Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 18, no. 4, pp. 577–602 (2014)
DOI: 10.7155/jgaa.00338

# Practical SAHN Clustering for Very Large Data Sets and Expensive Distance Metrics

*Nils Kriege Petra Mutzel Till Schäfer*

Dept. of Computer Science, Technische Universität Dortmund, Germany

**Abstract**

Sequential agglomerative hierarchical non-overlapping (SAHN) clustering techniques belong to the classical clustering methods applied heavily in many application domains, e.g., in cheminformatics. 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 metric distance measures in order to obtain a best-case runtime 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.

This aspect is demonstrated in our extensive experimental evaluation, where we apply our algorithm to large graph databases in combination with computationally demanding graph distance metrics. We compare our approach to exact state-of-the-art SAHN algorithms in terms of quality and runtime on real-world and synthetic instances including vector and graph data. The evaluations show a subquadratic runtime in practice and a very low memory footprint. Our approach yields high-quality clusterings and is able to rediscover planted cluster structures in synthetic data sets.

| Submitted: | Reviewed: | Revised: | Accepted: | Final: |
|---|---|---|---|---|
| March 2014 | July 2014 | July 2014 | August 2014 | December 2014 |

Published:
December 2014

| Article type: | Communicated by: |
|---|---|
| Regular paper | S. Pal and K. Sadakane |

A preliminary version of this paper [22] was published in the proceedings of WALCOM 2014. This work was supported by the German Research Foundation (DFG), priority programmes "Algorithm Engineering" (SPP 1307) and "Algorithms for Big Data" (SPP 1736).

*E-mail addresses:* nils.kriege@cs.tu-dortmund.de (Nils Kriege) petra.mutzel@cs.tu-dortmund.de (Petra Mutzel) till.schaefer@cs.tu-dortmund.de (Till Schäfer)

## 1 Introduction

Clustering is a generic term for methods to identify homogeneous subsets, so-called *clusters*, in a set of objects. It is a key technique in exploratory data analysis and widely used in many fields like drug discovery, storage and retrieval, network analysis and pattern recognition [11, 28]. 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.

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) [1, 9, 28]. These approaches start with singleton clusters and iteratively merge two clusters with minimum 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 [27]. A unique advantage of hierarchical methods is that the result can naturally be visualized as a *dendrogram*, a rooted binary tree where each inner 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 SAHN clustering allows for iterative refinement of clusters making these methods especially suitable for an interactive exploration process, even for very large data sets [7].

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 [11] 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 measures have a worst-case runtime of $\mathcal{O}(n^2)$ [13] and are optimal since the general problem requires time $\Omega(n^2)$ [26]. 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 for large data sets. In case of low-dimensional vector data the runtime required for distance computations is often neglected. For expensive distance measures a substantial amount of the total runtime is actually due to the quadratic number of pairwise distance computations. In particular, when complex structured objects like graphs are clustered this can be a crucial bottleneck.

**Related Work.** Several exact algorithms with a quadratic worst-case runtime are known, some of which are limited to specific linkage methods, e.g., the NN-Chain algorithm [27], the single linkage minimum spanning tree algorithm [31] and methods based on dynamic closest pairs [13]. 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 speed up clustering have been proposed. Data summarization is a common acceleration technique. An easy approach is to draw a random sample and cluster it instead of the whole data set. However, using random sampling leads to distortions in the clustering results. The 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. [30] use an accelerated leaders algorithm to draw a better sampling for average linkage. Another example is the Data Bubble summarization technique [3, 40], which was originally developed for OPTICS clustering [2], but is also suitable for single linkage SAHN clustering.

Further acceleration is possible when using heuristic methods. Koga et al. [21] 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 functions 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 [29] and to speedup $k$-means [12]. To accelerate OPTICS a pivot based approach for heuristic $k$-close neighbor rankings was proposed by Zhou and Sander [39, 41]. 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 [26], Conga Line data structure [13]), which can be accelerated for metric distance functions [38]. 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).

Clustering methods based on distances can be used to cluster structured objects like graphs since various graph distance measures exist (see Sect. 4). However, clustering graphs remains challenging, because these measures are often computationally demanding and typically have a high intrinsic dimensionality [34] limiting the effectiveness of index structures. Several methods especially dedicated to graphs have been proposed: Based on concepts to summarize a set of graphs by a single representative, $k$-means clustering was generalized to graphs [15, 19]. A different variant referred to as kernel $k$-means can be applied to graphs utilizing the large number of available graph kernels [35]. Graph mining techniques were used to perform clustering in a suitable low-dimensional feature space, cf. [34]. In [32] a streaming algorithm for overlapping clustering was proposed that iteratively tries to assign a graph to each cluster using algorithmic techniques from frequent subgraph mining.

**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 runtime 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 requiring only linear space. Our approach is broadly applicable since it is not limited to the Euclidean vector space, but supports arbitrary metric distance measures. Many known distance measures for complex objects like graphs actually are a metric [10] and their computation often is expensive. The new method requires only a linear number of distance computations and can therefore cluster large data sets even when distance computations are expensive. To demonstrate the effectiveness and versatility of our approach we apply the heuristic SAHN algorithm in the domain of graphs using various distance metrics for graphs. Our extensive experimental evaluation on a real-world data set from cheminformatics and synthetic data sets shows that the new method yields high-quality results comparable to exact algorithms, in particular when the data sets indeed contain a nested cluster structure. To our knowledge there are no other competing heuristic SAHN algorithms for general metric space supporting centroid or median linkage.

## 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 by cutting the associated dendrogram at different heights. We refer to a clustering that results from such a cut and contains $i$ 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*.

The triangle inequality in combination with the symmetric property fulfilled by metric distance functions yields lower and upper bounds for the distance between two objects based on the exact distances between the objects and a distinguished object $p \in \mathcal{X}$ referred to as *pivot*:

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

A nearest neighbor (NN) of a cluster $C_i$ is the cluster $C_j \neq C_i$ to which $C_i$ has the smallest distance. In the following we use the tilde and the hat to denote heuristic methods, e.g., $\widetilde{\text{NN}}$ and $\widehat{\text{NN}}$ refer to a 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 [26] and utilizes an index structure for NN search. To efficiently determine heuristic NNs we adopt the approach of [41, 39] based on best frontier search combined with a pivot tree and generalize it to support the specific requirements arising from 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 [26] and is close to the lower bound. Our modified version of the generic clustering algorithm (Alg. 1) has complexity $\Omega(n\ (\log n + k + m))$ for arbitrary linkage and distance measures and therefore the lower bound directly depends on 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 centroid linkage, median linkage and heuristic NN searches. Note that HSAHN requires a 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.

Besides the runtime improvement, we modified the generic clustering algorithm in order to minimize distance distortions caused by our heuristic NN search, which is called in line 25. Let $\widehat{\mathrm{NN}}(C)$ denote the heuristic NN of $C$. As a side effect of calculating the NN also a non-symmetric, heuristic distance $\widehat{D}(C, \widehat{\mathrm{NN}}(C))$ is calculated without extra computational costs (see Sect. 3.3 for more details). Since we use the lower bound of (1) for the heuristic, we know that the real distance is greater than or equal to the heuristic distance in any direction:

$$D(C, \widehat{\mathrm{NN}}(C)) \geq \max\{\widehat{D}(C, \widehat{\mathrm{NN}}(C)), \widehat{D}_{\mathrm{rev}} := \widehat{D}(\widehat{\mathrm{NN}}(C), C)\}$$

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. A problem is, that the priority queue used by the generic clustering algorithm always returns the weaker lower bound first, since it has a higher priority. To overcome this issue without calculating a computationally more demanding symmetric distance, we can use the reverse distance, which is already computed in case of a reciprocal NN pair, and improve the quality of our heuristic by reinserting the tuple $(C, \widehat{\mathrm{NN}}(C))$ into the priority queue with the distance $\max\{\widehat{D}, \widehat{D}_{\mathrm{rev}}\}$ (lines 16-19 of Alg. 1).

