
Maximizing Agreements for Ranking, Clustering and Hierarchical Clustering via MAX-CUT

Vaggos Chatziafratis
Google Research NY

Mohammad Mahdian
Google Research NY

Sara Ahmadian
Google Research NY

Abstract

In this paper, we study a number of well-known combinatorial optimization problems that fit in the following paradigm: the input is a collection of (potentially inconsistent) local relationships between the elements of a ground set (e.g., pairwise comparisons, similar/dissimilar pairs, or ancestry structure of triples of points), and the goal is to aggregate this information into a global structure (e.g., a ranking, a clustering, or a hierarchical clustering) in a way that maximizes agreement with the input. Well-studied problems such as rank aggregation, correlation clustering, and hierarchical clustering with triplet constraints fall in this class of problems. We study these problems on stochastic instances with a hidden embedded ground truth solution. Our main algorithmic contribution is a unified technique that uses the maximum cut problem in graphs to approximately solve these problems. Using this technique, we can often get approximation guarantees in the stochastic setting that are better than the known worst case inapproximability bounds for the corresponding problem. On the negative side, we improve the worst case inapproximability bound on several hierarchical clustering formulations through a reduction to related ranking problems.

1 Introduction

In many learning/optimization problems, the input data is in the form of a number of *ordinal* judgements about the local relationships among a set of n items. A prominent example is the problem of ranking n al-

ternatives, where the input is often pairwise comparisons between these items. For example, sports teams are often ranked by aggregating the results of matches played between pairs of teams, and election outcomes are decided by aggregating individual votes.

Learning from comparisons has been prevalent across different domains, as humans are typically good at quickly answering *ordinal* questions (“which movie/restaurant/candidate do you prefer”), but often respond slowly and inaccurately to *cardinal* questions (“how much do you like this option”). In the psychology literature, the method of *paired comparisons* that has been in use since the 1920’s is based on this principle (see (Thurstone, 1959, Chapter 7)). Moreover, modern online platforms can organically extract such ordinal preferences by observing the users (e.g., “which movie did they first watch”, or “did they skip a search result and click on the next one”) and later use them for improving search or recommendation rankings (see, for example, Joachims (2002)). The same principle applies to settings other than ranking. For example, when trying to learn a clustering of n items, it is easier for a human judge to answer questions of the form “should x and y be in the same cluster” than to measure the similarity of x and y . Or, to reconstruct the evolutionary tree (also known as the phylogenetic tree) between n species, biologists often start by answering questions of the form “between three species x, y , and z , which two are evolutionarily closer”.

At the heart of each of these examples is the non-trivial algorithmic task of reconciling potentially inconsistent judgements into a global solution. This defines a number of algorithmic problems that we study in this paper. Though seemingly unrelated, all of these problems seek to find a global structure that has the maximum number of agreements with the given collection of local ordinal relationships. As we shall see later in the paper, the problems are also linked in that we can apply a common technique (based on graph max cuts) to them all. The problems, shown in Figure 1, fall under the three categories of ranking, clustering, and hierarchical clustering:

- **Ranking:** The goal is to find an ordering of n items. In the *Maximum Acyclic Subgraph* (MAS), the input is a number of pairwise comparisons of the form $a < b$. In *Betweenness*, the input is a number of triples $a|b|c$ meaning that b is between a and c in the ordering. In *Non-Betweenness*, the input is a number of triples $b|ac$ meaning that b is not between a and c .
- **Clustering:** In the *Correlation Clustering* problem, the goal is to find a partitioning of n items, and the input is a number of pairs of the form ab , meaning that a and b should be in the same cluster, and a number of pairs of the form $a|b$, meaning that a and b should be in different clusters.
- **Hierarchical clustering:** The goal is to find a (rooted or unrooted) tree with the set of n items as its leaves. In the *Desired Triplets* problem, the input is a number of triplets $ab|c$, meaning that the least common ancestor of a and b is a descendant of the least common ancestor of a, b , and c . In the *Desired Quartets* problem, the input is a number of quartets $ab|cd$, meaning that the unique path connecting a and b in the tree does not intersect with the unique path connecting c and d . The *Forbidden Triplets* and *Forbidden Quartets* problems are defined similarly with the opposite requirements.

These problems come from a variety of applications: MAS is a formulation of the rank aggregation problem and has many applications, e.g., in search ranking. Correlation Clustering is a central problem in unsupervised learning and data analysis (Bansal et al., 2004). Hierarchical clustering problems are motivated by applications in reconstructing phylogenetic trees (Felsenstein, 2004), and are also related to the objective-driven formulations of Dasgupta (2016), Moseley and Wang (2017) and Cohen-Addad et al. (2019) for hierarchical clustering. In fact, the Desired Triplets formulation described above is tightly connected with objective-based approaches for Hierarchical Clustering as can be seen in Charikar et al. (2019a,b). Betweenness and Non-Betweenness are motivated by applications in genome sequencing in bioinformatics (Slonim et al., 1997). We are interested in algorithms that can provide an approximation guarantee, i.e., a provable bound on the multiplicative factor between the solution found by the algorithm and the optimal solution. We will consider this problem both in the worst case and under a stochastic model with an embedded ground-truth solution. Our contribution is two-fold (see Table 1 for a summary):

On the positive side, in Section 3, under a simple stochastic model akin to the well-known *stochastic*

block model, we are able to improve upon worst-case approximations for all problems and in some cases (e.g., for problems on rankings and hierarchies) even overcome impossibility results. Interestingly, our algorithms are all based on variants of MAXCUT on graphs that can have both positive and negative weights and may also be directed. Some approaches for tree reconstruction based on MAXCUT had been used in previous experimental works (Snir and Rao, 2006, 2008, 2012), and in this way our work provides concrete proof for why these heuristics are reported to perform well on “real-world” instances. Our natural stochastic model captures “real-world” instances via an embedded ground-truth from which we generate “noisy” constraints, similar to the Stochastic Block Model (Mossel et al., 2012) in community detection.

On the negative side, we obtain new hardness of approximation results for four problems on hierarchical clustering: Forbidden Triplets, Desired Triplets, Forbidden Quartets, Desired Quartets. Briefly, we may refer to them as triplets/quartets consistency problems. These are instances of Constraint Satisfaction Problems (CSP) on trees (Bodirsky and Mueller, 2010; Bodirsky et al., 2016), analogous to SAT formulas in complexity. Even though such problems on hierarchies have been studied for decades, the current best approximations are achieved by trivial baseline algorithms. Our hardness results give some explanation why previous approaches were not able to obtain anything better. Our result on the Forbidden Triplets problem is tight and is the first tight hardness for CSPs on trees, extending analogous hardness results by Guruswami et al. (2011) from linear orderings (i.e., rankings) to trees. This is carried out in Section 4.

Our stochastic model for collecting information is the simplest form of embedded model on n items, and is motivated by crowdsourcing and biological applications (Vaughan, 2017; Kleindessner and von Luxburg, 2017; Ghoshdastidar et al., 2019; Snir and Yuster, 2012). We simply choose items at random and include a pairwise/triplet/quartet constraint depending on the task. For example, to generate constraints for the MAS problem on rankings, let π^* denote a ground-truth ranking (e.g., of chess players or ads to show a user). We select uniformly at random m pairs of items a_i, b_i and then we generate m pairs $a_i < b_i$; if a_i precedes b_i in π^* the constraint is included with probability $(1 - \epsilon)$, otherwise the opposite constraint is generated. Thus, some fraction of the constraints can be erroneous. After generating m (noisy) constraints in this way, our goal is to find a global solution (ranking, partition, or tree) that satisfies as many as possible.

Techniques: Our hardness reductions for Maximum Forbidden Triplets consistency are based on mapping

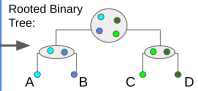
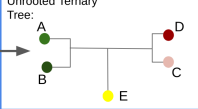
Problem Name	Types of Constraints	Types of Solutions
Max Acyclic Subgraph (MAS) Betweenness (BTW): Non-Betweenness (Non-BTW):	A<B, B<C, B<D, D<C A B C, B C D, D C A, B D C A BC, B DC, D CA, C DB	Ranking: A, B, C, D Ranking: A, B, C, D Ranking: A, B, C, D
Correlation Clustering:	Must-Link & Cannot-Link (AB) (A D) (BC) (B D) (CD) (D E)	Partition: A, B, C, D, E
Desired/Forbidden Triplets:	Desired & Forbidden AB C AC B CD B AD C AC D AB C	Rooted Binary Tree: 
Desired/Forbidden Quartets:	Desired & Forbidden AB CD AC BD AE CD AD CB ED AB AE CB AE BC AB ED AC BE AB CD	Unrooted Ternary Tree: 

Figure 1: A schematic representation of all problems considered in the paper. The left column has the problem names, the middle the types of constraints and the right column has a candidate solution. With green are constraints that are correctly resolved in the given candidate solution, whereas with red are those that are incorrect. For more examples, see Section 2.

trees to permutations on their leaves and back, and showing that any constant factor improvement over trivial baselines would refute the Unique Games Conjecture¹ (UGC) (Khot, 2002). Regarding our MAXCUT algorithm (see Algorithm 1), it is based on MAXCUT variations on directed and undirected graphs with negative weights and is conceptually simple. Briefly, given an instance for any of the problems we consider, we map it to a graph where edges encode the underlying constraints; perhaps the most intuitive such construction is for Correlation Clustering where a “must-link” or “cannot-link” constraint between items i, j is captured by a negative or positive edge (i, j) respectively. Then, we show how large (positive) cuts in this graph yield partitions that satisfy many of the constraints. The existence of a large cut can be guaranteed by analyzing our stochastic model and so an approximate MAXCUT algorithm can yield improvements over previous results. An interesting ingredient that we need for the case of MAS, is how to approximate the MAXCUT problem on *directed* graphs with both positive and negative weights which, to the best of our knowledge, hadn’t been analyzed before.

More broadly, we justify theoretically why prior experimental heuristics work and we extend them to work for new problems with provable approximation guarantees. Our work also presents the first case of a CSP on trees that is approximation resistant; recall that many important CSPs, including Max3SAT, are approximation resistant, i.e., it is NP-hard to approximate them better than a random assignment. This echoes the striking result by Håstad (2001) on ap-

¹Khot’s UGC is a major open question in complexity. We will not define it here as we only use some of its consequences on ordering problems (Guruswami et al., 2011).

proximation resistance of boolean CSPs to CSPs on trees and shows why no algorithmic improvement had been made in the worst-case, despite significant efforts (Byrka et al., 2010; Jiang et al., 2001; Bryant, 1997; He et al., 2006; Steel, 1992).

Table 1: Shown in bold are our improved hardness (column “Hardness”) and approximations under our stochastic model (column “Stochastic”). Column “Approx.” has prior approx. ratios. Also see Section 3 and Appendix A for the dependence on error parameter ϵ .

