An objective function for order preserving hierarchical clustering

Daniel Bakkelund
11th December 2024

Abstract

We present a theory and an objective function for similarity-based hierarchical clustering of probabilistic partial orders and directed acyclic graphs (DAGs). Specifically, given elements $x \leq y$ in the partial order, and their respective clusters $[x]$ and $[y]$, the theory yields an order relation $\leq'$ on the clusters such that $[x] \leq' [y]$. The theory provides a concise definition of order-preserving hierarchical clustering, and offers a classification theorem identifying the order-preserving trees (dendrograms). To determine the optimal order-preserving trees, we develop an objective function that frames the problem as a bi-objective optimization, aiming to satisfy both the order relation and the similarity measure. We prove that the optimal trees under the objective are both order-preserving and exhibit high-quality hierarchical clustering. Since finding an optimal solution is NP-hard, we introduce a polynomial-time approximation algorithm and demonstrate that the method outperforms existing methods for order-preserving hierarchical clustering by a significant margin.

1 Introduction

Clustering is one of the oldest and most popular techniques for exploratory data analysis and classification, and methods for hierarchical clustering dates back almost a century. While clustering methods for graphs and other types of structured data are common, it is also common that the structure is lost during the clustering process; the clustering does not retain the structure of the original data. Alternatively, the methods that do retain the structure of the data are not easily combinable with a general notion of similarity to influence the quality of the individual clusters. In this article, we show how to do order preserving hierarchical clustering for partially ordered data that is equipped with a measure of similarity between elements. The goal is to produce hierarchical clusterings where the clusters themselves are ordered to retain the original ordering of the data, while simultaneously producing high quality clusters.

The objective function we present is an extension of the objective function for similarity based hierarchical clustering by Dasgupta (2016), where we introduce an additional component representing the level of comparability according to the partial order. This allows us to optimise on similarity and order preservation simultaneously, obtaining a hierarchical clustering that balances the two objectives.

While several works exist that perform order preserving clustering, and although some of these are also hierarchical in nature (Herrmann et al., 2019), there exists, to the authors’ knowledge, no attempts to provide a general definition of the concept. As foundation for the presented objective, we therefore suggest a formal definition of order preserving hierarchical clustering motivated by order theory and classical hierarchical clustering.
Since optimisation turns out NP-hard, we provide a polynomial time approximation algorithm with a relative performance guarantee of $O(\log^{3/2} n)$. We close the paper with a demonstration of the efficacy of the approximation on the benchmark dataset (Bakkelund, 2021b), showing that the provided theory outperforms other methods with a large margin.

The main contributions from this paper are:

- A formal theory for order preserving hierarchical clustering.
- An objective function for order preserving hierarchical clustering of partially ordered sets equipped with a similarity or dissimilarity.
- A polynomial time algorithm approximating the objective function, with a relative performance guarantee of $O(\log^{3/2} n)$.

1.1 Motivating use case

The theory described in this paper is inspired by an industry database of machine parts. In the database, the machine parts are registered in part-of relations, so that a part has a reference to its containing part, according to the design. The result is a family of part trees where the nodes in the trees are parts that consist of parts that consist of parts and so on.

Over time, some designs have been copied with small alterations, and due to business reasons, all the parts of the new structure have been given new identifiers, with no reference to where from it was copied. In hindsight, it is desirable to discern which parts are equivalent to which other parts, to allow for, for example, spare part interchange across designs. We therefore seek a classification of part types into classes of interchangeable types.

Such a classification has to take into account the part-of relations: As an example, assume that $a$, $b$, $c$ and $d$ are machine parts, and that $a$ is a part of $b$ and $c$ is a part of $d$, as illustrated in Figure 1.1. If we classify $a$ and $c$ as interchangeable (Figure 1.2), then we also say that $a$ can be a part of $d$ and $c$ can be a part of $b$. It also implies that $a$ and $d$ cannot be interchanged, since you cannot replace a part by a sub-part. Figure 1 presents a selection of classifications of these four elements, together with possible interpretations of what the classifications mean in the domain of part-of relations.

While each piece of machinery constitutes a tree of machine parts, some part types are used across different types of machinery. As a result, the structure of machine part types and part-of relations make up a directed acyclic graph, or equivalently, a partially ordered set.

In the sought after classification, a cluster should consist of interchangeable parts. But equally important, the clusters should be ordered so that all the parts in the “lesser” cluster can be used as a sub-part of any element of a “greater” cluster. This is depicted in cases 2 and 4 of Figure 1.

This brings us to order preserving hierarchical clustering—a method that provides us with a hierarchical clustering that preserves the partial order. As a result, the corresponding ultrametric on the set of parts reflects both the similarities of the parts, as well as the part-of relations. And indeed, as we show in Section 2.3, the more elements that are strictly between two elements in the partial order, the larger the ultrametric distance between the elements in an order preserving hierarchical clustering. This makes perfect sense in the world of machine parts: if one part is deeply buried inside another part, such as a tire valve on a bicycle, there is little chance for the parts to be interchangeable.

While flat clustering provides partitions that can be considered as classifications of the data, hierarchical clustering provides tree structures over the data, where the leaves are the original
Figure 1: A selection of possible ordered clusterings of the set \{a, b, c, d\}. Possible interpretations in terms of the motivating use case are given together with the clusterings. All but 6) are examples of order preserving clusterings. In 6), the part-of relations constitute a cycle, implying that the parts are proper sub-parts of themselves, which is a contradiction.

1. The original data. \(a\) is a part of \(b\) and \(c\) is a part of \(d\).

2. \(a\) and \(c\) are equivalent, and both can be sub-parts of both \(b\) and \(d\), allowing for spare part interchange.

3. \(b\) and \(d\) are equivalent, and both contain \(a\) and \(c\) as sub-parts.

4. \(a\) and \(c\) are equivalent, and \(b\) and \(d\) are equivalent. It is likely that one pair is a copy of the other.

5. \(b\) and \(c\) are equivalent, and both \(a\) and \(b\) are sub-parts of \(d\). \(b\) and \(c\) can be interchanged as spare parts in \(d\).

6. Not a valid clustering since the induced part-of relations are cyclic.

Data points, and the bifurcations in the trees are where elements are joined into successively larger clusters in the hierarchy. In the context of machine learning, a significant benefit of these trees is that they correspond to ultrametrics (Jardine and Sibson, 1971). This means that in addition to classification, hierarchical clustering yields a distance function on the data. In the above use case, we are looking for items similar to a given item \(x_0\). The ultrametric is ideal for this purpose, allowing the user to explore items contained in successively larger neighbourhoods about \(x_0\) until a suitable element is found.

1.2 Related work

An earlier work on order preserving hierarchical clustering can be found in (Bakkelund, 2021a), providing a means for clustering strictly partially ordered data. The work in this paper provides significant generalisations of several key concepts:

- Since (Bakkelund, 2021a) requires the hard constraint \(a < b \Rightarrow [a] < [b]\), it follows that comparable elements can never be clustered together, regardless of their similarity. Consider, for example, the sections of a book ordered by order of appearance. An order preserving hierarchical clustering will place similar sections together, while at the same time maintain the ordering of the sections. Hence, the order preserving hierarchical clustering can be seen to produce a table of contents for the book. When the constraint \(a < b \Rightarrow [a] < [b]\) is applied, none of the sections can be clustered together, since they are all comparable. Hence, the only hierarchical clustering produced by (Bakkelund, 2021a) for linear orders is the singleton clustering. In this paper, we treat the order relation as non-strict, alleviating this problem.

- A hard, binary constraint for order preservation, such as that of (Bakkelund, 2021a), means that any data where the relation has cycles, for example due to errors in the data, cannot be clustered. In this paper, the order relation is considered probabilistic, and is weighted against the similarity measure. This allows us to cluster also data where the order relation is not a perfect partial order, providing a significant increase in availability and robustness in applications.
Finally, the theory in (Bakkelund, 2021a) only covers order preserving agglomerative hierarchical clustering. This paper provides a general theory for order preserving hierarchical clustering that is independent of methods or algorithms, and covers all models.

The objective function presented in this paper extends on the work by Dasgupta (2016). The model set forth by Dasgupta has been the subject of several publications, whereof Roy and Pokutta (2017), Moseley and Wang (2017), Chatziafratis et al. (2018), Cohen-Addad et al. (2019) are of particular interest with regards to this exposition. Similarly to the method we present in this paper, the method by Chatziafratis et al. (2018) extends on Dasgupta’s model in order to incorporate additional constraints from the domain. Their method can be sorted in the category of clustering with constraints (Basu et al., 2008; Davidson and Ravi, 2005), where the constraints can be seen as a particular type of structure imposed on the data. As such, this is indeed a variation of what we could call structure preserving hierarchical clustering. However, the type of structural constraints do not cover order relations. And, while the constraints in clustering with constraints are provided, for example, by domain experts in order to adjust the clustering, the order relations used in order preserving hierarchical clustering emanate directly from the data. It is the data itself that is ordered, and it is this order relation we wish to preserve.

Carlsson et al. (2014) present a hierarchical clustering regime called quasi clustering for asymmetric networks. An asymmetric network is the same as a weighted directed graph, and is covered by what we define as relaxed order relations in this paper. We can therefore say that, at a high level, the methods operate on similar data. However, the method presented by Carlsson et al. is almost opposite of what we define as order preserving: objects are clustered together if there is significant “flow” between the elements. In order preserving clustering, this is exactly when elements are not placed together.

A field that does order preserving clustering is that of acyclic partitioning of graphs. Given a directed acyclic graph, the goal is to partition the vertices so that the graph that arises when you preserve the edges going between partitions, is also a directed acyclic graph; for an example, see the article by Nossack and Pesch (2014). This is a topic that has been studied intensively with regards to applications ranging from railway planning to parallel processing, compiler theory and VLSI. Hierarchical methods have also been developed of late. While some of the new developments are of a more general nature (Herrmann et al., 2017; Herrmann et al., 2019), a lot of the work focuses on satisfying domain specific constraints and objectives. However, the methods do not easily combine with a similarity or dissimilarity to support more classical notions of clustering.

Several works have been published on clustering of ordered data. We include two classes of methods in this group. The first is that of comparison based clustering, where the degree of similarity of elements is derived from an order relation. An example is using pairwise comparisons of elements done by users for preference ranking. See the quite recent article by Ghoshdastidar et al. (2019) for more examples and references. In this category, we also find the works of (Janowitz, 2010), providing a wholly order theoretic approach to hierarchical clustering, including the case where the dissimilarity is replaced by a partially ordered set. The other class is that of partitioning a family of ordered sets, so that sets that are similar to each other are co-located in clusters. One example is the work by (Kamishima and Fujiki, 2003), presenting a method for clustering sets containing preference data. Another example of methods in this category is that of clustering of families of time series, such as the model described by Luczak (2016), producing clusters consisting of similar times series.

Whereas all the mentioned work touch upon one or more concepts involved in order preserving hierarchical clustering, none of the methods offer a means to provide hierarchical clus-
terings where the order relation and the similarity are combined, and where one seeks to find a hierarchical clustering that attempts to satisfy both.

1.3 Layout of the article

Section 1.4 recalls the required background of graph theory, order relations and hierarchical clustering, as well as recalling Dasgupta’s cost function. Section 2 gives a formal definition of order preserving hierarchical clustering, and precisely identifies the binary trees over a set that correspond to order preserving hierarchical clusterings. In Section 3, we introduce our objective function. Section 4 provides an investigation of the properties of the proposed model, looking only at the effect of the order relation, keeping the similarity out of the equation. The combined value function is the topic of Section 5, where we view the clustering problem in terms of bi-objective optimisation. Section 6 describes the polynomial time approximation algorithm, while Section 7 provides a demonstration of the efficacy of the method on data from the machine parts database described in Section 1.1. Section 8 provides a concise summary and presents a short list of future research topics.

1.4 Background

A graph is a pair \( G = (V, E) \) of vertices and edges. Unless otherwise is specified, we assume that all graphs are directed, and we refer to directed edges as arcs. The graph \( G \) is transitive if, for every pair of elements \( a, b \in V \) for which there is a path from \( a \) to \( b \), there is an edge \( (a, b) \in E \). A graph isomorphism \( \theta : G \to G' \) for graphs \( G = (V, E) \) and \( G' = (V', E') \) is a bijection \( \theta : V \to V' \) for which \((x, y) \in E \Leftrightarrow (\theta(x), \theta(y)) \in E'\).

For a set \( X \), we write \( A \subset X \) if \( A \) is a subset of \( X \), proper or not. A binary relation on a set \( X \) is a subset \( E \subset X \times X \). A partial order on a set \( X \) is a binary relation \( E \) on \( X \) that is reflexive, antisymmetric and transitive, and in that case we call the pair \((X, E)\) a partially ordered set. We usually denote the partially order by \( \leq \), writing \( x \leq y \) for \((x, y) \in E\), and \( x < y \) to mean \( x \leq y \land x \neq y \). We write \( \mathcal{I}_\leq \) to represent the indicator function of the partial order \( \leq \), in which case we have \( \mathcal{I}_\leq((x, y)) = 1 \) if and only of \( x \leq y \), and zero otherwise. For subsets \( A, B \subset X \), we use the notation \( \mathcal{I}_{\leq}(A, B) \) to denote the magnitude

\[
\mathcal{I}_{\leq}(A, B) = \sum_{(a, b) \in A \times B} \mathcal{I}_{\leq}((a, b)).
\]

We refer to two elements \( x, y \in X \) as comparable if either \( x \leq y \) or \( y \leq x \), and a pair of elements that are not comparable are called incomparable. A linear order is a partial order in which any two elements are comparable. A chain in a partially ordered set \((X, \leq)\) is a subset \( Y \subset X \) that is linearly ordered with respect to \( \leq \). Given two partial orders \( \leq, \leq' \) on \( X \), we say that \( \leq' \) is an extension of \( \leq \) if \( x \leq y \Rightarrow x \leq' y \). If in addition \( \leq' \) is a linear order, we call \( \leq' \) a linear extension of \( \leq \). For two order relations where one is an extension of the other, we say that the orders are compatible. The transitive closure of a binary relation \( E \) on \( X \) is the smallest transitive relation on \( X \) that extends \( E \).

A map \( h : (X, \leq) \to (Y, \leq') \) between partially ordered sets is said to be order preserving if \( x \leq y \Rightarrow h(x) \leq' h(y) \). Notice that a composition of order preserving maps is also order preserving. See (Schröder, 2003) for a general introduction to order theory.

