Data Min Knowl Disc
DOI 10.1007/s10618-015-0406-1

Beyond rankings: comparing directed acyclic graphs

Eric Malmi · Nikolaj Tatti · Aristides Gionis

Received: 31 August 2014 / Accepted: 9 February 2015

Abstract Defining appropriate distance measures among rankings is a classic area of study which has led to many useful applications. In this paper, we propose a more general abstraction of preference data, namely directed acyclic graphs (DAGs), and introduce a measure for comparing DAGs, given that a vertex correspondence between the DAGs is known. We study the properties of this measure and use it to aggregate and cluster a set of DAGs. We show that these problems are **NP**-hard and present efficient methods to obtain solutions with approximation guarantees. In addition to preference data, these methods turn out to have other interesting applications, such as the analysis of a collection of information cascades in a network. We test the methods on synthetic and real-world datasets, showing that the methods can be used to, e.g., find a set of influential individuals related to a set of topics in a network or to discover meaningful and occasionally surprising clustering structure.

Keywords Directed acyclic graphs · Aggregation · Clustering · Preferences · Information cascades

Responsible editors: Joao Gama, Indre Zliobaite, Alipio Jorge, Concha Bielza.

E. Malmi (✉) · N. Tatti · A. Gionis
HIIT and Department of Computer Science, Aalto University, Espoo, Finland
e-mail: eric.malmi@aalto.fi

N. Tatti
e-mail: nikolaj.tatti@aalto.fi

A. Gionis
e-mail: aristides.gionis@aalto.fi

Published online: 14 March 2015

1 Introduction

Rankings and partial rankings are used in real-life applications to model associations in data. Examples include ranking documents for information-retrieval applications, ranking user preferences for modeling users and making recommendations, ranking experts for learning applications, and many more. Defining appropriate distance measures among rankings is a classic area of study (Kendall 1938, 1976; Goodman and Kruskal 1972) and it provides useful tools to analyze datasets that contain rankings.

The problems of *aggregating* and *clustering* rankings are motivated by many application scenarios and they have been widely studied in the literature (Madden 1995; Murphy and Martin 2003; Borda 1781; Ailon et al. 2008). For example, the rank-aggregation problem arises in meta-search engines, where a number of search engines return different rankings over a set of documents and the goal is to find a new ranking that incorporates the rankings of the individual search engines in the best possible way. Similarly, the problem of clustering rankings is encountered when users state their preferences with respect to a set of items (commercial products, movies, restaurants, etc.) and one needs to segment the user base so that in each market segment the users agree as much as possible with respect to their rankings.

A number of authors convincingly argue that rankings encountered in practical applications are not total orders over the full set of items (Ailon 2010; Fagin et al. 2003, 2006). Instead, it is more common to be confronted with missing information, and rankings become partial rankings (rankings with ties) and top-k lists. Consequently, the problem of defining distance measures among partial rankings and top-k lists was studied in the literature, as well as the problem of aggregating such rankings with missing information.

In this paper, we go one step further towards modeling and managing data with missing information. We assume that associations in data are modeled not just by partial rankings or by top-k lists, but by the more general abstraction of *directed acyclic graphs* (DAGs). We view DAGs as a useful abstraction to model interesting associations between data items in many modern applications. We thus consider the problem of developing methods that can be used to cope with data represented as DAGs. We start by addressing the problem of devising a meaningful distance measure between DAGs, and we then consider the classic problems of *aggregation* and *clustering* on DAG data. We should point out that in our problem setting, the correspondence between the vertices of different graphs is known, as will become clear later in the examples and experiments that we provide. This makes the comparison of two graphs computationally much cheaper than in methods based on *graph edit distances* (see e.g., Bunke and Shearer 1998; Jiang et al. 2001).

The notion of a distance between partial orders, that is, transitively closed DAGs, has been previously introduced by Brandenburg et al. (2012, 2013). However, computing the two distance measures they propose results in an **NP** or **coNP**-complete problem, so more efficient methods are called for.

Before discussing our methods and our contributions in more detail, we provide further motivation for our approach by giving examples of real-world datasets modeled as DAGs.

1.1 User preferences

Humans are better at making comparative rather than absolute judgements (Laming 2003). In many cases, preferences of users over a set of items can be recorded via user-feedback mechanisms. For example, when a user is selecting an item among a small number of choices provided by the application, the user, implicitly states a preference for that item over the rest. In such applications, a user can be modeled by a *preference graph*, a directed graph that represents preferences between pairs of data items. Such preference graphs are likely to be DAGs, or near-DAGs, as the underlying preference relation tends to be transitive. The crucial observation here is that since we are typically recording preferences over small sets of items, a large number of edges is missing, and thus, the preference graph resembles a DAG rather than a full or a partial ranking.

1.2 Information cascades

Our second example arises in the analysis of information cascades in social media (Anagnostopoulos et al. 2008; Barbieri et al. 2013; Gomez-Rodriguez et al. 2011, 2012; Goyal et al. 2010; Kempe et al. 2003; Macchia et al. 2013; Saito et al. 2008; Su et al. 2014). In this case we consider "actions" performed in a social network. Due to social influence, copies of such actions are performed by neighboring nodes in the network, e.g., retweets in the Twitter network, and thus information cascades are observed. Due to time ordering, the cascade of an action in a network is a DAG. Analysis of information cascades is important in understanding dynamic phenomena in the social network, for example, who are the influential nodes, how information propagates, and so on. Therefore, one needs appropriate tools to compare, aggregate, and cluster cascades (DAGs) of different actions.

Motivated from the above applications, we develop techniques for managing data represented as DAGs. We first propose a sound distance measure for comparing DAGs. Our measure extends the Kendall-tau measure for full rankings and partial rankings. Similar to the work of Fagin et al. (2006), our measure uses parameters $0 \leq p, q \leq 1$ to penalize for pairs of items for which one, or both, of the two DAGs do not make a clear ordering decision. We study the proposed distance measure with respect to its metric properties. We are able to show that the measure satisfies a relaxed version of the triangle inequality, where the "slackness" factor depends on the penalty parameters p and q.

The proposed distance measure is then used to define the problems of aggregating and clustering a set of DAGs. We show that the DAG aggregation and DAG clustering problems are **NP**-hard and we present efficient methods to obtain solutions with approximation guarantees. Our solutions rely crucially on the relaxed triangle inequality property of the proposed distance measure.

We test our methods on synthetic and real-world datasets. Our experiments show that a simple greedy approach yields good performance. It can be used to recover ground-truth DAGs when data are inflicted with high levels of noise, as well as to identify meaningful clusters in real-world applications where data are represented with DAGs.

The rest of the paper is organized as follows. In Sect. 2 we discuss basic concepts and we introduce our notation. Our distance measure is presented in Sect. 3. Sections 4 and 5 discuss the problems of aggregating and clustering DAGs. In Sect. 6 we discuss other work related to the problems we consider. Our experimental evaluation on synthetic and real datasets are presented in Sect. 7, while Sect. 8 is a short conclusion.

2 Preliminaries and notation

We will discuss basic concepts and we establish the notation that we will use throughout the remaining paper.

A directed graph $G = (V, E)$ is a tuple of vertices V and edges E, which is a set of ordered pairs $(i, j) \in V \times V$. A directed acyclic graph (DAG) is a directed graph that has no directed cycles.

Without loss of generality, we assume that graphs share the same set of vertices. Indeed, if two graphs have different set of vertices, the missing vertices for each graph can be added as singletons. Furthermore, as we will see, our DAG-comparison measure penalizes appropriately for such missing vertices added as singletons. Consequently, we assume that when comparing two DAGs we can just compare their edges. Given two DAGs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ we say that a vertex pair (i, j) is a *discordant* if $(i, j) \in E_1$ and $(j, i) \in E_2$. On the other hand, if $(i, j) \in E_1$ and $(i, j) \in E_2$, then we say that the pair is *concordant*.

A special case of DAG is a *total order*, in which the vertices are assumed to have an order and the edge set E consists of $\binom{|V|}{2}$ edges, so that $(i, j) \in E$ if and only if i occurs before j in the total order. We should point out that we do not assume that our graphs are transitively closed. This means that while we can apply our distance to a partial order, which is essentially a transitively closed DAG, we also consider DAGs that are not partial orders.

Given two total orders G_1 and G_2, a Kendall-tau distance $K(G_1, G_2)$ between the two orders is the number of all discordant vertex pairs between them. Kendall-tau is a distance measure that takes values between 0 and $\binom{|V|}{2}$.[1] It becomes 0 when the two orders are the same, and it becomes $\binom{|V|}{2}$ when G_1 is a reversed version of G_2. Kendall-tau is a distance measure that is widely used to compare rankings. Fagin et al. (2006) extend the Kendall-tau distance (as well as other ranking distances) for *partial rankings*, that is, rankings with ties. Their work is not only theoretically interesting but of great practical importance, since orders (rankings) encountered in practice are rarely total orders (full rankings). In this paper, we take one further step on the problem of devising measures to compare rankings: we extend the Kendall-tau distance to general DAGs. Consequently, our methods provide a basis to deal with datasets in which DAGs is the appropriate data abstraction.