	Approx.	Hardness	Stochastic
MAS	1/2	1/2	0.642
BTW	1/3	1/3	0.402
NON-BTW	2/3	2/3	0.84
Correl. Cl.	0.76	APX-hard	0.82(*)
Forb. Triplet	2/3	2/3 (tight)	0.78(*)
Des. Triplet	1/3	2/3	0.64(*)
Forb. Quartet	2/3	8/9	0.672
Des. Quartet	1/3	2/3	0.425

Remark 1. We want to point out that all our approximation results here hold with high probability as a standard concentration argument about the stochastic process guarantees that the weight of the cuts is well-concentrated around its mean (as long as the number of generated constraints $m \geq \Omega(\log n)$).

Remark 2. Our results for ranking and quartets hold with no assumption on the optimal solution. For the positive results (denoted with $(*)$ in Table 1) via MAXCUT for correlation clustering and triplets however, we need a mild balancedness assumption, roughly stating that the optimal solution contains a relatively balanced $(\frac{1}{3} : \frac{2}{3})$ partition, to ensure the existence of a good cut in the ground-truth (see Appendix, Assumption 1). Usually, such assumptions are common in generative graph models for clustering, e.g., the Stochastic Block Model (Mossel et al., 2012; Abbe et al., 2015) and for hierarchical clustering, e.g., the Hierarchical Stochastic Block Model (Lyzinski et al., 2016; Cohen-Addad et al., 2019; Ghoshdastidar et al., 2019), where we expect to see at least two large communities emerge.

2 Background and Related Work

As the paper discusses multiple problems on rankings, partitions and hierarchies, we devote this section in describing the multitude of problems. A familiar reader can skip this section and proceed to Section 3.

There are 3 categories of problems we study here, depending on the type of the output: ranking (also called a *permutation* or a *leaf ordering* in biology (Bar-Joseph et al., 2001)), clustering (partitioning of the data points) and hierarchical clustering (also called

phylogenetic tree). There has been significant amounts of work on each of these tasks, that we only partially cover here as we go over our problems and results.

2.1 Optimization Problems and Types of Constraints

In all problems, we are given m constraints and we want to maximize the number of constraints satisfied by our output, whether it be a ranking, a partition or a hierarchy. We describe below the types of different constraints (see also Figure 1):

Ranking (i.e., a permutation or leaf ordering): Given n labels $\{1, 2, \dots, n\}$, we want to find a permutation that maximizes the number of satisfied constraints of the following form:

- **Pairwise comparisons:** A constraint here is of the form “ $a < b$ ”, indicating that in the output permutation, item a should precede b . If this information is encoded as a directed graph G with arcs $a \rightarrow b$, this gives rise to the Maximum Acyclic Subgraph (MAS) or Feedback Arc Set (FAS), two fundamental problems in computer science (Karp, 1972).
- **Betweenness (BTW) and Non-Betweenness (Non-BTW) constraints:** In the BTW problem (Opatrny, 1979; Chor and Sudan, 1998; Makarychev, 2012), we are given relative ordering constraints of the form $a|b|c$ indicating “ b should be between a and c ”. This allows for abc or cba out of the 6 possible orderings for the 3 labels. As the name suggests, NON-BTW is the complement of BTW, where a constraint $bc|a$ (equivalently $a|bc$) indicates that in the output permutation “ a should *not* lie between b and c ”. This allows for 4 valid relative orderings abc, acb, bca, cba . Generally, these are the two most common examples of ordering Constraint Satisfaction Problems (ordering CSPs) of arity 3 and are mainly motivated by applications in bioinformatics (Slonim et al., 1997). They have also played a major role in complexity (Guruswami et al., 2011; Austrin et al., 2013).

Just to give a sense of the approximability of these problems in the worst-case, the current best constant factor is a $\frac{1}{2}$ -approximation for MAS, a $\frac{1}{3}$ -approximation for BTW, and a $\frac{2}{3}$ -approximation for NON-BTW, all achieved by a *random* permutation. We also know that under the Unique Games Conjecture (UGC) of Khot (2002), the first two results are tight, whereas the third is tight under $P \neq NP$. Such problems, where a random output is provably the best, are called *approximation resistant* and have been studied extensively by theoreticians (Charikar et al., 2009; Guruswami et al., 2008; Hästad, 2001; Austrin

and Mossel, 2009). Our work gives strong evidence pointing to the fact that important CSPs on trees (triplets/quartets) may be approximation resistant.

Clustering: Here we want to maximize agreements with **Must-Link/Cannot-Link** constraints: The input is a graph with “+” or “-” edges indicating if the two endpoints should belong to the same cluster or not. Such constraints give rise to Correlation Clustering, an important paradigm for data analysis both in practice (Davidson and Basu, 2007; Wagstaff and Cardie, 2000; Wagstaff et al., 2001) and theory (Bansal et al., 2004; Ailon et al., 2008; Charikar et al., 2005; Swamy, 2004). The current best for maximizing agreements is a 0.7666 multiplicative approximation via semidefinite programs (Swamy, 2004) and an APX-hardness is known (Charikar et al., 2005). Here we will improve upon 0.7666, under our stochastic model for generating constraints.

Hierarchical Clustering (i.e., phylogenetic trees): There are two common types of trees: rooted and unrooted. Given n data points, a rooted binary tree on n leaves, where each leaf corresponds to a data point, is usually called a *hierarchical clustering* and is a standard tool for data analysis across different disciplines (Steinbach et al., 2000; Leskovec et al., 2014; Tumminello et al., 2010; Sørlie et al., 2001). Unrooted ternary trees (all nodes have degree 3, except the leaves that have degree 1) are usually called *phylogenetic trees* and are prevalent in computational biology as they describe speciation events throughout the evolution of species (Bryant, 1997; Felsenstein, 2004). Here we will use the two terms interchangeably to describe hierarchies on n leaves. Since in a hierarchy all data are eventually separated at the leaves, pairwise constraints no longer make sense and the analogue of “must-link/cannot-link” are so-called “must-link-before/cannot-link-before” constraints:

- **Desired/Forbidden Triplets:** The output here is a rooted binary tree T on n leaves. We say a triplet relation “ $t = ab|c$ ” is *obeyed* by T (or T *obeys* t), if the lowest common ancestor (LCA) of a, b is a descendant of the LCA of a, c in T . Otherwise T *disobeys* $ab|c$. A triplet can be *desired* (we write $t \in \mathcal{T}_D$) and we want the output T to obey it² or *forbidden* (we write $t \in \mathcal{T}_F$) and we want T to disobey/avoid it, giving rise to important optimization problems studied in computational biology and graph theory under the name of rooted triplets consistency (Steel, 1992; Bryant, 1997; Byrka et al., 2010; He et al., 2006). Notice that a forbidden triplet $ab|c$ is less restrictive, since it only specifies that T should either obey

²For example, “penguin, dolphin| tiger” could be a desired triplet as the tiger is the least relevant item.

$ac|b$ or $bc|a$, but not $ab|c$. This is reflected in the complexity of the problems: given a set of forbidden triplets, it is NP-complete to check consistency (i.e., if there is a tree avoiding all of them), whereas checking consistency of desired triplets in polynomial time was established long ago by Aho et al. (1981).

- **Desired/Forbidden Quartets:** The desired output here is a ternary unrooted tree T . We say a quartet $q = ab|cd$ is *obeyed* by T (or T *obeys* q) if the (unique) path from a to b in T does not share any vertices with the (unique) path from c to d in T . Otherwise T *disobeys* q . Similarly to triplets, a quartet can be *desired* ($q \in \mathcal{Q}_D$) or *forbidden* ($q \in \mathcal{Q}_F$), giving rise to important quartets consistency problems in biology and graph theory (Felsenstein, 2004; Bryant, 1997; Jiang et al., 2001; Snir and Rao, 2006). For both problems, even if the input is consistent, checking consistency is NP-complete.

Once again, just to give a sense of the approximability, for desired triplets or quartets, the current best is a $\frac{1}{3}$ -approximation and for forbidden triplets or quartets, the current best is a $\frac{2}{3}$ -approximation. Embarrassingly, in all four cases these are achieved by a random (rooted or unrooted) tree or a simple greedy construction (He et al., 2006).

2.2 Further Motivation and Related Work

Here, we further make a comparison to other relevant works. For ranking, many different types of probabilistic models have been considered (Braverman and Mossel, 2009; Shah et al., 2016; Shah and Wainwright, 2017; Negahban et al., 2012; Falahatgar et al., 2017) giving statistical guarantees for reconstructing the desired permutation. Instead of pairwise comparisons, the problem has also been studied in the case where partial rankings or complete information (“tournaments”) is provided (Fagin et al., 2006; Ailon, 2010; Kenyon-Mathieu and Schudy, 2007). Clustering with constraints and qualitative information (both max and min versions) were studied in Bansal et al. (2004); Charikar et al. (2005) where approximations via linear programs were derived or practical improvements were made possible (Wagstaff et al., 2001; Wagstaff and Cardie, 2000). In crowdsourcing and biological applications, both triplet and quartets queries have been deployed (Vinayak and Hassibi, 2016; Vaughan, 2017; Kleindessner and von Luxburg, 2017; Ghoshdastidar et al., 2019; Snir and Rao, 2006; Bryant, 1997) as they can be more intuitive for non-expert users compared to pairwise comparisons. Semi-supervised models, where triplet queries depend on answers to previous queries have been studied in Emamjomeh-Zadeh and Kempe (2018); Vikram and Dasgupta (2016).

To further motivate our stochastic model and results, we include a slightly more detailed comparison with 3 important prior works Braverman and Mossel (2009); Emamjomeh-Zadeh and Kempe (2018); Snir and Yuster (2012) that study “ground-truth” stochastic models similar to ours. The authors in Braverman and Mossel (2009) study the ranking problem and assume that there exists a ground-truth ranking π^* , as we do. However, their stochastic model assumes either that we have access to *all* $\binom{n}{2}$ pairwise comparisons, or that we have access to *complete* rankings σ on the n items, where each complete ranking σ is generated with probability inverse exponential in the Kemeny distance between π^* and σ (Kemeny distance is the number of inversions, i.e., the number of pairs ordered in π^* differently from σ).

As it will become obvious, their assumptions are much stricter than our simple stochastic model that generates m pairwise comparisons uniformly at random. Moreover, notice that our approximation guarantees hold for *any* number m of given constraints without requiring it to be $\Omega(n^2)$. Given their more refined model, they are of course in a position to analyze the maximum likelihood estimator and prove approximate recovery results, e.g., that no element is misplaced by more than $\log n$ positions with high probability; however no guarantees are given for the number of violated pairwise constraints, which is the focus of our paper.

For triplets hierarchical clustering, the authors in Emamjomeh-Zadeh and Kempe (2018) assume there exists a ground-truth binary tree T , as we do. However, they are allowed *adaptive* triplet queries and show that $\approx n \log n$ such queries suffice to recover T using a clever partition algorithm similar to Quicksselect and Quicksort. Once again, our model is not adaptive, and we do not pose any constraints on the number m of given constraints. For quartets hierarchical clustering, our model is similar to Snir and Yuster (2012), but we generalize their results to hold both for forbidden and desired quartets.

Finally, our constrained version of Hierarchical Clustering based on triplet constraints was studied in Chatziafratis et al. (2018) under the assumption that the input contains pairwise similarities as well as triplet constraints.