A (flat) clustering of a set \( X \) is a partition \( C = \{C_i\}_{i=1}^k \) of \( X \) into disjoint subsets. We do not consider overlapping clusters; every clustering is a partition and corresponds to an equivalence relation \( \sim \) on \( X \). A cluster is a block in the partition, equivalently, an equivalence class under
the equivalence relation. We employ the usual bracket notation for equivalence classes based on representatives, writing \([x]_C\) for the cluster of \(x\) in the clustering \(C\), possibly leaving out the subscript if there is no risk of ambiguity. For every clustering \(C\) of \(X\), the unique map \(q : X \to C\) defined by \(q(x) = [x]_C\), sending an element to its cluster, is called the quotient map of \(C\).

Starting with a non-empty, finite set \(X\), a split of \(X\) is a partition \((A, B)\) of \(X\) into two non-empty, disjoint subsets. Each of these sets may again be split, giving rise to a hierarchical decomposition of \(X\) that can be drawn as a tree, where every node has its split components as children until no more splits can be made. This process produces a binary tree that has the set \(X\) at the root, and the singleton sets \(\{x\} \mid x \in X\) as leaf nodes, and defines a hierarchical clustering of \(X\).

We consider every split \((A, B)\) to be oriented, meaning that the split \((A, B)\) is considered to be different from the split \((B, A)\). We refer to this concept as oriented splits, and the trees as oriented trees. If a node in a tree splits into \((A, B)\), we refer to \(A\) as the left child, and \(B\) as the right child of the parent node. We denote the set of all oriented binary trees over \(X\) that can be generated according to the above procedure by \(\mathcal{B}(X)\). And, for a tree \(T \in \mathcal{B}(X)\), if \(S\) is a node in \(T\) that splits into \((A, B)\), we denote this by writing \(S \xrightarrow{T} (A, B)\). Notice that for \(T \in \mathcal{B}(X)\), every node \(S\) in \(T\) is a subset of \(X\). Let \(x \vee y\) denote the smallest node in \(T\), considered as a subset of \(X\), containing both \(x\) and \(y\), and let \(|T[x \vee y]|\) be the subtree of \(T\) rooted at \(x \vee y\).

In particular, \(|T[x \vee y]|\) is the smallest subtree rooted at an internal node containing both \(\{x\}\) and \(\{y\}\) as leaves. Finally, we write \(|T[x \vee y]|\) to denote the number of leaf nodes of \(T[x \vee y]\), noting that this number coincides with the cardinality of the root node of \(T[x \vee y]\), namely \(|x \vee y|\). An example of a hierarchical clustering is depicted in Figure 2.

An ultrametric on \(X\) is a metric \(d\) on \(X\) for which

\[
d(x, z) \leq \max\{d(x, y), d(y, z)\} \quad \forall x, y, z \in X.
\]

We refer to the above equation as the ultrametric inequality, and note that it implies the triangle inequality. The set of all ultrametrics over \(X\) is denoted \(\mathcal{U}(X)\).

Given a binary tree \(T \in \mathcal{B}(X)\), Roy and Pokutta (2017) show that the map \(\mathcal{U}_X : \mathcal{B}(X) \to \mathcal{U}(X)\) given by

\[
\mathcal{U}_X(T)(x, y) = |T[x \vee y]| - 1
\]

is an embedding for non-ordered trees. That is, every non-ordered binary tree over \(X\) maps to a unique ultrametric over \(X\) via \(\mathcal{U}_X\). Unless there is any chance of ambiguity, we denote the ultrametric \(\mathcal{U}_X(T)\) by \(u_T\).

Given an ultrametric \(u\) on \(X\), for every \(t \in \mathbb{R}_+\), the relation \(\sim_t\) given by \(x \sim_t y \iff u(x, y) \leq t\) is an equivalence relation on \(X\), and provides us with a flat clustering of \(X\). (See Carlsson and Mémoli (2010) or Jardine and Sibson (1971) for details.) When we refer to the clustering of an equivalence relation \(\sim_t\) in the context of a tree \(T \in \mathcal{B}(X)\), unless otherwise is stated, this is always in relation to the ultrametric \(u_T\). In particular, for \(T \in \mathcal{B}(X)\), the flat clusterings provided by \(T\) are the clusterings defined by the equivalence relations \(\{\sim_t \mid t \in \mathbb{R}_+\}\).

For example, considering the tree in Figure 2 and the ultrametric \(u_T\), the corresponding flat clusterings are presented in Table 1.
### 1.4.1 The Dasgupta cost model

A **similarity** on a set $X$ is a symmetric function $s : X \times X \to [0, 1]$. Given a similarity $s$ on $X$, the **Dasgupta cost function** is the function $\text{cost}_s : \mathcal{B}(X) \to \mathbb{R}_+$ defined as

$$\text{cost}_s(T) = \sum_{\{x, y\}} |T[x \lor y]|s(x, y), \quad (2)$$

where the sum is over all distinct pairs of elements of $X$. The optimisation problem is to find a tree $T \in \mathcal{B}(X)$ minimising (2). For a list of desirable properties of optimal trees under this model, please consult the references given in the related work section.

Dasgupta defines similarity measures to have codomain all of $\mathbb{R}_+$, but since minimising (2) is invariant with respect to positive scaling of $s$, and since $X$ is finite, the above definition implies no loss of generality.

**Dual formulation.** The dual formulation of Dasgupta’s cost function allows us to solve the optimisation by maximisation. The motivation for introducing the dual is that when we introduce order relations, it is easier to discuss the optimisation problem in the context of maximisation.

Define the dual of $s$ to be the function $s_d : X \times X \to [0, 1]$ given by

$$s_d(x, y) = 1 - s(x, y), \quad (3)$$

and define the **value function** $\text{val}_{s_d} : \mathcal{B}(X) \to \mathbb{R}_+$ by

$$\text{val}_{s_d} = \sum_{\{x, y\}} |T[x \lor y]|s_d(x, y).$$

As of (Dasgupta, 2016, §4.1) any binary tree over $X$ maximising $\text{val}_{s_d}$ is a tree that minimises $\text{cost}_s$.

## 2 Order preserving hierarchical clustering

The contributions from this section are two-fold: first, we provide a formal definition of order preserving hierarchical clustering. Second, we define a class of binary trees called **order preserving trees**, and prove that these trees are exactly the oriented binary trees over $X$ that correspond to order preserving hierarchical clusterings.

Towards the end of the section, we present a result describing how the partial order is reflected in the ultrametric $u_T$ of an order preserving tree $T$: that the more elements there are between two elements in the partial order, the more different they are under the ultrametric.
2.1 Formal definition

We start by recalling the order theoretical notion of an order preserving flat clustering, and then provide an extended definition that includes hierarchical clustering.

The next two definitions and following theorem can be found in (Blyth, 2005, §3.1). We recall them here for completeness, and also to rephrase the concepts in the context of clustering. We start by defining what it means for a flat clustering to be order preserving.

**Definition 1.** Let \((X, \leq)\) be a partially ordered set, and let \(C\) be a clustering of \(X\). We say that the clustering \(C\) is **order preserving** (with respect to \(\leq\)) if there exists a partial order \(\leq'\) on \(C\) so that 

\[
x \leq y \Rightarrow [x]_C \leq' [y]_C.
\]

The next definition and theorem classify exactly the order preserving clusterings for a partially ordered set.

**Definition 2.** Let \((X, \leq)\) be an ordered set, and let \(C = \{C_i\}_{i=1}^k\) be a clustering of \(X\). Let \(E\) be the binary relation on \(C\) satisfying 

\[
(C_i, C_j) \in E \iff \exists x, y \in X : x \leq y \land x \in C_i \land y \in C_j.
\]

We define the **induced relation on** \(C\), denoted \(\leq'_C\), to be the transitive closure of \(E\).

An instructive illustration of what the induced relation looks like, is that of a \(C\)-fence (Blyth, 2005), or just fence, for short:

\[
\begin{array}{cccc}
| & | & | & | \\
\text{b}_1 & \text{b}_2 & \ldots & \text{b}_{n-1} & \text{b}_n \\
| & | & | & | \\
\text{a}_1 & \text{a}_2 & \ldots & \text{a}_{n-1} & \text{a}_n \\
\end{array}
\]  

(4)

Triple lines indicate elements in the same cluster, and the arrows represent comparability in \((X, \leq)\). The fence allows traversal from \(b_1\) to \(a_n\) along arrows and through clusters, in which case we say that the fence **links** \(b_1\) to \(a_n\). The induced relation \(\leq'_C\) has the property that \(x \leq'_C y\) if and only if there exists a \(C\)-fence linking \(x\) to \(y\).

**Theorem 3** (Blyth, 2005, Thm.3.1). Let \((X, \leq)\) be an ordered set, and let \(C\) be a clustering of \(X\). Then the following two statements are equivalent:

1. The induced relation \(\leq'_C\) is a partial order on \(C\).
2. The quotient map \(p : (X, \leq) \to (C, \leq'_C)\) is order preserving.

Notice that Item 2 of the theorem implies that \(C\) is an order preserving flat clustering. Our task now, is to define what it means for a hierarchical clustering to be order preserving. We start by recalling the definition of a hierarchical clustering, and then propose an extension that includes order preservation.

For two clusterings \(C = \{C_i\}_{i=1}^m\) and \(D = \{D_j\}_{j=1}^n\) of \(X\), we say that \(C\) is a **refinement** of \(D\) if, for every \(C_i \in C\), there is a \(D_j \in D\) so that \(C_i \subset D_j\). If this is the case, then the map \(q : C \to D\) defined by \(q(C_i) = D_j \iff C_i \subset D_j\) is also a quotient map. We denote the refinement relation by \(C \sqsubset D\), and refer to the map \(q\) as the **quotient map induced by the refinement**.
A hierarchical clustering over $X$ is, thus, a sequence $\{C_i\}_{i=0}^k$ of clusterings of $X$ for which

\[ C_0 \sqsubseteq \cdots \sqsubseteq C_k. \]  

(5)

A sequence $\{C_i\}_{i=0}^k$ satisfying (5) corresponds to a persistent set as of Carlsson and Memoli (2013). If we require that we have $C_k = \{X\}$, the sequence corresponds to a dendrogram in the nomenclature of Jardine and Sibson (1971), and, recalling the singleton partition $S(X) = \bigcup_{x \in X} \{\{x\}\}$, if $C_0 = S(X)$, the sequence corresponds to a definite dendrogram, also according to Jardine and Sibson. All of the mentioned concepts define hierarchical clustering in the classical sense.

We now have the following observation.

**Theorem 4.** Given a partially ordered set $(X, \leq)$ and a hierarchical clustering $H = \{C_i\}_{i=0}^k$ of $X$, then the following statements are equivalent:

1. All quotient maps $p_i : X \to C_i$, for $0 \leq i \leq k$, are order preserving with respect to $\leq$;
2. All quotient maps $q_i : C_i \to C_{i+1}$ for $0 \leq i \leq k-1$, induced by the refinements, are order preserving with respect to the induced order relations;
3. The following diagram is commutative and all quotient maps are order preserving:

\[
\begin{array}{cccc}
C_0 & q_0 & C_1 & q_1 & \cdots & q_{k-1} & C_k \\
p_0 & & p_1 & & \cdots & & p_k \\
X & & & & & & .
\end{array}
\]

Before presenting the proof, we suggest the following definition:

**Definition 5.** Given a partially ordered set $(X, \leq)$, an order preserving hierarchical clustering of $X$ with respect to $\leq$ is a hierarchical clustering of $X$ satisfying any and all of the statements of Theorem 4.

**Proof of Theorem 4.** We start by proving the equivalence of of statements 1 and 2. Assume first that the quotient maps $p_i : X \to C_i$ are order preserving. If we can show that for every $A, B \in C_i$, we have $A \leq'_i B \Rightarrow q_i(A) \leq'_{i+1} q_i(B)$, we are done.

Since $A \leq'_i B$, there are elements $a \in A$ and $b \in B$ that are linked via a $C_i$-fence. Now, $C_i$ is a refinement of $C_{i+1}$, so every pair of co-clustered elements in $C_i$ are also co-clustered in $C_{i+1}$. Since this implies that all in-cluster links in fences are maintained, there is also a $C_{i+1}$-fence linking $a$ and $b$, and we must therefore have $A \leq'_{i+1} B$. Since $p_{i+1}$ is order preserving, $\leq'_{i+1}$ is an order relation on $C_{i+1}$, so according to Theorem 3, $q_i$ is order preserving too.

Now assume that all the $q_i$ are order preserving. Since the maps $p_i$ must be quotient maps sending elements to their clusters, there is only one possible definition of these maps; namely

\[ p_i = q_{i-1} \circ \cdots \circ q_0 \circ p_0. \]

Since $p_0$ is order preserving, the result follows by induction on $i$, as all maps on the right hand side are order preserving, and since a composition of order preserving maps is an order preserving map.

Finally, the diagram commutes due to the above definition of the $p_i$, and the maps are all order preserving if and only of both statements 1 and 2 hold.
2.2 Order preserving binary trees

We now define order preserving binary trees and show that these trees are exactly the binary trees that correspond to order preserving hierarchical clusterings.

All the concepts on binary trees given in this section extend straightforwardly to the class of all oriented trees over \( X \), regardless of the arity of the splits. However, since the rest of the paper deals exclusively with binary trees, we have chosen to stick to binary trees also for this part.

Recall that for a split \((A, B)\), the order of the components is significant; the split is ordered.

**Definition 6.** Let \((X, \leq)\) be a partially ordered set. A split \((A, B)\) of \(X\) is order preserving (with respect to \(\leq\)) if

\[
x \leq y \Rightarrow (y, x) \not\in A \times B \quad \forall x, y \in X.
\]

An order preserving tree is a tree in which every split is order preserving.

The idea is that, in an order preserving split there are no “reversals” of ordered pairs; the order of the elements shall coincide with the order of the sets they reside in. For example, if \(x \leq y\), we can have \((x, y) \in A \times B\), \(x, y \in A\) or \(x, y \in B\). The only illegal constellation is that of equation (6).

Recalling the indicator function \(I_\leq\) of the partial order, we could replace equation (6) by requiring that \(I_\leq(B, A) = 0\), a statement that is equivalent to that of Definition 6.

Notice that if we draw a tree \(T \in \mathcal{B}(X)\) on a piece of paper, with the root at the top and the leaves at the bottom, since all splits are oriented, the tree induces a unique linear ordering on the leaves as they appear from left to right. We write \(\leq_T\) to denote this ordering, and refer to \(\leq_T\) as the linear order on \(X\) induced by \(T\).