[1] Most often the Kendall-tau distance is defined to be a value between 0 and 1 by normalizing with the total number of vertex pairs $\binom{|V|}{2}$.

3 Measuring DAG distance

Our goal is to develop a meaningful and well-founded distance measure between DAGs. We first define such a distance measure as a generalization of Kendall-tau, and then study its properties. In particular we show that the proposed measure satisfies a *relaxed version* of the triangle inequality. This *near-metric* property is particularly useful as it allows to develop approximation algorithms for aggregation and clustering tasks.

3.1 Generalization of Kendall-tau for DAGs

As with the Kendall-tau distance, we consider a measure defined over pairs of vertices in the two DAGs. As opposed to full rankings (total orders), in DAGs a pair of vertices may not be directly comparable. As a result, given two directed acyclic graphs, a pair of vertices may be neither concordant nor discordant. Thus, a distance measure that considers pairs of vertices in two DAGs, should consider the following two cases of assigning a penalty:

Discordant pairs:	We should penalize an edge (i, j) if i precedes j in one graph while j precedes i in the other graph.
Potentially discordant pairs:	If one or both of the input DAGs are not fully connected then there is a pair of vertices (i, j) for which we do not know if i precedes j or vice versa, so we have a potentially discordant pair, which should be penalized less heavily than a pure discordant pair.

The proposed DAG distance measure satisfies the above two requirements. Our approach is similar to the approach by Fagin et al. (2006), where they generalize the Kendall-tau distance to partial rankings by assigning a penalty $0 < p < 1$ to pairs for which the two partial rankings do not have clear agreement or disagreement. In a similar way, our DAG distance measure is defined with respect to partial penalty parameters p and q, where $0 \leq p, q \leq 1$. As it will become clear below we also assume $q \leq p$.

For our definition, we also assume an arbitrary ordering of the vertices of the two DAGs. Such an ordering is only assumed for notational simplicity and is not related to the concept of a total order. This ordering is only used to ensure that each pair of vertices i and j is considered only once, so any arbitrary bijection from the set of vertices V to $\{1, \ldots, |V|\}$ can be used. We should stress that even though the definition of the DAG distance is based on an ordering of vertices, the final result does not depend on it.

Let $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ be two DAGs and let $0 \leq p, q \leq 1$ be fixed parameters. Let i and j be two vertices of V such that $i < j$. Let $e = (i, j)$ and $f = (j, i)$ be e with reversed direction. We define $K_{i,j}^{(p,q)}(G_1, G_2)$ to be a penalty associated to vertices i and j for the DAGs G_1 and G_2 with respect to the parameters p and q. We consider four cases:

Case 1: ($e \in E_1$ and $e \in E_2$) or ($f \in E_1$ and $f \in E_2$). In this case, G_1 and G_2 agree on e, and we set the distance to be $K_{ij}(G_1, G_2) = 0$.

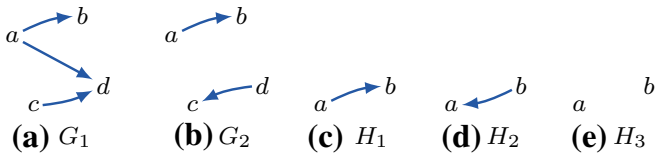

Fig. 1 Toy graphs related to Examples 1–2

Case 2: ($e \in E_1$ and $f \in E_2$) or ($e \in E_2$ and $f \in E_1$). In this case, G_1 and G_2 completely disagree on e, and we set the distance to be $K_{ij}(G_1, G_2) = 1$.

Case 3: (e or $f \in E_1$ and $e, f \notin E_2$) or (e or $f \in E_2$ and $e, f \notin E_1$). In this case, an edge between i and j exists in one graph but not in the other. We set $K_{ij}(G_1, G_2) = p$.

Case 4: $e, f \notin E_1$ and $e, f \notin E_2$. In this last case there is no edge between i and j in either of the graphs. We set $K_{ij}(G_1, G_2) = q$. The motivation for setting $q > 0$ is that otherwise two empty DAGs are as similar as two identical complete DAGs even though one might argue that the latter two have much more in common than the empty DAGs.

We are now ready to define our DAG distance measure.

Definition 1 Let $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ be two DAGs and let $0 \leq p, q \leq 1$ be fixed parameters. We define the DAG distance $K^{(p,q)}(G_1, G_2)$ to be the sum of distances over pairs of vertices $(i, j) \in V \times V$ such that $i < j$,

$$K^{(p,q)}(G_1, G_2) = \sum_{\substack{(i,j) \in V \times V \\ i<j}} K_{i,j}^{(p,q)}(G_1, G_2).$$

Whenever clear from the context, we will drop p and q from the notation and write $K(G_1, G_2)$ instead of $K^{(p,q)}(G_1, G_2)$.

Example 1 Let us compute the distance $K^{(p,q)}(G_1, G_2)$ of the graphs given in Fig. 1. There is one concordant edge, namely (a, b), one discordant pair, (c, d), one pair of type Case 3, (a, d), and the remaining three pairs are of Case 4. Hence, the distance is equal to

$$K^{(p,q)}(G_1, G_2) = 0 + 1 + p + 3q.$$

A few observations are in order: First, it is easy to see that in the case that the two DAGs are total orders, Definition 1 reduces to the standard Kendall-tau.

Second, it is reasonable to assume that Case 3, when an edge exists in one graph but not the other, is a case of stronger or equal disagreement compared to Case 4, when an edge is missing from both graphs. Therefore, we will assume that the penalty associated to Case 3 is greater than or equal to the penalty associated to Case 4, that is, $q \leq p$. In fact, the case $q > p$ has undesirable consequences. For example, for the problem of DAG aggregation, which we discuss in the next section, $q > p$ implies that the optimal centroid of two empty DAGs will be a complete DAG.

3.2 Relaxed triangle inequality

Our next step is to examine whether the proposed distance measure K is a metric. We remind that for a distance measure $d : X \times X \to \mathbb{R}$ to be a metric, the following properties should hold: (i) non-negativity: $d(x, y) \geq 0$; (ii) symmetry: $d(x, y) = d(y, x)$; (iii) identity: $d(x, y) = 0$ if and only if $x = y$; and (iv) triangle inequality: $d(x, z) \leq d(x, y) + d(y, z)$. The reason for undertaking this study is twofold: First, satisfying the metric properties provides additional evidence that our distance measure is meaningful and well founded. Second, many algorithms have been designed to take advantage of metric properties with respect to efficiency, e.g., via effective pruning rules, or with respect to satisfying certain quality guarantees. Indeed, after we show that the proposed distance measure K satisfies relaxed metric properties, we are able to exploit those properties in order to obtain high-quality solutions for the problems of aggregating and clustering DAGs.

Back to the question whether the proposed distance K satisfies the metric properties, it is immediate that the distance is non-negative and symmetric. If $q = 0$, then the distance satisfies the identity property, namely, $K(G, G) = 0$, however this property does not hold for $q > 0$ as penalty q is induced by edges that are missing from both input graphs. Note that this was done by design as we want to punish potential discordant pairs. The most important property with respect to designing approximation algorithms is the triangle inequality. Our next result shows that the distance measure K satisfies a *relaxed* version triangle inequality, meaning that for any directed acyclic graphs G_1, G_2, G_3 it holds $K(G_1, G_2) \leq c(K(G_1, G_3) + K(G_3, G_2))$, for some constant $c \geq 1$, which depends on the parameters p and q.

Proposition 1 *Let $0 \leq q \leq p \leq 1$ and $0 < p$. Then distance measure K satisfies relaxed triangle inequality,*

$$K(G_1, G_2) \leq c(K(G_1, G_3) + K(G_2, G_3)),$$

where $c = \max\left(4p^2, q + \max(2p, 1)\right)/2p$.

Proof We will prove the result by upper bounding the distance $K(G_1, G_2)$ and lower bounding the distance measures $K(G_1, G_3)$ and $K(G_2, G_3)$.

Consider $K(G_1, G_2)$, and let k_i be the number of Case i edges as defined in Sect. 3.1. The distance is equal to

$$K(G_1, G_2) = k_2 + pk_3 + qk_4.$$

Define also $n_1, \ldots, n_4$ for $K(G_1, G_3)$ and $m_1, \ldots, m_4$ for $K(G_2, G_3)$.

