by adding the cluster to the queue that is next to C_x in the list L_i from which C_i was added to the queue. The cluster C_j which is counted $|P|$ times first is the heuristic NN with respect to (2). The rationale is that the lower bounds induced by the clusters retrieved from the queue are monotonically increasing. That means all lower bounds which will be obtained in the future are greater than the heuristic distance $\tilde{D}(C_i, C_j)$ and therefore C_j must be the heuristic NN of C_i.

To combine the pivot tree with the best frontier search, Zhou and Sanders run a k-close neighbor ranking for each node of the pivot tree and joins the different close neighbors for each object afterwards. This approach is not possible for SAHN clustering since we cannot efficiently calculate which of the heuristic NNs found for each node is the best. Furthermore it is possible that the NNs found for each node are not correct with respect to (2), while the best frontier search in general can guarantee to find the correct heuristic NN. For that reason we need to use a different technique which will be described below.

Our integration of the best frontier search into the pivot tree runs a NN query for cluster C_i over all pivots $p \in P_x$ where $C_i \in X_x$. While searching for the NN of C_i the cardinality of $P_{i \cup j}$ is not fixed for an arbitrary cluster C_j. Therefore it must be calculated for each cluster C_j that is retrieved from the frontier separately. The value can be calculated by finding the LCA of C_i and C_j in the pivot tree and counting the number of all pivots on the path between the LCA and the root node. To avoid unnecessary calculations, it is calculated on demand and cached for the time of the NN query.

Because the asymptotic worst case complexity of a NN query with the best frontier search is not better than linear (see Sect. 3.4), a search depth bound s

is used. After s clusters are retrieved from the priority queue the best frontier search is stopped and the cluster that is counted most often is returned as NN.

3.4 Theoretical Complexity

Time complexity. To initialize the pivot tree data structure n clusters are assigned to f pivots on each level of the tree. The construction of the pivot tree therefore needs $\Theta(d\ f\ n)$ time where d represents the tree depth. Since the number of required pivots for a achieving certain quality does not depend on the input size (Sect. 4), f and d can be seen as constant values and the overall construction time is linear.

As mentioned before, the generic clustering algorithm has a runtime of $\Omega(n\ (\log(n) + k + m))$ and $\mathcal{O}(n^2\ (\log(n) + k))$, if k is the complexity of the NN search and m the complexity of the merge process.
The merging consists of two deletions from and one insertion into the pivot tree. Therefore, the complexity for merging is $\mathcal{O}(d\ \log(n))$ as it takes $\mathcal{O}(\log(n))$ time to insert a cluster into a sorted list L_i.
The search of a NN is limited by $\mathcal{O}(p\ s)$ if p is the number of the used pivot elements and s the search depth bound. The runtime includes $\mathcal{O}(s)$ extractions from the frontier queue with length $\mathcal{O}(p)$. Pushing the frontier and finding the initial elements in L_i for each pivot takes constant time. With the same rationale as before p can be seen as a constant. It is shown in the experimental evaluation in Sect. 4 that s can be chosen as a constant value, too.

The overall time complexity for HSAHN clustering is therefore limited by $\mathcal{O}(n^2 \log(n))$ in the worst case and $\mathcal{O}(n \log(n))$ in the best case.

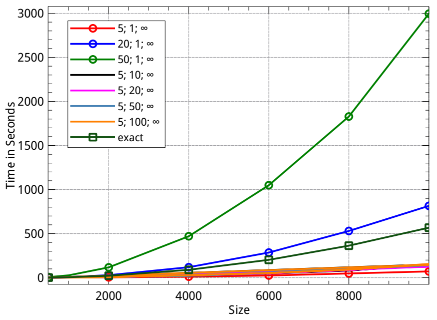
Space requirements. Since the tree depth d is a constant factor and therefore the number of nodes is also a constant, the pivot tree needs only linear space. Each node stores a constant number of sorted lists L_i which store at most n clusters. Also the hash table (to find the leaf nodes), the priority queue in the generic clustering algorithm and the frontier queue need $\mathcal{O}(n)$ space. Therefore the overall space requirements are linear with respect to the input size.