For an order preserving tree, the order \(\leq_T\) is a linear extension of the partial order:

**Lemma 7.** If \((X, \leq)\) is a partially ordered set, then \(T \in \mathcal{B}(X)\) is an order preserving tree with respect to \(\leq\) if and only if

\[
x \leq y \Rightarrow x \leq_T y \quad \forall x, y \in X.
\]

**Proof.** Assume first that \(T\) is order preserving, and pick \(x, y \in X\) for which \(x \leq y\). Let \(S = x \vee y\) and \(S_\leq(A, B)\), so that \(x\) and \(y\) end up in different split components of \(S\). Since \(T\) is order preserving, we must have \(x \in A\) and \(y \in B\), but this means that \(x \leq_T y\) too, so the implication holds.

For the opposite direction, let \(S_\leq(A, B)\) be a split in \(T\). Furthermore, let \(a, b \in X\) be comparable, and let \(a\) and \(b\) end up in different components in the split of \(S\); that is: \(a \leq b\), and \(a \vee b = S\). Due to the assumption \(a \leq b \Rightarrow a \leq_T b\), we get \(a \in A\) and \(b \in B\). Since this covers all comparable pairs across \(A\) and \(B\), the split \(S_\leq(A, B)\) is order preserving. \(\square\)

Recall that for a tree \(T \in \mathcal{B}(X)\) and a non-negative real number \(t\), there is a flat clustering \(C_t\) of \(X\) defined by the equivalence relation \(\sim_t\) (Section 1.4). For every flat clustering under a tree \(T\), the relation \(\leq_T\) also induces an order on the clusters:

**Lemma 8.** For a partially ordered set \((X, \leq)\), let \(T \in \mathcal{B}(X)\) and \(v_T = \sqcup_X(T)\). If \(t \in \mathbb{R}_+\), and if \(C_t = \{C_i\}_{i=1}^k\) are the clusters of \(\sim_t\), then the enumeration on the clusters can be chosen so that

\[
x \leq_T y \land x \in C_i \land y \in C_j \Rightarrow i \leq j \quad \forall x, y \in X.
\]

\[\text{(7)}\]
Proof. We prove this by induction. Let \( \{t_1 > \cdots > t_m\} \) be a maximal set of real numbers so that each equivalence relation \( \sim_{t_i} \) is distinct for \( 1 \leq i \leq m \). Clearly, the statement holds for \( t_1 \), where there is only one cluster. Assume that the statement holds for \( t_k \) when \( k \geq 1 \), and consider the case \( t_{k+1} \). Let \( C_i, C_j \) be distinct clusters under \( \sim_{t_{k+1}} \). If there is a cluster \( C \) under \( \sim_{t_k} \) for which \( C \rightarrow (C_i, C_j) \), then the split defines an ordering of \( C_i \) and \( C_j \) that is compatible with \( \leq_T \). If no such \( C \) exists, then \( C_i \) and \( C_j \) are subsets of distinct clusters under \( \sim_{t_k} \). Due to the induction hypothesis, these equivalence classes are already ordered compatibly with \( \leq_T \), and this order propagates to \( C_i \) and \( C_j \), maintaining compatibility. \( \square \)

We accept the risk of using \( \leq_T \) also to denote the relation on the clusters, writing \( C_i \leq_T C_j \).

Remark 9. In order theoretic jargon, Lemma 8 implies that all the clusters are convex with respect to \( \leq_T \); if \( x, z \in C_i \) and \( x \leq_T y \leq_T z \), then \( y \in C_i \) too.

We are now ready to state the main result of this section.

Theorem 10. Let \((X, \leq)\) be a partially ordered set, and let \( T \in \mathcal{B}(X) \). Then \( T \) is an order preserving tree with respect to \( \leq \) of and only if the hierarchical clustering corresponding to \( T \) is an order preserving hierarchical clustering of \( X \) with respect to \( \leq \).

Proof. Recall that the hierarchical clustering corresponding to \( T \) is the sequence of clusters \( \mathcal{H} = \{C_i\}_{i=0}^m \) corresponding to the equivalence relations \( \sim_t \) under \( u_T \).

Assume that \( T \) is order preserving, and let \( C = \{C_i\}_{i=1}^m \) be a flat clustering in \( \mathcal{H} \). We shall show that \( C \) is an order preserving clustering, implying that the quotient map \( p : X \rightarrow C \) is order preserving, where after the theorem follows from Theorem 4.

By combining Lemmas 7 and 8, we can enumerate the clusters of \( C \) according to \( \leq_T \) so that \( C_1 \leq_T \cdots \leq_T C_m \), and moreover, \( x \leq y \Rightarrow [x]_C \leq_T [y]_C \). Let \( \leq' \) be the induced relation on \( C \) (Definition 2). Then \( x \leq y \Rightarrow [x]_C \leq' [y]_C \). But this means that the linear order \( \leq_T \) on \( C \) is an extension of \( \leq' \), so \( \leq' \) must be a partial order on \( C \).

For the only-if part, let \( \mathcal{H} \) be the clustering corresponding to \( T \), and assume that \( \mathcal{H} \) is an order preserving hierarchical clustering of \( X \) with respect to \( \leq \). In particular, we have \( C_0 = S(X) \) as an order preserving clustering, and since \( \leq_0 \) has \( \leq_T \) as a linear extension, it follows that \( \leq_T \) is a linear extension also of \( \leq \). According to Lemma 7, this means that \( T \) is order preserving with respect to \( \leq \). \( \square \)

Now we know that we are looking for order preserving trees. What remains is simply to devise a method to identify the best of them. Before we close the section, we present a result that links order preserving trees to ultrametrics, and shows how the partial order is reflected in the ultrametric distances.

2.3 Order preserving trees and ultrametric distances

We now demonstrate that for an order preserving tree \( T \in \mathcal{B}(X) \) over a partially ordered set \((X, \leq)\), given the ultrametric \( u_T = \bigcup_X(T) \), the distance between two elements \( u_T(x, y) \) is bounded below by the number of elements strictly between \( x \) and \( y \) in \( \leq \).

For an ordered set \((X, \leq)\) and \( x, y \in X \), define the function \( \text{jmp} : X \times X \rightarrow \mathbb{N} \) so that if \( x \leq y \), then \( \text{jmp}(x, y) \) is one less than the cardinality of a maximal chain having \( x \) as minimal element and \( y \) as maximal element, and zero if \( x \not\leq y \). The magnitude of \( \text{jmp}(x, y) \) is equal to the maximal number of “jumps” one would have to make, jumping one element at the time,
starting at \( x \) and stopping at \( y \), when jumping only in the direction of strictly larger elements. We define the ordered separation of \( x, y \in X \) to be the magnitude
\[
\text{sep}(x, y) = \max\{\text{jmp}(x, y), \text{jmp}(y, x)\}.
\]
Notice that \( \text{sep}(x, y) = 0 \) if and only of either \( x = y \), or \( x \) and \( y \) are incomparable.

**Theorem 11.** Let \((X, \leq)\) be a partially ordered set, let \( T \in \mathcal{B}(X) \) be order preserving with respect to \( \leq \), and let \( \{x_i\}_{i=1}^n \) be the enumeration of elements of \( X \) corresponding to the order \( \leq_T \) so that \( i \leq j \iff x_i \leq_T x_j \). Let \( u_T = \bigcup X(T) \). Then
\[
u_T(x_i, x_j) = |i - j| \geq \text{sep}(x_i, x_j).
\]

**Proof.** The right hand side inequality follows from the fact that every element between \( x_i \) and \( x_j \) under \( \leq \), must also be between \( x_i \) and \( x_j \) under \( \leq_T \), due to Lemma 7. The left hand side inequality follows from the definition of \( u_T \), namely \( u_T(x_i, x_j) = |T[x_i \vee x_j]| - 1 \), and the fact that every leaf between \( x_i \) and \( x_j \) under \( \leq_T \) must be a leaf of \( T[x_i \vee x_j] \). To see why this must be true, it is sufficient to consider the planar drawing of \( T \), and realising that every element between \( x_i \) and \( x_j \) must join either \( x_i \) or \( x_j \) before \( x_i \) and \( x_j \) are joined. \( \square \)

This essentially tells us that the more elements that lie between two elements, the more similar they are under an ultrametric that corresponds to an order preserving tree. Our choice of ultrametric definition makes the result easily quantifiable, but the observation holds for all ultrametrics. The intuition behind this is straight forward: The more elements that are between two elements in the partial order, the more elements there are between the elements in \( \leq_T \). And for any ultrametric (equivalently, dendrogram) that is based on an order preserving tree, this means that the more elements there are between two elements in the partial order, the higher in the tree (or dendrogram) they join, leading to a higher ultrametric distance.

Looking back at the motivating industry use case, where we have parts that consists of parts and so on, Theorem 11 simply reflects the fact that the more nested levels of composition there is between two parts, the less similar they are.

### 3 An objective function for trees over ordered data

We now turn our attention to the task of devising a method for identifying “good” order preserving trees. In this section, we define our representation of ordered data, and present a value function for binary trees that incorporates the idea of order preservation. And, as we show in the next section, the function is a realisation of order preserving hierarchical clustering in the sense that it correctly identifies order preserving trees when the data is partially ordered.

In the continuation, we treat the order relation as a stochastic object. Let the function \( \omega : X \times X \to [0, 1] \) represent a family of random binary relations \( E \) on \( X \), where \((x, y) \in E \) with probability \( \omega(x, y) \). We refer to \( \omega \) as a **relaxed binary relation** on \( X \).

**Example 1.** If \((X, \leq)\) is a partially ordered set, and if \( 0 \leq q < p \leq 1 \), we can define
\[
\omega(x, y) = \begin{cases} 
p & \text{if } x \leq y, \\
q & \text{otherwise.}
\end{cases}
\]

Then \( \omega \) is a relaxed binary relation. Moreover, for any random relation \( E \) in the family of relations represented by \( \omega \), recalling the indicator function \( I_E \) of \( E \), the expected value of \( I_E \) is 
\[
\mathbb{E}(I_E(x, y)) = \omega(x, y).
\]
However, we cannot in general expect \( E \) to be a partial order on \( X \).
To remind ourselves that we are working with ordered sets, we shall mostly refer to $\omega$ as a relaxed order, keeping in mind that it is the same thing as a relaxed binary relation.

Next, define the antisymmetrisation of $\omega$ as the function $g : X \times X \to [-1, 1]$, given by

$$g(x, y) = \omega(x, y) - \omega(y, x).$$

(8)

As the name suggests, the function is antisymmetric, and it computes the signed net comparability between $x$ and $y$. We see that $g(x, y)$ takes on a large positive value if there is strong evidence for $x < y$, and a large negative value if there is strong evidence for $x > y$. Moreover, if the comparability is ambiguous, with $\omega(x, y)$ and $\omega(y, x)$ being very similar, $g(x, y)$ will take on a value close to zero.

To see why this is useful, consider Example 1, and assume that $q$ and $p$ are very close in magnitudes, meaning that the probability of $(x, y) \in E$ is very close to the probability of $(y, x) \in E$. If we are producing a split $(A, B)$ of $X$ and trying to decide where to place $x$ and $y$ in order to make the split order preserving (Definition 6), it makes little sense to strongly favour one placement over the other in this situation. The close to zero value of $g(x, y)$ reflects this fact. The function $g$ can be used for hierarchical clustering on its own without any similarity, and the properties of $g$ is the subject of study of Section 4.

However, to achieve order preserving hierarchical clustering, we must combine $g$ with a similarity. Define an ordered similarity space to be a triple $(X, s, \omega)$, where $X$ is a set, $s$ is a similarity, and $\omega$ is a relaxed order on $X$. For a given $(X, s, \omega)$, recall the similarity dual $s_d$ from (3). Let the split value function $f : X \times X \to [-1, 2]$ be defined as

$$f(x, y) = s_d(x, y) + g(x, y).$$

(9)

This function will attain its maximal value on $(x, y)$ if both $s_d(x, y)$ and $g(x, y)$ are maximal, meaning that $x$ and $y$ are considered to be both highly dissimilar and we have $x < y$ with high confidence; both being evidence for splitting $x$ and $y$ into separate clusters.

On the other hand, the function will attain its minimal value on $(x, y)$ only if $g(x, y)$ has a large negative value. Since $s_d$ is symmetric and $g$ is antisymmetric, this means that by swapping the arguments, $f(y, x)$ will attain a large positive value, again being evidence for splitting the elements apart.

And finally, if the elements are not comparable, then $g(x, y)$ is close to zero, and if the elements are not dissimilar, then $s_d$ is close to zero, so $f$ is close to zero if there is little evidence for splitting the elements apart.

**Definition 12.** Given $(X, s, \omega)$ and a tree $T \in \mathcal{B}(X)$, the value of $T$ under $(X, s, \omega)$ is given by the function $\text{val}_f : \mathcal{B}(X) \to \mathbb{R}$, defined as

$$\text{val}_f(T) = \sum_{x \leq_T y} |T[x \lor y]|f(x, y),$$

(10)

where the iteration order is dictated by the linear order $\leq_T$ induced by $T$. Furthermore, an optimal hierarchical clustering of $(X, s, \omega)$ is a binary tree $T^* \in \mathcal{B}(X)$ for which

$$\text{val}_f(T^*) = \max_{T \in \mathcal{B}(X)} \text{val}_f(T).$$

The multiplier $|T[x \lor y]|$ of $\text{val}_f$ encourages pairs of elements for which $f(x, y)$ attains a large positive value to be split apart close to the root of the tree. Looking back at the discussion preceding the definition, this means that elements that are strongly indicated to belong to different clusters will be split apart close to the root.
But the maximisation also favors an orientation on elements: If $\omega$ is as in Example 1, and if $x < y$, then $g(x, y) > 0$ and $g(y, x) < 0$. This intuitively suggests that a tree $T \in \mathcal{B}(X)$ in which $x \leq_T y$ will have higher value compared to if $y \leq_T x$, and therefore suggests that a maximal tree will be order preserving, as of Lemma 7; a property that will be proven formally in Section 4.

There is an alternative formulation of (10) which we will make occasional use of. Recall that every node in a tree $T \in \mathcal{B}(X)$ is a subset of $X$, and an internal node $S \in T$ splits according to $S \rightarrow (A, B)$. If we define $f(A, B) = \sum_{(a, b) \in A \times B} f(a, b)$, we can formulate $\text{val}_f$ as

$$\text{val}_f(T) = \sum_{S \rightarrow (A, B)} |S| f(A, B),$$

where the sum is over all the splits $S \rightarrow (A, B)$ in $T$.

## 4 Properties of $g$

In this section, we study the value function

$$\text{val}_g(T) = \sum_{x \leq_T y} |T[x \vee y]| g(x, y),$$

where we only take into account the order relation, ignoring the similarity. The first part, Section 4.1, is concerned with the behaviour on ideal inputs. That is, we show that when $\omega$ is a binary partial order, then the optimal trees under $\text{val}_g$ are order preserving. Next, in Section 4.2, we turn to relaxed orders, and present a quantitative analysis of $\text{val}_g$, showing that the efficacy of $\text{val}_g$ with respect to order preservation is in the same class as Dasgupta’s model is with respect to clustering. Finally, Section 4.3 presents a worked example, showing how (11) can be used to analyse migration patterns between states.