Consider a set $D = (E_1 \setminus E_2) \cup (E_2 \setminus E_1)$. For every Case 2 edge, say $e = (i, j)$, there are two edges $(i, j) \in D$ and $(j, i) \in D$. The remaining edges in D correspond to Case 3 edges possibly with a reversed direction. This implies that $|D| = 2k_2 + k_3$. Let $k = 2k_2 + k_3$, $n = 2n_2 + n_3$, and $m = 2m_2 + m_3$. The previous argument shows that k is the number of directed edges where G_1 and G_2 disagree. This is a known metric and it follows that $k \leq n + m$.

Note that any edge of Case 4 in $K(G_1, G_2)$ can only be Case 3 or Case 4 edge in $K(G_1, G_3)$. Hence, $k_4 \leq n_3 + n_4$ and similarly $k_4 \leq m_3 + m_4$.

We consider three cases, depending on values of p and q.

First, let us assume that $p \leq 1/2$. Since $k_2 \leq k/2$ we can upper bound $K(G_1, G_2)$ by

$$\begin{aligned} K(G_1, G_2) &= k_2 + pk_3 + qk_4 = k_2 + p(k - 2k_2) + qk_4 \\ &= (1 - 2p)k_2 + pk + qk_4 \\ &\leq (1 - 2p)k/2 + pk + qk_4 = k/2 + qk_4. \end{aligned}$$

Define $x = p/(1 + q)$. Note that $x + qx = p$ and $x \leq 1/2$. Consequently,

$$\begin{aligned} K(G_1, G_3) &= n_2 + pn_3 + qn_4 = n_2 + xn_3 + q(n_4 + xn_3) \\ &= n_2 + x(n - 2n_2) + q(n_4 + xn_3) \\ &= (1 - 2x)n_2 + xn + q(n_4 + xn_3) \\ &\geq xn + q(n_4 + xn_3). \end{aligned}$$

Define $c = (1 + q)/(2p)$. Note that $cx = 1/2$ and that $c \geq 1/2$. Multiplying the previous inequality with c gives us

$$\begin{aligned} cK(G_1, G_3) &\geq cxn + cq(n_4 + xn_3) = n/2 + q(cn_4 + n_3/2) \\ &\geq n/2 + q/2(n_4 + n_3). \end{aligned}$$

Similarly, we can lower bound the distance $K(G_2, G_3)$ by

$$cK(G_2, G_3) \geq m/2 + q/2(m_4 + m_3).$$

We can now combine the inequalities,

$$\begin{aligned} K(G_1, G_2) &\leq k/2 + qk_4 \\ &\leq (n + m)/2 + q/2(m_3 + m_4 + n_3 + n_4) \\ &\leq c(K(G_2, G_3) + K(G_1, G_3)). \end{aligned}$$

Assume now that $p \geq 1/2$. We can upper bound $K(G_1, G_2)$ by setting $k_2 = 0$,

$$K(G_1, G_2) = (1 - 2p)k_2 + pk + qk_4 \leq pk + qk_4.$$

Assume that $4p^2 - q \leq 2p$. Let $z = p/(q + 2p)$ and $x = p - qz = 2p^2/(q + 2p)$. Note that the assumption implies that $x \leq 1/2$, which allows us to lower bound

$$\begin{aligned} K(G_1, G_3) &= (1 - 2x)n_2 + xn + q(n_4 + zn_3) \\ &\geq xn + q(n_4 + zn_3). \end{aligned}$$

Let $c = (q + 2p)/2p$. Then $cx = p$ and $cz = 1/2$. Since $c \geq 1/2$, we have

$$cK(G_1, G_3) \geq pn + q/2(n_4 + n_3).$$

Combining the inequalities, similar to the first case, proves the proposition for the second case.

Finally, assume that $4p^2 - q \geq 2p$. Let $z = 1/(4p)$ and set $x = p - qz = (4p^2 - q)/(4p^2)$. Since $x \geq 1/2$, we obtain a lower bound

$$\begin{aligned} K(G_1, G_3) &= (1 - 2x)n_2 + xn + q(n_4 + zn_3) \\ &\geq n/2 + q(n_4 + zn_3). \end{aligned}$$

Let $c = 2p$. Then $c/2 = p$ and $cz = 1/2$. Since $c \geq 1/2$,

$$cK(G_1, G_3) \geq pn + q/2(n_4 + n_3).$$

Combining the inequalities, similar to the first case, proves the proposition for the third case.

By setting $c = \max\left(4p^2, q + \max(2p, 1)\right)/2p$, we are able to handle all three cases simultaneously. □

Note that if we set $p = 1/2$ and $q = 0$, then the distance satisfies the triangle inequality. In fact, $K^{(1/2,0)}(G_1, G_2)$ is a half of symmetric difference between edge sets E_1 and E_2,

$$K^{(1/2,0)}(G_1, G_2) = \frac{1}{2}|(E_1 \setminus E_2) \cup (E_2 \setminus E_1)|.$$

If we set $p = q = 0$, then we are penalizing only Case 1 edges, that is, discordant pairs. Proposition 1 holds only when $p > 0$, but it suggests that when $p = 0$, K does not satisfy even a relaxed version of the triangle inequality. In order to prove this intuition, we show a simple example.

Example 2 Consider H_1, H_2, and H_3 given in Fig. 1. If we set $p = 0$, then $K(H_1, H_2) = 1$ but $K(H_1, H_3) = K(H_2, H_3) = 0$. Consequently, there is no $c > 0$ such that

$$K(H_1, H_2) \leq c(K(H_1, H_3) + K(H_2, H_3)).$$

4 DAG aggregation

We now consider the problem of DAG aggregation, that is, summarizing a set of DAGs with a single DAG. We define the problem and demonstrate that it is computationally intractable. We then propose two different methods for solving the DAG-aggregation problem, and we show that both methods provide approximations to the optimal solution. While one method provides a better theoretical bound than the other, a simple

heuristic based on the second approach is shown to be the algorithm that works best in practice.

4.1 Problem definition

We formulate the problem of aggregating DAGs as the following *median-type* optimization problem.

Problem 1 (*DAG Aggregation*) Given a set of M directed graphs $G_1, \ldots, G_M$, find a DAG C minimizing

$$\sum_{i=1}^{M} K^{(p,q)}(G_i, C).$$

We can view DAG AGGREGATION as an extension of RANK AGGREGATION, if we view ranks as graphs. It is known that RANK AGGREGATION is **NP**-hard (Dwork et al. 2001), however this does not imply directly that DAG AGGREGATION is also **NP**-hard, since in DAG AGGREGATION the output is not required to be a full ranking.

In order to prove hardness we reduce a known **NP**-hard problem called FEEDBACK ARC SET, where the goal is to construct a DAG with least amount of edges.

Problem 2 (*Feedback Arc Set (FAS)*) Given a directed graph $G = (V, E)$, find the smallest set of edges $F \subset E$ such that $(V, E \setminus F)$ is a DAG.

The **NP**-hardness of FAS (Karp 1972) implies the following result.

Proposition 2 DAG AGGREGATION *is **NP**-hard.*

Proof We will prove the hardness by reducing FAS to DAG AGGREGATION. Fix $p, q > 0$.

Assume that we are given a directed graph $G = (V, E)$. We can safely assume that there are no 2-cycles in G, otherwise we can modify G by splitting edge into two edges by adding at most $|E|$ vertices.

Our first step is to split G into two DAGs. In order to do that select and fix an arbitrary order over the vertices. Define $E_1 = \{(i, j) \in E \mid i < j\}$ and $E_2 = E \setminus E_1$. Clearly $E_1 \cap E_2 = \emptyset$ and $E_1 \cup E_2 = E$. Set $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$. Both G_1 and G_2 are DAGs.

Let $C = (V, H)$ be the solution for DAG AGGREGATION for G_1, G_2. We claim that $F = E \setminus H$ solves FAS.

In order to prove this, let us first show that $H \subseteq E$. Assume otherwise and let $e \in H \setminus E$. Then this edge contributes either $2p$ or $p + 1$ to the cost, depending whether the reversed edge is in E. However, if we delete e from H, then the cost is either $2q$ or $p+q$. Consequently, we can always decrease the cost by deleting e. Thus, we can safely assume that $H \subseteq E$.

Since we have no 2-cycles in E, an edge $e \in H$ contributes p to the cost of DAG AGGREGATION because $e \in E_1$ and $e \notin E_2$, or vice versa. On the other hand, an edge

$e \in E \setminus H$ contributes $p + q$ to the cost. Hence, the cost of DAG AGGREGATION is equal to

$$\begin{aligned} K(G_1, C) + K(G_2, C) &= q(|V|(|V| - 1) - 2|E|) + p|H| \\ &\quad + (p + q)|E \setminus H| \\ &= q(|V|(|V| - 1) - 2|E|) - q|H| + (p + q)|E|. \end{aligned}$$

Consequently, minimizing the cost is equal to maximizing $|H|$ which is equal to minimizing $|E \setminus H|$. □

Proposition 2 implies that solving DAG AGGREGATION optimally is computationally intractable. Hence, in the rest of this section we consider two approximation algorithms.