Our benchmarks show that this approach is faster than calculating an improved heuristic distance with our pivot based algorithm and does not harm

```
function GenericClustering($\mathcal{X}$)
    $\mathcal{L} \leftarrow$ SingletonClusters($\mathcal{X}$)          ▷ $\mathcal{L}$: clusters of the actual level
    $Q \leftarrow \emptyset$                                  ▷ priority queue of clusters
    $NN \leftarrow \emptyset$                                 ▷ mapping from cluster to its NN
    $DistNN \leftarrow \emptyset$                 ▷ mapping from cluster to the distance to its NN
    for all $C \in \mathcal{L}$ do                                 ▷ initialization of $Q$
        InsertNn($C$)
    while $|\mathcal{L}| > 1$ do                                        ▷ main loop
        $C_i \leftarrow Q$.ExtractMin()
        $C_j \leftarrow NN[C_i]$
        while $C_i \notin \mathcal{L}$ or $C_j \notin \mathcal{L}$ do                         ▷ invalid entry
            if $C_i \in \mathcal{L}$ then
                InsertNn($C_i$)
            $C_i \leftarrow Q$.ExtractMin()
            $C_j \leftarrow NN[C_i]$
        if $NN[C_j] = C_i$ and $DistNN[C_i] < DistNN[C_j]$ then
            $Q$.Insert($C_i, DistNN[C_j]$)
            $DistNN[C_i] \leftarrow DistNN[C_j]$
            continue
        $C_k \leftarrow$ MergeCluster($C_i, C_j$)              ▷ $(C_i, C_j)$ minimal NN pair
        $\mathcal{L} \leftarrow \mathcal{L} \setminus \{C_i, C_j\} \cup C_k$
        InsertNn($C_k$)
    return $\mathcal{L}$                              ▷ return root node of the dendrogram
function InsertNn($C$)
    $(C_{\mathrm{NN}}, dist) \leftarrow$ NnSearch($C$)   ▷ NN search strategy can be plugged in
    $Q$.Insert($C, dist$)                                         ▷ $Q$ is sorted by dist
    $NN[C] \leftarrow C_{\mathrm{NN}}$
    $DistNN[C] \leftarrow dist$
```

Algorithm 1: Modified Generic Clustering Algorithm

the observed clustering quality. The runtime improvement is particularly apparent in case of computationally cheap distances, otherwise the total runtime is typically dominated by exact distance computations.

## 3.2 Pivot Tree

Given a set of pivots $P \subset \mathcal{X}$, the lower bound from (1) can be used to determine a heuristic distance according to:

$$\widetilde{D}(C_i, C_j) = \max_{p \in P} |D(\{p\}, C_i) - D(\{p\}, C_j)| \qquad (2)$$

To increase the effectiveness of the pivots for close or NN queries Zhou and Sander proposed a pivot tree data structure [41]. The main idea behind this structure is that the heuristic distance to close objects must be more precise

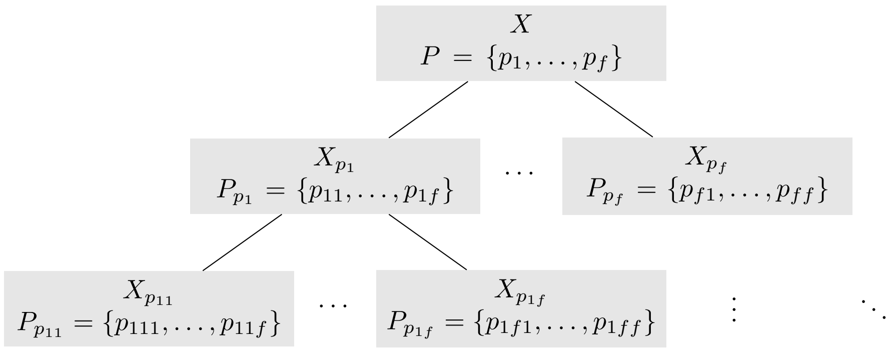

$X$
$P = \{p_1, \dots, p_f\}$
$X_{p_1}$
$P_{p_1} = \{p_{11}, \dots, p_{1f}\}$
$X_{p_f}$
$P_{p_f} = \{p_{f1}, \dots, p_{ff}\}$
$X_{p_{11}}$
$P_{p_{11}} = \{p_{111}, \dots, p_{11f}\}$
$X_{p_{1f}}$
$P_{p_{1f}} = \{p_{1f1}, \dots, p_{1ff}\}$

Figure 1: Pivot Tree

than the distance to objects that are further away in order to calculate nearest neighbors with a high approximation quality. In our case we determine our heuristic NNs with the formula:

$$\widetilde{\mathrm{NN}}(C_i) = \underset{C_j}{\operatorname{argmin}}\{\widetilde{D}(C_i, C_j)\} \tag{3}$$

The original pivot tree is a static data structure. SAHN clustering merges clusters, i.e., we delete the two clusters that are merged and create a new cluster. Therefore, we extend the data structure to allow deletions and insertions of objects (i.e., clusters in our case). Additionally we use a different strategy to calculate the heuristic distances within the pivot tree and a simplified notation.

As shown in Fig. 1 each node of the pivot tree is linked to a set of singleton clusters $X$ and a set of pivots $P \subseteq X$. The set of pivots is randomly chosen from $X$. One child node is created for each pivot. The set $X$ belongs to the root node and contains all clusters, while the set $X_{pi}$ of a child node contains all clusters 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 $\widetilde{D}(C_i, C_j)$ is performed according to (2) based on the common pivots in $P_{i\cup j}$:

$$P_{i\cup j} = \{p \in P_k \mid C_i, C_j \in X_k\}$$

It is computationally cheap to compute $P_{i\cup j}$ since we know that each cluster is present only on the 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 the 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 is reached. Each split operation creates the child nodes for a leaf node with maximum number of clusters. At the start the data structure contains only 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 distance to all pivots efficiently and therefore cannot insert a new cluster top down. However, the distances can be computed efficiently to some pivots by incremental distance updates using the Lance-Williams update formula [23]:

$$D(C_{i\cup j}, C_k) = \alpha_i \; D(C_i, C_k) + \alpha_j \; D(C_j, C_k) + \beta \; D(C_i, C_j) + \gamma \; |D(C_i, C_k) - D(C_j, C_k)| \quad (4)$$

By setting the parameters $\alpha_i$, $\alpha_j$, $\beta$ and $\gamma$ to specific values, one can obtain different linkage strategies. In our use case $C_k$ always coincides with a pivot. For the LCA of $C_i$ and $C_j$ and all ancestors in the pivot tree, we know the distance of both clusters to the nodes' pivots and we can use $\widetilde{D}$ as distance between them. For this reason, we insert the merged cluster $C_{i\cup j}$ into these nodes. This approach has the drawback that the depth of the pivot tree decreases over a series of merge operations. However, this happens relatively late in the merge process, because close clusters are merged first and are located in the same branch of the pivot tree with a high probability. Also the locality property of clusters in $X$ is not be violated, since the two clusters that are merged are relatively close to the pivot and therefore the merged cluster is also close with respect to centroid and median linkage.