3 Using MaxCut on instances with embedded ground-truth

We present our main strategy MAXCUT behind our positive results. As we will see, by modifying the graphs, our method is flexible to allow for combinations of constraints, e.g., both BTW and NON-BTW constraints for rankings, or both desired and forbidden triplets (or quartets) for trees.

Stochastic Model for Generating Constraints:

Since our goal is to beat the worst-case approximation and hardness results, we use a simple stochastic model with an embedded ground-truth solution on n items. The form of the ground-truth changes depending on which problem we consider; it can be a ranking (for MAS, BTW, NON-BTW), a partition (for Correlation Clustering) or a hierarchical tree (rooted for Triplets and unrooted for Quartets). For generating the m input constraints, we simply choose items at random and with probability $(1 - \epsilon)$ we add a pairwise/triplet/quartet constraint that is consistent with the ground-truth, otherwise with probability ϵ we add an erroneous constraint on the selected items. For example, in the introduction, we saw the MAS constraints. Similarly, for BTW, we would uniformly at random pick m triples of items a, b, c and then add w.p. $(1 - \epsilon)$ the constraint $a|b|c$ if b appears in between a and c in the ground-truth ordering. Also, for the Triplets Consistency problem, we would again uniformly at random pick m triples of items a, b, c and then add w.p. $(1 - \epsilon)$ the constraint $ab|c$ if c is separated first from a, b in the ground-truth (rooted binary) tree. For all problems, after getting m (noisy) constraints in the analogous manner, our goal is to find a global solution that satisfies as many constraints as possible.

Positive Results: Using our stochastic model we can escape worst-case impossibility results and for all 3 categories of problems, we present improved approximation algorithms. At a high-level, we first construct a graph by encoding each of the local constraints on the items as a set of positive or negative edges between them. The graph captures the desired relationships and then, we find a good first split maximizing the ratio of satisfied over violated constraints by the cut. Naturally, our algorithm MAXCUT (see Algorithm 1) is based on variants of MAXCUT on graphs with negative weights. An interesting building block in our analysis when solving for better Maximum Acyclic Subgraphs, is the directed MAXCUT problem on graphs with negative weights which, to the best of our knowledge, hadn't been analyzed before. We note that for the triplets problem on trees, analogous MAXCUT heuristics had been successfully used before in experimental work for computational biology, however with no theoretical guarantees (Snir and Rao, 2006, 2012, 2008). An exception is the work of Snir and Yuster (2012), where they focus only on the desired quartets problem, however their analysis is a special case of ours for when $\mathcal{Q}_{\mathcal{F}} = \emptyset$ (i.e., the input contains no forbidden quartets). Our final approximations circumvent known hardness results for the case of rankings (Guruswami et al., 2011) and our new hardness results for trees described in detail later in Section 4.

3.1 Better Approximations for MAS

We start with MAS as it is perhaps the easiest to describe (see also Algorithm 1):

Theorem 1. *Given m constraints generated according to our stochastic model on n items, MAXCUT satisfies at least $(0.642 - 0.4285\epsilon)m$ on average, where ϵ is the fraction of erroneous comparisons. If moreover $m \geq \Omega(\log n)$, the result holds w.h.p.*

Remark 3. *For example, if the error parameter $\epsilon = 0.1$, hence 10% of the m generated constraints are erroneous, we still satisfy $\approx 60\%$ of them, and we still beat the previous best $\frac{1}{2}$ -approximation together with the known hardness (Guruswami et al., 2008).*

Our general proof template has 5 steps:

- Building a graph: For a sampled constraint $a < b$ indicating that a should precede b in the ranking, we add two directed edges:

+1 directed from $a \rightarrow b$, -1 directed from $b \rightarrow a$

Since the problem has orientation, we define the weight of a directed cut (S, \bar{S}) as the sum of all (positively or negatively) weighted arcs going from S to \bar{S} (and we ignore the arcs going from \bar{S} to S).

- Cuts and constraints: The goal of constructing the graph is to use information about its cuts and relate them to the pairwise constraints. Notice that a cut (S, \bar{S}) can either obey, disobey or leave unaffected the status of a $a < b$ constraint, depending on if a or b belongs to S or \bar{S} . Let m_s, m_v denote the satisfied, violated constraints by the cut, respectively. The weight of any directed (S, \bar{S}) cut is thus:

$$w(S, \bar{S}) = m_s(S, \bar{S}) - m_v(S, \bar{S}) \quad (1)$$

as satisfied pairs m_s (with $a \in S, b \in \bar{S}$) contribute $+1$ and violated pairs m_v (with $a \in \bar{S}, b \in S$) contribute -1 .

- Lower Bounding MAXCUT: The constructed graph from the first step, is directed and has both positive and negative weights. Based on eq. (1), we should find a large cut in this graph as this translates to many satisfied constraints. In order to find the cut, we use a MAXCUT variant that finds a cut comparable to the optimal max cut in graphs that are directed and contain both positive and negative weights. However, we cannot use the standard Goemans-Williamson algorithm and guarantees Goemans and Williamson (1995), as the graph is directed with positive and negative weights. A new ingredient in our proof is a semidefinite programming relaxation and analysis for this variant that achieves:

$$\mathbb{E}(w(S, \bar{S})) \geq 0.857w(\text{OPT}) - 0.143 \cdot W^- \quad (2)$$

where $w(\text{OPT})$ is the weight of the optimum cut and W^- is the total negative weight in the graph in absolute value. Based on the graph construction in this case, $W^- = m$ as every constraint contributed a -1 edge. We just note that the numerical values 0.143 and 0.857 sum to 1, and they just arise from the rounding scheme used to obtain an integral solution from the relaxation.

- Now that we have a lower bound for $w(S, \bar{S})$ based on the optimum cut, in order to conclude the algorithm’s cut is large (and hence satisfies many constraints), we need to lower bound the optimum’s cut weight $w(\text{OPT})$. To do this we consider the weight of a *median* directed cut: the median cut is defined to be the one that assigns the first $n/2$ labels in the optimum ordering for MAS, on one side of the cut, and the rest $n/2$ labels to the other side of the cut. Since the labels for the constraints according to our stochastic model were chosen at random, a counting argument implies that with high probability $\approx \frac{1}{2}m$ of the generated constraints are satisfied by the median cut and hence also by OPT . To see this, observe that for nearly half of the $a < b$ constraints, a belongs to the first $n/2$ labels of the median cut, whereas b belongs to the remaining $n/2$ labels. Since OPT is by definition even better than the median cut, we get that it has a large cut value. If we wanted to be slightly more precise, we should say that due to errors in an ϵ fraction of the generated constraints, we actually lose a small ϵ fraction of the constraints (we defer details to Appendix A) but this discounts the optimum cut only by a small amount.
- Output of MAXCUT : Finally, we need to find a good permutation overall, not just a good top split. Our algorithm starts by finding an approximate MAXCUT (S, \bar{S}) in G and then proceeds by outputting a *random* permutation on the items in S and in \bar{S} and concatenating them. Finally, we can compute the overall value of ALG (dropping the notation with (S, \bar{S})):

$$\begin{aligned} \text{ALG} &= m_s + \frac{1}{2}m_u = \\ &= m_s + \frac{1}{2}(m - m_s - m_v) = \frac{1}{2}m + \frac{1}{2}(w(S, \bar{S})) \end{aligned} \quad (3)$$

where m_u are the constraints that were unaffected by the (S, \bar{S}) cut. By eq. (3), we already see that we get some advantage over the $\frac{1}{2}m$ baseline which is optimal in the worst-case (and is achieved by a random permutation on all n items).

Remark 4. A natural question is to attempt to use MaxCut repeatedly on each of the two generated parts

of the first split. However analyzing the repeated MaxCut approach is not that simple, as once the first approximate MaxCut is performed, there is no randomness in the two generated subgraphs that we can exploit. Analogous difficulties arise in dissimilarity-based and quartets-based hierarchical clustering Charikar et al. (2019a); Snir and Yuster (2012); Ahmadian et al. (2020). Finally, we want to point out that such analyses are also known to be challenging from the literature on Random Forests for decision trees (e.g., Scornet et al. (2015)) where a similar (data-dependent) two-step analysis has been elusive.

3.2 Extensions to Other Problems

The same proof template as presented here can be modified to deal with the remaining problems: BTW, NON-BTW, forbidden and desired triplets, forbidden and desired quartets. As each of these constraints, involve 3 or 4 points, the construction and analyses become more involved. We present briefly the main modifications for the graph construction (see Appendix A for details).

For a BTW constraint $\{a|b|c\}$, we add undirected edges: $+2$ for (a, c) and -1 for $(b, a), (b, c)$. The edges capture that a cut violates the constraint if it separates b from a, c . For a NON-BTW constraint $\{ab|c\}$ indicating that c should not be between a, b in the final ordering, we add the following 3 undirected edges: $+1$ for pairs $(c, a), (c, b)$ and -2 for the pair (a, b) . Recall, that for BTW and NON-BTW, the ultimate goal is to beat the factors $\frac{1}{3}$ and $\frac{2}{3}$ which are currently optimal in the worst-case:

Theorem 2. Given $m = \Omega(\log n)$ noisy constraints on n items, variations of MAXCUT satisfy at least $(0.402 - 0.329\epsilon)m$ and $(0.845 - 0.329\epsilon)m$ constraints w.h.p. for BTW and NON-BTW, respectively, where ϵ is the fraction of erroneous constraints.

For Correlation Clustering, for each CANNOT-LINK constraint ab , we add a $+1$ for (a, b) , and for each MUST-LINK constraint ab , we add -3.2735 for edge (a, b) . The chosen numerical value -3.2735 depends on the current best 0.766-approximation for Correlation Clustering (Swamy, 2004) (see Appendix A).

Theorem 3. Given $m = \Omega(\log n)$ noisy “must-link/cannot-link” constraints on n items, MAXCUT (modified appropriately) satisfies at least $(0.8226 - 0.775\epsilon)m$ constraints w.h.p., where ϵ is the fraction of erroneous constraints.

Analogous theorems hold for the Triplets/Quartets consistency problems. Due to space constraints, we omit the statements but we refer the reader to Table 1 for the final ratios and to Appendix A for the proofs.

Algorithm 1 Our MAXCUT template as instantiated for MAS.

Input: m pairwise constraints for MAS.

1. For each $a < b$ constraint, insert a $+1$ arc directed from $a \rightarrow b$ and another arc with negative weight -1 directed from $b \rightarrow a$. Call the resulting graph G .
 2. Run our approximate MAXCUT algorithm suitable for directed graphs with negative weights to get a first split (S, \bar{S}) , satisfying eq. (2).
 3. Construct a random permutation π_1 on the nodes in S and a random permutation π_2 on the nodes in \bar{S} . Let π be the ranking obtained by concatenating π_1 and then π_2 .
 4. Return π .
-

4 Hardness for CSPs on Trees

Negative Results: As mentioned, previous work (Byrka et al., 2010; Jiang et al., 2001; Bryant, 1997; He et al., 2006; Steel, 1992) tried to get better approximations for triplets/quartets consistency compared to trivial baselines. Recall, that the trivial baseline is to simply output a random tree (either rooted or unrooted depending on the problem). In our paper, near optimal hardness of approximation results for the maximum desired/forbidden triplets/quartets consistency problems (4 problems in total) are presented shedding light to why, despite significant efforts from different communities, no improvement had been made for nearly thirty years. As a consequence, we get the first tight hardness for an ordering problem on trees, thus extending the work of Guruswami et al. (2011) from orderings on the line to hierarchical clustering.