4.2 Approximating DAG aggregation by a median centroid

Our first approach to select a centroid is simply picking the centroid from the input graphs instead of constructing it from scratch. Remarkably, the fact that the distance is almost a metric gives us an approximation ratio guarantee.

More formally, assume that we are given $G_1, \ldots, G_M$. We select the centroid *from* the input graphs $G_1, \ldots, G_M$ minimizing the distance, that is,

$$\text{MEDIAN}\ (G_1, \ldots, G_M) = \arg\min_{G_j} \sum_{i=1}^{M} K^{(p,q)}(G_i, G_j).$$

Since K satisfies a relaxed triangle inequality, we can achieve a constant approximation ratio. The proof of the following proposition is standard and it is omitted for lack of space.

Proposition 3 *Assume a set of DAGs $G_1, \ldots, G_M$. For $0 \leq q \leq p \leq 1$ with $0 < p$,* MEDIAN $(G_1, \ldots, G_M)$ *yields an approximation ratio of $2c$, where c is a constant as defined in Proposition 1.*

4.3 Greedy approximation of DAG aggregation

Our next approach is based on the fact that we can express DAG AGGREGATION as an instance of WEIGHTED FEEDBACK ARC SET problem.

In order to do that, assume that we are given $G_1, \ldots, G_M$ DAGs and let $C = (V, E)$ be a candidate centroid. Let $b(i, j)$ define the cost of not having (i, j) as an edge in the centroid,

$$b(i, j) = \sum_{m=1}^{M} K_{ij}(G_m, H_{empty}), \tag{1}$$

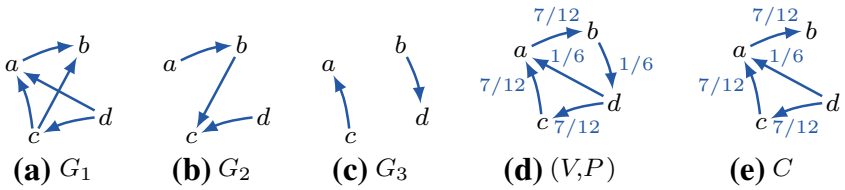

Fig. 2 Toy graphs related to Example 3

where H_{empty} is a graph with no edges. Similarly, define $w(i, j)$ to be the cost of having (i, j) as an edge in the centroid,

$$w(i, j) = \sum_{m=1}^{M} K_{ij}(G_m, H_{full}), \tag{2}$$

where H_{full} is a full directed graph. This gives us

$$\sum_{m=1}^{M} K(G_m, C) = \sum_{e \in E} w(e) + \sum_{e \notin E} b(e).$$

Let us now define $P = \{e \in V \times V \mid w(e) < b(e)\}$ to be the set of edges that ideally we would like to have in the centroid. We can safely assume that the centroid edges are all in P, $E \subseteq P$. However, since P may contain cycles, we cannot have all of these edges. For each edge e not included in the centroid, we define *regret* $r(e) = b(e) - w(e)$ to be the difference we need to pay for not having the edge in C. We can write the cost as

$$\sum_{m=1}^{M} K(G_m, C) = \sum_{e \in P} w(e) + \sum_{e \notin P} b(e) + \sum_{e \in P \setminus E} r(e).$$

Since the first two terms do not depend on C, we see that for selecting optimal centroid we need to minimize the third sum. This is in fact an instance of WEIGHTED FEEDBACK ARC SET (WFAS) problem, with an input graph (V, P, r).

Example 3 Consider G_1, G_2, and G_3 given in Fig. 2. Set $p = 1/3$ and $q = 1/4$ and let us compute the optimal centroid. In order to do that, let us compute P. The regret for an edge (a, b) is equal to

$$r(a, b) = 2/3 + 1/4 - 1/3 = 7/12.$$

Similarly, $r(a, c) = r(c, d) = 7/12$. The regret for (a, d) and (b, d) is

$$r(a, d) = r(b, d) = 1/3 + 2/4 - 2/3 = 1/6.$$

Finally, the regret for (b, c) is

$$r(b, c) = 2/3 + 1/4 - 1/3 - 1 = -5/12$$

which is negative and hence (b, c) is not included in P. The rest of the vertex pairs also have negative regrets, and are not included in P. The graph (V, P) is given in Fig. 2d. This graph has a cycle and the optimal DAG, C, is obtained by removing the weakest edge (b, d). The optimal centroid is given in Fig. 2e.

We can estimate WFAS with $O(\log|V|\log\log|V|)$ with an algorithm based on Linear Programming techniques (Even et al. 1995).

Algorithm 1: AGGRFAS, solves DAG AGGREGATION problem using Feedback arc set solver

compute w and b for each pair (i, j) using Eqs. 1–2;
$P \leftarrow \{e \in V \times V \mid w(e) < b(e)\}$;
$r(e) \leftarrow b(e) - w(e)$ for all $e \in P$;
$F \leftarrow \text{WFAS}(V, P, r)$;
return $(V, P \setminus F)$;

Proposition 4 AGGRFAS *yields an approximation ratio of* $O(\log|V|\log\log|V|)$.

Proof Let $G_1, \ldots, G_m$ be the input graphs and let r be the regret. We showed earlier that we can write the cost function as

$$\sum_{m=1}^{M} K(G_m, C) = \text{const} + \sum_{e \in P \setminus E} r(e),$$

for any graph $C = (V, E)$, where const ≥ 0 is a constant and does not depend on C. Let OPT be the optimal solution of the second term and let Q be the cost of a solution of weighted FAS found by an algorithm given in Even et al. (1995). Then since const ≥ 0 and $Q \geq OPT$, we have

$$\frac{\text{const} + Q}{\text{const} + OPT} \leq \frac{Q}{OPT}.$$

The solver in Even et al. (1995) has an approximation ratio guarantee of

$$Q/OPT \in O(\log|V|\log\log|V|)$$

which proves the result. □

As approximation algorithms based on LP-approaches are rarely practical, we consider a significantly simpler but efficient approach, given in Algorithm 2.

Algorithm 2: GREEDY, estimates optimal centroid given a set of DAGs $G_1, \ldots, G_M$

$Q \leftarrow \left\{\bigcup_{i=1}^{M} E_i\right\}$;
compute w and b for each edge $e \in Q$ using Eqs. 1–2;
$P \leftarrow \{e \in Q \mid w(e) < b(e)\}$;
$r(e) \leftarrow b(e) - w(e)$ for all $e \in P$;
$E \leftarrow \emptyset$;
foreach $e \in P$ *sorted by regret* **do**
  if $e \cup E$ *has no cycle* **then**
    add e to E;
return (V, E);

The idea behind this algorithm is straightforward. We order the edges based on regret, edges with smallest regret first. Note that we only need to consider edges that have appeared in at least one of the input DAGs since $w(e) < b(e)$ can not hold for the other edges. Then we keep adding edges into a centroid in an order, ignoring the edges that create cycles. Even though this is a very simple approach, in our experiments it outperforms MEDIAN , an algorithm for which we have a constant approximation guarantee.

4.3.1 Computational complexity

Calculating the proposed distance measure K between DAGs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ only requires counting the number of concordant and discordant pairs as the potentially discordant pairs can be computed based on these two numbers and the number of vertices $|V|$. If we store edges in a hash table as a preprocessing step, this yields a complexity of $O(\min(|E_1|, |E_2|))$.

Assume that we are given M input DAGs. Let k be the total number of edges in the input graphs, $k = \sum_{i=1}^{M} |E(G_i)|$. Computing the cost of a centroid can be done in $O(k)$ time. Hence, the complexity cost for MEDIAN is $O(Mk)$.

In GREEDY , we first need to form sets $Q = \left\{\bigcup_{i=1}^{M} E_i\right\}$ and $P = \{e \in Q \text{ such that } w(e) < b(e)\}$. Taking the union of all edges for Q takes $O(k)$ steps. We can see that $|P| \leq |Q|$. Second, we need to detect cycles in an incrementally increasing graph. This can be done in $O(\min(|P|^{\frac{1}{2}}, |V|^{\frac{2}{3}})|P|)$ steps using the method described by Bender et al. (2011), which gives us a running time of $O(k + \min(|P|^{\frac{1}{2}}, |V|^{\frac{2}{3}})|P|)$. However, in our experiments, we implemented the cycle detection simply by keeping track of the transitive closure of the centroid while adding new edges to it since term $O(k)$ seemed to be the bottleneck of the algorithm.

Finally, we would like to point out that the main computational challenge in DAG AGGREGATION is due to the requirement of having an *acyclic* centroid. Indeed, if we ignored the acyclicity constraint, we could compute the optimal centroid in polynomial time by taking all edges for which it holds that $w(e) < b(e)$. Nevertheless, in some cases it is useful to have a DAG centroid, for example, if you want to also get a topological ordering of the vertices. Such a scenario is exemplified by our last experiment on clustering preference data.