4 Experimental Results

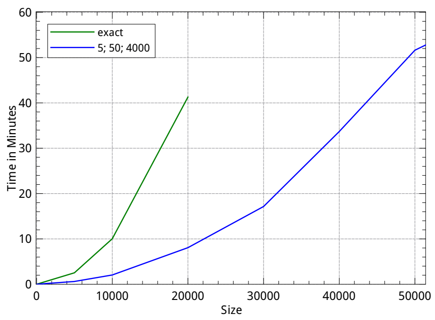
This section will cover performance and quality measurements of our Java implementation. All tests were performed on an Intel Core i7 CPU 940 (2.93GHz) with a Linux operating system and a Java HotSpot virtual machine (version 1.6) which was limited to 5 GiB of heap space. The implementation as well as the evaluation framework is publicly available at the Scaffold Hunter Website[1] and licensed under the GPLv3.

We used real-world as well as synthetic datasets[2] for the evaluation. The real-world dataset (SARFari kinase) stems from the ChEMBL[3] database and contains

[1] `http://scaffoldhunter.sourceforge.net`
[2] `http://ls11-www.cs.tu-dortmund.de/staff/publication_data_schaefer`
[3] `https://www.ebi.ac.uk/chembldb`



(a) Unbounded Search Depth



(b) Complete Kinase Dataset

Fig. 2. Runtime / Kinase Dataset / Tanimoto Distance

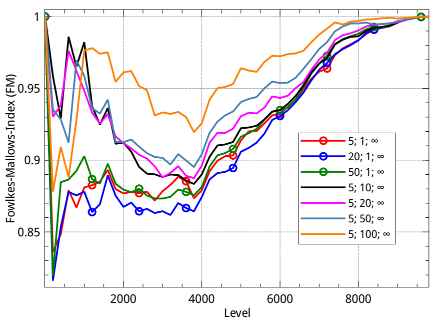
a set of ≈ 50 000 molecules. The Euclidean and Tanimoto distance are utilized for this dataset, while the latter is applied on the Daylight bit fingerprint (1024 bits) from the CDK toolkit[4] which represent structural information of the molecules. Two additional synthetic euclidean datasets are used to analyze the impact of the clusterability on the quality. The first contains uniformly distributed data and the second 32 normally distributed and well-separated clusters.

All quality measurements are done by comparing each level of the HSAHN results with the exact results. The Fowlkes Mallows Index and the Normalized Variation of Information measurement is employed to measure each single level. All test results (performance and quality) are averaged over three runs, because the random selection of pivots results in a non-deterministic behavior of the algorithm. Without an exception the differences were very small and the results were stable over different runs. However one might not over-interpret small differences in the plots. In the legends of the plots the parameters of the best frontier search are noted as $(f; l; s)$, where f is the number of pivots per node, l is the number of leaf nodes and s is the search depth bound. Note that $l = 1$ means that the pivot tree is deactivated.

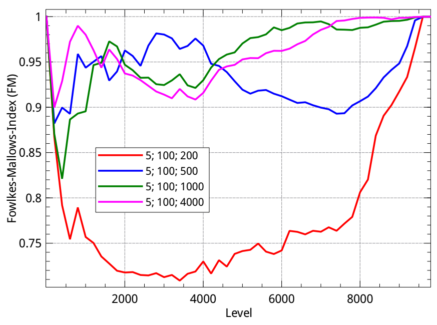
Speed. As shown in Fig. 2(a) the performance of the algorithm scales linear with the number of pivots. For the unbounded search depth the asymptotic runtime behavior is clearly quadratic and the absolute runtime for a high pivot count even exceeds the runtime of the exact algorithm. It is noteworthy that the number of leafs in the pivot tree does not have a major influence on the overall performance. With a reasonably set of parameters (the rationale follows in the quality evaluation) the time to cluster the whole kinase dataset (Fig. 2(b)) is much lower than for the exact case. The heuristic curve flattens in the higher levels. Therefore the asymptotic behavior is subquadratic.