### 4.1 Optimality on partially ordered data

The goal of this section is to show that if $(X, \leq)$ is a partially ordered set, and if we define $\omega = I \leq$, then the optimal trees under $\text{val}_g$ are order preserving. This is important, for it shows that when the input is well formed, then order preserving hierarchical clustering works as intended. Therefore, in what follows, let that $\omega = I \leq$.

Before we embark on the main theorem, we introduce some new tools. For a partially ordered set $(X, \leq)$ and a tree $T \in \mathcal{B}(X)$, every pair of elements in $X$ are evaluated in $\text{val}_g$ according to how they are ordered under $\leq_T$. We define the **T-symmetrisation of $g$** to be the function $\gamma_T : X \times X \rightarrow \mathbb{R}$, defined as

$$\gamma_T(x, y) = \begin{cases} g(x, y) & \text{if } x \leq_T y, \\ g(y, x) & \text{otherwise.} \end{cases}$$

That is, $\gamma_T$ is a symmetric function that computes the value under $g$ consistent with the orientation on the arguments induced by $T$.

Moreover, given a partially ordered set $(X, \leq)$ and a tree $T \in \mathcal{B}(X)$, we define the **cluster graph over $(X, \leq)$ and $T$** to be the weighted undirected complete graph $G_T = (X, E, \nu)$, where the weight function is given by

$$\nu(x, y) = |T[x \vee y]| \gamma_T(x, y).$$
That is, every edge \((x, y)\) in \(G_T\) has a weight that corresponds to the value of the pair \(\{x, y\}\) in \(\text{val}_\gamma(T)\). The \(T\)-symmetrisation ensures that the orientation induced by \(T\) is adhered to. The following lemma points up the purpose of the above construction; it allows us to replace the formula for the value \(\text{val}_\gamma\) of a tree by a sum over the edge weights of the cluster graph.

**Lemma 13.** If \(G_T = (X, E, \nu)\) is the cluster graph over \((X, \leq)\) and \(T\), then

\[
\text{val}_\gamma(T) = \sum_{e \in E} \nu(e).
\]

**Proof.** Every distinct pair \(\{x, y\} \in X \times X\) is summed over exactly once, and since \(\gamma_T\) ensures that the pair is associated with the value under \(\nu\) corresponding to the orientation induced by \(T\), the lemma holds. \(\Box\)

The next lemma allows us to manipulate the cluster graph in a controlled manner. Recall that for a tree \(T \in \mathcal{B}(X)\) with \(T = (V, E)\), every node \(S \in V\) is a subset of \(X\), and every edge \(e \in E\) is a subset inclusion.

**Lemma 14.** Let \(\phi : X \to X\) be a bijection, and let \(T = (V, E)\) be a binary tree over \(X\). Then there exists a binary tree \(T' = (V', E')\) over \(X\) that is isomorphic to \(T\), and where the isomorphism \(\phi_T : V \to V'\) is given by

\[
\phi_T(S) = \{ \phi(x) \mid x \in S \}.
\]

In particular, for all \(x, y \in X\), we have

\[
|T[x \lor y]| = |T'[\phi(x) \lor \phi(y)]|.
\]

**Proof.** The set \(V\) is a subset of the power set of \(X\), and the map \(\phi_T\) is the mapping from the power set of \(X\) to itself induced by \(\phi\), restricted to \(V\). Since \(\phi\) is a bijection, the map \(\phi_T : V \to V'\) is also a bijection. Moreover, since \(\phi_T\) preserves subset inclusion, it follows that the tree structure is preserved under \(\phi_T\). Thus, \(T'\) is a well-defined binary tree over \(X\), and is isomorphic to \(T\). Equation (12) follows since \(\phi_T\) preserves cardinalities. \(\Box\)

Given \(\phi\) and \(T\) from Lemma 14, the tree \(T'\) is necessarily unique. We refer to \(T'\) as the tree induced by \(T\) and \(\phi\).

We are now ready to state our theorem, telling us that for a partially ordered set, the optimal trees under \(\text{val}_\gamma\) are order preserving:

**Theorem 15.** If \((X, \leq)\) is a partially ordered set and \(T \in \mathcal{B}(X)\) is not order preserving, then there exists a bijection \(\phi : X \to X\) so that the tree \(T'\) induced by \(T\) and \(\phi\) is order preserving and has strictly higher value than \(T\). In particular, the optimal trees over \(X\) are order preserving.

**Proof.** If \(T\) is not order preserving, we can pick distinct \(a, b \in X\) for which \(a \leq b\) and \(b \leq_T a\). Let \(\sigma\) be the permutation on \(X\) swapping \(a\) and \(b\), and leaving all other elements fixed, and let \(T'\) be the tree induced by \(T\) and \(\sigma\). We show that \(\text{val}_\gamma(T) < \text{val}_\gamma(T')\).

Let \(G_T = (X, E, \nu)\) be the cluster graph over \((X, \leq)\) and \(T\), and let \(G_{T'} = (X, E, \nu')\) be the cluster graph over \((X, \leq)\) and \(T'\). We claim that both of the following holds:

\[
\exists e \in E : \nu'(e) > \nu(e),
\]

\[
\forall e \in E : \nu'(e) \geq \nu(e).
\]

(13)

(14)
If so, according to Lemma 13, $T'$ has a strictly higher value than $T$. We may then repeat the process of permuting elements and generating induced trees of strictly higher value until we reach an order preserving tree. Since a sequence of permutations is a bijection, it follows that the order preserving tree is induced by $T$ and this sequence of permutations.

To prove the claim, since no element outside the subtree $T[a \lor b]$ is affected by the permutation, we can assume that $a$ and $b$ are joined at the root of $T$, without loss of generality.

To prove (13), notice that

\[ \nu(a, b) = |X| \gamma_T(a, b) = -|X| < |X| = |X| \gamma_{T'}(a, b) = \nu_\sigma(a, b). \]

To prove (14), we partition the edges $E$ into five disjoint sets:

\[
\begin{align*}
E_1 &= \{(x, y) \in E \mid x, y \neq a, b\} & E_4 &= \{(a, x) \in E \mid a \leq x \land b \notin x\} \\
E_2 &= \{(x, a) \in E \mid x \leq a\} & E_5 &= \{(x, b) \in E \mid x \leq a \land x \leq b\} \\
E_3 &= \{(b, x) \in E \mid b \leq x\}
\end{align*}
\]

Case $E_1$: Since all edges in $E_1$ are between fixed points under $\sigma$, there are no changes of the edge weights.

Case $E_2$: Let $(x, a) \in E_2$. Then $x \leq a \leq y$, since $\leq$ is transitive. From Lemma 14, and since $\gamma_T(x, a) = \gamma_T'(x, b)$ and $\gamma_T(x, b) = \gamma_T'(x, a)$, we get

\[ \nu(x, a) = |T[x \lor a]| \gamma_T(x, a) = |T'[\sigma(x) \lor \sigma(b)]| \gamma_{T'}(x, b) = \nu'(x, b), \]
\[ \nu(x, b) = |T[x \lor b]| \gamma_T(x, b) = |T'[\sigma(x) \lor \sigma(a)]| \gamma_{T'}(x, a) = \nu'(x, a). \]

Hence, the edge weights are merely swapped around by $\sigma$. Case $E_3$ is proven by a symmetric argument.

Case $E_4$: Let $X$ split into $(A, B)$ at the root of $T$ so that $b \in A$ and $a \in B$, and let $(a, x) \in E_4$. Now, if $x \in A$, then $\nu(a, x) = -|X|$, which is the lowest possible value, so $\nu'(a, x) \geq \nu(a, x)$. And if $x \in B$, then $\nu'(x, a) = |X|$, which is the largest possible value, so also in this case, $\nu'(a, x) \geq \nu(a, x)$.

A symmetric argument covers case $E_5$.

Recalling the random family of graphs over a partially ordered set given in Example 1, the following corollary states that in expectation, the trees of maximal value are order preserving.

**Corollary 16.** Let $(X, \leq)$ be a partially ordered set, and let $G = (V, E)$ be a graph according to the random family of Example 1. If we define $\omega = I_E$, and if $T$ is a non-order preserving tree with respect to $\leq$, then there exists an order preserving tree $T' \in \mathcal{F}(X)$ with $\text{Eval}_g(T') > \text{Eval}_g(T)$.

**Proof.** First, for $\alpha > 0$, because $\text{val}_{ag} = \alpha \text{val}_g$, the optimal trees under $\text{val}_{ag}$ coincide with the optimal trees under $\text{val}_g$. Hence, optimisation under $\text{val}_g$ is invariant to positive scaling of $g$.

Second, given $(X, \omega)$, the optimal trees under $\text{val}_g$ are invariant with respect to a large class of affine transformations of $\omega$. This is because if $\alpha, \beta \in \mathbb{R}$ with $\alpha > 0$, and if we define $\omega' = \alpha \omega + \beta$, we get

\[ g'(x, y) = \omega'(x, y) - \omega'(y, x) = \alpha \omega(x, y) + \beta - \alpha \omega(y, x) - \beta = \alpha g(x, y). \]

Now, given $\omega$ from Example 1, and defining $\omega' = \frac{1}{p-q} (\omega - q)$, we get $\omega' = I_{\leq}$. That is, the optimal trees under $\text{val}_g$ for $\omega = I_{\leq}$ coincide with the optimal trees under $\text{val}_g$ for $\omega$ as given in Example 1. Since the latter is the expected value of $\omega$ for a graph from the random family, the corollary follows. \qed
4.2 Efficacy on planted partial orders

In this section, we provide quantitative results on the difference in order preservation between optimal trees and suboptimal trees for a simple class of partial orders. We show that our model partitions partially ordered input with power comparable to how Dasgupta’s model partitions cliques.

For the duration of this section, we fix $|X| = n$ even.

4.2.1 Bounding the number of reversed pairs for planted bipartite partial orders

We start by defining our notion of a planted partial order.

**Definition 17.** Let $X$ be a set, $(A^*, B^*)$ a split of $X$ into equally large blocks, and $p, q$ real numbers for which $0 \leq q < p \leq 1$. Let $\leq^*$ denote the smallest partial order on $X$ satisfying $(x, y) \in A^* \times B^* \Rightarrow x \leq^* y$, and let $\Gamma$ denote the stochastic family of directed graphs $G = (X, E)$ for which

$$
\Pr((x, y) \in E) = \begin{cases} 
p & \text{if } x \leq^* y, 
q & \text{otherwise}.
\end{cases}
$$

We refer to the partially ordered set $(X, \leq^*)$ as the **planted bipartite partial order defined by** $X$, $A^*$, $B^*$, $p$ and $q$.

Notice that if $G = (X, E)$ is drawn from $\Gamma$, then $G$ gives rise to a relaxed order on $X$ defined by $\omega_G = I_{\leq^*}$. In particular, we have $E[\omega_G(x, y)] = \Pr((x, y) \in E)$, so that if we have $g(x, y) = \omega_G(x, y) - \omega_G(y, x)$, we get

$$
Eg(x, y) = \begin{cases} 
p - q & \text{if } x <^* y, 
q - p & \text{if } y <^* x, 
0 & \text{otherwise}.
\end{cases}
$$

As a consequence, any tree that splits into $(A^*, B^*)$ at the root is of maximal expected value.

The next definition serves to define the quality of a binary tree in terms of order preservation. For a partially ordered set $(X, \leq)$, we write $[<]$ to denote the number of comparable pairs of distinct elements under $\leq$; that is $[<] = |\{(x, y) \in X \times X \mid x < y\}|$.

**Definition 18.** Let $(X, \leq)$ be a partial order, and let $\delta \in [0, \frac{1}{2}]$. A tree $T \in \mathcal{B}(X)$ is $\delta$-good (with respect to $\leq$) if, for every split $(A, B)$ in $T$, we have

$$\frac{\mathcal{I}_{\leq}(B, A)}{[<]} \leq \delta.$$ 

That is, the fraction of reversed pairs over the split against the total number of comparable pairs in $(X, \leq)$ is no larger than $\delta$.

Clearly, a tree is order preserving if and only if it is 0-good. Also, for the bipartite planted partial order, any tree that splits into $(A^*, B^*)$ at the root is 0-good with respect to $\leq^*$.

The first result of this section shows that if we define our value function based on a random graph from $\Gamma$, then there is a significant difference in expected value between an optimal tree and one that is not $\delta$-good.
Lemma 19. Let $G = (X, E)$ be drawn from $\Gamma$, and define $g(x, y) = I_E(x, y) - I_E(y, x)$. If $T^*$ is a tree that splits $X$ into $(A^*, B^*)$ at the root, and if $T$ is a tree that is not $\delta$-good with respect to $\leq^*$, then
\[
\text{Eval}_g(T^*) > \text{Eval}_g(T) + (n + 2)(p - q)\delta n^2/4.
\]

Proof. Since $T$ is not $\delta$-good, there is a split $(A, B)$ in $T$ where $I_{\leq^*}(B, A) / [\leq^*] > \delta$. By construction, we have $[\leq^*] = |A^*||B^*| = n^2$, and since the split $(A, B)$ involves reversed pairs, we must have $|A \cup B| \geq 2$. Hence, the reversed pairs contribute negatively in the split with magnitude
\[
(p - q)|A \cup B|I_{\leq^*}(B, A) > |A \cup B|(p - q)\delta[\leq^*] \geq 2(p - q)\delta[\leq^*] = 2(p - q)\delta n^2/4.
\]

Moreover, the pairs contributing to $I_{\leq^*}(B, A)$ fail to contribute positively to the value of the root split by a magnitude
\[
|X|(p - q)I_{\leq^*}(B, A) \geq n(p - q)\delta[\leq^*] = n(p - q)\delta n^2/4.
\]

Adding the magnitudes yields the statement of the lemma. \hfill \square

The above result is on the expected value $\text{Eval}_g(T)$. The next result bounds the difference between $\text{Eval}_g(T)$ and $\text{val}_g(T)$.