## 3.3 Best Frontier Search

The best frontier searchhas been used to accelerate the OPTICS clustering algorithm in [39, 41]. We briefly describe the main idea below.

Let us assume that we only use a single pivot $p$ and all clusters are ordered by their distances to $p$. Furthermore, let $\mathrm{id}(C)$ be the index of cluster $C$ in this ordering. With respect to (2) the heuristic distances to a fixed cluster $C_i$ are monotonically increasing when we move away from $C_i$ in the ordering:

$$\forall C_i, C_j, C_k \in \mathcal{C} : \; \mathrm{id}(C_i) < \mathrm{id}(C_j) < \mathrm{id}(C_k) \vee \mathrm{id}(C_i) > \mathrm{id}(C_j) > \mathrm{id}(C_k) \Rightarrow \widetilde{D}(C_i, C_j) < \widetilde{D}(C_i, C_k)$$

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. The best frontier search solves this problem by keeping a sorted list $L_p$ of clusters for each pivot $p$. To find the position of a cluster in $L_p$ in constant time, the clusters are linked to their list elements. 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 an 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 extracted from the queue and a count is maintained 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 $C_x$ was added to the queue. The cluster $C_j$ that is first counted $|P|$ times is the heuristic NN with respect to (3). The rationale is that the lower bounds induced by the clusters retrieved from the queue are monotonically increasing. That means all lower bounds that would be obtained subsequently are greater than the heuristic distance $\widetilde{D}(C_i, C_j)$ and therefore $C_j$ must be the heuristic NN of $C_i$.

To combine the pivot tree with the best frontier search, Zhou and Sander performed a $k$-close neighbor ranking for each node of the pivot tree and joined the different close neighbors for each object afterwards. This approach is not feasible for SAHN clustering since we cannot efficiently determine which of the heuristic NNs found for each node is the best. Furthermore it is possible that the heuristic NNs are not correct with respect to (3), while the best frontier search in general can find the correct heuristic NN. For this reason we need to use a different technique which is described below.

Our integration of the best frontier search into the pivot tree runs an 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 $P_{i\cup j}$ for an arbitrary cluster 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 computed on demand and cached for the time of the NN query.

Because the asymptotic worst-case complexity of an 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. We use the terms $\widehat{\mathrm{NN}}$ and $\widehat{D}$ for the search bounded NN search and distance. Note that the search bound is also the reason for the asymmetric behavior of $\widehat{D}$.

## 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 pivots that is required to achieve a certain quality does not depend on the input size (Sect. 5), $f$ and $d$ can be considered constant and the overall construction time is linear.

As mentioned before, the modified generic clustering algorithm has a runtime of $\Omega(n\ (\log n + k + m))$ and $\mathcal{O}(n^2\ (\log n + k))$, where $k$ is the complexity of the NN search and $m$ the complexity of the merge process. The merge step consists of two deletions from and one insertion into the pivot tree. Therefore, the complexity of merge 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 an NN is bounded by $\mathcal{O}(p\ s)$, where $p$ is the number of pivot elements used 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 considered constant. It is shown in the experimental evaluation in Sect. 5 that $s$ can be chosen as a constant value, too.

The overall time complexity for HSAHN clustering is therefore bounded 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 and the number of nodes is 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. The hash table (to find the leaf nodes), the priority queue in the generic clustering algorithm and the frontier queue require $\mathcal{O}(n)$ space. Therefore the overall space requirements are linear with respect to the input size.

## 4 Graph Distance Metrics

Graphs are a versatile data structure used to represent structured objects in many application domains such as biology, chemistry or pattern recognition. To show the flexibility of our clustering algorithms based on arbitrary metric distances, we apply our algorithm to graphs. Comparing graphs and developing meaningful similarity measures between graphs is of utmost importance for almost all data mining algorithms when applied to graphs. Consequently various approaches for this task have been developed and are widely used in practice. Methods may be based on classical graph theoretical concepts like the maximum common subgraph, driven by concrete specific applications, e.g., molecular fingerprints, or developed for specific data mining algorithms like graph kernels.

We briefly present several distance metrics that have been used previously for various purposes and derive distance metrics from graph kernels. These measures have varying characteristics and differ in terms of computational complexity, dimensionality and their general notion of similarity. We discuss their applicability to our clustering algorithm before reporting on our experimental evaluation.

**Molecular Fingerprints** Molecular fingerprints are binary or numerical feature vectors, which represent graphs associated with chemical compounds. Predefined features of a molecular graph (e.g., certain substructures) are mapped to distinct positions in this vector where the number of its occurrences is stored. In case of binary fingerprints just the presence of certain features is indicated by bits. Fingerprints are used for various applications in cheminformatics including clustering and are often compared with computationally cheap distance measures.

The Tanimoto coefficient is most commonly used for chemical compound comparisons based on binary fingerprints [16]. It is equivalent to the Jaccard coefficient when fingerprints are represented by sets. For two binary fingerprints $F_1$ and $F_2$ it is defined as

$$\textsc{Tanimoto}(F_1, F_2) = \frac{N_{11}}{N_{11} + N_{10} + N_{01}}, \tag{5}$$

where $N_{11}$ refers to the number of shared on bits, while $N_{10}$ and $N_{01}$ refer to exclusive on bits in $F_1$ and $F_2$, respectively. The Tanimoto coefficient is a normalized similarity measure in the interval $[0, 1]$. The distance $1-\textsc{Tanimoto}(F_1, F_2)$ is a metric [24].

Fingerprints are commonly used to cluster molecules, because of the cheap distance computations. Furthermore, the fingerprints can be precomputed and reused for different clustering settings. Because of their fixed feature set, they are less versatile than graph kernel distances. However, the features can be selected carefully for a specific task and imply an interpretable way to abstract from the complex graph structure.

**Maximum Common Subgraph Distances** Given two graphs $G$ and $H$, the maximum common induced subgraph problem (MCIS) is to determine the size (in terms of vertices) of the largest graph that is isomorphic to an induced subgraph of $G$ and $H$. In case of labeled graphs the isomorphism between the subgraphs typically not only has to preserve adjacencies, but also vertex and edge labels. Note that the common subgraph is not necessarily connected. Furthermore, multiple common subgraphs of the same maximum size may exist. In many applications it is beneficial to consider arbitrary subgraphs (without isolated vertices) and to quantify the size of graphs in terms of edges (MCES). Both variants of the problem are known to be $\mathcal{NP}$-complete and the computation is demanding even for small graphs in practice [8].