It it important to know that we were unable to cluster a dataset with 30 000 structures with the exact algorithm due to memory constraints. On the contrary,

[4] `http://cdk.sourceforge.net`

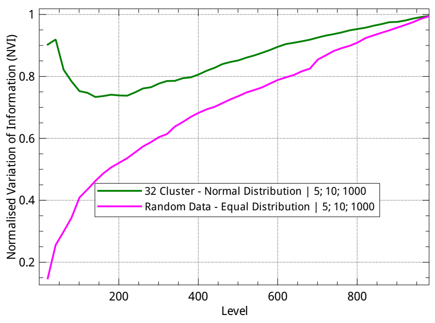


(a) Quality: Number of Pivots

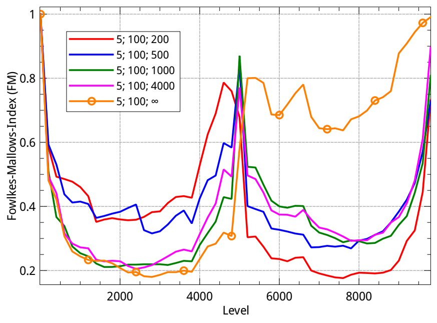


(b) Quality: Search Depth

Fig. 3. Kinase Dataset / Euclidean Distance (5 Dimensions)



(a) Synthetic Datasets



(b) Ambiguity: Kinase / Tanimoto

Fig. 4. Influence of the Data Distribution

the heuristic clustering algorithm used less than 1 GiB of memory to cluster the whole kinse dataset.

Quality - Pivot Count and Pivot Tree. From the theoretical point of view more pivots should result in a better quality of the best frontier search. However our test results do not show significant differences in quality if the number of pivots are increased over a certain threshold. From our observations the main aspect that influences this threshold is the intrinsic dimensionality of the data and not the input size. Figure 3(a) clearly shows that there is no significant difference in quality if no pivot tree and 5, 20 or 50 pivots are used. Anyway, when using the pivot tree data structure, the quality can be enhanced further. This is a remarkable result, as the runtime of the setting $(50; 1; \infty)$ is more than 10 times higher than the runtime of the setting $(5; 100; \infty)$.

Quality - Search Depth and Ambiguity. The search depth is limiting the runtime of the best frontier search. Therefore it is very important to know, if the search depth can be chosen in a sublinear relation to the input size, while retaining a constant quality. Our tests revealed that this search bound can be chosen constant. For this we calculated an average quality score over all levels but the lowest 10% (e.g. level 9 001 to 10 000 for the input size 10 000) and compared this values for different fixed search depths over a series of different input sizes. Also for very low search depths the quality was constant over all input sizes. The reason to not use the lowest 10% is that this levels are strongly influenced by the starting situation where both clusterings contain only singletons.

Experimental evaluation showed that the search depth can be chosen about 500 in the low dimensional euclidean space (Fig. 3(b)). Lower values significantly harm the quality of the results. The search depth seems to be sensitive to the number of pivots used and the dimensionality of the data. The first observations are not surprising, since an increased amount of pivots increases the exit condition in the best frontier search loop. The second observation can be explained by the distribution of the distances. For high dimensional space the distances become more equal to each other. Equal distance means that even a small deviation of the lower distance bound (1) results in a higher probability, that this item is retrieved falsely from the frontier.

This observation also explains why the quality of the Tanimoto measurement (Fig. 4(b)) is lower than the quality of the Euclidean measurement and why the limitation of the search depth has such a huge impact. The number of different distances for the Tanimoto distance is limited by the length of the Farey sequence which is $\frac{3n^2}{\pi^2}$, where n is the bit count. For a dataset size of 10 000 this means that the number of object pairs is about 300 times more than the number of distinct distance values. This leads to a high ambiguity in the clustering process and makes the results even unstable when comparing two exact algorithms. It is a general problem when comparing clustering by their structure, that alternative good clusterings cannot be covered by such a measure.

The peak in Fig. 4(b) at level 5 000 is also explainable when having a look at Fig. 4(a). If we have separated clusters inside the dataset, the quality is good at exactly the level which corresponds to the number of clusters. This implies that we are able to identify real clusters in a dataset with the HSAHN algorithm albeit the clustering quality might not be very well over all levels. This conclusion is very important because it proves the practicability of our approach.