Lemma 20. Let $G = (X, E)$ be a random graph from $\Gamma$, let $g(x, y) = I_E(x, y) - I_E(y, x)$, and pick $\varepsilon \in (0, 1)$. Then we have, for any $T \in \mathcal{B}(X)$, with probability $1 - \varepsilon$,
\[
|\text{val}_g(T) - \text{Eval}_g(T)| < n^2 \sqrt{2n \ln 2n + \ln \frac{2}{\varepsilon}}.
\]

Proof. The proof is identical to that of (Dasgupta, 2016, Lemma 8) with one modification: If, for every pair $x <_T y$ of elements in $X$, we consider the functions $g(x, y)$ to be separate independent random variables, then the magnitude of change of $\text{val}_g(T)$ is bounded by $2n$ whenever only one of these random variables are allowed to change. This because $g \in [-1, 1]$. The result now follows from an application of McDiarmid’s inequality and Dasgupta’s proof. \hfill \square

We now combine the two above lemmas, showing that for a random graph drawn from the family corresponding to the bipartite planted partial order, that a tree that is optimal with respect to that graph has an upper bound on the fraction of reversed pairs given by $O(\sqrt{(\log n)/n})$.

Theorem 21. Let $G = (X, E)$ be a random graph from $\Gamma$, let $g(x, y) = I_E(x, y) - I_E(y, x)$, and pick $\varepsilon \in (0, 1)$. If $T$ is an optimal tree for $\text{val}_g$, then $T$ is $\delta$-good with respect to $\leq^*$ with probability $1 - \varepsilon$ for
\[
\delta = \frac{8}{p - q} \sqrt{\frac{2n \ln 2n}{n} + \frac{1}{n^2 \ln \frac{2}{\varepsilon}}}.
\]

Proof. Let $T^*$ be a tree that splits into $(A^*, B^*)$ at the root. Since $\text{val}_g(T) \geq \text{val}_g(T^*)$, we can deduce that
\[
\text{Eval}_g(T^*) - \text{Eval}_g(T) = \text{Eval}_g(T^*) - \text{val}_g(T^*) + \text{val}_g(T^*) - \text{Eval}_g(T)
\leq \text{Eval}_g(T^*) - \text{val}_g(T^*) + \text{val}_g(T) - \text{Eval}_g(T)
\leq |\text{Eval}_g(T^*) - \text{val}_g(T^*)| + |\text{val}_g(T) - \text{Eval}_g(T)|
\leq 2n^2 \sqrt{2n \ln 2n + \frac{2}{\varepsilon}}.
\]
with probability $1 - \varepsilon$. Hence, we have
\[
\text{Eval}_g(T^*) \leq \text{Eval}_g(T) + 2n^2\sqrt{2n \ln 2n + \frac{2}{\varepsilon}}
\]
with probability $1 - \varepsilon$ too.

Now, if $T$ is not $\delta$-good, Lemma 19 yields
\[
\text{Eval}_g(T) + (n + 2)(p - q)\delta[<^*] \leq \text{Eval}_g(T) + 2n^2\sqrt{2n \ln 2n + \frac{2}{\varepsilon}},
\]
giving us
\[
\delta \leq \frac{2n^2\sqrt{2n \ln 2n + \frac{2}{\varepsilon}}}{(n + 2)(p - q)[<^*]} < \frac{2n^2\sqrt{2n \ln 2n + \frac{2}{\varepsilon}}}{(p - q)^{n^2/\tau}} = \frac{8}{p - q} \sqrt{\frac{2 \ln 2n}{n} + \frac{1}{n^2} \ln \frac{2}{\varepsilon}}.
\]

On the number of comparable pairs in the bipartite planted partial order. By construction, the number of comparable pairs between $A^*$ and $B^*$ in the planted partial order is $n^2/\tau$, which is as high as it can be. An interesting question is to how many pairs we need in order to be able to separate the two blocks. Re-tracing the steps in the above calculations while letting $[<^*]$ denote the number of comparable pairs between the blocks, gives us a fraction of reversed pairs of $O\left(\frac{n^2}{[<^*]}\right)\sqrt{\log(n)/n}$. Indeed, if we choose to have as few comparable pairs as possible while still separating the two blocks, we can manage with $[<^*] = \frac{n}{\tau}$. This yields an asymptotic fraction of $O\left(\sqrt{n \log n}\right)$ reversed pairs.

For the fraction of reversed pairs to diminish as $n \to \infty$, we need $[<^*] = \Omega(n^2)$. This can be achieved, for example, by having all points in a fixed fraction $\alpha$ of the elements of $A^*$ being related to all points of an equally large fraction of $B^*$, while the remaining pairs of points are related to exactly one point in the other set. This yields
\[
[<^*] = \alpha^2 n^2/4 + (1 - \alpha)n/2 = \Omega(n^2)
\]
for any choice of $\alpha > 0$.

### 4.3 A worked example – migration between states

This section presents a worked example, analysing migration flow between states in the USA. The purpose of the example is to illustrate how order preserving hierarchical clustering with val$_g$ behaves on data that is not partially ordered, but has a directed nature. As the example shows, the produced hierarchical clustering is similar to what can be obtained by applying other methods already in use for hierarchical clustering, such as MAXDiCut (Feige and Goemans, 1995) and DIRECTEDSPARSESTCUT (Chuzhoy and Khanna, 2006). However, for these methods, it is not obvious how to combine them with a similarity.

The states being subject to this analysis, and the migration data, is presented in Table 2. The values are obtained from http://www.census.gov, and represent migration data between US states in the year 2011.

We let the function $\omega$ represent the flow of people moving from one state to another in the course of one year. The hierarchical clustering proceeds by splitting the set of states in two,
Table 2: Migration data normalised so that the total migration sums to one. The table displays the migration magnitudes in “from row to column”-fashion.

| State     | 1  | 2  | 3  | 4  | 5  | 6  | 7  |
|-----------|----|----|----|----|----|----|----|
| Arizona   | 1  | –  | 0.064 | 0.006 | 0.018 | 0.014 | 0.012 | 0.022 |
| California| 2  | 0.089 | –  | 0.016 | 0.072 | 0.061 | 0.033 | 0.069 |
| Idaho     | 3  | 0.004 | 0.009 | –  | 0.007 | 0.011 | 0.014 | 0.020 |
| Nevada    | 4  | 0.016 | 0.065 | 0.006 | –  | 0.013 | 0.008 | 0.009 |
| Oregon    | 5  | 0.008 | 0.033 | 0.013 | 0.003 | –  | 0.004 | 0.052 |
| Utah      | 6  | 0.019 | 0.016 | 0.011 | 0.006 | 0.006 | –  | 0.009 |
| Washington| 7  | 0.025 | 0.065 | 0.016 | 0.008 | 0.039 | 0.009 | –  |

Figure 3: Figure showing the result of the clustering of the migration data. The binary tree is displayed to the left, and the sequence of splits of the states are shown to the right. The arrows on the splits indicate the direction of net migration.
providing the two blocks of states having maximal net flow of migrants from one block to the other. Then these blocks will be split accordingly, and so on, until only single states remain.

The result of the hierarchical clustering of this data is presented in Figure 3. We see that the first state to be split off is California, indicating that the largest flow of migration is from California to all the other states. Second, Washington is split off from the remaining states, and the orientation (arrow) tells us that the net flow of migration is from Arizona, Idaho, Oregon, Nevada and Utah to Washington. Third, Oregon and Idaho is split off from Nevada, Utah and Arizona, with net migration from the latter group to the former, and so on. The linear order on the states induced by the binary tree is

$\text{Ca} \leq_T \text{Ut} \leq_T \text{Az} \leq_T \text{Nv} \leq_T \text{Or} \leq_T \text{Id} \leq_T \text{Wa}$,

indicating the general direction of net migration flow.

5 Properties of $f = s_d + g$

We now turn to study the full objective function $f$ as defined in (9); that is, the sum of the dissimilarity $s_d$ and the antisymmetrisation $g$ of the relaxed order relation.

We start by looking at an example, showing how we can use order preserving hierarchical clustering to recover the ancestral tree of former U.S. president John F. Kennedy. The example illustrates how the dissimilarity and the order relation combine to successfully recover the ancestral tree in a situation where neither objective could have managed alone.

The example also shows us that the two objectives must be balanced in order to achieve both good clustering and good order preservation. This is the topic of the second part of the section, where we consider the problem of finding the Pareto optimal trees in the context of bi-objective optimisation. We show that due to the linearity of the objectives, we can recover the entire Pareto front by studying the convex combinations of $s_d$ and $g$.

5.1 Analysing the JFK ancestral tree

We will now illustrate how the dissimilarity and the order relation combine to successfully recover the ancestral tree of John F. Kennedy in a situation where neither of the objectives can do this alone. The idea is to cluster the individuals in the tree so that closely related individuals are grouped together, while at the same time keeping the generations apart. The dissimilarity is derived from name differences, and the order relation is based on the ancestor-descendant relations among the individuals. The ancestral tree is depicted in Figure 4, and the dissimilarities are listed in Table 3.

| 1: John F. Kennedy | 2 | 3 | 4 | 5 | 6 | 7 |
|--------------------|---|---|---|---|---|---|
| 2: Joseph P. Kennedy | 1 | 0.29 | 0.31 | 0.53 | 0.53 | 0.50 | 0.63 |
| 3: Rose F. Kennedy | 2 | 0.40 | 0.50 | 0.59 | 0.62 | 0.73 |
| 4: Patrick J. Kennedy | 3 | 0.61 | 0.53 | 0.65 | 0.70 |
| 5: Mary A. Kennedy | 4 | | 0.44 | 0.50 | 0.47 |
| 6: John F. Fitzgerald | 5 | | 0.65 | 0.56 |
| 7: Mary J. Fitzgerald | 6 | | | | 0.28 |

Table 3: Dissimilarities of names in the ancestral tree. On the left are mappings from individuals to indices, and on the right is the table of name dissimilarities, rounded to two decimal places. The dissimilarities correspond to the Jaccard distances between the names.
Notice that, in the ancestral tree, the Kennedy name is present throughout, and in particular across the generations, yielding high similarities between descendant- and ancestor names. Due to this, attempting to cluster using only the dissimilarity will cause descendants and ancestors to be placed in the same cluster.

On the other hand, defining $\omega$ so that $\omega(x, y) = 1$ if and only if $x$ is a descendant of $y$ (and zero otherwise), this will keep generations apart, but $\omega$ holds no information about how to identify groups of individuals within the same generation.

If we combine the two objectives as in $f = s_d + g$, the corresponding optimal hierarchical clustering is presented in Figure 5. We see that the generations are nicely split apart, and the grandparents are nicely grouped together, as we wished for. Hence, by combining the dissimilarity and the order relation, the result turns out correctly.

Looking back at our motivating use case (Section 1.1), the database of machinery exhibits the exact same tendency: The high similarity between parts and sub-parts makes clustering difficult, but we can mitigate this problem by taking the part-of relation into account.

### 5.2 Balancing clustering against order preservation

In the above example, the dissimilarity $s_d$ and the order relation $\omega$ are, in a sense, competing: the dissimilarity wants to place similar elements together, in spite of them being in different generations, and the order relation tries to keep the generations apart. In the field of operations
research, this is termed a bi-objective optimisation problem, having two distinct objectives where ideally both shall be optimised for. Multi-objective optimisation is a thoroughly studied area of research, and several methods exists to approach this class of problems. See, for example, the survey by Marler and Arora (2004) for an overview.

The material presented in this section are known results within multiobjective optimisation. We still choose to write it out, for the sake of completeness and also because it reveals an efficient approach to explore different optimal solutions.

We have chosen to focus on Pareto optimality (Zadeh, 1963) in the context of bi-objective maximisation. In order to do that, for an ordered similarity space \((X, s, \omega)\), we decompose \(v_f\) into its sum components \(v_{sd}\) and \(v_g\):

\[
\text{Definition 22. Let } (X, s, \omega) \text{ be an ordered similarity space. For trees } T, T' \in \mathcal{B}(X) \text{ we say that } T \text{ is Pareto dominated by } T' \text{ if }
\]

1. \(v_{\theta}(T') \geq v_{\theta}(T)\) for \(\theta \in \{s, g\}\), and
2. for at least one objective \(v_{\theta}\) we have \(v_{\theta}(T') > v_{\theta}(T)\).

A solution is Pareto optimal if there are no solutions dominating it, and the family of all Pareto optimal solutions is referred to as the Pareto front.

Thus, a Pareto optimal solution has the property that for any other candidate solution, at least one objective will deteriorate. An illustration of the Pareto front is given in Figure 6.

Figure 6: The Pareto front of a bi-objective optimisation problem with objectives \((v_{sd}, v_g)\). The squares are the different candidate solutions, and the solid line connects the candidates at the Pareto front. The shaded wedge illustrates why the black square belongs on the Pareto front: there are no solutions above or to the right of this point. The dashed line indicates the convex hull of the Pareto front.

To identify the Pareto front for a bi-objective optimisation problem, the naive approach is to identify the optimal solutions for all linear combinations \(\gamma v_{sd} + \delta v_g\) for \(\gamma, \delta > 0\).\(^2\) However, as the following definition and lemma shows, in our case, we can limit the study to convex combinations of the objectives. We start by introducing a new value function \(v_{\alpha}\), which is a convex combination of the two objectives.

\(^2\)More efficient methods exist, such as (Kim and de Weck, 2005).
Lemma 24. For a suitable value of the order relation, and \( \alpha \), the generalisation of the optimisation problem is thus to find an \( \alpha \)-optimal tree. For example, binary search can be applied for exploring the Pareto front. Since there is no canonical best solution on the Pareto front, and therefore no canonically best tree, the objectives given the current context. For multi-objective optimisation in general, there is significant amount of research has focused on the topic of aiding the domain expert in identifying the best solution; see Miettinen’s book (Miettinen, 1998, pp. 131–213) for an overview.

Definition 23. Given \((X, s, \omega)\) and \( \alpha \in [0, 1] \), define \( \text{val}_\alpha : \mathcal{B}(X) \to \mathbb{R} \) as
\[
\text{val}_\alpha(T) = \sum_{x \leq y} |T[x \vee y]| \left[ \alpha s_d(x, y) + (1 - \alpha) g(x, y) \right].
\]

Notice that due to the linearity of \( \text{val}_{s_d} \) and \( \text{val}_g \), we have
\[
\text{val}_\alpha(T) = \alpha \text{val}_{s_d}(T) + (1 - \alpha) \text{val}_g(T).
\]

Optimising a linear combination \( \gamma \text{val}_{s_d} + \delta \text{val}_g \) can be replaced by optimising \( \text{val}_\alpha \) for a suitable value of \( \alpha \):

Lemma 24. For \( \gamma, \delta > 0 \), let \( \alpha = \frac{\gamma - \delta}{\gamma + \delta} \). Then \( T \in \mathcal{B}(X) \) maximises \( \text{val}_\alpha \) if and only if \( T \) also maximises \( \gamma \text{val}_{s_d} + \delta \text{val}_g \).