Specifically, for maximizing forbidden triplets, we show that no polynomial time algorithm can achieve a constant better than $\frac{2}{3}$ -approximation. Similar to Guruswami et al. (2008, 2011) this is assuming the Unique Games Conjecture, however for maximizing desired triplets, we show a threshold of $\frac{2}{3}$, assuming $P \neq NP$. The above also implies that forbidden triplets is approximation resistant as a random tree also achieves a $\frac{2}{3}$ factor. In fact our hardness results for all 4 problems are stronger, as we show it's not possible to distinguish almost perfectly consistent inputs from inputs where the optimum solution achieves almost the same as a random solution.

Technically, in order to get the hardness results, we give algorithms to obtain permutations on the leaves of a tree, such that if the tree obeyed many triplet/quartet constraints, then the permutation would also obey a large fraction of them when viewed as appropriate ordering constraints. Specifically, we prove that under the UGC, it is hard to approximate the Forbidden Triplets Consistency problem

better than a factor of $\frac{2}{3}$, even in the unweighted case.

Fact 1. Let K be the total number of triplets constraints in an instance of BTW. For any $\epsilon > 0$, it is UGC-hard to distinguish between BTW instances of the following two cases:

YES: $val(\pi^*) \geq (1 - \epsilon)K$, i.e. the optimal permutation satisfies almost all constraints.

NO: $val(\pi^*) \leq (\frac{1}{3} + \epsilon)K$, i.e. the optimal permutation does not satisfy more than $1/3$ fraction.

Given the above fact from Guruswami et al. (2011), we prove our $\frac{2}{3}$ -inapproximability result for Forbidden Triplets:

Theorem 4. Let K be the total number of the triplet constraints in an instance of Forbidden Triplets Consistency. For any $\delta > 0$, it is UGC-hard to distinguish between the following two cases:

YES: $val(T^*) \geq (1 - \delta)K$, i.e. the optimal tree satisfies almost all the triplet constraints.

NO: $val(T^*) \leq (\frac{2}{3} + \delta)K$, i.e. the optimal tree does not satisfy more than $\frac{2}{3}$ fraction of triplets.

Proof. Start with a YES instance of the BTW problem with optimal permutation π^* and $val(\pi^*) \geq (1 - \epsilon)K$. Viewing each BTW constraint $a|b|c$ as a forbidden triplet $ac|b$, we show how to construct a tree T such that $val(T) \geq (1 - \delta(\epsilon))K$. In fact, the construction is straightforward: simply assign the n labels, in the order they appear in π^* , as the leaves of a caterpillar tree (every internal node has its left child being a leaf). Observe that this caterpillar tree satisfies: $val(T) \geq (1 - \epsilon)K$. This is because if a BTW constraint $a|b|c$ was obeyed by π^* , it will also be avoided (viewed as a forbidden triplet $ac|b$) by the caterpillar tree above: if a appears first in the permutation then the caterpillar will avoid $ac|b$ as a gets separated first, otherwise if c appears first, then again the caterpillar tree will avoid $ac|b$ as c gets separated first.

The NO instance is more challenging. Start with a NO instance of the BTW problem with optimal π^* of value $val(\pi^*) \leq (\frac{1}{3} + \epsilon)K$. Viewing the BTW constraints as forbidden triplets, we show that the optimum tree T^* cannot achieve better than $> (2/3 + 2\epsilon)K$, because this would imply that $val(\pi^*) > (\frac{1}{3} + \epsilon)K$, which is a contradiction. For this, assume that some tree T scored a value $val(T) > (2/3 + 2\epsilon)K$. We will construct a permutation π from the tree T with value $val(\pi) > (1/3 + \epsilon)K$, a contradiction. Notice that there are forbidden triplets that may be avoided by the tree, yet obeyed by the permutation: for example for a forbidden triplet $t = ac|b$, the tree R that first removes a and then splits b, c will successfully avoid t , however the permutation acb can come from R by projection, however acb does not obey the BTW constraint $a|b|c$. Hence directly projecting the leaves of T onto a line may not satisfy $> (1/3 + 2\epsilon)K$, since every

forbidden triplet $ac|b$ avoided by T , can be ordered by this projected permutation in a way that would not obey the corresponding BTW constraint $a|b|c$. However, just by randomly swapping each left and right child for every internal node in the tree before we do the projection to the permutation, would satisfy $1/2 \cdot (2/3 + 2\epsilon)K = (1/3 + \epsilon)K$ number of constraints. To see this, note that with probability $\frac{1}{2}$ a forbidden $ac|b$ avoided by T will be mapped to the desired abc (and not acb) or cba (and not cab) ordering.

Finally, we get $val(\pi^*) \geq val(\pi) > (1/3 + \epsilon)K$, a contradiction that we were given a NO instance. To conclude, $\frac{2}{3}$ -inapproximability follows from the gap of these two instances. \square

For the Desired Triplets problem, the proof proceeds in a similar fashion. One main difference is that we prove hardness of $\frac{2}{3}$ under $P \neq NP$, without assuming UGC. The reason is that we reduce from the NON-BTW problem that is known to be approximation resistant, subject only to $P \neq NP$. Of course, one open question is to close the gap between this $\frac{2}{3}$ factor and the current best approximation of $\frac{1}{3}$.

Theorem 5. *Let K be the total number of the triplet constraints in an instance of Desired Triplets Consistency. For any $\delta > 0$, it is NP-hard to distinguish:*

YES: $val(T^*) \geq (\frac{1}{2} - \delta)K$

NO: $val(T^*) \leq (\frac{1}{3} + \delta)K$

Switching to quartet problems, our reductions are more challenging. The first challenge is that constraints are on 4 items so we need to resort to an ordering CSP of arity 4, that we term 4-SEPARATEDNESS. Next, trees are unrooted and we want to generate an ordering on their leaves. To do this we first root the tree at some internal node and then follow a similar strategy for randomly reordering their children. For desired quartets we show hardness of $\frac{2}{3}$ and for forbidden quartets a hardness of $\frac{8}{9}$ (see App. A for statements). Recall that the best approximations are $\frac{1}{3}$ and $\frac{2}{3}$ respectively, achieved by a random (unrooted) tree.

Remark 5. *Note that our hardness results give optimal results when restricted to (rooted or unrooted) caterpillar trees, an important tree family, where each internal node has at least one leaf as a child.*

5 Conclusion

We studied ranking, correlation clustering and hierarchical clustering under qualitative constraints and we presented a simple algorithm based on MAXCUT that is able to overcome known hardness results under a random model. We also provided the first tight hardness of approximation for CSPs on trees shedding light to basic problems in computational biology and extending previous results by Guruswami et al. (2011)

from ordering CSPs to trees. We believe that a nice open question is to prove that the two most important families of CSPs on trees (triplets and quartets consistency) are approximation resistant. Here we showed this for the case of forbidden triplets. More generally, it is conceivable that all non-trivial CSPs on trees are in fact approximation resistant, implying that the inapproximability results of Guruswami et al. (2011) can be extended from linear orderings to trees.

References

- E. Abbe, A. S. Bandeira, and G. Hall. Exact recovery in the stochastic block model. *IEEE Transactions on Information Theory*, 62(1):471–487, 2015.
- S. Ahmadian, V. Chatziafratis, A. Epasto, E. Lee, M. Mahdian, K. Makarychev, and G. Yaroslavtsev. Bisect and conquer: Hierarchical clustering via max-uncut bisection. *The 23rd International Conference on Artificial Intelligence and Statistics*, 2020.
- A. V. Aho, Y. Sagiv, T. G. Szymanski, and J. D. Ullman. Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions. *SIAM Journal on Computing*, 10(3):405–421, 1981.
- N. Ailon. Aggregation of partial rankings, p-ratings and top-m lists. *Algorithmica*, 57(2):284–300, 2010.
- N. Ailon, M. Charikar, and A. Newman. Aggregating inconsistent information: ranking and clustering. *Journal of the ACM (JACM)*, 55(5):1–27, 2008.
- P. Austrin and E. Mossel. Approximation resistant predicates from pairwise independence. *Computational Complexity*, 18(2):249–271, 2009.
- P. Austrin, R. Manokaran, and C. Wenner. On the NP-hardness of approximating ordering constraint satisfaction problems. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 26–41. Springer, 2013.
- N. Bansal, A. Blum, and S. Chawla. Correlation clustering. *Machine Learning*, 56(1-3):89–113, 2004.
- Z. Bar-Joseph, D. K. Gifford, and T. S. Jaakkola. Fast optimal leaf ordering for hierarchical clustering. *Bioinformatics*, 17(suppl_1):S22–S29, 2001.
- M. Bodirsky and J. K. Mueller. The complexity of rooted phylogeny problems. In *Proceedings of the 13th International Conference on Database Theory*, pages 165–173, 2010.
- M. Bodirsky, P. Jonsson, and T. Van Pham. The complexity of phylogeny constraint satisfaction. In *33rd Symposium on Theoretical Aspects of Computer Science*, 2016.
- M. Braverman and E. Mossel. Sorting from noisy information. *arXiv preprint arXiv:0910.1191*, 2009.