5 Conclusions

Our tests show that the HSAHN algorithm can greatly expand the size of datasets which can be clustered in a reasonable amount of time. Furthermore the memory usage is lowered dramatically, which often sets a hard limit to the dataset size. The linear dependence on exact distance calculations makes it possible to use computationally expensive distance measures even on huge datasets.

Our approach was integrated in the software Scaffold Hunter [15], a tool for the analysis and exploration of chemical space, and has been proven a valuable alternative to exact approaches in practice.

References

1. Ankerst, M., Breunig, M.M., Kriegel, H.P., Sander, J.: Optics: ordering points to identify the clustering structure. SIGMOD Rec. 28(2), 49–60 (Jun 1999)
2. Breunig, M.M., Kriegel, H.P., Kröger, P., Sander, J.: Data bubbles: quality preserving performance boosting for hierarchical clustering. SIGMOD Rec. 30(2), 79–90 (May 2001)
3. Chen, J., MacEachren, A.M., Peuquet, D.J.: Constructing overview + detail dendrogram-matrix views. IEEE Transactions on Visualization and Computer Graphics 15, 889–896 (2009)
4. Downs, G.M., Barnard, J.M.: Clustering Methods and Their Uses in Computational Chemistry, pp. 1–40. John Wiley & Sons, Inc. (2003)
5. Elkan, C.: Using the triangle inequality to accelerate k-means. In: ICML'03. pp. 147–153 (2003)
6. Eppstein, D.: Fast hierarchical clustering and other applications of dynamic closest pairs. J. Exp. Algorithmics 5 (Dec 2000)
7. Koga, Hisashi, Ishibashi, Tetsuo, Watanabe, Toshinori: Fast agglomerative hierarchical clustering algorithm using Locality-Sensitive Hashing. Knowledge and Information Systems 12(1), 25–53 (2007)
8. Lance, G.N., Williams, W.T.: A general theory of classificatory sorting strategies 1. hierarchical systems. The Computer Journal 9(4), 373–380 (1967)
9. Murtagh, F.: Multidimensional clustering algorithms. In: COMPSTAT Lectures 4. Physica-Verlag, Wuerzburg (1985)
10. Murtagh, F., Contreras, P.: Algorithms for hierarchical clustering: an overview. Wiley Interdisc. Rew.: Data Mining and Knowledge Discovery 2(1), 86–97 (2012)
11. Müllner, D.: Modern hierarchical, agglomerative clustering algorithms (2011), arXiv:1109.2378v1
12. Nanni, M.: Speeding-up hierarchical agglomerative clustering in presence of expensive metrics. In: Advances in Knowledge Discovery and Data Mining. pp. 378–387. Springer Berlin (2005)
13. Patra, B.K., Hubballi, N., Biswas, S., Nandi, S.: Distance based fast hierarchical clustering method for large datasets. In: Proceedings of the 7th international conference on Rough sets and current trends in computing. pp. 50–59. RSCTC'10, Springer-Verlag (2010)
14. Rohlf, F.J.: Hierarchical clustering using the minimum spanning tree. Comput. Journal 16, 93–95 (1973)
15. Wetzel, S., Klein, K., Renner, S., Rauh, D., Oprea, T.I., Mutzel, P., Waldmann, H.: Interactive exploration of chemical space with Scaffold Hunter. Nature Chemical Biology 5(8), 581–583 (Jun 2009)
16. Zezula, P., Amato, G., Dohnal, V., Batko, M.: Similarity Search: The Metric Space Approach, Advances in Database Systems, vol. 32. Springer (2006)
17. Zhou, J.: Efficiently Searching and Mining Biological Sequence and Structure Data. Ph.D. thesis, University of Alberta (2009)
18. Zhou, J., Sander, J.: Speedup clustering with hierarchical ranking. In: Proceedings of the Sixth International Conference on Data Mining. pp. 1205–1210. ICDM '06, IEEE Computer Society, Washington, DC, USA (2006)
19. Zhou, J., Sander, J.: Data bubbles for non-vector data: Speeding-up hierarchical clustering in arbitrary metric spaces. In: Proceedings of the 29th international conference on very large data bases - Volume 29. pp. 452–463. VLDB '03, VLDB Endowment (2003)