5 Clustering DAGs

A natural application for distance measure is clustering. More formally, consider the following problem.

Problem 3 Given a set of DAGs $G_1, \ldots, G_M$ and a number k, find k clusters $P_1, \ldots, P_k$ and centroids $C_1, \ldots, C_K$ such that

$$\sum_{i=1}^{k} \sum_{G \in P_i} K^{(p,q)}(G, C_i)$$

is minimized.

Note that since we require to discover centroids along with the clusters, the clustering problem becomes automatically **NP**-hard. In fact, if we set $k = 1$, then the problem reduces into finding a single centroid for all input graphs, a DAG AGGREGATION problem. This contrasts standard clustering problems where finding the centroid is typically a straightforward computation.

We approach this problem by running a k-means type algorithm. Given a set of centroids, we group the input DAGs into clusters minimizing the distance. Once the groups are selected, we then select a centroid from each cluster. In order to select a centroid, we use MEDIAN and GREEDY algorithms introduced in the previous section.

6 Related work

DAG AGGREGATION is an extension of the rank aggregation problem. The latter task arises typically from aggregating search engine results. However, this optimization problem has been studied in the context of voting, centuries before first computers, see for example (Borda 1781). When using Kendall-tau distance, the rank aggregation problem is also known as Kemeny-Young rank aggregation problem. The problem is **NP**-hard (Dwork et al. 2001), however it admits a PTAS scheme (Kenyon-Mathieu and Schudy 2007). Extensions of rank aggregations have suggested such as partial rankings, that is, rankings with ties (Ailon 2010; Fagin et al. 2006), and top-k rankings (Ailon 2010; Fagin et al. 2003), where only top-k elements are ranked and the remaining objects are left unranked.

Extensions of rank aggregation to DAGs have been considered by Brandenburg et al. (2012, 2013). Here the extension is done by representing a DAG with a set of linear extensions. This allows to use set distances with Kendall-tau or Spearman footrule as a base distance. Unfortunately, the number of extensions may be exponential and indeed the problem of computing certain set distances becomes **NP**-hard.

In case the vertex correspondence between the graphs to be compared is unknown, the main computational challenge is to find an alignment between vertices. Several studies are based on this premise (see e.g., Bunke and Shearer 1998; Jiang et al. 2001). However, since our starting point is rankings, it is natural to assume that the vertex correspondence is known. Thus we focus on the problem of defining a meaningful distance measure given the correspondence, which makes the distance measure evaluation computationally much cheaper.

As we have seen in previous sections, finding the optimal solution for DAG AGGREGATION is intimately related to the FEEDBACK ARC SET (FAS) problem. The reduction of **NP**-hardness is done by reducing FAS to DAG AGGREGATION and one we can get an approximation algorithm by using a solver given in Even et al. (1995). FAS is known to be APX-hard with a known coefficient of $c = 1.3606$ (Kann 1992; Dinur and Safra 2005), that is, given that $\mathbf{P} \neq \mathbf{NP}$ there is no approximation algorithm with a constant approximation guarantee of c. On the other hand, selecting a centroid from the input graphs gives us a guarantee ratio of $(1 + q)/p$. Solving FAS for tournaments, graphs having edges for each vertex pair, is substantially easier as this problem admits a PTAS scheme (Kenyon-Mathieu and Schudy 2007). Note that

in order to get a tournament graph we need to have dense input graphs. This hints that DAG AGGREGATION may be easier to solve if the input graphs are dense.

7 Experimental evaluation

7.1 Experiments on synthetic data

In this section, we present our experiments with synthetic datasets. The objective is to test the ability of the K distance measure to distinguish DAGs generated from different distributions, and also to study the sensitivity of the distance measure with respect to its parameters. For the aggregation and clustering tasks, knowing the ground truth in the generated data allows to compare the MEDIAN and the GREEDY algorithms against baselines. We start our discussion by describing how we generate the data.

7.1.1 Synthetic DAG generation

In order to generate synthetic DAGs having a cluster structure, we use the following approach. We start by generating a seed DAG, adding edges (i, j), where $i < j$, with probability p_{edge}. From this seed we create N corrupted DAGs, by deleting edges with a probability p_{remove}, adding new edges with a probability of p_{add}, and swapping vertices N_{swaps} times. The probability p_{add} is determined from p_{edge} and p_{remove} so that the corrupted graphs have on average the same amount of edges as the seed. This leaves us three parameters, p_{edge}, p_{remove} and N_{swaps}. Increasing p_{remove} and N_{swaps} will make graphs more corrupt. The parameter p_{edge} determines the density of the DAGs, so that $p_{\text{edge}} = 0$ will produce an empty DAG while $p_{\text{edge}} = 1$ will produce a total order. In all experiments with the synthetic data, we set the number of vertices to 50 and we assume transitivity, thus taking the transitive closures of the DAGs once all of them have been generated. Note that transitivity is not required for the proposed distance measure and algorithms to work. Indeed, in Sect. 7.2, we present experiments on real-world information cascade data where transitivity does not hold.

7.1.2 Selecting distance measure parameters

A good distance measure should be able to differentiate between samples from different distributions while recognizing the similarity of the samples from the same distribution. In order to measure the goodness of the distance, we can vary parameters p and q of the proposed measure K, and see how well K can differentiate two sets of DAGs, given that they are drawn from two different distributions. To measure the similarity of the sets, we use the Friedman-Rafsky MST-based runs test (Friedman and Rafsky 1979). The idea is that we calculate the pairwise distances between all samples from both sets and find the minimum spanning tree. Test statistic R is the number of edges that connect samples from different sets, and a low value of R indicates that we can reject the null hypothesis of the sets being from the same distribution.

Our experimental setting is the following. First, we generate two sets of DAGs with the same parameters but with different seed DAGs. Then, we use the Friedman-Rafsky

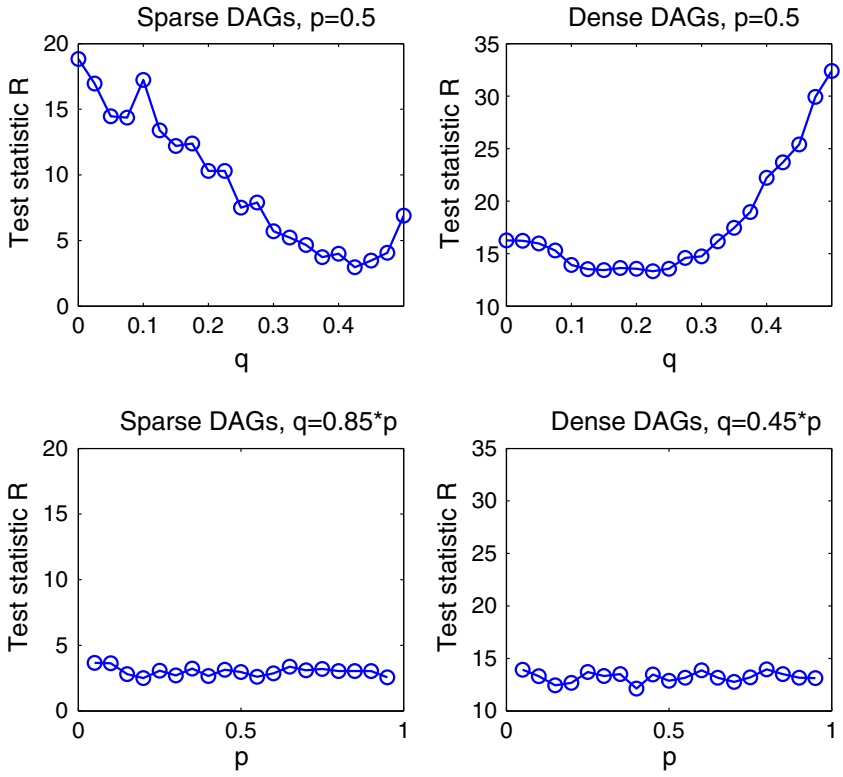

Fig. 3 Differentiation capability of the proposed measure K with different *values* of q (*top*) and different *values* of p keeping the ratio of p and q fixed (*bottom*)

test to see whether these two sets follow the same distribution, while varying p and q. Since we know that the two sets are from different distributions, we can simply see how the variation in the parameters p and q affects the statistics R. The analysis is repeated for sparse DAGs ($p_{\text{edge}} = 0.03$) and dense DAGs ($p_{\text{edge}} = 0.08$) and for different levels of corruption. Results for a single level of corruption ($p_{\text{remove}} = 0.7$, $N_{\text{swaps}} = 0$) while varying p and q are shown in Fig. 3. From the top figures, we have computed the optimal ratios of p and q for sparse DAGs ($q/p = 0.85$) and for dense DAGs ($q/p = 0.45$) when p is fixed. In the bottom figures, these ratios have been fixed after which we have varied p and q accordingly.