Several metrics for graphs have been proposed that take the size of a maximum common subgraph into account:

$$d(G,H) = 1 - \frac{|\textsc{Mcs}(G,H)|}{\max\{|G|,|H|\}} \qquad \text{BS [5]} \tag{6}$$

$$d(G,H) = 1 - \frac{|\textsc{Mcs}(G,H)|}{|G|+|H|-|\textsc{Mcs}(G,H)|} \qquad \text{WSKR [36]} \tag{7}$$

$$d(G,H) = |G|+|H|-2|\textsc{Mcs}(G,H)| \qquad \text{FV [14]} \tag{8}$$

These distances can be combined with both definitions of maximum common subgraph and we refer to (6), for example, either by MCIS-BS or MCES-BS to distinguish between both variants. In our experimental evaluation below we focus on the distance FV (8), which has been proposed on several occasions and was shown to yield a metric for different definitions of common subgraphs [14, 10]. The size of a maximum common subgraph is closely related to the graph edit distance [4] and (8) actually corresponds to the minimum number of edit operations to turn one graph into the other. For example, one may first delete the parts of $G$ that do not belong to the maximum common subgraph and than add parts of the other graph to finally obtain $H$, where deletion and addition of either vertices or edges is considered as edit operation.

Since we want to combine this distance with centroid linkage one may ask for the intuition of a centroid under this distance measure. Given two graphs $G$ and $H$, let $C$ be a maximum common subgraph of $G$ and $H$. It is easy to verify that $d(G,H) = 2d(G,C) = 2d(H,C)$ suggesting that one way to think of a centroid is the common subgraph.

**Graph Kernel Distances** Various kernels for graphs have been proposed in recent years, see [35] and references therein, which allow to apply the wealth of so-called *kernel methods*, including Support Vector Machines as the most prominent example, to graphs. A graph kernel is a function $k : \mathcal{G} \times \mathcal{G} \to \mathbb{R}$ on the set of graphs $\mathcal{G}$ that can be expressed as a dot product $k(G,H) = \langle \phi(G), \phi(H) \rangle$, where $\phi : \mathcal{G} \to \mathcal{H}$ is a mapping into a *feature space* $\mathcal{H}$. Although a kernel may be efficiently computable, a corresponding feature space may be of high or even indefinite dimension. Therefore, it may be prohibitive to explicitly map elements into feature space. In this respect kernels substantially differ from vector representations like molecular fingerprints. However, a distance metric in feature space can be computed in terms of kernel values avoiding such a mapping. We determine the Euclidean distance in the feature space according to

$$d(G,H) = \sqrt{\sum_{i=1}^{n} (g_i - h_i)^2} = \sqrt{k(G,G) + k(H,H) - 2k(G,H)}, \tag{9}$$

where $(g_1, \dots, g_n)^T = \phi(G)$ and $(h_1, \dots, h_n)^T = \phi(H)$. Therefore, we can easily derive metrics from arbitrary graph kernels. We employed graph kernels based on random walks of fixed length (FLRW), which basically count the common walks contained in both graphs, where each walk is associated with the sequence of vertex and edge labels encountered on the walk. Kernels based on random walks are widely used and efficiently computable [18, 35]. We have implemented an algorithm based on product graphs, which is similar to the approach described in [18], and consider walks of length 4. In addition we used Weisfeiler-Lehman graph kernels [33] that are scalable on large graphs with discrete labels and were shown to perform well in classification tasks. Different kernels based on the classical method for isomorphism testing have been developed and we have implemented the Weisfeiler-Lehman subtree (WL-ST) and shortest path (WL-SP) kernel. Both kernels perform two iterations of label refinement and WL-SP requires shortest-path distances to match exactly.

In contrast to maximum common subgraph distances, (9) as well as the distance based on fingerprints yield metrics in a feature space, but in general a pseudo metric for graphs, since $d(G,H) = 0$ may also hold for two non-isomorphic graphs $G$ and $H$. Nevertheless, such distances can be useful for specific tasks and can be used for clustering graphs with our approach.

## 5 Experimental Results

In this section we 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 GB 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.

**Data Sets.** We used a real-world as well as various synthetic data sets[2] for the evaluation.

The real-world data set (SARFari kinase) stems from the ChEMBL[3] database and contains a set of $\approx 50\,000$ molecules. The Euclidean distance, Tanimoto coefficient, various kernels and various maximum common subgraph distances were utilized for this data set. The Euclidean distance was applied on 5 numerical properties of the molecules. The Tanimoto distance was applied to Daylight bit fingerprints (1024 bits), which were generated with the chemistry development kit[4].

For the above data set, quality was measured by comparing the exact and the heuristic results. This method always suffers from the possibility that there might be two different clusterings that represent some ground truth equally well. If these alternative clusterings differ a lot, the comparison suggests a bad clustering quality for the heuristic approach. To overcome this evaluation weakness, we generated a data set of connected graphs hierarchically, such that we were able to compare the generated hierarchy (ground truth) with the cluster hierarchy. The generation of this data set is based on random edge extensions and is described below. We started with an empty graph (the root node in the hierarchy) and extended this graph $m$ times with $o$ edges, such that we get $m$ graphs with $o$ edges on the second level of the hierarchy. We proceeded with the leaf nodes and randomly extended each of them. This procedure was repeated until a certain depth was reached. The labels of the vertices and edges were chosen randomly from a set of labels with size $q$ and the probability for a backward extensions (connecting two existing vertices) was set to 0.7. An obvious choice of the parameter $m$ seems to be 2, since the clustering result is a binary tree. However, choosing $m = 2$ is problematic, because the data set diversity is low (i.e., cluster separation is low on medium to low levels of the hierarchy). Adding more edges in one step (i.e., incrementing $o$) to overcome this issue for $m = 2$ is also not practicable, since the graphs become too large to compute the distances between them in a reasonable amount of time for meaningful data set sizes. Furthermore, the presence of more than two clusters at the same level is common in real world data. For this reasons, we chose $m = 6, o = 3$ and a depth of 4 to create the data set. The hierarchy was than

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

made binary, by replacing each node by a random binary tree with $m$ leaves. To further reduce the graph size, we chose $o = 1$ for the last level of extensions. For the performance evaluations we used $m = 10$ to generate data sets larger than 1024.

In addition, we used two synthetic 5 dimensional Euclidean data sets to analyze the impact of the clusterability on the quality of the resulting clustering. The first contained uniformly distributed data and the second contained 32 normally distributed and well-separated clusters.

**Test Setup.** The quality measurements were done by comparing each level of the HSAHN results either with the exact results or the generated hierarchy which represents the ground truth. The Fowlkes Mallows Index (FMI) [17] and the Normalized Variation of Information (NVI) [25] measurements were employed to measure each single level. The FMI is based on a counting pair approach, i.e., it counts the pairs of objects, that are in the same cluster or in different clusters in both clusterings. The NVI is an information theoretic measure, that uses only the cluster sizes for comparison. We observed that the FMI produces more differentiated results for different settings than NVI. On the other hand NVI is less biased by the number of clusters and was therefore utilized as a alternative measure.

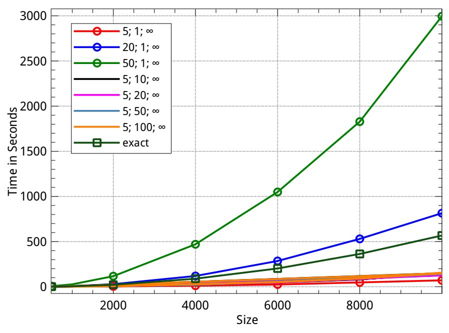
Please note that, to our knowledge, there exist no other competing heuristic SAHN algorithms for general metric space supporting centroid or median linkage. Therefore, we compared our algorithm with state of the art exact algorithms in the case of performance measurements and cannot compare the quality with competing approaches.