- D. Bryant. Building trees, hunting for trees, and comparing trees: theory and methods in phylogenetic analysis. *PhD Thesis*, 1997.
- J. Byrka, S. Guillelot, and J. Jansson. New results on optimizing rooted triplets consistency. *Discrete Applied Mathematics*, 158(11):1136–1147, 2010.
- M. Charikar and V. Chatziafratis. Approximate hierarchical clustering via sparsest cut and spreading metrics. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 841–854. SIAM, 2017.
- M. Charikar, V. Guruswami, and A. Wirth. Clustering with qualitative information. *Journal of Computer and System Sciences*, 71(3):360–383, 2005.
- M. Charikar, V. Guruswami, and R. Manokaran. Every permutation csp of arity 3 is approximation resistant. In *2009 24th Annual IEEE Conference on Computational Complexity*, pages 62–73. IEEE, 2009.
- M. Charikar, V. Chatziafratis, and R. Niazadeh. Hierarchical clustering better than average-linkage. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2291–2304. SIAM, 2019a.
- M. Charikar, V. Chatziafratis, R. Niazadeh, and G. Yaroslavtsev. Hierarchical clustering for euclidean data. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2721–2730, 2019b.
- V. Chatziafratis, R. Niazadeh, and M. Charikar. Hierarchical clustering with structural constraints. In *International Conference on Machine Learning*, pages 774–783, 2018.
- B. Chor and M. Sudan. A geometric approach to betweenness. *SIAM Journal on Discrete Mathematics*, 11(4):511–523, 1998.
- V. Cohen-Addad, V. Kanade, F. Mallmann-Trenn, and C. Mathieu. Hierarchical clustering: Objective functions and algorithms. *Journal of the ACM (JACM)*, 66(4):1–42, 2019.
- S. Dasgupta. *A Cost Function for Similarity-Based Hierarchical Clustering*, page 118–127. Association for Computing Machinery, New York, NY, USA, 2016.
- I. Davidson and S. Basu. A survey of clustering with instance level constraints. *ACM Transactions on Knowledge Discovery from data*, 1(1-41):2–42, 2007.
- E. Emamjomeh-Zadeh and D. Kempe. Adaptive hierarchical clustering using ordinal queries. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 415–429. SIAM, 2018.
- R. Fagin, R. Kumar, M. Mahdian, D. Sivakumar, and E. Vee. Comparing partial rankings. *SIAM Journal on Discrete Mathematics*, 20(3):628–648, 2006.
- M. Falahatgar, A. Orlitsky, V. Pichapati, and A. T. Suresh. Maximum selection and ranking under noisy comparisons. In *International Conference on Machine Learning*, pages 1088–1096. PMLR, 2017.
- 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. IEEE, 1995.
- J. Felsenstein. *Inferring phylogenies*, volume 2. Sinauer associates Sunderland, MA, 2004.
- D. Ghoshdastidar, M. Perrot, and U. von Luxburg. Foundations of comparison-based hierarchical clustering. In *Advances in Neural Information Processing Systems*, pages 7454–7464, 2019.
- M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145, 1995.
- V. Guruswami, R. Manokaran, and P. Raghavendra. Beating the random ordering is hard: Inapproximability of maximum acyclic subgraph. In *2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 573–582. IEEE, 2008.
- V. Guruswami, J. Håstad, R. Manokaran, P. Raghavendra, and M. Charikar. Beating the random ordering is hard: Every ordering csp is approximation resistant. *SIAM Journal on Computing*, 40(3):878–914, 2011.
- J. Håstad. Some optimal inapproximability results. *Journal of the ACM (JACM)*, 48(4):798–859, 2001.
- Y.-J. He, T. N. Huynh, J. Jansson, and W.-K. Sung. Inferring phylogenetic relationships avoiding forbidden rooted triplets. *Journal of Bioinformatics and Computational Biology*, 4(01):59–74, 2006.
- T. Jiang, P. Kearney, and M. Li. A polynomial time approximation scheme for inferring evolutionary trees from quartet topologies and its application. *SIAM Journal on Computing*, 30(6):1942–1961, 2001.
- T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, page 133–142, New York, NY, USA, 2002. Association for Computing Machinery.
- R. M. Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

- C. Kenyon-Mathieu and W. Schudy. How to rank with few errors. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 95–103, 2007.
- S. Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the thirty-fourth annual ACM symposium on Theory of computing*, pages 767–775. ACM, 2002.
- M. Kleindessner and U. von Luxburg. Kernel functions based on triplet comparisons. In *Advances in Neural Information Processing Systems*, pages 6807–6817, 2017.
- J. Leskovec, A. Rajaraman, and J. D. Ullman. *Mining of massive datasets*. Cambridge university press, 2014.
- V. Lyzinski, M. Tang, A. Athreya, Y. Park, and C. E. Priebe. Community detection and classification in hierarchical stochastic blockmodels. *IEEE Transactions on Network Science and Engineering*, 4(1):13–26, 2016.
- Y. Makarychev. Simple linear time approximation algorithm for betweenness. *Operations research letters*, 40(6):450–452, 2012.
- B. Moseley and J. Wang. Approximation bounds for hierarchical clustering: Average linkage, bisecting k-means, and local search. In *Advances in Neural Information Processing Systems*, pages 3094–3103, 2017.
- E. Mossel, J. Neeman, and A. Sly. Stochastic block models and reconstruction. *arXiv preprint arXiv:1202.1499*, 2012.
- S. Negahban, S. Oh, and D. Shah. Iterative ranking from pair-wise comparisons. In *Advances in neural information processing systems*, pages 2474–2482, 2012.
- J. Opatrny. Total ordering problem. *SIAM Journal on Computing*, 8(1):111–114, 1979.
- E. Scornet, G. Biau, J.-P. Vert, et al. Consistency of random forests. *The Annals of Statistics*, 43(4):1716–1741, 2015.
- N. Shah, S. Balakrishnan, A. Guntuboyina, and M. Wainwright. Stochastically transitive models for pairwise comparisons: Statistical and computational issues. In *International Conference on Machine Learning*, pages 11–20, 2016.
- N. B. Shah and M. J. Wainwright. Simple, robust and optimal ranking from pairwise comparisons. *The Journal of Machine Learning Research*, 18(1):7246–7283, 2017.
- D. Slonim, L. Kruglyak, L. Stein, and E. Lander. Building human genome maps with radiation hybrids. *Journal of Computational Biology*, 4(4):487–504, 1997.
- S. Snir and S. Rao. Using max cut to enhance rooted trees consistency. *IEEE/ACM transactions on computational biology and bioinformatics*, 3(4):323–333, 2006.
- S. Snir and S. Rao. Quartets maxcut: a divide and conquer quartets algorithm. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 7(4):704–718, 2008.
- S. Snir and S. Rao. Quartet maxcut: a fast algorithm for amalgamating quartet trees. *Molecular phylogenetics and evolution*, 62(1):1–8, 2012.
- S. Snir and R. Yuster. Reconstructing approximate phylogenetic trees from quartet samples. *SIAM Journal on Computing*, 41(6):1466–1480, 2012.
- T. Sørlie, C. M. Perou, R. Tibshirani, T. Aas, S. Geisler, H. Johnsen, T. Hastie, M. B. Eisen, M. Van De Rijn, S. S. Jeffrey, et al. Gene expression patterns of breast carcinomas distinguish tumor subclasses with clinical implications. *Proceedings of the National Academy of Sciences*, 98(19):10869–10874, 2001.
- M. Steel. The complexity of reconstructing trees from qualitative characters and subtrees. *Journal of classification*, 9(1):91–116, 1992.
- M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In *KDD workshop on text mining*, volume 400, pages 525–526. Boston, 2000.
- C. Swamy. Correlation clustering: maximizing agreements via semidefinite programming. In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 526–527. Society for Industrial and Applied Mathematics, 2004.
- L. L. Thurstone. *The Measurement of Values*. The University of Chicago Press, 1959.
- M. Tumminello, F. Lillo, and R. N. Mantegna. Correlation, hierarchies, and networks in financial markets. *Journal of economic behavior & organization*, 75(1):40–58, 2010.
- J. W. Vaughan. Making better use of the crowd: How crowdsourcing can advance machine learning research. *The Journal of Machine Learning Research*, 18(1):7026–7071, 2017.
- S. Vikram and S. Dasgupta. Interactive bayesian hierarchical clustering. In *International Conference on Machine Learning*, pages 2081–2090, 2016.
- R. K. Vinayak and B. Hassibi. Crowdsourced clustering: Querying edges vs triangles. In *Advances in Neural Information Processing Systems*, pages 1316–1324, 2016.
- K. Wagstaff and C. Cardie. Clustering with instance-level constraints. *AAAI/IAAI*, 1097:577–584, 2000.

K. Wagstaff, C. Cardie, S. Rogers, and S. Schrödl.
Constrained k-means clustering with background
knowledge. In *ICML*, volume 1, pages 577–584,
2001.

A Omitted Proofs - Improved Approximations via MaxCut

In this first section of the Appendix, we present the omitted details for our positive results. Specifically, we show how to overcome impossibility results (see also Appendix B) by going beyond the hardness of approximation thresholds ρ for each of the problems considered in the paper. As noted, to escape the worst-case analysis, we will assume the input is given as a set of m noisy constraints generated according to our stochastic model and the goal is to obtain a solution with strictly more than ρm satisfied constraints.

Recall that in Table 1, only for the results on Correlation Clustering and on Triplets Consistency marked with an asterisk (*), we required a mild balancedness assumption. The assumption here on the balancedness of the ground truth partition or ground truth hierarchical clustering is used in our reduction, and specifically when analyzing our MAXCUT approach. It is needed in order to ensure that based on our stochastic model, our MAXCUT approach can find a large cut in the constructed graph which later translates into a large portion of satisfied constraints.

Assumption 1. *For a tree with n leaves, a split (L, R) at an internal node is called balanced if $|L| = cn, |R| = (1 - c)n$ with $\frac{1}{3} \leq c \leq \frac{2}{3}$. We assume that in the optimum tree there exists one split that is balanced. Similarly, for a clustering on n nodes, if there exists a partition of the clusters into two sides (L, R) such that $|L| = cn, |R| = (1 - c)n$ with $\frac{1}{3} \leq c \leq \frac{2}{3}$, we say the clustering is balanced.*

This is a reasonable assumption since hierarchical clusterings tend to be balanced and indeed recursive balanced cuts tend to recover good hierarchies (Charikar and Chatziafratis, 2017). In essence, we exclude caterpillar trees or more generally highly skewed trees that are generated by always removing tiny pieces out of a giant component. Moreover, such assumptions are common in generative graph models for clustering, e.g., the Stochastic Block Model (Mossel et al., 2012; Abbe et al., 2015) and for hierarchical clustering, e.g., the Hierarchical Stochastic Block Model (Lyzinski et al., 2016; Cohen-Addad et al., 2019; Ghoshdastidar et al., 2019), where we expect to see at least two large communities emerge. For example, recent generative models like the Hierarchical Stochastic Block Model in Ghoshdastidar et al. (2019) satisfy the balancedness assumption with $c = \frac{1}{2}$.

A.1 Quartets Consistency from Noisy constraints

Let $\mathcal{Q}_{\mathcal{F}}, \mathcal{Q}_{\mathcal{D}}$ be the set of forbidden and desired quartet constraints with sizes $|\mathcal{Q}_{\mathcal{F}}| = m_1, |\mathcal{Q}_{\mathcal{D}}| = m_2$

respectively. The total number of generated constraints according to our stochastic model is denoted by $m = m_1 + m_2$. Out of those constraints, let ϵ_1, ϵ_2 denote the fraction of the erroneous forbidden and erroneous desired quartet constraints respectively. Our main theorem here is:

Theorem 6. *Given $m = m_1 + m_2$ constraints as above on n items, our algorithm MAXCUT satisfies at least $(0.425 - 0.261\epsilon_1)m_1 + (0.672 - 0.296\epsilon_2)m_2$ on average, where ϵ_1, ϵ_2 are as above. If moreover $m_1, m_2 \geq \Omega(\log n)$, the result holds w.h.p.*

For example, if the constraints are not erroneous (i.e., $\epsilon_1 = \epsilon_2 = 0$), we satisfy 42.5% of the desired quartets, while avoiding 67.2% of the forbidden quartets, improving upon prior best approximations.

In order to prove Theorem 6, we will require several intermediate lemmas and constructions.