We omit detailed illustration of our results, however, the main conclusions drawn are the following: (i) When setting $p = 0.5$, the optimal value of q depends on the density of the DAGs so that for sparse DAGs q should be near to p, whereas for denser DAGs q can be smaller in order to be able to distinguish DAGs from different distributions. (ii) When the ratio of p and q has been fixed, the distance measure is not sensitive to the value of p. (iii) The optimal value of q depends on how corrupted the DAGs are.

All in all, these results show that the optimal values of p and q are problem dependent and it remains an open problem how to best select them. Nevertheless, in our experiments with real-world data, the following simple strategy proved to be useful: set $p = 0.5$, $q = 0$ and increase q until the obtained aggregated centroids become nonempty. Furthermore, an expert might be able to use his or her domain knowledge

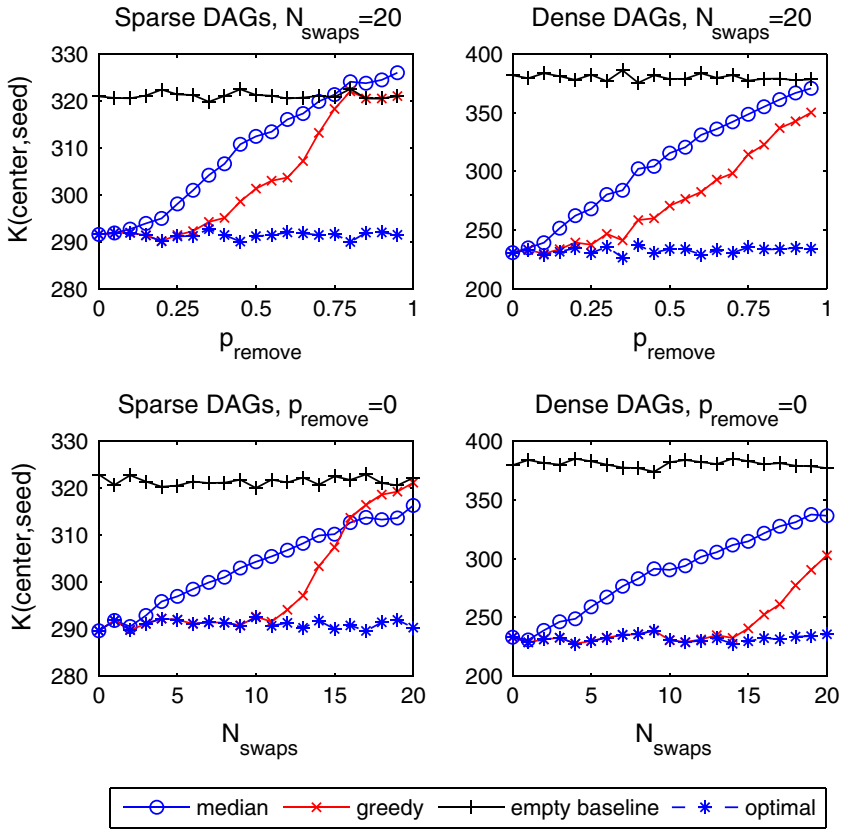

Fig. 4 DAG aggregation performance measured as the distance from the planted seed DAG to the obtained centroid

to decide how much potentially discordant pairs should be penalized and select p and q accordingly.

7.1.3 DAG aggregation

Next we evaluate our DAG aggregation methods, by testing how well they are able to uncover an underlying seed DAG. Our methodology here is to generate a set of input DAGs from a fixed seed, apply our DAG aggregation methods to compute a centroid, and then calculate the distance between the centroid and the seed DAG.

We compare MEDIAN and GREEDY along with two other methods. The baseline method uses an empty DAG as the centroid, while the optimal method outputs the seed DAG itself giving a lower bound for the distance. We set the parameters of the measure to $p = \frac{1}{2}$ and $q = \frac{1}{4}$. The results are shown in Fig. 4 for different values of p_{remove} and N_{swaps}.

From the results we see that in most cases GREEDY performs significantly better than MEDIAN . However, the drawback of GREEDY is that if the data generation parameters are increased above a certain point so that the input DAGs become very diverse, then the centroid converges into an empty DAG. In our other experiments, we notice that this happens even more easily if the value of q is decreased.

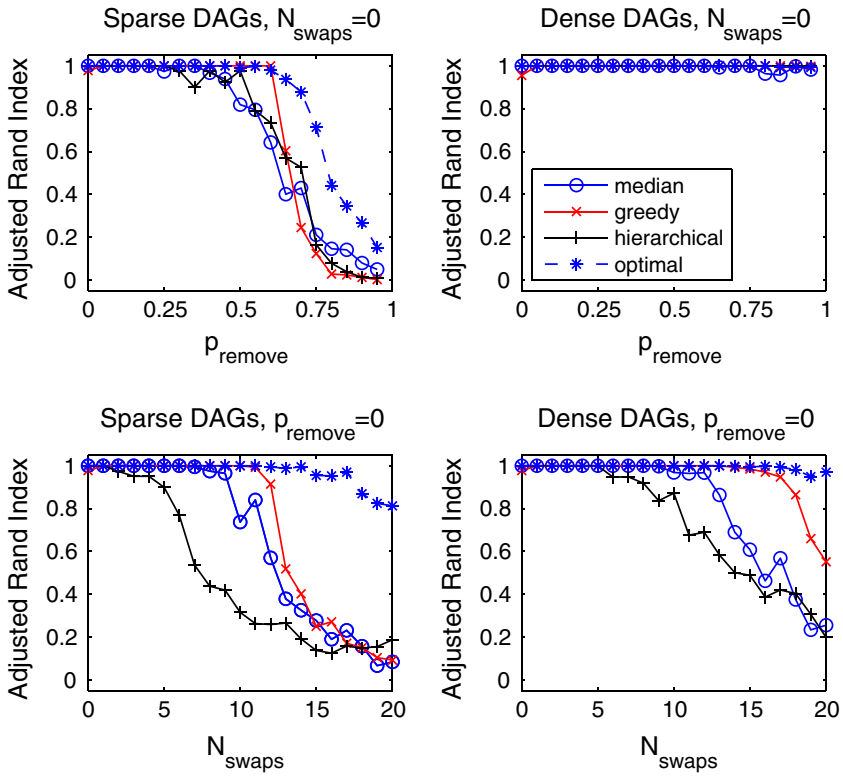

Fig. 5 Clustering method comparison for the synthetic data. Higher *values* are better

7.1.4 DAG clustering

In our final experiment with synthetic data we cluster DAGs. We generate 5 clusters using our synthetic-data generator, each cluster containing 20 graphs. Furthermore, we make 20 additional vertex index swaps for each seed creating discordant pairs between clusters, and furthering clusters away from each other.

Evaluating the performance of a clustering algorithm is generally a difficult problem due to lack of ground truth but with our synthetic dataset we know the correct partitioning of the DAGs. Thus we can measure performance using the Adjusted Rand Index (ARI) (Hubert and Arabie 1985).

We use k-means type algorithm which updates the cluster centroids using either MEDIAN or GREEDY . These approaches are compared to a *hierarchical* clustering method with complete linkage criterion and to *optimal* method which sets the seed DAGs as the initial cluster centroids and then assigns each input DAG to the closest centroid, giving us an upper bound for the performance. For the k-means approaches, we run ten restarts with random initial cluster assignments and select the run which minimizes the clustering cost defined in Problem 3. For all methods, we run ten repetitions with newly generated datasets and calculate the average ARI. We set the parameters for the distance measure to $p = \frac{1}{2}$ and $q = \frac{1}{4}$. The results are shown in Fig. 5.

When varying p_{remove}, GREEDY yields the correct clustering up to $p_{\text{remove}} = 0.6$ for sparse DAGs, but after this the performance drops. MEDIAN and the hierarchical methods perform comparably. For dense DAGs all methods are able to find the correct

clustering on almost every occasion. On the other hand, when varying parameter N_{swaps} and potentially inducing discordant pairs within a cluster, the differences are more clear. GREEDY has the best overall performance, whereas the hierarchical method performs the worst.

7.2 Experiments on information cascades

In this section, we apply the proposed DAG-aggregation methods to real-world music-listening data from Last.fm. This type of analysis can provide valuable insights, e.g., for artists who want to advertise their music to targeted individuals that are influential among their peers. Our experimental setting is inspired by the large body of work in social-network analysis, where one observes information cascades in social networks and the goal is to infer the underlying influence model (Anagnostopoulos et al. 2008; Barbieri et al. 2013; Gomez-Rodriguez et al. 2011, 2012; Goyal et al. 2008, 2010; Macchia et al. 2013; Saito et al. 2008; Su et al. 2014). This line of work focuses on learning influence probabilities on the edges of the social network, but also the roles of the network users in the information-diffusion process.