In addition to the runtime and quality measurements, we calculated the intrinsic dimensionality [6] for each data set in combination with each distance measure. The intrinsic dimensionality is defined as

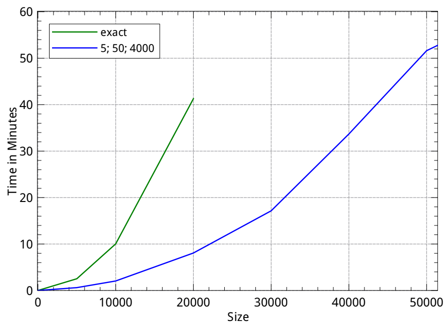
$$\rho = \frac{\mu^2}{2\ \sigma^2},$$

where $\mu$ is the average distance and $\sigma$ is the standard deviation. It plays an important role in any pivot approach, as the number of pivots required to approximate a distance with a certain quality increases with the dimensionality of the data set. Since the data points in an $n$-dimensional vector space might be embeddable into a much lower dimensional space preserving all pairwise distances, it is important to distinguish between the vector space dimension and the intrinsic dimension.

All heuristic test results were averaged over three runs, because the random selection of pivots leads to a non-deterministic behavior of the algorithm. Without an exception the differences were very small and the results were stable over different runs. In 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.



(a) Unbounded Search Depth



(b) Complete Kinase Data Set

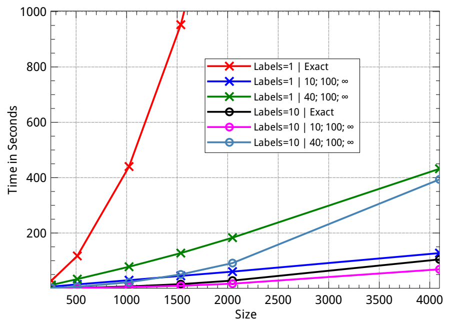
Figure 2: Runtime for Tanimoto Distance: Kinase Data Set

**Runtime.** The runtime behavior differs for different distance measures. For computationally demanding distance measures and data sets of feasible size, the runtime is dominated by the initialization of the pivot tree data structure. For computationally cheap distance measures the runtime is dominated by the clustering process itself. We will focus on computationally cheap distance measures to investigate the performance of the best frontier search and advance to more complex distances afterwards.

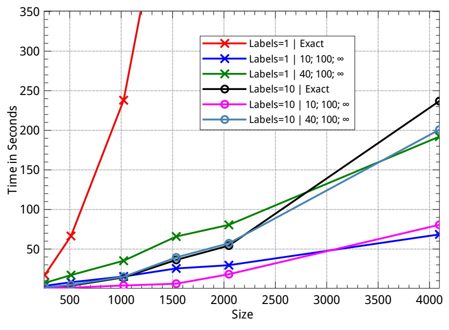
As shown in Fig. 2(a) the performance of the algorithm scales linearly with the number of pivots if we use the computationally cheap Tanimoto distance. For the unbounded search depth the empirical runtime behavior is clearly quadratic and the absolute runtime for a high pivot count even exceeds the runtime of the exact algorithm. It is noteworthy that the number of leaves in the pivot tree does not have a major influence on the overall performance. If we limit the search depth and use a reasonable set of parameters (the rationale follows in the quality evaluation paragraph) the time to cluster the whole kinase data set (Fig. 2(b)) is much lower than in the exact case. The heuristic curve flattens at the higher levels and the observed behavior is slightly subquadratic.

The evaluation on the kinase data set also shows the memory advantage of our algorithm for larger data sets, since we were unable to cluster 30 000 structures with the exact algorithm due to memory constraints. On the other hand, the heuristic clustering algorithm used less than 1 GB of memory to cluster the whole kinase data set.

Figures 3(a) and 3(b) show the runtime of the exact and the heuristic algorithm for generated hierarchical data sets (1 and 10 labels). The heuristic algorithm was run with two different settings for the number of pivots and an unbounded search depth. As previously expected, the use of computationally demanding distances hurts the runtime of the exact algorithm a lot more than the runtime of the heuristic algorithm. In an interactive environment, where fast response times are essential, even clustering small data sets (with a size $\approx$ 4000) with the exact algorithm can be prohibitive. On the other hand the



(a) MCS-FV



(b) Kernel-FLRW

Figure 3: Runtime for Kernel and MCS Distances: Hierarchy Data Set

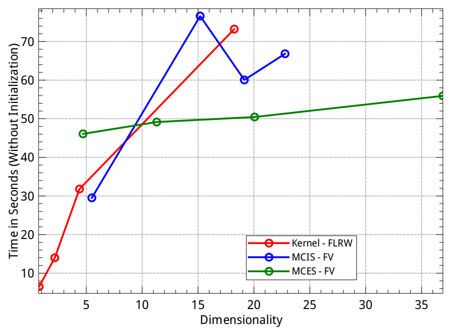


Figure 4: Influence of the Dimensionality on the Clustering Runtime

heuristic clustering is up to 60 times faster and this speed up increases with the data set size. The number of labels used for the generation of the data set has a huge impact on the clustering performance. The higher the number of labels the less demanding is a single distance computation. It is remarkable that for the heuristic clustering, this does not generally lead to a lower overall runtime. Figures 3(a) and 3(b) show a curve that is superlinear for the 10-label settings and a near to linear curve for the 1-label settings, where the linear number of distance computations dominates the overall runtime. We can therefore expect that the 10-label performance will get worse for larger data sets compared to the 1-label performance.

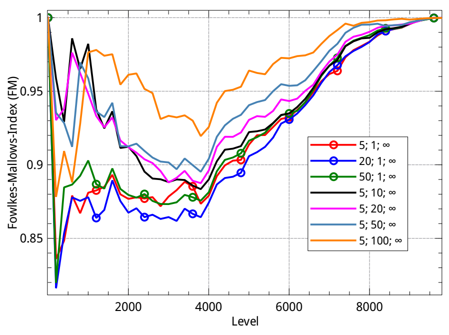
This observation is explainable if we consider the effect of the intrinsic dimensionality of the data sets, cf. Table 1, on the runtime behavior. With increasing dimensionality also the probability for a pivot to give a bad approximation of the real distance increases. Furthermore, with an increasing number of pivots, that are not able to approximate the distance well, the number of false elements that will be processed by the best frontier search before the heuristic NN is found increases.