Recall that forbidden quartets should be avoided, whereas desired quartets should be satisfied by the tree our algorithm finds. We use the following notation: Let $\mathcal{Q}_{\mathcal{A}}(\text{ALG})$ denote the number of quartets $q \in \mathcal{Q}_{\mathcal{F}}$ avoided, $\mathcal{Q}_{\mathcal{F}}(\text{ALG})$ the number of quartets $q \in \mathcal{Q}_{\mathcal{F}}$ not avoided (of course, $\mathcal{Q}_{\mathcal{F}} = m_2 = \mathcal{Q}_{\mathcal{A}}(\text{ALG}) + \mathcal{Q}_{\mathcal{F}}(\text{ALG})$) and $\mathcal{Q}_{\mathcal{D}}(\text{ALG})$ the number of quartets $q \in \mathcal{Q}_{\mathcal{D}}$ satisfied by the output phylogenetic tree. For the case of no errors $\epsilon_1 = 0, \epsilon_2 = 0$, the best approximation under worst-case analysis is:

$$\mathcal{Q}_{\mathcal{D}}(\text{ALG}) - \mathcal{Q}_{\mathcal{F}}(\text{ALG}) \geq \frac{1}{3}(|\mathcal{Q}_{\mathcal{D}}| - |\mathcal{Q}_{\mathcal{F}}|) \iff$$

$$\mathcal{Q}_{\mathcal{D}}(\text{ALG}) + \mathcal{Q}_{\mathcal{A}}(\text{ALG}) \geq \frac{1}{3}m_1 + \frac{2}{3}m_2$$

In fact, the guarantees hold separately $\mathcal{Q}_{\mathcal{D}}(\text{ALG}) \geq \frac{1}{3}|\mathcal{Q}_{\mathcal{D}}|$ and $\mathcal{Q}_{\mathcal{A}}(\text{ALG}) \geq \frac{2}{3}|\mathcal{Q}_{\mathcal{F}}|$ and are achieved either by a simple greedy algorithm or by a random tree (He et al., 2006). Our goal is to find a tree beating the above guarantees, i.e., satisfying strictly more than $\frac{1}{3}$ fraction of desired quartets and strictly more than $\frac{2}{3}$ fraction of forbidden quartets. Our approach is based on extending a previous analysis from Snir and Rao (2006) that only handled the case with $\mathcal{Q}_{\mathcal{F}} = \emptyset$.

The end result of our algorithm ALG, which is based on MAXCUT, is a tree with the following guarantees:

$$\mathcal{Q}_{\mathcal{D}}(\text{ALG}) + \mathcal{Q}_{\mathcal{A}}(\text{ALG}) \geq$$

$$\geq (0.425 - 0.261\epsilon_1)m_1 + (0.672 - 0.296\epsilon_2)m_2 \quad (4)$$

We start by instantiating our general algorithmic template in Algorithm 1 to the case of the Quartets Consistency problem, and we describe the necessary changes for the appropriate graph construction below:

Graph Construction from constraints: The goal here is to construct a graph encoding the qualitative information from the generated quartets so that a MAXCUT subroutine can yield a reasonable first split of the output phylogenetic tree. Quartets $q \in \mathcal{Q}_{\mathcal{F}}$ needs to be handled differently from quartets $q \in \mathcal{Q}_{\mathcal{D}}$. For each forbidden $q = \{ab|cd\} \in \mathcal{Q}_{\mathcal{F}}$ we add the following six + or - weighted edges:

$$\begin{aligned} &+2 \text{ for pairs } (a, b), (c, d) \text{ and} \\ &-1 \text{ for pairs } (a, c), (a, d), (b, c), (b, d) \end{aligned}$$

and for a $q = \{ab|cd\} \in \mathcal{Q}_{\mathcal{D}}$ we add the following + or - edges:

$$\begin{aligned} &-2 \text{ for pairs } (a, b), (c, d) \text{ and} \\ &+1 \text{ for pairs } (a, c), (a, d), (b, c), (b, d) \end{aligned}$$

Let G be the undirected weighed multigraph constructed from the constraints as above and let (S, \bar{S}) denote any graph cut into two parts. We say that a quartet $q = \{ab|cd\} \in \mathcal{Q}_{\mathcal{F}} \cup \mathcal{Q}_{\mathcal{D}}$ is *unaffected* by the cut (S, \bar{S}) if all four labels a, b, c, d end up in one of the two parts. For quartets whose endpoints are separated by the cut, we distinguish 3 cases: if one of the labels goes to one of the two parts while the remaining 3 labels go to the other part, we say that q is *postponed*. If precisely a, b are contained in some part, while the other part contains precisely c, d , we say q is *obeyed*. In any other case, q is *disobeyed* (e.g., $a, c \in S$ and $b, d \in \bar{S}$ or the symmetric split $a, d \in S$ and $b, c \in \bar{S}$). The perhaps more natural terms *satisfied* and *violated* were not used as we deal both with desired and forbidden quartets and would be misleading when accounting for the maximization objective:

Lemma 1. *The weight of any cut (S, \bar{S}) can be computed based on the status of the quartets as:*

$$\begin{aligned} w(S, \bar{S}) &= 2m_d^{\mathcal{Q}_{\mathcal{F}}}(S, \bar{S}) - 4m_o^{\mathcal{Q}_{\mathcal{F}}}(S, \bar{S}) + \\ &+ 4m_o^{\mathcal{Q}_{\mathcal{D}}}(S, \bar{S}) - 2m_d^{\mathcal{Q}_{\mathcal{D}}}(S, \bar{S}) \end{aligned} \quad (5)$$

where $m_d^{\mathcal{Q}_{\mathcal{F}}}, m_d^{\mathcal{Q}_{\mathcal{D}}}$ is the number of disobeyed quartets by the cut that belong to $\mathcal{Q}_{\mathcal{F}}, \mathcal{Q}_{\mathcal{D}}$ respectively and similarly $m_o^{\mathcal{Q}_{\mathcal{F}}}, m_o^{\mathcal{Q}_{\mathcal{D}}}$ is the number of obeyed quartets from $\mathcal{Q}_{\mathcal{F}}, \mathcal{Q}_{\mathcal{D}}$ respectively.

Proof. Note that by our choice for the edge weights, if $q = \{ab|cd\}$ is postponed or unaffected by the cut (S, \bar{S}) , its contribution to $w(S, \bar{S})$ is 0 regardless of $q \in \mathcal{Q}_{\mathcal{F}}$ or $q \in \mathcal{Q}_{\mathcal{D}}$. Now, if a forbidden $q \in \mathcal{Q}_{\mathcal{F}}$ is obeyed, that counts as a mistake and it decreases the weight of the cut by -4, whereas if it is disobeyed, that counts as a correct choice and it increases the weight of the cut by +2. Accordingly we compute the contribution for the desired quartets $q \in \mathcal{Q}_{\mathcal{D}}$ as +4 if obeyed and -2 if disobeyed. Summing over all constraints gives us the lemma. \square

The final step is to compute the overall quartets our algorithm had success on, relative to the sample sizes m_1, m_2 :

Lemma 2. *If (S, \bar{S}) is the first split of ALG, the total number of quartets decomposed correctly is:*

$$ALG = \mathcal{Q}_{\mathcal{A}}(ALG) + \mathcal{Q}_{\mathcal{D}}(ALG) \geq \frac{2}{3}m_1 + \frac{1}{3}m_2 + \frac{1}{6}w(S, \bar{S})$$

Proof. Let $m_p^{\mathcal{Q}_{\mathcal{F}}}, m_u^{\mathcal{Q}_{\mathcal{F}}}$ denote the number of postponed or unaffected by the cut forbidden quartets, and $m_p^{\mathcal{Q}_{\mathcal{D}}}, m_u^{\mathcal{Q}_{\mathcal{D}}}$ denote the number of postponed or unaffected by the cut desired quartets. Our algorithm first uses an approximation to MAXCUT and then proceeds greedily (or randomly) to achieve the baseline guarantees by building a tree on S and on \bar{S} :

$$\begin{aligned} ALG &\geq m_d^{\mathcal{Q}_{\mathcal{F}}}(S, \bar{S}) + \frac{2}{3}(m_u^{\mathcal{Q}_{\mathcal{F}}}(S, \bar{S}) + m_p^{\mathcal{Q}_{\mathcal{F}}}(S, \bar{S})) + \\ &+ m_o^{\mathcal{Q}_{\mathcal{D}}}(S, \bar{S}) + \frac{1}{3}(m_u^{\mathcal{Q}_{\mathcal{D}}}(S, \bar{S}) + m_p^{\mathcal{Q}_{\mathcal{D}}}(S, \bar{S})) \end{aligned}$$

For notation purposes, from now on we drop the parentheses (S, \bar{S}) from the terms since we always refer to the (S, \bar{S}) cut. Observe that $m_1 = m_d^{\mathcal{Q}_{\mathcal{F}}} + m_o^{\mathcal{Q}_{\mathcal{F}}} + m_u^{\mathcal{Q}_{\mathcal{F}}} + m_p^{\mathcal{Q}_{\mathcal{F}}}$ and similarly $m_2 = m_d^{\mathcal{Q}_{\mathcal{D}}} + m_o^{\mathcal{Q}_{\mathcal{D}}} + m_u^{\mathcal{Q}_{\mathcal{D}}} + m_p^{\mathcal{Q}_{\mathcal{D}}}$. By substituting the terms for unaffected and postponed quartets we get:

$$\begin{aligned} ALG &\geq \\ m_d^{\mathcal{Q}_{\mathcal{F}}} &+ \frac{2}{3}(m_1 - m_d^{\mathcal{Q}_{\mathcal{F}}} - m_o^{\mathcal{Q}_{\mathcal{F}}}) + m_o^{\mathcal{Q}_{\mathcal{D}}} + \frac{1}{3}(m_2 - m_d^{\mathcal{Q}_{\mathcal{D}}} - m_o^{\mathcal{Q}_{\mathcal{D}}}) \\ &= \frac{2}{3}m_1 + \frac{1}{3}m_d^{\mathcal{Q}_{\mathcal{F}}} - \frac{2}{3}m_o^{\mathcal{Q}_{\mathcal{F}}} + \frac{1}{3}m_2 + \frac{2}{3}m_o^{\mathcal{Q}_{\mathcal{D}}} - \frac{1}{3}m_d^{\mathcal{Q}_{\mathcal{D}}} \\ &= \frac{2}{3}m_1 + \frac{1}{3}m_2 + \frac{1}{6}(2m_d^{\mathcal{Q}_{\mathcal{F}}} - 4m_o^{\mathcal{Q}_{\mathcal{F}}} + 4m_o^{\mathcal{Q}_{\mathcal{D}}} - 2m_d^{\mathcal{Q}_{\mathcal{D}}}) \end{aligned}$$

From equation (5), the last term is equal to the weight of the (S, \bar{S}) cut and this finishes the proof. \square

Now we need to show that there is a good cut with high weight in the graph. Recall that the graph has positive and negative edges. For such graphs, the guarantee of the rounding algorithm of Goemans and Williamson (1995) is as follows:

Fact 2. *For graphs with both positive and negative weights, one can efficiently find a cut (S, \bar{S}) with weight:*

$$w(S, \bar{S}) \geq 0.878w(S^*, \bar{S}^*) - 0.122W^-$$

where (S^*, \bar{S}^*) is the optimum solution for MAX-CUT and W^- is the absolute sum of all negative edge weights.