Proof. For a fixed \( \beta > 0 \), a tree that maximises \( \text{val}_{s_d} + \text{val}_g \) also maximises \( \beta (\text{val}_{s_d} + \text{val}_g) \). And since
\[
\frac{1}{\gamma + \delta} (\gamma \text{val}_{s_d} + \delta \text{val}_g) = \frac{\gamma}{\gamma + \delta} \text{val}_{s_d} + \frac{\delta}{\gamma + \delta} \text{val}_g = \alpha \text{val}_{s_d} + (1 - \alpha) \text{val}_g,
\]
and since \( 0 \leq \frac{\gamma}{\gamma + \delta}, \frac{\delta}{\gamma + \delta} \leq 1 \) and \( \frac{\gamma}{\gamma + \delta} + \frac{\delta}{\gamma + \delta} = 1 \), the lemma holds.

According to the above Lemma, the Pareto front is convex in the sense that all Pareto optimal solutions are located on the convex hull indicated by the dashed line in Figure 6. Also, this means that if \( 0 \leq \beta \leq \beta' \leq 1 \) and \( \text{val}_\beta \) and \( \text{val}_{\beta'} \) have the same optimal trees, then optimisation of \( \text{val}_\alpha \) is constant for \( \alpha \in [\beta, \beta'] \). This is useful, since it gives a well defined basis on which to apply, for example, binary search to explore the Pareto front.

The objective \( \text{val}_\alpha \) has the property that \( \text{val}_{\alpha=0} = \text{val}_g \), optimising only with respect to the order relation, and \( \text{val}_{\alpha=1} = \text{val}_{s_d} \), optimising only with respect to the similarity. Part of the optimisation problem is thus to find an \( \alpha \in [0, 1] \) providing the best balance between the objectives given the current context. For multi-objective optimisation in general, there is no canonical best solution on the Pareto front, and therefore no canonically best \( \alpha \) for \( \text{val}_\alpha \). A significant amount of research has focused on the topic of aiding the domain expert in identifying the best solution; see Miettinen’s book (Miettinen, 1998, pp. 131–213) for an overview.

While we believe it is possible to produce a theorem along the lines of Theorem 21 also in the presence of both a similarity and an order relation, we consider this to be a major task, and postpone this for future research.

5.3 A note on idempotency

A function \( h : X \to X \) is idempotent if \( h \circ h = h \). Jardine and Sibson (1971) define a clustering method as appropriate if it is idempotent. This is considered to be a key feature of any clustering methodology. Cohen-Addad et al. (2019) show that Dasgupta’s model is appropriate, and the below theorem shows that this still holds for order preserving hierarchical clustering. An important difference is, of course, that we must pass along the induced order relation for the second invocation. Notice that for a normalised ultrametric \( u_T \) on \( X \), the function \( 1 - u_T \) is a similarity on \( X \).

Theorem 25. Assume that \((X, s, \omega)\) has \( T \) as an optimal tree under \( \text{val}_f \), and let \( u_T \) be the normalised ultrametric corresponding to \( T \). Then \((X, 1 - u_T, \leq_T)\) has \( T \) as an optimal tree under \( \text{val}_f \). That is; \( \text{val}_f \) is appropriate.
Proof. Since \( \text{val}_{s_d} \) is appropriate, clustering the similarity space \((X, 1 - u_T)\) without any order relation will yield a binary tree \(T'\) that reproduces \(u_T\) under \(\text{val}_{u_T}\). This tree may not be order preserving, but it is isomorphic to \(T\), and this tree isomorphism must necessarily be a pairwise swapping of elements so that the isomorphism is induced by \(T'\) and these swaps. Hence, according to Theorem 15, \(T\) has at least the value of \(T'\) under \(\text{val}_f\).

The following corollary expresses the same result from a different angle: if we do not care about the induced order relation but only about the ultrametric, we can omit the order relation all together, given that we have a suitable similarity.

**Corollary 26.** For every ordered similarity space \((X, s, \omega)\), if \(T\) is an optimal tree with respect to \(\text{val}_f\), then there exists a similarity \(s'\) on \(X\) for which \(T'\) is an optimal tree of \(\text{val}_{s'_d}\) and where \(u_{T'} = u_T\).

### 6 Approximation

Since optimisation of \(\text{val}_{s_d}\) is NP-hard, optimisation of \(\text{val}_f\) is NP-hard too. In this section, we present a polynomial time approximation algorithm with a relative performance guarantee of \(O(\log^{3/2} n)\). The method is based on successive applications of \textsc{DirectedSparsestCut} (Chuzhoy and Khanna, 2006) to produce an order preserving tree. The method, thus, falls in the category of divisive hierarchical clustering. The \textsc{DirectedSparsestCut} is similar to the more common \textsc{SparsestCut} (Leighton and Rao, 1999), with the obvious difference that in \textsc{DirectedSparsestCut}, one attempts to have as many arcs as possible pointing in the same direction across the cut.

Notice that graph cuts, and what we have called splits, are the same thing. In a split, we focus on splitting a set in two, while for a graph cut, we focus on cutting the arcs spanned by the split. However, since every relaxed order relation \(\omega : X \times X \to [0, 1]\) corresponds to a complete directed weighted graph, the concepts coincide.

**Definition 27.** Given a complete directed weighted graph over \(X\) with weight function \(\nu\), the **directed cut density** of a split \((A, B)\) of \(X\) is the magnitude

\[
\frac{\nu(A, B)}{|A||B|} = \frac{\sum_{a,b} \nu(a,b)}{|A||B|}.
\]

In particular, a **directed sparsest cut** is a split of minimal cut density, taken over all possible binary splits \((A, B)\) of \(X\).

Notice that, just as for splits in trees, a directed cut is oriented, in the sense that \((A, B)\) may have a different cut density than \((B, A)\).

Also notice that, for an undirected graph, the weight function is symmetric, so that the cut densities of \((A, B)\) and \((B, A)\) coincide. In this case, we drop the prefix, and refer to \((A, B)\) as a possibly **sparsest cut**.

### 6.1 Duality

To use \textsc{DirectedSparsestCut} for our problem, we must solve it as a minimisation problem. We therefore introduce the dual to \(\text{val}_f\), namely \(\text{cost}_{f_d}\), and show that maximisation under \(\text{val}_f\) is equivalent to minimisation under \(\text{cost}_{f_d}\). We start by showing that there is a dual function to \(g\), denoted \(g_d\), that allows us to maximise \(\text{val}_g\) by minimising \(\text{cost}_{g_d}\). Thereafter, we combine \(\text{cost}_{g_d}\) with the non-dual \(\text{cost}_s\) to make up \(\text{cost}_{f_d}\).
For an ordered set \((X, \omega)\), we define the **dual of the antisymmetrisation** of \(\omega\) to be the function
\[
g_d(x, y) = 1 - g(x, y).
\] (18)
Notice that \(g = 1 - g_d\), meaning that the two functions are each others duals. Notice also that \(g_d : X \times X \to [0, 2]\), eliminating the negative coefficients in the optimisation problem.

**Lemma 28.** Let \((X, \omega)\) be given, and let \(g_d\) be the dual of the antisymmetrisation of \(\omega\). Then the function \(\text{cost}_{g_d} : B(X) \to \mathbb{R}_+\) defined as
\[
\text{cost}_{g_d}(T) = \sum_{x \leq y} |T[x \lor y]| g_d(x, y)
\] (19)
is dual to \(\text{val}_g\) in the sense that any tree that maximises \(\text{val}_g\) also minimises \(\text{cost}_{g_d}\).

**Proof.** Recalling (Dasgupta, 2016, Theorem 3), stating that
\[
\sum_{x \leq y} |T[x \lor y]| = \frac{|X|^3 - |X|}{3}
\]
for all trees \(T \in B(X)\), this yields
\[
\text{cost}_{g_d}(T) = \sum_{x \leq y} |T[x \lor y]|(1 - g(x, y))
\]
\[
= \sum_{x \leq y} |T[x \lor y]| - \sum_{x \leq y} |T[x \lor y]| g(x, y)
\]
\[
= \frac{|X|^3 - |X|}{3} - \text{val}_g(T).
\]
Hence, any tree that maximises \(\text{val}_g\) is a tree that minimises \(\text{cost}_{g_d}\).

For an ordered similarity space \((X, s, \omega)\), we define the **dual split value function** \(f_d : X \times X \to [0, 3]\) as
\[
f_d(x, y) = 2 - f(x, y).
\]

**Theorem 29.** Given \((X, s, \omega)\) where \(s\) is a similarity and \(\omega\) is a relaxed order, the maximisation problem of Definition 12 can be solved by minimising
\[
\text{cost}_{f_d}(T) = \sum_{x \leq y} |T[x \lor y]| f_d(x, y).
\]

**Proof.** Since \(f_d = 2 - f = (1 - s_d) + (1 - g) = s + g_d\), it follows that \(\text{cost}_{f_d} = \text{cost}_s + \text{cost}_{g_d}\). If we let \(M = \frac{|X|^3 - |X|}{3}\), we have
\[
2M - \text{cost}_{f_d} = (M - \text{cost}_s) + (M - \text{cost}_{g_d}) = \text{val}_{s_d} + \text{val}_g = \text{val}_f.
\]
Hence, a tree that maximises \(\text{val}_f\), also minimises \(\text{cost}_{f_d}\).

**A remark on the dual of the antisymmetrisation.** The similarity \(s\) can be defined in terms of an undirected graph with weight function \(s\), and the dual \(s_d = 1 - s\) is also a graph. Indeed, if \(s\) has unit weights, then \(s_d\) is the graph with complementary edge set. However, for the order relation, we have a directed weighted graph with weight function \(\omega\), and with antisymmetrisation \(g(x, y) = \omega(x, y) - \omega(y, x)\). The dual of the antisymmetrisation \(g_d = 1 - g\) is dual to \(g\), but does not correspond to an antisymmetrisation of a relaxed order or weight function, since \(g_d\) itself is not antisymmetric. It is the dual graph to the complete graph over \(X\) with weight function \(g\), but what this dual graph represents is not obvious.
6.2 Approximation algorithm

This section describes the approximation algorithm, and provides an analysis of its relative performance guarantee. In the algorithm, we assume the existence of a directed cut approximation function; that is, a function that takes as arguments a vertex set together with a weight function, and produces an approximation of an optimal cut that comes with a relative performance guarantee.

Concretely, given an ordered similarity space \((X, s, \omega)\), if \(\theta\) is an approximation of \(\text{DirectedSparsestCut}\) with a relative performance guarantee \(\alpha\) and if \(\theta(X, f_d) = (A, B)\), then \(f_d(A, B)/(|A||B|)\) is no more than a factor \(\alpha\) off from a true sparsest directed cut of \((X, f_d)\).

The currently best known approximation algorithm of \(\text{DirectedSparsestCut}\) has a relative performance guarantee of \(O(\sqrt{\log n})\) (Agarwal et al., 2005).

**Definition 30.** Let \((X, s, \omega)\), and let \(\theta\) be an approximation algorithm of \(\text{DirectedSparsestCut}\). The following algorithm produces the approximate order preserving hierarchical clustering of \((X, f_d)\).

```plaintext
procedure MakeTree(X, f_d)
    if |X| = 1 then
        return X
    else
        Let \((A, B)\) be the cut approximation returned by \(\theta(X, f_d)\)
        Let LEFT_TREE be the tree returned by MakeTree(A, f_d)
        Let RIGHT_TREE be the tree returned by MakeTree(B, f_d)
        return (LEFT_TREE, RIGHT_TREE)
    end if
end procedure
```

Given an approximation of \(\text{DirectedSparsestCut}\) with an approximation guarantee of \(O(\sqrt{\log n})\), according to the following theorem, the above algorithm produces a binary tree that has a cost that is no higher than a factor \(O(\log^{3/2}n)\) of an optimal tree.

**Theorem 31.** Given \((X, E)\) be a complete directed weighted graph with weight function \(f_d\), and let \(\theta\) be an approximation algorithm of \(\text{DirectedSparsestCut}\). The following algorithm produces the approximate order preserving hierarchical clustering of \((X, f_d)\).

**Proof (sketch).** Since a \(\text{DirectedSparsestCut}\) is equivalent to a \(\text{SparsestCut}\) for an undirected graph, the above algorithm produces a binary tree that has a cost that is no higher than a factor \(O(\log^{3/2}n)\) of an optimal tree.

**Lemma 32.** Given \((X, s, \omega)\), for every binary tree \(T \in B(X)\), there exists a split \((A, B)\) of \(X\) for which

\[
\frac{f_d(A, B)}{|A||B|} \leq \frac{27}{2n^3} \text{cost}_{f_d}(T).
\]
Proof. The key is to control the cardinalities of the components of the split \((A, B)\), while at the same time making sure that the split is ordered according to the order induced by the tree.

For a tree \(T \in \mathcal{B}(X)\), we define a **maximum cardinality path** to be a path in \(T\) starting at the root, and for every node, the next node on the path is a child of the current node of maximal cardinality. Based on a maximum cardinality path \(\{N_i\}_{i=1}^m\), we can create the required split over the following steps:

1. Construct two sequences \(\{L_i\}_{i=1}^{m-1}\) and \(\{R_i\}_{i=1}^{m-1}\) of subsets of \(X\) as follows: For \(1 \leq i \leq m - 1\), let \(\ell(N_i)\) and \(r(N_i)\) be the left and right children of \(N_i\), respectively. Now, if \(|\ell(N_i)| < |r(N_i)|\), set \(L_i = \ell(N_i)\) and \(R_i = \emptyset\), otherwise set \(L_i = \emptyset\) and \(R_i = r(N_i)\). We refer to these sets as the \(i\)-th left split-off and \(i\)-th right split-off, respectively.

2. Define the set sequences \(\{A_k\}_{k=1}^{m-1}\) and \(\{B_k\}_{k=1}^{m-1}\) as

\[
A_k = \bigcup_{i=1}^k L_i \quad \text{and} \quad B_k = \bigcup_{i=1}^k R_i.
\]

We refer to \(A_k\) as the \(k\)-th accumulated left split-off, and \(B_k\) as the \(k\)-th accumulated right split-off.

3. Let \(K\) be the smallest natural number for which \(\max\{|A_K|, |B_K|\} \geq n/3\).

4. Finally, if \(|A_K| \geq n/3\), set \(A = A_K\) and \(B = X - A\). Otherwise, set \(B = B_K\) and \(A = X - B\).

We refer to \((A, B)\) as a **balanced** \(T\)-split, and call \(\{N_i\}_{i=1}^K\) the head sequence of the balanced \(T\)-split. The key points to take away from this construction are the following:

\begin{itemize}
  \item [b1.] \(|N_K| \geq n/3\),
  \item [b2.] \((a, b) \in A \times B \Rightarrow a \leq_T b\),
  \item [b3.] \(\cup_{i=1}^K \ell(N_i) = A\) and \(\cup_{i=1}^K r(N_i) = B\).
\end{itemize}

We prove each property in turn:

**b1:** Since \(\max\{|A_{K-1}|, |B_{K-1}|\} < n/3\), and since, by construction, \(X = \bigcup_{k=1}^m A_k \cup B_k\), and since these sets are disjoint, it follows that \(|N_K| \geq n/3\).