As a consequence, the search depth as well as the number of pivots have

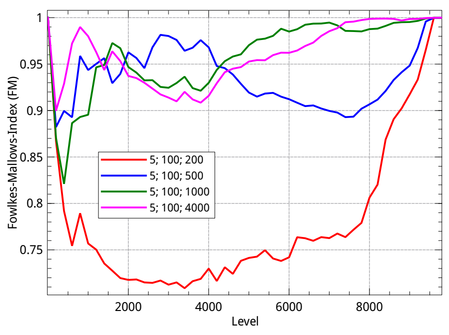
| Label | MCIS-FV | MCES-FV | MCIS-BS | MCES-BS | MCIS-WSKR |
|---|---|---|---|---|---|
| 1 | 4.52 | 3.98 | 4.31 | 4.74 | 7.78 |
| 2 | 13.08 | 7.82 | 12.65 | 9.12 | 29.15 |
| 3 | 15.73 | 16.16 | 17.00 | 12.95 | 36.77 |
| 10 | 18.45 | 20.52 | 23.64 | 20.64 | 47.38 |

| Label | MCES-WSKR | Kernel-FLRW | Kernel-WL-ST | Kernel-WL-SP |
|---|---|---|---|---|
| 1 | 8.28 | 0.80 | 9.88 | 6.44 |
| 2 | 20.36 | 1.96 | 39.75 | 9.93 |
| 3 | 37.12 | 6.54 | 40.00 | 14.95 |
| 10 | 46.07 | 15.94 | 75.43 | 38.59 |

Table 1: Dimensionality of the Hierarchy Data Set



(a) Quality: Number of Pivots

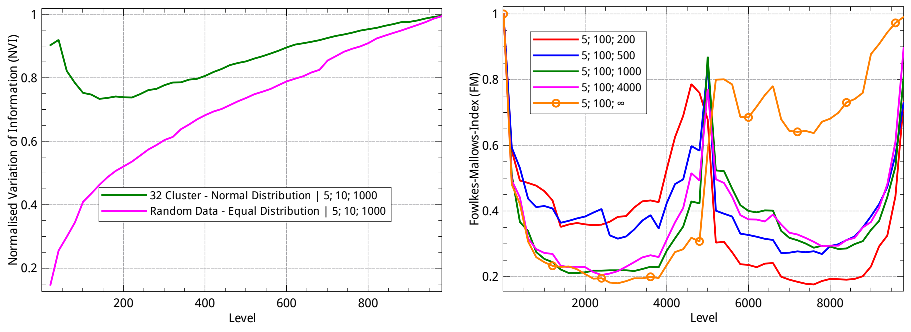


(b) Quality: Search Depth

Figure 5: Comparing the Exact and the Heuristic Results: Kinase Data Set; Euclidean Distance (5 Dimensions)

to be increased for high dimensional data sets. Even if we keep the number of pivots fixed, the runtime of an unbounded best frontier search will increase. This effect is shown in Fig. 4. We generated four fixed sized data sets with $1, 2, 3$ and 10 labels. We then plotted the dimensionality and the runtime of the clustering for a selected set of distances. We left out the runtime of the initialization in this plot, to be independent of the computational complexity of the distance measure and show only the effect of the structure of the data on the clustering performance. While there is a trend that the runtime increases with the dimensionality, the plot also shows that there are other factors influencing the runtime. Furthermore, the runtime is influenced by the dimensionality to a different degree depending on the distance measure.

**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.



(a) Synthetic Data Sets (b) Ambiguity: Kinase / Tanimoto

Figure 6: Comparing the Exact and the Heuristic Results: Influence of the Data Distribution

However, our test results do not show significant differences in the quality when the number of pivots exceeds a certain threshold. From our observations the main aspect that influences this threshold is the structure of the data and not the input size. The number of pivots can therefore be seen as a constant with respect to the asymptotic runtime behavior. Figure 5(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 at least 10 times higher than the runtime of the setting $(5; 100; \infty)$.

**Quality - Search Depth and Ambiguity.** The search depth limits the runtime of the best frontier search. Therefore it is very important to know, if the search depth can be sublinear in 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 NVI 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 these 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 these levels are strongly influenced by the starting situation where both clusterings contain only singletons.

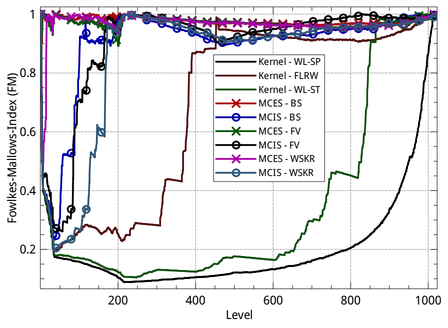
The experimental evaluation showed that the search depth can be chosen about 500 in the low dimensional Euclidean space, see Fig. 5(b). Lower values significantly harm the quality of the results. A much higher search depth (e.g. $\approx 5000$ for MCES-FV and MCIS-FV) has to be chosen for the maximum common subgraph and the Weisfeiler-Lehman graph kernel distances using the hierarchical data sets. We observed that the search depth is sensitive to the number of pivots $|P|$ and the dimensionality of the data. The first observation is not surprising, since the best frontier search loop is stopped after the first object is counted $|P|$ times. The second observation can be explained by the already mentioned higher probability for a pivot to lay on a disadvantageous position relative to two objects. Furthermore the distribution of the distances plays an important role as the pairwise objects tend to be equidistant in high dimensional space. Lower deviation in the pairwise distances means that even a small deviation in the lower distance bound (1) results in a higher probability, that an object is falsely retrieved from the frontier.

This observation also explains why the quality of the Tanimoto distance (Fig. 6(b)) is lower than the quality of the Euclidean distance and why a limit on 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 data set size of 10 000 and 1024 bits 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 unstable, even if one is comparing two exact algorithms.

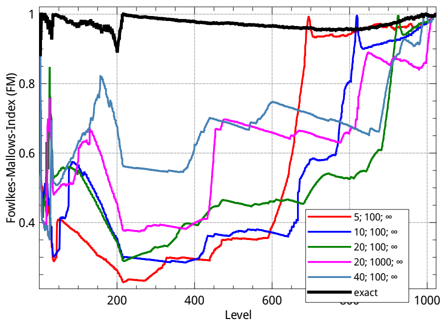
The peak in Fig. 6(b) at level 5 000 is also explainable when we look at Fig. 6(a). If the data set contains well-separated clusters, the quality is good at exactly the level which corresponds to the number of clusters. This implies that we are able to identify real clusters in a data set with the HSAHN algorithm albeit the clustering quality might not be very well over all levels. This conclusion is very important because it establishes the practicability of our approach.

**Quality - Maximum Common Subgraph and Kernel Distances** As mentioned earlier, the alternative clusterings of the same quality are not adequately scored when compared to the exact results. The quality measurements considered here will therefore compare the results with the generated ground truth. The problem with ambiguous merge operations still remains for some levels, but for the levels which contain a real cluster structure we can expect a high quality. We will give a short example to make this aspect clear. Let us consider a binary tree with four leaves. The root node has a dissimilarity of two and both children have a dissimilarity of one. It is not defined which of the children should be merged first. The two left leaves and the two right leaves have the same dissimilarity and therefore the ordering is arbitrary. That means, we can expect a high quality score for the second level, while there might be a lower quality score for the third level.

The number of labels we use for the data set generation does also have an influence on the possibility to reconstruct the hierarchy unambiguously. The less labels we use for the data set generation the higher is the probability, that we randomly create the same (sub-)graph in two different branches of our hierarchy. It is impossible to distinguish between these graphs in the latter recreation of the hierarchy by the clustering algorithm and we might add a subtree into the wrong branch. Therefore, we can expect a higher clustering quality for a higher label count for the exact algorithm. As the intrinsic dimensionality increases with the label count, the last aspect might not be observable using the heuristic clustering.



(a) Exact SAHN Clustering



(b) MCES-FV: All Settings

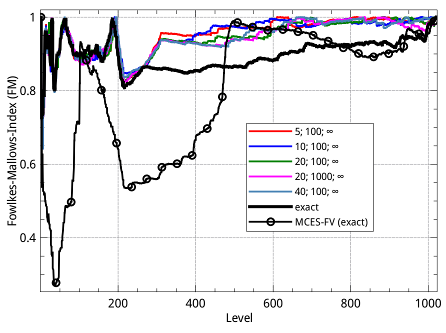
Figure 7: Comparison to Ground Truth: 10 Labels

Another important aspect of this part of the evaluation is to demonstrate, which maximum common subgraph distances and which kernel distances are suitable for centroid linkage and SAHN clustering in general and which distances are a good choice for our heuristic approach.