The cut (S, \bar{S}) is produced in the same manner as in the standard Goemans-Williamson algorithm via random hyperplane rounding on their semidefinite relaxation for MAXCUT. We will use this fact to prove the following:

Lemma 3. *The weight of the top split relative to the sizes of the quartet constraints is:*

$$w(S, \bar{S}) \geq (0.03229 - 1.56\epsilon_1)m_1 + (0.5525 - 1.56\epsilon_2)m_2$$

Proof. Observe that in the constructed graph, the total negative weight is $W^- = 4m_1 + 4m_2$ as each quartet adds a total negative weight of -4. In order to use Fact 2, we require a lower bound on the optimum value $w(S^*, \bar{S}^*)$.

Notice that for any phylogenetic tree, since all internal vertices have three neighbors each (a trivalent tree), we can always find an edge that induces a *balanced* cut. For n leaves, a cut (L, R) is called balanced if $|L| = cn$, $|R| = (1 - c)n$ with $\frac{1}{3} \leq c \leq \frac{2}{3}$. From our uniform generating model, recall that the number of quartet constraints the cut (L, R) succeeds at is:

$$\mathbb{E}(m_d^{\mathcal{Q}_{\mathcal{F}}}) = 6c^2(1 - c)^2(1 - \epsilon_1)m_1$$

$$\mathbb{E}(m_o^{\mathcal{Q}_{\mathcal{D}}}) = 6c^2(1 - c)^2(1 - \epsilon_2)m_2$$

and the number of constraints the cut (L, R) fails at, due to the erroneous constraints is:

$$\mathbb{E}(m_o^{\mathcal{Q}_{\mathcal{F}}}) = 6c^2(1 - c)^2\epsilon_1m_1, \quad \mathbb{E}(m_d^{\mathcal{Q}_{\mathcal{D}}}) = 6c^2(1 - c)^2\epsilon_2m_2$$

The quantity $c^2(1 - c)^2$ with $\frac{1}{3} \leq c \leq \frac{2}{3}$ attains a minimum value of $\frac{4}{81}$ when $c = \frac{1}{3}$; hence, from Lemma 1, the weight of the cut (L, R) on the constructed graph is:

$$\begin{aligned} w(L, R) &\geq 2m_d^{\mathcal{Q}_{\mathcal{F}}}(L, R) - 4m_o^{\mathcal{Q}_{\mathcal{F}}}(L, R) + \\ &\quad + 4m_o^{\mathcal{Q}_{\mathcal{D}}}(L, R) - 2m_d^{\mathcal{Q}_{\mathcal{D}}}(L, R) \\ &\geq \frac{16}{27}(1 - \epsilon_1)m_1 - \frac{32}{27}\epsilon_1m_1 + \frac{32}{27}(1 - \epsilon_2)m_2 - \frac{16}{27}\epsilon_2m_2 \quad (6) \end{aligned}$$

Of course the optimum cut has even larger weight than the specific balanced (L, R) cut so: $w(S^*, \bar{S}^*) \geq w(L, R)$. Substituting equation (6) in Fact 2 yields the lemma. \square

Proof of Theorem 6. From Lemma 2, we have a lower bound on our algorithm's performance via the approximate max cut. Substituting the quantity $w(S, \bar{S})$ based on Lemma 3, yields the theorem. \square

From the above, notice that we can still beat the prior best baselines as long as the error rates are not too big ($\epsilon_1 \leq 3.4\%$ and $\epsilon_2 \leq 35.4\%$).

A.2 Triplets Consistency from Noisy constraints

Here we show a similar approximation result but for Triplets. Let $\mathcal{T}_{\mathcal{F}}$, $\mathcal{T}_{\mathcal{D}}$ be the set of forbidden and desired triplet constraints with sizes $|\mathcal{T}_{\mathcal{F}}| = m_1$, $|\mathcal{T}_{\mathcal{D}}| = m_2$ respectively. The total number of generated constraints

is denoted by $m = m_1 + m_2$. Out of those constraints, let ϵ_1, ϵ_2 denote the fraction of the erroneous forbidden and erroneous desired triplet constraints respectively.

Theorem 7. *Given $m = m_1 + m_2$ constraints as above on n items, our algorithm MAXCUT satisfies at least $(\frac{2}{3} + 0.11378 - 0.5853\epsilon_1)m_1 + (\frac{1}{3} + 0.30886 - 0.5853\epsilon_2)m_2$ on average, where ϵ_1, ϵ_2 are as above. If moreover $m_1, m_2 \geq \Omega(\log n)$, the result holds w.h.p.*

For example, if the constraints are not erroneous (i.e., $\epsilon_1 = \epsilon_2 = 0$), we satisfy 64% of the desired triplets, while avoiding 78% of the forbidden triplets. This latter ratio beats our worst-case inapproximability results for triplets (see also Appendix B).

The reason we stated the numerical values in this form is that the trivial baselines achieve ratios of $\frac{2}{3}$ and $\frac{1}{3}$ for m_1 and m_2 respectively.

Recall that forbidden triplets should be avoided, whereas desired triplets should be satisfied by the tree our algorithm finds. We use the following notation: Let $\mathcal{T}_{\mathcal{A}}(\text{ALG})$ denote the number of triplets $t \in \mathcal{T}_{\mathcal{F}}$ avoided, $\mathcal{T}_{\mathcal{F}}(\text{ALG})$ the number of triplets $t \in \mathcal{Q}_{\mathcal{F}}$ not avoided (of course, $\mathcal{T}_{\mathcal{F}} = m_2 = \mathcal{T}_{\mathcal{A}}(\text{ALG}) + \mathcal{T}_{\mathcal{F}}(\text{ALG})$) and $\mathcal{T}_{\mathcal{D}}(\text{ALG})$ the number of triplets $t \in \mathcal{T}_{\mathcal{D}}$ satisfied by the output rooted binary hierarchical tree. For the case of no errors $\epsilon_1 = 0, \epsilon_2 = 0$, the best approximation under worst-case analysis is:

$$\begin{aligned} \mathcal{T}_{\mathcal{D}}(\text{ALG}) - \mathcal{T}_{\mathcal{F}}(\text{ALG}) &\geq \frac{1}{3}(|\mathcal{T}_{\mathcal{D}}| - |\mathcal{T}_{\mathcal{F}}|) \iff \\ \mathcal{T}_{\mathcal{D}}(\text{ALG}) + \mathcal{T}_{\mathcal{A}}(\text{ALG}) &\geq \frac{1}{3}m_1 + \frac{2}{3}m_2 \end{aligned}$$

In fact, the guarantees hold separately $\mathcal{T}_{\mathcal{D}}(\text{ALG}) \geq \frac{1}{3}|\mathcal{T}_{\mathcal{D}}|$ and $\mathcal{T}_{\mathcal{A}}(\text{ALG}) \geq \frac{2}{3}|\mathcal{T}_{\mathcal{F}}|$ and are achieved either by a simple greedy algorithm or by a random tree (He et al., 2006). Our goal is to find a tree beating the above guarantees, i.e., satisfying strictly more than $\frac{1}{3}$ fraction of desired triplets and strictly more than $\frac{2}{3}$ fraction of forbidden triplets.

The end result of our algorithm ALG, which is based on MAXCUT, is a tree with the following guarantees:

$$\begin{aligned} \mathcal{T}_{\mathcal{D}}(\text{ALG}) + \mathcal{T}_{\mathcal{A}}(\text{ALG}) &\geq \\ (\frac{2}{3} + 0.11378 - 0.5853\epsilon_1)m_1 + (\frac{1}{3} + 0.30886 - 0.5853\epsilon_2)m_2 &\quad (7) \end{aligned}$$

We proceed by describing the necessary changes to be made in our algorithmic template in Algorithm 1, in order to handle the triplet constraints.

Graph Construction from constraints: The goal here is to construct a graph encoding the qualitative information from the generated triplets so that a MAXCUT subroutine can yield a reasonable first split of the output binary hierarchical tree. Triplets $t \in \mathcal{T}_{\mathcal{F}}$ need to be handled differently from triplets $t \in \mathcal{T}_{\mathcal{D}}$. For

each forbidden $t = \{ab|c\} \in \mathcal{T}_{\mathcal{F}}$ we add the following 3 + or - undirected weighted edges:

+2 for the pair (a, b) and -1 for pairs $(c, a), (c, b)$

and for a $t = \{ab|c\} \in \mathcal{T}_{\mathcal{D}}$ we add the following + or - edges:

-2 for the pair (a, b) and +1 for pairs $(c, a), (c, b)$

Let G be the undirected weighed multigraph constructed from the constraints as above and let (S, \bar{S}) denote any graph cut into two parts. We say that a triplet $t = \{ab|c\} \in \mathcal{T}_{\mathcal{F}} \cup \mathcal{T}_{\mathcal{D}}$ is *unaffected* by the cut (S, \bar{S}) if all three labels a, b, c end up in one of the two parts. For triplets whose endpoints are separated by the cut, we distinguish 2 cases: if precisely a, b are contained in some part, while the other part contains precisely c , we say t is *obeyed*. In any other case, t is *disobeyed* (e.g., $a, c \in S$ and $b \in \bar{S}$ or the symmetric split $b, c \in S$ and $a \in \bar{S}$). The perhaps more natural terms *satisfied* and *violated* were not used as we deal both with desired and forbidden quartets and would be misleading when accounting for the maximization objective:

Lemma 4. *The weight of any cut (S, \bar{S}) can be computed based on the status of the triplets as:*

$$w(S, \bar{S}) =$$

$$m_d^{\mathcal{T}_{\mathcal{F}}}(S, \bar{S}) - 2m_o^{\mathcal{T}_{\mathcal{F}}}(S, \bar{S}) + 2m_o^{\mathcal{T}_{\mathcal{D}}}(S, \bar{S}) - m_d^{\mathcal{T}_{\mathcal{D}}}(S, \bar{S}) \quad (8)$$

where $m_d^{\mathcal{T}_{\mathcal{F}}}, m_d^{\mathcal{T}_{\mathcal{D}}}$ is the number of disobeyed triplets by the cut that belong to $\mathcal{T}_{\mathcal{F}}, \mathcal{T}_{\mathcal{D}}$ respectively and similarly $m_o^{\mathcal{T}_{\mathcal{F}}}, m_o^{\mathcal{T}_{\mathcal{D}}}$ is the number of obeyed triplets from $\mathcal{T}_{\mathcal{F}}, \mathcal{T}_{\mathcal{D}}$ respectively.