**b2:** Notice that the \(L_i\) and \(R_i\) are left- and right children of the head sequence elements. If \(x\) is the leaf of the maximum cardinality path, this means that \(a \in L_i \Rightarrow a \leq_T x\), and likewise, \(b \in R_i \Rightarrow x \leq_T b\), so \((A_K, B_K)\) is order preserving with respect to \(\leq_T\). Now, if \(|A_K| \geq n/3\), this means that \(A = A_K\) and \(B = B_K \cup r(N_K)\), which again means that \((A, B)\) is order preserving with respect to \(\leq_T\). A symmetric argument covers the case \(|B_K| \geq n/3\).

**b3:** If \(|A_K| \geq n/3\), then \(R_K = \emptyset\), so

\[
A = \bigcup_{i=1}^K L_i = \bigcup_{i=1}^K \ell(N_i),
\]

\[
B = \left(\bigcup_{i=1}^K R_i\right) \cup r(N_K) = \left(\bigcup_{i=1}^{K-1} r(N_i)\right) \cup r(N_K).
\]

And again, a symmetric argument attests \(b3\) for \(|B_K| \geq n/3\).
That proves the properties of the balanced $T$-split, and we are now in position to prove the statement of the lemma: Let $(A, B)$ be a balanced $T$-split, and let $\{N_i\}_{i=1}^K$ be the head sequence of the split. Then

$$\text{cost}_{f_d}(T) \geq \sum_{i=1}^K |N_i| f_d(\ell(N_i), r(N_i)) \geq \sum_{i=1}^K \frac{n}{3} f_d(\ell(N_i), r(N_i)) \geq \frac{n}{3} f_d(A, B).$$

From this, we deduce that

$$\frac{f_d(A, B)}{|A||B|} \leq \frac{3}{n} \cdot \frac{\text{cost}_{f_d}(T)}{(2n/3)(n/3)} = \frac{27}{2n^3} \text{cost}_{f_d}(T).$$

**Recovering intuition—DirectedDensestCut:** While the dual of the antisymmetrisation defies intuitive interpretation, we can recover our intuitive understanding of the approximation as follows. Assume that $(A, B)$ is a sparsest cut under $f_d$. Since

$$\frac{f_d(A, B)}{|A||B|} = \frac{\sum_{a \in A, b \in B} 2 - f(a, b)}{|A||B|} = \frac{2|A||B| - f(A, B)}{|A||B|} = 2 - \frac{f(A, B)}{|A||B|},$$

it follows that $(A, B)$ maximises $f(A, B)/(|A||B|)$. It seems natural to call a cut that maximises $f(A, B)/(|A||B|)$ a **directed densest cut (under $f$)**. From what the authors can find, this cut is not previously mentioned in the literature, but one may still assume the existence of an approximation algorithm for DirectedDensestCut. It is, thus, clear that the above approximation algorithm of cost $f_d$ is equivalent to successive applications of a DirectedDensestCut approximation to approximate a maximal tree under $val_f$.

### 7 Demonstration

In this section, we demonstrate the efficacy of the theory in recovering copy-paste relations from the machine parts dataset described in Section 1.1.

The demonstration makes use of the approximation scheme described in Section 6. As mentioned, there exist good approximation algorithms for DirectedSparsestCut. However, as we discuss further in Section 8, the availability of these approximations is scarce, so for the demonstration, we have settled on optimal DirectedSparsestCut. The downside is that the algorithm can only deal with fairly small sets, but, as we show below, the sample sizes still suffice to provide a valuable demonstration. For industrial applications, a proper DirectedSparsestCut approximation will be required.

#### 7.1 A model for planted partitions over machine parts data

The dataset we use is described in detail in the data article (Bakkelund, 2021b), and is the same dataset that was used for the demonstration in (Bakkelund, 2021a). The data article is backed by an open source implementation\(^3\) that produces planted partitions simulating the copy-paste mechanism described in Section 1.1.

In brief, the planted partitions are generated as follows: First, a sample $X_0$ of size $n$ is drawn from the original data, giving us an ordered similarity space $(X_0, s_0, \leq_0)$. We then duplicate $(X_0, \leq_0)$ a total of $m$ times. These duplications represent the copy-paste relations.

\(^3\)https://pypi.org/project/machine-parts-pp/
We denote the copies by \((X_j, \leq j)\) for \(1 \leq j \leq m\), and enumerate the sets as \(X_j = \{x^j_1, \ldots, x^j_n\}\) for \(0 \leq j \leq m\). This means that, for \(1 \leq i \leq n\) we have that \(x^j_i\) and \(x^k_i\) are copies of each other whenever \(j \neq k\). Now, set \(X = \bigcup_{j=0}^{m} X_j\), and let the order relation \(\leq\) on \(X\) be the union of the \(\leq_j\), so that \((X, \leq)\) is a partially ordered set. To define the similarity \(s : X \times X \to [0, 1]\), we proceed as follows: If \(x\) and \(y\) are in the same copy-paste component, then their similarity is the same as in the original component. And, if they are in different copy-paste components, then their similarity is that of the original similarity minus a stochastic value. Concretely, let \(N(\mu, \sigma^2)\) be the normal distribution located at \(\mu\) with scale \(\sigma^2\), and let \(Y \sim N(\mu, \sigma^2)\). We then define

\[
s(x^i_p, x^j_q) = \begin{cases} 
  s_0(x^0_p, x^0_q) & \text{if } i = j, \\
  s_0(x^0_p, x^0_q) - Y & \text{otherwise},
\end{cases}
\]

applying rejection sampling to ensure \(s(x^i_p, x^j_q) \in [0, 1]\). In this way, the internal cohesion in the copy-paste components are maintained, while the coupling between the copy-paste related items is reduced as \(\mu\) and \(\sigma^2\) increase.

The above process leaves us with two pieces of data:

1. The ordered similarity space \((X, \leq, s)\) to be clustered;
2. The hidden clustering (planted partition) representing the copy-paste relations, given by the clusters \(C_i = \{x^j_i\}_{j=0}^{m}\) for \(1 \leq i \leq n\).

### 7.2 Measures of clustering quality

To evaluate the quality of a binary tree \(T \in \mathcal{B}(X)\) relative a planted partition, we follow the method of (Bakkelund, 2021a, Sections 8 and 9), applying the following measures of clustering quality:

**Adjusted Rand index.** Adjusted Rand indices are some of the most popular measures of clustering quality, and the one-sided Rand indices are good indicators of the success in recovering planted partitions. To report to which degree the methods can recover the planted clustering, we use the one-sided adjusted Rand index (ARI) assuming that the candidate clustering is drawn from the family of all possible clusterings over the set (Gates and Ahn, 2017).

**Element wise adjusted order Rand index.** Bakkelund (2021a, §8.2.2) introduces a quality measure to determine to which degree the induced order relation between two different clusterings of the same ordered set coincide. We use this quality measure, \(\bar{oARI}\), to decide to which degree the reported clustering has an induced order relation that coincides with the induced order relation of the planted partition.

**Loops.** This quality measure computes the fraction of elements of \(X\) participating in cycles in the induced relation of the clustering. While simple, is pins down the error in order preservation for a clustering. For easier comparison alongside the other measures, we report on \(1 - \text{fraction}\), so that loops = 1 if there are no cycles, and loops = 0 if all elements participate in an cycle. The loops measure is described in detail in (Bakkelund, 2021a, §9.3).

Given a planted partition \(\{C_j\}_j\) over \(X\) and a tree \(T \in \mathcal{B}(X)\), to compute the quality of \(T\) with respect to the above measures, we start by identifying the flat clustering of \(T\) with the highest ARI,\(^4\) and then proceed to compute the other quality measures for this flat clustering.

\(^4\)We employed the clusim python library, version 0.4, to compute the ARI.
Although this puts a higher emphasis on ARI compared to the other measures, we would like to emphasise that both $\bar{o}$ARI and loops are central measures for order preserving clustering, and in particular for applications where order preservation is key.

7.3 Methods evaluated in the demonstration

To demonstrate the method described in this paper, we use the approximation method from Section 6 to approximate a good tree for $\val_\alpha$ using optimal \textsc{DirectedSparsestCut}. We denote this method by $\val_\alpha$.

For comparison, we run a selection of other methods on the same problem instances. From (Bakkelund, 2021a), we know that the method provided by that article\textsuperscript{5} is currently best in class for this type of problems. This method is denoted by $\HC_{30,\varepsilon}^C$, representing a 30-fold approximation using complete linkage (Bakkelund, 2021a, Definition 14).

Also, to compensate for the lack of hierarchical clustering methods that support do-not-cluster constraints, Bakkelund (2021a, §9.2) presents a method termed \textit{simulated constrained clustering} based on the following trick. The initial similarity is modified so that every pair of comparable elements have similarity zero. As demonstrated there, when used with classical hierarchical clustering and complete linkage, this provides a significant improvement on the results, compared to just classical hierarchical clustering and complete linkage without the trick. We do the same here, modifying the similarity and approximating a good tree using Dasgupta’s method and no order relation. We denote the method by $\val_1^0$, since $\alpha = 1$ means we ignore the order relation, and the zero indicates that all comparable elements have zero similarity. As we demonstrate below, the trick has a significant effect, but it does not compete with the objective function we have presented in this work.

Finally, we also compare with classical hierarchical clustering with complete linkage,\textsuperscript{6} denoting this method by $\HC^C$.

7.4 Execution and results

A number of experiments were run with different parameter settings and on different parts of the dataset. While the magnitudes of the quality measures varied between the different parameter settings, the trend was the same in all cases. Even the order of the methods with respect to quality of results hardly changed between experiments. We have picked a subset of the experiments, presented in Table 4, to represent the overall observations. For each parameter set in Table 4, we ran 50 experiments,\textsuperscript{7}

where each experiment can be outlined as

1. Given the parameters, create a planted partition based on the procedure in Section 7.1;
2. Run each of the methods on the generated partially ordered similarity space;
3. Evaluate the output of each method according to the listed quality measures.

The mean values reported in the below plots are the means over these 50 executions.

7.4.1 Varying $\alpha$

Figure 7 shows the effect of varying $\alpha$ through $[0, 1]$. Recall that for $\alpha = 0$, it is only the order relation that determines the tree. The abrupt change for arbitrary small $\alpha > 0$ is due to

\textsuperscript{5}We employed the \texttt{ophac python} library version 0.4.5 for this demonstration.

\textsuperscript{6}We employed \texttt{scipy.cluster.hierarchy} from \texttt{scipy} version 1.7.1.

\textsuperscript{7}To ensure that convergence was reached with 50 experiments, we ran additional tests with 200 experiments for a small selection of parameter combinations.
Table 4: Parameters for the presented experiments. If the range column contains more than one value, this means that we present plots where this parameter varies over the specified range. In that case, all other parameters are set to the value in the default column. Notice that for $\alpha$, we have no default value, but present the result for the $\alpha$ with the best mean ARI value.

| par. | range | default | explanation |
|------|-------|---------|-------------|
| cc   | 6     | 6       | connected component no. to draw from |
| $n$  | 5     | 5       | size of drawn sample |
| $m$  | 4     | 4       | number of copies to make |
| $\alpha$ | 0 ··· 1 | best | weight of order relation vs similarity |
| $\sigma^2$ | 0.05 ··· 0.4 | 0.15 | variance of subtracted noise |
| $\mu$ | 0.0 ··· 0.4 | 0.075 | location of subtracted noise |
| $d_{\min}$ | 1.0 | 1.0 | minimum average degree of the drawn sample |

Figure 7: The mean performance of $\hat{v}_{\alpha}$ for different values of $\alpha$, plotted together with the mean values of the other methods. Notice that the other methods are constant with respect to $\alpha$, explaining the horizontal plot lines. The bottom of the shaded region indicates the minimum observed value of the loops measure for $\hat{v}_{\alpha}$. 
tie resolution; in the experiment, the order relation is binary, and all relations have the same weight. This may in some cases lead to several equally valued, optimal binary trees. As soon as we introduce the similarity, the ties are broken, and the number of viable solutions drops. Additionally, since the similarity contains information identifying good trees, the average quality increases. But, as $\alpha \to 1$, the information provided by the order relation is gradually phased out, and the quality of the clustering decreases until it reaches the value on the right end of the plot.

Notice that for $\alpha = 1$, the ARI, $\bar{oARI}$ and loops values are those that would be obtained for the Dasgupta objective function, since $\bar{\text{val}}_1 = \bar{\text{val}}_{s_d}$. From the plot, we see that for ARI, the Dasgupta objective function outperforms $HC_{30,\varepsilon}$ with a small margin, even though $\bar{\text{val}}_{s_d}$ does not take the order relation into account.

It is also interesting to notice that $\bar{\text{val}}_{0}^1$, which neither uses the order relation, outperforms $HC_{30,\varepsilon}$ with significant margin with respect to ARI; an observation that is consistent through all the experiments we have done. We choose to conclude that these observations merely underline the significance of the Dasgupta objective function.

The loops measure deserves a separate comment. First, we note that $HC_{30,\varepsilon}$ has zero loops, which is guaranteed by that method. Second, from Theorem 15, we know that $\bar{\text{val}}_{\alpha=0}$, has no loops for an optimal tree. But what we see from the plots is that even the approximation $\bar{\text{val}}_{\alpha}$ returns nothing but loop-free trees for $\alpha$ significantly larger than zero. This is a very promising observation, indicating that $\bar{\text{val}}_{\alpha}$ can be used for applications where loops are unacceptable, such as classic use cases requiring acyclic partitioning of graphs. We note that further research is required to establish for which $\alpha > 0$ and under which assumptions on the order relation and the similarity such a guarantee can be set forth.

It should also be mentioned that for all the experiments, the best results were obtained for values of $\alpha$ close to zero. An open question is to whether this would also be the case for a weighted partial order without uniform weights.

### 7.4.2 Varying the variance

Figure 8 shows the effect of varying $\sigma^2$ through $[0.05, 0.4]$; that is, the variance of the noise added to the copy-paste similarities in (20). While each point in the plot of $\bar{\text{val}}_{\alpha}$ corresponds to the optimal value over all $\alpha \in [0, 1]$, from what we see across the experiments, more or less equally good results would be obtained by simply fixing $\alpha = 0.1$ or some other low value (see the comment at the end of Section 7.4.1).