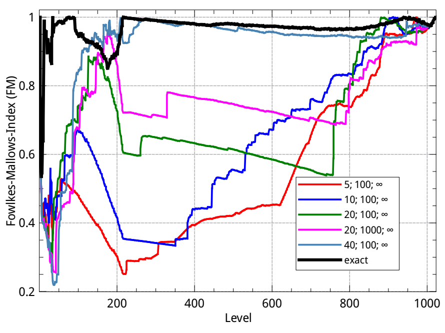
Figure 7(a) shows the quality of the exact SAHN algorithm for different distances and 10 labels. We can see that the maximum common subgraph distances consistently have a higher quality than the kernel distances. The Weisfeiler-Lehman graph kernels seem to be incompatible with centroid linkage or are not suitable for the specific clustering task. Only the fixed length random walk kernel shows some evidence, that the first level of the hierarchy was merged correctly, but shows a bad quality for the levels 0 to 400. On the other hand, the maximum common subgraph distances performed very well. Some were able to perfectly recreate the hierarchy aside from the ambiguity between the hierarchy levels explained above. It is conspicuous, that the MCES distances performed better than MCIS distances. The different ways to calculate the distance from the maximum common subgraph does not seem to have a huge influence. This behavior can be explained given the way we generated the data set. We added random edges, and therefore the dissimilarities should correspond to the hierarchy level of the LCA if we use the MCES measures. Therefore, it cannot be said that the MCIS distances perform any less in general.

Figure 7(b) shows the quality using HSAHN for the MCES-FV distance and 10 labels. The plot demonstrates, that a high number of pivots is needed for a good quality, but HSAHN is able to recreate the hierarchy with some deviation. MCES-BS performed similar to MCES-FV, while MCES-WSKR performed poor for the Heuristic clustering. These observations are in line with the intrinsic dimensionality from Table 1.

If we use less labels, the heuristic and the exact algorithms produce similar results (see Fig. 8). On the one hand, Fig. 8(b) met our expectations: the quality of the exact algorithm is slightly below the 10-label setting and the quality for the heuristic algorithm rises because of the lower dimensionality.



(a) Kernel-FLRW: 1 Label



(b) MCES-FV: 3 Label

Figure 8: Comparison to Ground Truth: 1 and 3 Labels

On the other hand it was surprising for us that the fixed length random walk kernel performed so well for unlabeled (i.e., 1 label) graphs (see Fig. 8(a)). It outperformed the MCES-FV distance measure in the exact and the heuristic case. The heuristic algorithm performed well, regardless of number of pivots used. In the lower levels the quality of the heuristic algorithm was even higher than the quality of the exact algorithm. We have some evidence, that these results can be transfered to other data sets as well, because we observed similar results regarding the different distance measures for the exact vs. heuristic comparisons on the real world data set.

## 6 Conclusions

Our tests have shown that the HSAHN algorithm can greatly increase the size of data sets which can be clustered in a reasonable amount of time. In addition, the memory usage is lowered dramatically, which often sets a hard limit on the data set size using exact algorithms. The linear dependence on exact distance calculations makes it possible to use computationally expensive distance measures even on large data sets. For the applicability of our algorithm, it is important to understand the influence of the algorithm parameters. In Sect. 5 we have discussed the selection of adequate parameters for the clustering in detail.

In addition to proposing the HSAHN algorithm, we have analyzed which graph distances are suitable for SAHN and HSAHN clustering in combination with centroid linkage and demonstrated the huge impact of the intrinsic dimensionality on the HSAHN results. We have shown that our algorithm is able to recreate the planted cluster structures and therefore is applicable to graph databases. Based on our experimental study, it seems advisable to use the FLRW kernel for unlabeled data and the MCIS-FV or MCES-FV distance for labeled graphs.

A starting point for further improvement is the calculation of the distances between the merged clusters and the pivots. Our approach to employ the Lance-Williams update formula based on heuristic distances has been shown to be convenient, but results may be biased by slight heuristic distance errors, which then accumulate over a series of merge operations. For the application of HSAHN on graph databases, it might be possible to calculate maximum common subgraphs, minimum common supergraphs or median graphs as representatives. We can use this representatives for exact distance calculations to the pivots. In this case it is also possible to insert the new cluster top down into the pivot tree data structure. Doing so, will only double the amount of exact distance calculations. However, there are some problems to solve, e.g., there might exist more than one common subgraph. Keeping all of them would require some extra distance computations and the need to aggregate the different distances to a common metric value.

Another aspect that should be investigated in future work, is the local splitting of the pivot tree. The hard boundaries of the local areas can lead to the situation, that only a few pivots are used for close clusters from different areas. It might be sufficient to use more than one pivot tree at the same time, because the random sampling strategy implies random borders. However, we are unsure about the effect this problem has in practice and there might be more efficient strategies to overcome this problem.

When clustering huge datasets with SAHN methods the visual representation becomes difficult, especially in terms of information overload. Therefore we suggest to use more flexible and dynamic representations of the dendrogram as in [7]. The HSAHN algorithm was integrated in the software Scaffold Hunter [37, 20], a tool for the analysis and exploration of chemical space, and has been proven a valuable alternative to exact approaches in practice.

## References

[1] M. R. Anderberg. *Cluster Analysis for Applications.* Academic Press, 1973.

[2] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander. OPTICS: Ordering Points To Identify the Clustering Structure. *SIGMOD Rec.*, 28(2):49–60, 1999. doi:10.1145/304181.304187.

[3] M. M. Breunig, H.-P. Kriegel, P. Kröger, and J. Sander. Data bubbles: quality preserving performance boosting for hierarchical clustering. *SIGMOD Rec.*, 30(2):79–90, 2001. doi:10.1145/376284.375672.

[4] H. Bunke. On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters*, 18(8):689–694, 1997. doi:10.1016/S0167-8655(97)00060-3.

[5] H. Bunke and K. Shearer. A graph distance metric based on the maximal common subgraph. *Pattern Recognition Letters*, 19(3-4):255–259, 1998. doi:10.1016/S0167-8655(97)00179-7.

[6] E. Chávez and G. Navarro. A probabilistic spell for the curse of dimensionality. In *ALENEX*, pages 147–160, 2001. doi:10.1007/3-540-44808-X_12.

[7] J. Chen, A. MacEachren, and D. Peuquet. Constructing overview + detail dendrogram-matrix views. *Visualization and Computer Graphics, IEEE Transactions on*, 15(6):889–896, Nov 2009. doi:10.1109/TVCG.2009.130.

[8] D. Conte, P. Foggia, and M. Vento. Challenging complexity of maximum common subgraph detection algorithms: A performance analysis of three algorithms on a wide database of graphs. *J. Graph Algorithms Appl.*, 11(1):99–143, 2007. doi:10.7155/jgaa.00139.

[9] W. H. Day and H. Edelsbrunner. Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of Classification*, 1(1):7–24, 1984. doi:10.1007/BF01890115.