7.2.1 Music listening data

The dataset we use is collected from Last.fm, a music service that tracks user music listening and provides recommendations. The dataset contains 1 372 users who have listened to a total of 1.2 million tracks (51 495 unique tracks) from 4 322 different artists between Jan 1, 2010 and Nov 10, 2010. The dataset also includes the social network of the users. Since we want to analyze information cascades, we study how the listening of different artists is propagated in the social network. When user A starts listening to an artist that her friend B is already listening to, we say that A is following B, and draw an edge (B, A). We note that we do not have complete user histories, so user A may have listened to the same artist previously (or in fact from another platform), but as all studies of social influence we ignore this effect. This process gives us a DAG for each artist whose vertices are the Last.fm users. We limit ourselves to 196 artists who have at least 100 listeners.

7.2.2 Artist clustering

We apply k-means, using GREEDY and MEDIAN for selecting centroids, with ten restarts. We set the number of clusters to 5, and $p = 1/2$ and $q = 0.4$.

GREEDY outperforms MEDIAN both in terms of the average running time and the average cost of the clustering. The results are shown in Table 1. The relative differences of the costs are small since the costs mainly consists of the penalties caused by Case 4 pairs as the DAGs are very sparse. Nevertheless, GREEDY consistently outperforms MEDIAN .

The clustering that obtained the lowest cost using GREEDY is shown in Table 2. We can see that different clusters capture different music genres. Clusters 2 and 5 contain mostly rock artists with the difference that the former is focused on classic

Table 1 Clustering results for information cascades

	Clustering cost ± std	Time (s)	Iterations
MEDIAN	73 746 000 ± 960	166	2.3
GREEDY	73 738 430 ± 350	63	8.3

Table 2 A clustering for artists in the Last.fm dataset

Cluster	#artists	Example artists	Top tags
1	25	Amy Winehouse, Kelly Rowland, Evanescence, Linkin Park, Jason Mraz	Pop, female vocalists, rnb, dance, soul
2	24	Pink Floyd, Black Sabbath, Joy Division, Led Zeppelin, Duran Duran	Rock, classic rock, 80s, new wave, alternative
3	12	Shakira, Taylor Swift, Lana Del Rey, Florence + the Machine, Madonna	Pop, female vocalists, rnb, dance, indie
4	16	Feist, La Roux, Mika, Bat for Lashes, Gossip	Indie, alternative, female vocalists, rock, indie rock
5	119	Johnny Cash, Placebo, Vampire Weekend, Air, Kiss	Rock, alternative, indie, pop, electronic

rock whereas the latter is more diverse. Cluster 3 captures female pop artists, whereas in Cluster 4, we have indie artists. Cluster 1 is an interesting mix of pop artists and alternative metal artists suggesting that there is a group of users listening to both genres. The results also show that the pop and rock genres are well represented in the data causing smaller genres to merge to these.

7.2.3 Influential users

We should note that our approach gives not only a clustering of artists based on cascades, but it can also help us identify the most influential users for each cluster. The users who are roots in the centroid DAG of each cluster and have large subtrees below are the potentially influential ones. Identifying influential users is very important for viral marking and campaign design.

7.3 Experiments on preference data

In this section, we apply our methods to analyze preference data of different users. This analysis shows how the proposed methods can be used to discover and visualize groups of people with different tastes.

7.3.1 Artist preference data

The data was collected through a Finnish music related website whose owner allowed us to display a survey for the visitors of the site for two days. We selected twenty popular Finnish/foreign artists from five different music genres (pop, rock, rap, metal,

Table 3 Clustering results for preference data

	Clustering cost ± std	Time (ms)	Iterations
MEDIAN	43 751 ± 22	650	2.0
GREEDY	43 097 ± 28	260	14.0

and electronic) and presented the visitors a series of questions of the form "Which artist do you like more: A or B" where A and B were two randomly selected, distinct artists. In total, we received data from 3 683 users (=IP addresses) out of whom 960 satisfy the following criteria: (i) answered at least 10 preference questions (ii) answered questions about their age and gender (iii) country is Finland (iv) preference graph is acyclic (6 % of the graphs contain a cycle).[2]

7.3.2 User clustering

A preference DAG is formed by taking all the pairwise preferences of a user and drawing an edge (a, b) if the user prefers artist a to b. We divide the users into 480 train users and 480 test users and cluster the train users using the greedy method with 3 clusters and parameters $p = 0.50, q = 0.48$. The value of q is selected by increasing it until the centroids obtained by GREEDY become nonempty. The results averaged over ten restarts are shown in Table 3. Again GREEDY obtains a better performance than MEDIAN in terms of the clustering cost and the running time even though it uses more iterations to converge.

The centroids of the clustering with the lowest cost are shown in Fig. 6. An analysis of the centroids reveals that the clusters capture very different types of preferences. The bottom cluster contains all metal fans, whereas the top-left cluster contains all those who like anything but metal. Quite interestingly, the users in the top-right cluster seem to be indifferent to genre, however, preferring Finnish artists over foreign ones. To make the structure of the DAGs more visible, we removed all edges $e = (u, v)$ if there was an alternative path from u to v via some intermediate vertices.

7.3.3 Preference prediction

To obtain further evidence that the obtained clusters are meaningful, we apply them to a music preference prediction problem. For each test user, we use k randomly selected preferences to determine the closest cluster centroid c and then predict all the remaining preferences by taking the majority vote over all train users in cluster c. As a baseline, we use a majority vote over the train users in all clusters. The results in Fig. 7 show that we obtain up to an 8 % absolute improvement over the baseline method. Note that further improvements might be attainable using more sophisticated methods but the purpose of this experiment is merely to show that the proposed clustering method is able to group people with similar preferences.

[2] The dataset can be downloaded at http://users.ics.aalto.fi/emalmi/artist_preference_data.zip.