As can be expected, all methods degrade in quality of the result when the variance in the similarities increases, but $\bar{\text{val}}_{\alpha}$ is clearly less affected by this than the other methods. Notice in particular that $\bar{\text{val}}_{\alpha}$ seems to be less affected by increase in variance than $\bar{\text{val}}_{1}^0$, although they are based on very much the same optimisation principles. At first, one may think that this is because the order relation is not modified by the noise, and this information is available to $\bar{\text{val}}_{\alpha}$, but not to $\bar{\text{val}}_{1}^0$. However, the order relation is utilised by $HC_{30,\varepsilon}$, and this method degrades just as severely as $\bar{\text{val}}_{1}^0$ when the variance increases. One possible explanation is that it is the combined objective function, utilising both the Dasgupta objective and the antisymmetrisation of the order relation (8) that causes $\bar{\text{val}}_{\alpha}$ to outperform the other methods.

Notice also that $\bar{\text{val}}_{\alpha}$ performs perfectly with respect to loops, regardless of the increasing noise variance.
Figure 8: The mean performances of the different methods for varying $\sigma^2$. The markers indicate for which values of $\sigma^2$ the experiments were conducted, and the other parameters are according to the default values of Table 4. For $\hat{\text{val}}_\alpha$, each point in the plot corresponds to the optimal value over all $\alpha \in [0, 1]$. Notice that for the loops-plots, the plot for $\mathcal{HC}^{\leq \epsilon}_{30, \epsilon}$ coincides with the plot of $\hat{\text{val}}_\alpha$. All trees returned by $\hat{\text{val}}_\alpha$ were loop-free for the chosen $\alpha$, hence, compared to Figure 7, there is no visible gray region in the plot.

Figure 9: The mean performances of the different methods for varying $\mu$. The markers indicate for which values of $\mu$ the experiments were conducted, and the other parameters are according to the default values of Table 4. For $\hat{\text{val}}_\alpha$, each point in the plot corresponds to the optimal value over all $\alpha \in [0, 1]$. Notice that for the loops-plots, the plot for $\mathcal{HC}^{\leq \epsilon}_{30, \epsilon}$ coincides with the plot of $\hat{\text{val}}_\alpha$. All trees returned by $\hat{\text{val}}_\alpha$ were loop-free for the chosen $\alpha$, hence, compared to Figure 7, there is no visible gray region in the plot.
7.4.3 Varying translation magnitude

Figure 9 shows the effect of varying $\mu$ through $[0, 0.4]$; that is, the location of the noise added to copy-paste similarities. As before, we present the result for the optimal $\alpha$ for $\text{val}_\alpha$.

We see that all methods, $\text{val}_\alpha$ included, are equally affected by increasing translation of similarities, but that $\text{val}_\alpha$ performs consistently better than all the other methods throughout all the experiments.

Also in this case, given a tree $T$ approximated by $\text{val}_\alpha$, the flat clustering with the best ARI is loop-free in every experiment.

8 Summary and future work

We have defined the concept of order preserving hierarchical clustering and classified the binary trees over a set that correspond to order preserving hierarchical clusterings. We have introduced the concept of a relaxed order relation, where the relations are weighted over $[0, 1]$, presented an objective function for hierarchical clustering of relaxed orders, and proven several beneficial properties of this function. In particular, whenever the relaxed order relation is a uniformly weighted partial order relation, an optimal tree is always order preserving. We have discussed the method in terms of bi-objective optimisation, where the order relation and the similarity are convexly combined, and provided an efficient basis on which to explore the Pareto front. We have provided a polynomial time approximation algorithm for the model, with a proven relative performance guarantee of at least $O(\log^{3/2} n)$, and, finally, we have demonstrated the method on the machine parts dataset, showing that our method outperforms existing methods with significant margin.

The following future research topics have been mentioned in the paper:

- Theorem 21 gives a probability bound for order preservation quality for $\text{val}_g$, and (Dasgupta, 2016, Theorem 9) gives a probability bound for clustering quality for $\text{val}_s$. We suggest to establish a probability bound for the combined order preservation- and clustering quality for $\text{val}_\alpha$.

- According to Theorem 15, if the order relation is a partial order, then the optimal trees for $\text{val}_g$ are order preserving. It follows that the same result holds for $\text{val}_\alpha$ when $\alpha = 0$. However, from the demonstration in Section 7.4.1, we see that the approximation $\text{val}_\alpha$ provides trees that are order preserving for $\alpha$ significantly larger than 0. For applications, a result that that can provide an order preservation guarantee for non-zero $\alpha$ would be of great value. It would allow the optimisation to take both the order relation and the similarity into account, and at the same time guarantee order preservation.

- The approximation bound $O(\log^{3/2} n)$ of $\text{val}_\alpha$ (Theorem 31) can most likely be improved, in the same manner that has been achieved for the Dasgupta objective function approximation (Charikar and Chatziafratis, 2017; Cohen-Addad et al., 2019; Roy and Pokutta, 2017).

Additionally, at the start of Section 7, we mentioned the lack of available implementations of approximations of $\textsc{DirectedSparsestCut}$. When trying to locate openly available implementations of such algorithms, we found none. A complicating factor is that the theory behind the $\textsc{DirectedSparsestCut}$ approximations is far from trivial, so implementing one’s own is not generally achievable unless already an expert in semidefinite programming. And finally, there are no known non-trivial approximations of $\textsc{DirectedSparsestCut}$ that provide useful
cuts. We therefore suggest a call for action within operations research to make available realisations of the existing approximation algorithms for \textsc{DirectedSparsestCut}, at least for the scientific community, and for not-unreasonably-small datasets.

Acknowledgements. I would like to express gratitude for funding to the DataScience@UiO innovation cluster (The Faculty of Mathematics and Natural Sciences, University of Oslo), the Department of Informatics (The Faculty of Mathematics and Natural Sciences, University of Oslo), and the SIRIUS Centre for Scalable Data Access (Research Council of Norway, project no.: 237898). I would also like to thank Henrik Forssell, Department of Informatics (University of Oslo), and Gudmund Hermansen, Department of Mathematics (University of Oslo), for invaluable comments, questions and discussions leading up to this work.

References

Amit Agarwal, Moses Charikar, Konstantin Makarychev, and Yury Makarychev. $O(\sqrt{\log n})$ approximation algorithms for min uncut, min 2cnf deletion, and directed cut problems. In Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing, STOC '05, page 573–581, New York, NY, USA, 2005. Association for Computing Machinery. ISBN 1581139608. doi: 10.1145/1060590.1060675. URL https://doi.org/10.1145/1060590.1060675.

Daniel Bakkelund. Order preserving hierarchical agglomerative clustering. Mach Learn, 2021a. URL https://doi.org/10.1007/s10994-021-06125-0.

Daniel Bakkelund. Machine part data with part-of relations and part dissimilarities for planted partition generation. Preprint, 2021b. URL https://bitbucket.org/Bakkelund/machine-parts-pp/src/master/latex/supp/machine-parts-pp.pdf.

Sugato Basu, Ian Davidson, and Kiri Wagstaff. Constrained Clustering: Advances in Algorithms, Theory, and Applications. Chapman & Hall/CRC, 1 edition, 2008. ISBN 1584889969, 9781584889960.

T.S. Blyth. Lattices and Ordered Algebraic Structures. Universitext. Springer London, 2005. ISBN 9781852339050. URL https://www.springer.com/gp/book/9781852339050.

Gunnar Carlsson and Facundo Mémoli. Characterization, stability and convergence of hierarchical clustering methods. J. Mach. Learn. Res., 11:1425–1470, August 2010. ISSN 1532-4435. doi: 10.1007/s10208-010-9141-9. URL http://www.jmlr.org/papers/v11/carlsson10a.html.

Gunnar Carlsson and Facundo Mémoli. Classifying clustering schemes. Foundations of Computational Mathematics, 13(2):221–252, Apr 2013. ISSN 1615-3383. doi: 10.1007/s10208-012-9141-9. URL https://doi.org/10.1007/s10208-012-9141-9.

Gunnar Carlsson, Facundo Mémoli, Alejandro Ribeiro, and Santiago Segarra. Hierarchical quasi-clustering methods for asymmetric networks. In Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32, ICML’14, pages II–352–II–360. JMLR.org, 2014. URL http://dl.acm.org/citation.cfm?id=3044805.3044932.

Moses Charikar and Vaggos Chatziafratis. Approximate hierarchical clustering via sparsest cut and spreading metrics. In Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA ’17, page 841–854, USA, 2017. Society for Industrial and Applied Mathematics.
Vaggos Chatziafratis, Rad Niazadeh, and Moses Charikar. Hierarchical clustering with structural constraints. In Jennifer Dy and Andreas Krause, editors, Proceedings of the 35th International Conference on Machine Learning, volume 80 of Proceedings of Machine Learning Research, pages 774–783. PMLR, 10–15 Jul 2018. URL http://proceedings.mlr.press/v80/chatziafratis18a.html.

Julia Chuzhoy and Sanjeev Khanna. Hardness of cut problems in directed graphs. In Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing, STOC ’06, page 527–536, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595931341. doi: 10.1145/1132516.1132593. URL https://doi.org/10.1145/1132516.1132593.

Vincent Cohen-Addad, Varun Kanade, Frederik Mallmann-Trenn, and Claire Mathieu. Hierarchical clustering: Objective functions and algorithms. J. ACM, 66(4), June 2019. ISSN 0004-5411. doi: 10.1145/3321386. URL https://doi.org/10.1145/3321386.

Sanjoy Dasgupta. A cost function for similarity-based hierarchical clustering. In Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing, STOC ’16, pages 118–127, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450341325. doi: 10.1145/2897518.2897527. URL https://dl.acm.org/doi/10.1145/2897518.2897527.

Ian Davidson and S. S. Ravi. Agglomerative hierarchical clustering with constraints: Theoretical and empirical results. In Alípio Mário Jorge, Luís Torgo, Pavel Brazdil, Rui Camacho, and João Gama, editors, Knowledge Discovery in Databases: PKDD 2005, pages 59–70, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

U. Feige and M. Goemans. Approximating the value of two power proof systems, with applications to max 2sat and max dicut. In Proceedings Third Israel Symposium on the Theory of Computing and Systems, pages 182–189, 1995. doi: 10.1109/ISTCS.1995.377033.

Alexander J. Gates and Yong-Yeol Ahn. The impact of random models on clustering similarity. Journal of Machine Learning Research, 18(87):1–28, 2017. URL http://jmlr.org/papers/v18/17-039.html.

Debarghya Ghoshdastidar, Michaël Perrot, and Ulrike von Luxburg. Foundations of comparison-based hierarchical clustering. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett, editors, Advances in Neural Information Processing Systems 32, pages 7456–7466. Curran Associates, Inc., 2019. URL http://papers.nips.cc/paper/8964-foundations-of-comparison-based-hierarchical-clustering.pdf.

J. Herrmann, J. Kho, B. Uçar, K. Kaya, and Ü. V. Çatalyürek. Acyclic partitioning of large directed acyclic graphs. In 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), pages 371–380, May 2017. doi: 10.1109/CCGRID.2017.101. URL https://hal.inria.fr/hal-01744603.

Julien Herrmann, M. Yusuf Özkaya, Bora Uçar, Kamer Kaya, and Ümit V. Çatalyürek. Multilevel algorithms for acyclic partitioning of directed acyclic graphs. SIAM J. Sci. Comput., 41(4):A2117–A2145, 2019. doi: 10.1137/18M1176865. URL https://doi.org/10.1137/18M1176865.

Melvin F. Janowitz. Ordinal and Relational Clustering. WORLD SCIENTIFIC, 2010. doi: 10.1142/7449. URL https://www.worldscientific.com/doi/abs/10.1142/7449.

Nicholas Jardine and Robin Sibson. Mathematical Taxonomy. Wiley series in probability and mathematical statistics. Wiley, 1971. ISBN 9780471440505.
Toshihiro Kamishima and Jun Fujiki. Clustering orders. In Gunter Grieser, Yuzuru Tanaka, and Akihiro Yamamoto, editors, Discovery Science, pages 194–207, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg. ISBN 978-3-540-39644-4. URL https://doi.org/10.1007/978-3-540-39644-4_17.

Il Yong Kim and Olivier de Weck. Adaptive weighted-sum method for bi-objective optimization: Pareto front generation. Structural and Multidisciplinary Optimization, 29:149–158, 2005. doi: 110.1007/s00158-004-0465-1. URL https://link.springer.com/article/10.1007/s00158-004-0465-1.

Donald E. Knuth. Big omicron and big omega and big theta. SIGACT News, 8(2):18–24, 1976. URL http://doi.acm.org/10.1145/1008328.1008329.

Tom Leighton and Satish Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. J. ACM, 46(6):787–832, November 1999. ISSN 0004-5411. doi: 10.1145/331524.331526. URL https://doi.org/10.1145/331524.331526.

Maciej Luczak. Hierarchical clustering of time series data with parametric derivative dynamic time warping. Expert Systems with Applications, 62:116 – 130, 2016. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2016.06.012. URL http://www.sciencedirect.com/science/article/pii/S0957417416302937.

R Timothy Marler and Jasbir S Arora. Survey of multi-objective optimization methods for engineering. Structural and multidisciplinary optimization, 26(6):369–395, 2004. URL https://link.springer.com/article/10.1007/s00158-003-0368-6.

Kaisa Miettinen. Nonlinear multiobjective optimization, volume 12 of International Series in Operations Research & Management Science. Springer US, 1998. doi: 10.1007/978-1-4615-5563-6. URL https://www.springer.com/gp/book/9780792382782.

Benjamin Moseley and Joshua R. Wang. Approximation bounds for hierarchical clustering: Average linkage, bisecting k-means, and local search. In Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS’17, page 3097–3106, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Jenny Nossack and Erwin Pesch. A branch-and-bound algorithm for the acyclic partitioning problem. Computers & operations research, 41(88):174–184, 2014. URL https://doi.org/10.1016/j.cor.2013.08.013.

Aurko Roy and Sebastian Pokutta. Hierarchical clustering via spreading metrics. Journal of Machine Learning Research, 18(88):1–35, 2017. URL http://jmlr.org/papers/v18/17-081.html.

Bernd Schröder. Ordered Sets, An Introduction. Springer Science + Business Media, LLC, 2003. URL https://link.springer.com/content/pdf/10.1007/978-3-319-29788-0.pdf.

Lotif Zadeh. Optimality and non-scalar-valued performance criteria. IEEE Transactions on Automatic Control, 8(1):59–60, 1963. doi: 10.1109/TAC.1963.1105511. URL https://ieeexplore.ieee.org/document/1105511.