[10] L. De Raedt and J. Ramon. Deriving distance metrics from generality relations. *Pattern Recogn. Lett.*, 30(3):187–191, Feb. 2009. doi:10.1016/j.patrec.2008.09.007.

[11] G. M. Downs and J. M. Barnard. *Clustering Methods and Their Uses in Computational Chemistry*, pages 1–40. John Wiley & Sons, Inc., New Jersey, USA, 2003. doi:10.1002/0471433519.ch1.

[12] C. Elkan. Using the triangle inequality to accelerate k-means. In *Proceedings of the 20th International Conference on Machine Learning*, ICML '03, pages 147–153, Menlo Park, California, USA, 2003. AAAI Press.

[13] D. Eppstein. Fast hierarchical clustering and other applications of dynamic closest pairs. *J. Exp. Algorithmics*, 5(1), 2000. doi:10.1145/351827.351829.

[14] M.-L. Fernàndez and G. Valiente. A graph distance metric combining maximum common subgraph and minimum common supergraph. *Pattern Recognition Letters*, 22(67):753 – 758, 2001. doi:10.1016/S0167-8655(01)00017-4.

[15] M. Ferrer, E. Valveny, F. Serratosa, I. Bardají, and H. Bunke. Graph-based *k*-means clustering: A comparison of the set median versus the generalized median graph. In *CAIP*, pages 342–350, 2009. doi:10.1007/978-3-642-03767-2_42.

[16] M. A. Fligner, J. S. Verducci, and P. E. Blower. A modification of the jaccardtanimoto similarity index for diverse selection of chemical compounds using binary strings. *Technometrics*, 44(2):110–119, 2002. doi:10.1198/004017002317375064.

[17] E. B. Fowlkes and C. L. Mallows. A method for comparing two hierarchical clusterings. *Journal of the American Statistical Association*, 78(383):553–569, 1983. doi:10.2307/2288117.

[18] Z. Harchaoui and F. Bach. Image classification with segmentation graph kernels. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1 –8, june 2007. doi:10.1109/CVPR.2007.383049.

[19] B. Jain and K. Obermayer. Elkans *k*-means algorithm for graphs. In G. Sidorov, A. Hernández Aguirre, and C. A. Reyes García, editors, *Advances in Soft Computing*, volume 6438 of *Lecture Notes in Computer Science*, pages 22–32. Springer Berlin Heidelberg, 2010. doi:10.1007/978-3-642-16773-7_2.

[20] K. Klein, O. Koch, N. Kriege, P. Mutzel, and T. Schäfer. Visual analysis of biological activity data with scaffold hunter. *Molecular Informatics*, 32(11-12):964–975, 2013. doi:10.1002/minf.201300087.

[21] H. Koga, T. Ishibashi, and T. Watanabe. Fast agglomerative hierarchical clustering algorithm using locality-sensitive hashing. *Knowledge and Information Systems*, 12(1):25–53, 2007. doi:10.1007/s10115-006-0027-5.

[22] N. Kriege, P. Mutzel, and T. Schäfer. SAHN clustering in arbitrary metric spaces using heuristic nearest neighbor search. In S. P. Pal and K. Sadakane, editors, *WALCOM*, volume 8344 of *Lecture Notes in Computer Science*, pages 90–101. Springer, 2014. doi:10.1007/978-3-319-04657-0_11.

[23] G. N. Lance and W. T. Williams. A general theory of classificatory sorting strategies 1. hierarchical systems. *The Computer Journal*, 9(4):373–380, 1967. doi:10.1093/comjnl/9.4.373.

[24] A. Lipkus. A proof of the triangle inequality for the tanimoto distance. *Journal of Mathematical Chemistry*, 26(1):263–265, Oct. 1999. doi:10.1023/A:1019154432472.

[25] M. Meilă. Comparing clusterings—an information based distance. *JMVA*, 98(5):873–895, 2007.

[26] D. Müllner. Modern hierarchical, agglomerative clustering algorithms. arXiv:1109.2378v1, 2011.

[27] F. Murtagh. Multidimensional clustering algorithms. In *COMPSTAT Lectures 4*, Wuerzburg, 1985. Physica-Verlag.

[28] F. Murtagh and P. Contreras. Algorithms for hierarchical clustering: an overview. *WIREs Data Mining Knowl Discov*, 2(1):86–97, 2012. doi:10.1002/widm.53.

[29] M. Nanni. Speeding-up hierarchical agglomerative clustering in presence of expensive metrics. In T. Ho, D. Cheung, and H. Liu, editors, *Advances in Knowledge Discovery and Data Mining*, volume 3518 of *Lecture Notes in Computer Science*, pages 378–387. Springer Berlin Heidelberg, 2005. doi:10.1007/11430919_45.

[30] B. K. Patra, N. Hubballi, S. Biswas, and S. Nandi. Distance based fast hierarchical clustering method for large datasets. In *Proceedings of the 7th international Conference on Rough Sets and Current Trends in Computing*, RSCTC'10, pages 50–59. Springer-Verlag, 2010. doi:10.1007/978-3-642-13529-3_7.

[31] F. J. Rohlf. Hierarchical clustering using the minimum spanning tree. *Computer Journal*, 16:9395, 1973.

[32] M. Seeland, T. Girschick, F. Buchwald, and S. Kramer. Online structural graph clustering using frequent subgraph mining. In J. L. Balczar, F. Bonchi, A. Gionis, and M. Sebag, editors, *Machine Learning and Knowledge Discovery in Databases*, volume 6323 of *Lecture Notes in Computer Science*, pages 213–228. Springer Berlin Heidelberg, 2010. doi:10.1007/978-3-642-15939-8_14.

[33] N. Shervashidze, P. Schweitzer, E. van Leeuwen, K. Mehlhorn, and K. Borgwardt. Weisfeiler-Lehman graph kernels. *Journal of Machine Learning Research*, 12:2539–2561, 2011.

[34] K. Tsuda and T. Kudo. Clustering graphs by weighted substructure mining. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 953–960, New York, NY, USA, 2006. ACM. doi:10.1145/1143844.1143964.

[35] S. V. N. Vishwanathan, N. N. Schraudolph, R. I. Kondor, and K. M. Borgwardt. Graph kernels. *Journal of Machine Learning Research*, 11:1201–1242, 2010.

[36] W. D. Wallis, P. Shoubridge, M. Kraetzl, and D. Ray. Graph distances using graph union. *Pattern Recognition Letters*, 22(6/7):701–704, 2001.

[37] S. Wetzel, K. Klein, S. Renner, D. Rauh, T. I. Oprea, P. Mutzel, and H. Waldmann. Interactive exploration of chemical space with Scaffold Hunter. *Nature Chemical Biology*, 5(8):581–583, 2009. `doi:10.1038/nchembio.187`.

[38] P. Zezula, G. Amato, V. Dohnal, and M. Batko. *Similarity Search: The Metric Space Approach*, volume 32 of *Advances in Database Systems*. Springer, 2006.

[39] J. Zhou. *Efficiently Searching and Mining Biological Sequence and Structure Data*. PhD thesis, University of Alberta, 2009.

[40] J. Zhou and J. Sander. 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*, VLDB '03, pages 452–463. VLDB Endowment, 2003.

[41] J. Zhou and J. Sander. Speedup clustering with hierarchical ranking. In *Data Mining, 2006. ICDM '06. Sixth International Conference on*, pages 1205–1210, 2006. `doi:10.1109/ICDM.2006.151`.