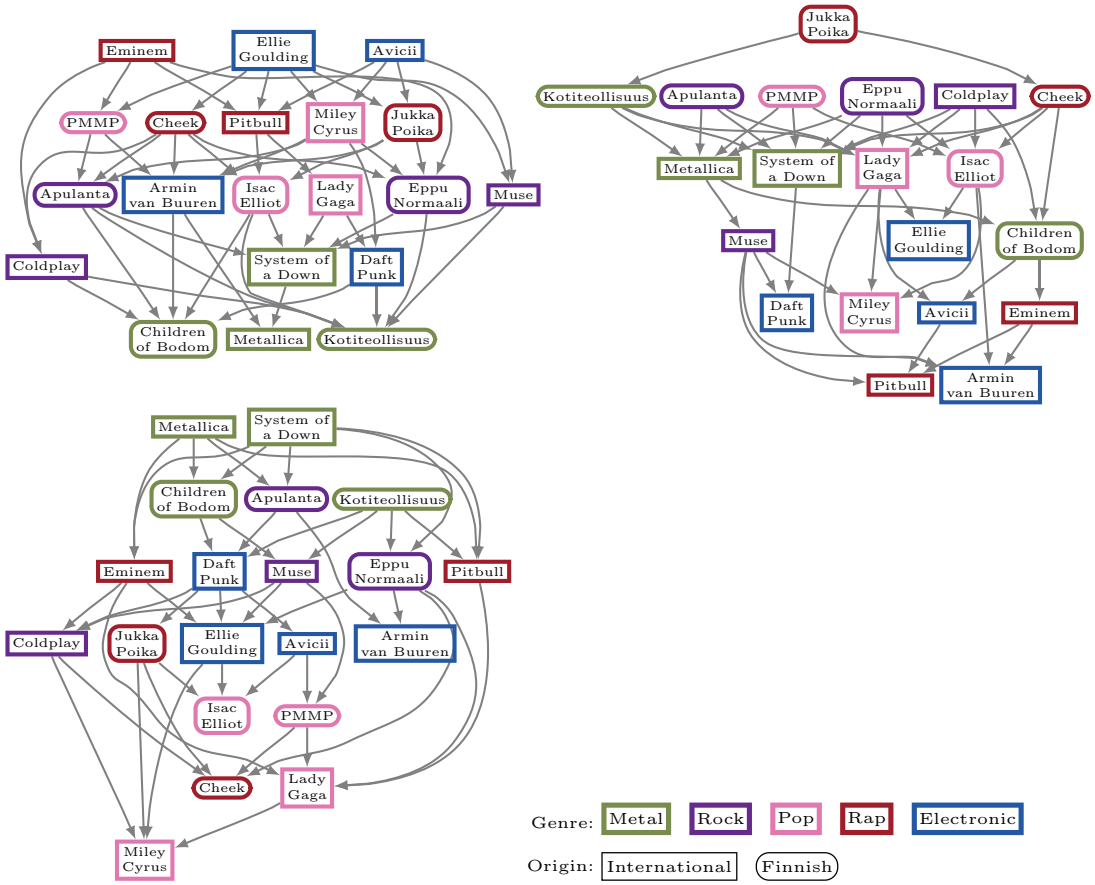

Fig. 6 Centroids for a clustering of the artist preference data

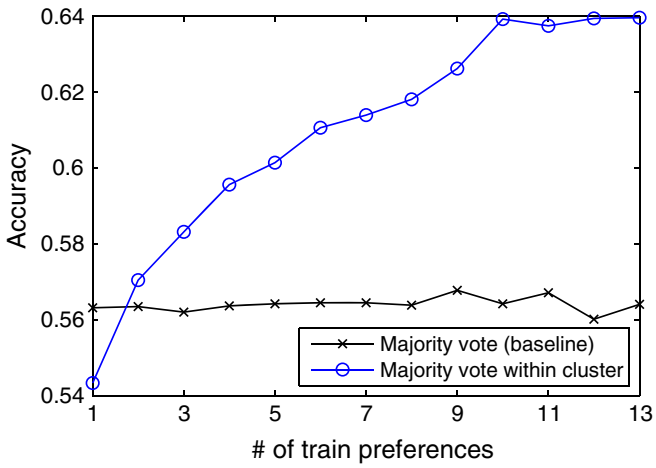

Fig. 7 Artist preference prediction results

8 Conclusions

In this paper, we suggested a natural generalization of Kendall-tau distance to directed acyclic graphs. We showed that this distance is a near-metric, that is, it satisfies a relaxed version of triangle inequality. We considered two applications of the distance: DAG aggregation and clustering. We were able to use the near-metric property to show that we can obtain a constant approximation guarantee for the DAG aggregation

problem using a median approach. DAG aggregation, in turn, can used to obtain a clustering by running a k-means type algorithm.

Our measure has potential applications in information-scarce rank analysis, where instead of full rankings we only have partial ranking information. Interestingly enough, DAGs also arise naturally from information cascades. The nature of the problem here is different: in rank analysis we have DAGs because we have missing information while in cascade analysis, DAGs contain all information that we hope to have. Here clustering and analyzing DAGs also have many applications, for example, by studying different clusters and their centroids, we can find the influential users in each cluster.

We run experiments both on synthetic data, measuring how well we are able to recover a planted clustering, and on real-world data regarding user preferences and information cascades, measuring the clustering cost and running time. In most cases a simple greedy approach outperformed MEDIAN , an algorithm for which we have a constant approximation guarantee. Furthermore, we showed that the obtained clusterings reveal meaningful and occasionally surprising information. For instance, a clustering of music preferences showed that while the preferences of some users are dictated by the genre of the music, for others they depend on the nationalities of the artists.

Our work opens several lines for future work. First, we could study how to rank items from a given set of DAGs. Second, we saw that DAG aggregation and FEEDBACK ARC SET problem are intimately related to each other. The latter is a well-studied problem and it would be fruitful to see whether this connection can be used further to establish new theoretical complexity results concerning both problems. Third, the standard Kendall-tau coefficient is often used as a test statistic for studying the dependency between two variables. Similarly, it would be interesting to look into whether the proposed measure could be used as a test statistic for assessing the dependency between two graphs.

A limitation of the proposed distance measure is that it is not clear how to find the best values of parameters p and q since these are problem dependent as our experiments show. In the experiments with real-world datasets, we have used a simple heuristic proposed in Sect. 7.1, but it remains an open problem how to select the parameter values optimally.

Acknowledgments The authors are grateful to Nicola Barbieri for providing the Last.fm dataset. We also thank the anonymous reviewers for their constructive feedback. This work was supported by Academy of Finland grant 118653 (ALGODAN).

References

Ailon N (2010) Aggregation of partial rankings, p-ratings and top-m lists. Algorithmica 57(2):284–300

Ailon N, Charikar M, Newman A (2008) Aggregating inconsistent information: ranking and clustering. J ACM 55(5):23

Anagnostopoulos A, Kumar R, Mahdian M (2008) Influence and correlation in social networks. In: Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining. pp 7–15

Barbieri N, Bonchi F, Manco G (2013) Cascade-based community detection. In: Proceedings of the sixth ACM international conference on Web search and data mining. pp 33–42

Bender MA, Fineman JT, Gilbert S, Tarjan RE (2011) A new approach to incremental cycle detection and related problems. arXiv:1112.0784

Borda J (1781) Mémoire sur les élections au scrutin. Histoire de l'Académie Royale des Sciences

Brandenburg F, Gleißner A, Hofmeier A (2012) Comparing and aggregating partial orders with Kendall tau distances. In: WALCOM: algorithms and computation. Lecture notes in computer science, vol 7157. Springer Berlin Heidelberg, pp 88–99

Brandenburg F, Gleißner A, Hofmeier A (2013) The nearest neighbor Spearman footrule distance for bucket, interval, and partial orders. J Comb Optim 26(2):310–332

Bunke H, Shearer K (1998) A graph distance metric based on the maximal common subgraph. Pattern Recognit Lett 19(3):255–259

Dinur I, Safra S (2005) On the hardness of approximating minimum vertex cover. Ann Math 162(1):439–485

Dwork C, Kumar R, Naor M, Sivakumar D (2001) Rank aggregation methods for the web. In: Proceedings of the 10th international conference on World Wide Web. pp 613–622

Even G, Naor J, Schieber B, Sudan M (1995) Approximating minimum feedback sets and multi-cuts in directed graphs. In: Proceedings of the 4th international conference on integer programming and combinatorial optimization. pp 14–28

Fagin R, Kumar R, Mahdian M, Sivakumar D, Vee E (2006) Comparing partial rankings. SIAM J Discrete Math 20(3):628–648

Fagin R, Kumar R, Sivakumar D (2003) Comparing top-k lists. SIAM J Discrete Math 17(1):134–160

Friedman JH, Rafsky LC (1979) Multivariate generalizations of the Wald-Wolfowitz and Smirnov two-sample tests. Ann Stat 7(4):697–717

Gomez-Rodriguez M, Balduzzi D, Schölkopf B (2011) Uncovering the temporal dynamics of diffusion networks. In: Proceedings of the 28th international conference on machine learning. pp 561–568

Gomez-Rodriguez M, Leskovec J, Krause A (2012) Inferring networks of diffusion and influence. ACM Trans Knowl Discov Data 5(4):21

Goodman LA, Kruskal WH (1972) Measures of association for cross classifications, iv: simplification of asymptotic variances. J Am Stat Assoc 67(338):415–421

Goyal A, Bonchi F, Lakshmanan LVS (2008) Discovering leaders from community actions. In: Proceedings of the 17th ACM conference on information and knowledge management. pp 499–508

Goyal A, Bonchi F, Lakshmanan LVS (2010) Learning influence probabilities in social networks. In: Proceedings of the third ACM international conference on Web search and data mining. pp 241–250

Hubert L, Arabie P (1985) Comparing partitions. J Classif 2(1):193–218

Jiang X, Munger A, Bunke H (2001) An median graphs: properties, algorithms, and applications. IEEE Trans Pattern Anal Mach Intell 23(10):1144–1151

Kann V (1992) On the approximability of np-complete optimization problems. Ph.D. thesis, KTH

Karp RM (1972) Reducibility among combinatorial problems. In: Complexity of computer computations. Springer, New York

Kempe D, Kleinberg J, Tardos É (2003) Maximizing the spread of influence through a social network. In: Proceedings of the ninth ACM SIGKDD international conference on knowledge discovery and data mining. pp 137–146

Kendall M (1938) A new measure of rank correlation. Biometrika 30:81–93

Kendall M (1976) Rank correlation methods, 4th edn. Hodder Arnold, London

Kenyon-Mathieu C, Schudy W (2007) How to rank with few errors. In: Proceedings of the 39th annual ACM symposium on theory of computing. pp 95–103

Laming D (2003) Human judgment: the eye of the beholder. Cengage Learning EMEA

Macchia L, Bonchi F, Gullo F, Chiarandini L (2013) Mining summaries of propagations. In: Proceedings of the 13th IEEE international conference on data mining. pp 498–507

Madden JI (1995) Analyzing and modeling rank data. Chapman & Hall, London

Murphy TB, Martin D (2003) Mixtures of distance-based models for ranking data. Comp Stat Data Anal 41(3–4):645–655

Saito K, Nakano R, Kimura M (2008) Prediction of information diffusion probabilities for independent cascade model. In: Knowledge-based intelligent information and engineering systems. pp 67–75

Su H, Gionis A, Rousu J (2014) Structured prediction of network response. In: Proceedings of the 31st international conference on machine learning. pp 442–450