Proof. Note that by our choice for the edge weights, if $t = \{ab|c\}$ is unaffected by the cut (S, \bar{S}) , its contribution to $w(S, \bar{S})$ is 0 regardless of $t \in \mathcal{T}_{\mathcal{F}}$ or $t \in \mathcal{T}_{\mathcal{D}}$. Now, if a forbidden $t \in \mathcal{T}_{\mathcal{F}}$ is obeyed, that counts as a mistake and it decreases the weight of the cut by -2, whereas if it is disobeyed, that counts as a correct choice and it increases the weight of the cut by +1. Accordingly we compute the contribution for the desired triplets $t \in \mathcal{T}_{\mathcal{D}}$ as +2 if obeyed and -1 if disobeyed. Summing over all constraints gives us the lemma. \square

The final step is to compute the overall quartets our algorithm had success on, relative to the sample sizes m_1, m_2 :

Lemma 5. *If (S, \bar{S}) is the first split of ALG, the total number of triplets decomposed correctly is:*

$$ALG = \mathcal{T}_{\mathcal{A}}(ALG) + \mathcal{T}_{\mathcal{D}}(ALG) \geq \frac{2}{3}m_1 + \frac{1}{3}m_2 + \frac{1}{3}w(S, \bar{S})$$

Proof. Let $m_u^{\mathcal{T}_{\mathcal{F}}}$ denote the number of unaffected by the cut forbidden triplets, and $m_u^{\mathcal{T}_{\mathcal{D}}}$ denote the number of unaffected by the cut desired triplets. Our algorithm first uses an approximation to MAXCUT and then proceeds greedily (or randomly) to achieve the baseline guarantees by building a tree on S and on \bar{S} :

$$ALG \geq m_d^{\mathcal{T}_{\mathcal{F}}}(S, \bar{S}) + \frac{2}{3}m_u^{\mathcal{T}_{\mathcal{F}}}(S, \bar{S}) + m_o^{\mathcal{T}_{\mathcal{D}}}(S, \bar{S}) + \frac{1}{3}m_u^{\mathcal{T}_{\mathcal{D}}}(S, \bar{S})$$

For notation purposes, from now on we drop the parentheses (S, \bar{S}) from the terms since we always refer to the (S, \bar{S}) cut. Observe that $m_1 = m_d^{\mathcal{T}_{\mathcal{F}}} + m_o^{\mathcal{T}_{\mathcal{F}}} + m_u^{\mathcal{T}_{\mathcal{F}}}$ and similarly $m_2 = m_d^{\mathcal{T}_{\mathcal{D}}} + m_o^{\mathcal{T}_{\mathcal{D}}} + m_u^{\mathcal{T}_{\mathcal{D}}}$. By substituting the terms for the unaffected triplets we get:

$$ALG \geq$$

$$\begin{aligned} m_d^{\mathcal{T}_{\mathcal{F}}} + \frac{2}{3}(m_1 - m_d^{\mathcal{T}_{\mathcal{F}}} - m_o^{\mathcal{T}_{\mathcal{F}}}) + m_o^{\mathcal{T}_{\mathcal{D}}} + \frac{1}{3}(m_2 - m_d^{\mathcal{T}_{\mathcal{D}}} - m_o^{\mathcal{T}_{\mathcal{D}}}) \\ = \frac{2}{3}m_1 + \frac{1}{3}m_d^{\mathcal{T}_{\mathcal{F}}} - \frac{2}{3}m_o^{\mathcal{T}_{\mathcal{F}}} + \frac{1}{3}m_2 + \frac{2}{3}m_o^{\mathcal{T}_{\mathcal{D}}} - \frac{1}{3}m_d^{\mathcal{T}_{\mathcal{D}}} \\ = \frac{2}{3}m_1 + \frac{1}{3}m_2 + \frac{1}{3}(m_d^{\mathcal{T}_{\mathcal{F}}} - 2m_o^{\mathcal{T}_{\mathcal{F}}} + 2m_o^{\mathcal{T}_{\mathcal{D}}} - m_d^{\mathcal{T}_{\mathcal{D}}}) \end{aligned}$$

From equation (8), the last term is equal to the weight of the (S, \bar{S}) cut and this finishes the proof. \square

Now we can use again Fact 2 to give a lower bound on the optimal cut. The cut (S, \bar{S}) is produced in the same manner as in the standard Goemans-Williamson algorithm via random hyperplane rounding on their semidefinite relaxation for MAXCUT. We will use the fact to prove the following:

Lemma 6. *The weight of the top split relative to the sizes of the triplet constraints is:*

$$w(S, \bar{S}) \geq (0.3413 - 1.756\epsilon_1)m_1 + (0.9266 - 1.756\epsilon_2)m_2$$

Proof. Observe that in the constructed graph, the total negative weight is $W^- = 2m_1 + 2m_2$ as each triplet adds a total negative weight of -2. In order to use Fact 2, we require a lower bound on the optimum value $w(S^*, \bar{S}^*)$.

Here is the first time where we require Assumption 1 about the balancedness of the ground truth tree. From our stochastic model, recall that the number of triplet constraints the cut (L, R) succeeds at is:

$$\mathbb{E}(m_d^{\mathcal{T}_{\mathcal{F}}}) = (3c^2(1-c) + 3c(1-c)^2)(1-\epsilon_1)m_1$$

$$\mathbb{E}(m_o^{\mathcal{T}_{\mathcal{D}}}) = (3c^2(1-c) + 3c(1-c)^2)(1-\epsilon_2)m_2$$

and the number of constraints the cut (L, R) fails at, due to the erroneous constraints is:

$$\mathbb{E}(m_o^{\mathcal{T}_{\mathcal{F}}}) = (3c^2(1-c) + 3c(1-c)^2)\epsilon_1m_1$$

$$\mathbb{E}(m_d^{\mathcal{T}_{\mathcal{D}}}) = (3c^2(1-c) + 3c(1-c)^2)\epsilon_2m_2$$

The quantity $c^2(1-c) + c(1-c)^2$ with $\frac{1}{3} \leq c \leq \frac{2}{3}$ attains a minimum value of $\frac{2}{9}$ when $c = \frac{1}{3}$; hence,

from Lemma 4, the expected weight of the cut (L, R) on the constructed graph is:

$$\begin{aligned}
 & w(L, R) \geq \\
 & m_d^{\mathcal{T}_F}(L, R) - 2m_o^{\mathcal{T}_F}(L, R) + 2m_o^{\mathcal{T}_D}(L, R) - m_d^{\mathcal{T}_D}(L, R) \\
 & \geq \frac{2}{3}(1 - \epsilon_1)m_1 - \frac{4}{3}\epsilon_1m_1 + \frac{4}{3}(1 - \epsilon_2)m_2 - \frac{2}{3}\epsilon_2m_2 \quad (9)
 \end{aligned}$$

Of course the optimum cut has even larger weight than the specific balanced (L, R) cut so: $w(S^*, \bar{S}^*) \geq w(L, R)$. Substituting equation (9) in Fact 2 yields the lemma. \square

Proof of Theorem 7. From Lemma 5, we have a lower bound on our algorithm’s performance via the approximate max cut. Substituting the quantity $w(S, \bar{S})$ based on Lemma 6, yields the theorem. \square

From the above, notice that we beat the trivial baselines as we avoid $\approx 78\% > \frac{2}{3}$ of the forbidden triplets and we satisfy $\approx 64\% > \frac{1}{3}$ of the desired triplets.

A.3 Rankings from Noisy constraints

Here we will show how to beat the approximability thresholds for 3 problems: MAS, BTW and NON-BTW, even though our techniques can be extended to handle many other ordering problems and combinations of desired or forbidden ordering constraints.

Non-BTW: The goal here is to beat the threshold of $\frac{2}{3}$ -approximation and as we will see a 0.84-approximation is possible. The main difference again is on the way we construct the graph based on the generated triplet constraints. For a query $\{ab|c\}$ indicating that c should not be between a, b in the final ordering we add the following 3 undirected edges:

+1 for pairs $(c, a), (c, b)$ and -2 for the pair (a, b)

The graph is as always constructed by inserting all these edges for each of the triplet constraints. We describe below the necessary changes for each of the steps of the template.

- Contrary to previous ordering problems, here a cut into two pieces can either satisfy, postpone or leave unaffected the status of a triplet $\{ab|c\}$. The weight of the cut is:

$$w(S, \bar{S}) = 2m_s(S, \bar{S}) - m_p(S, \bar{S})$$

as a satisfied triplet contributes +2 in the objective ($(c, a), (c, b)$ are cut) while a postponed triplet contributes a total of -1 (labels a and b are separated).

- Our algorithm **ALG**, starting with the (S, \bar{S}) cut and continuing randomly after that, scores a total objective (we drop the (S, \bar{S}) notation):

$$\text{ALG} = m_s + \frac{2}{3}m_u + \frac{1}{2}m_p$$

since even for postponed constraints there is still a $\frac{1}{2}$ probability of correctly placing c either first or last among the three labels. Substituting $m = m_s + m_p + m_u$ which is true for any cut:

$$\begin{aligned}
 \text{ALG} &= \\
 &= m_s + \frac{2}{3}(m - m_s - m_p) + \frac{1}{2}m_p = \\
 &= \frac{2}{3}m + \frac{1}{3}m_s - \frac{1}{6}m_p = \\
 &= \frac{2}{3}m + \frac{1}{6}(2m_s - m_d) = \frac{2}{3}m + \frac{1}{6}w(S, \bar{S}) \quad (10)
 \end{aligned}$$

- The graph’s total negative weight is $W^- = 2m$ so the Goemans-Williamson guarantee is:

$$\mathbb{E}(w(S, \bar{S})) = 0.878w(\text{OPT}) - 0.122 \cdot 2m \quad (11)$$

We lower bound the weight $w(\text{OPT})$ by the weight of the *median* cut: consider the median element q in the unknown optimum permutation and then let one part of the split be the elements that precede q . Generally, in permutation problems, ensuring that a balanced cut with large cut value exists, is easier than problems on trees, as the median cut guarantees a 50-50 split. Since the labels for the constraints were chosen at random, a simple counting argument implies that in expectation $\frac{3}{4}m$ (i.e., $3c^2(1 - c)m + 3(1 - c)^2cm$ with $c = \frac{1}{2}$) constraints are satisfied by the **OPT** cut, so $w(\text{OPT}) \geq 2 \cdot \frac{3}{4}(1 - \epsilon)m - \frac{3}{4}\epsilon m$ and we get $(0.845 - 0.329\epsilon)$ -approximation by substituting in equation (11) and then to (10). For example, even when $\approx 10\%$ are erroneous, we still get a 0.81-approximation.

BTW: The goal here is to beat the $\frac{1}{3}$ -approximation which is the current best for inconsistent instances of BTW. We will get a 0.402-approximation. If the instance is promised to be consistent, Makarychev [Makarychev \(2012\)](#) gave an algorithm achieving $\frac{1}{2}$ -approximation. It is a divide and conquer algorithm that is simple and runs in linear time. A significantly slower algorithm based on semidefinite program with the same approximation guarantee was previously proposed by Chor and Sudan [Chor and Sudan \(1998\)](#).

For a triplet $\{a|b|c\}$ indicating that b should be between a and c in the ordering we construct a graph with undirected edges:

+2 for the pair (a, c) and -1 for the pairs $(b, a), (b, c)$

The edges try to capture that a cut violates the constraint if it separates b from a, c . We give our main steps:

- Contrary to NON-BTW, a cut into two pieces here can either violate, postpone or leave unaffected the status of the triplet $\{a|b|c\}$. The weight of a cut is:

$$w(S, \bar{S}) = m_p(S, \bar{S}) - 2m_v(S, \bar{S})$$

as violated triplets contribute -2 and postponed triplets $+1$.

- Crucially, a postponed by the cut triplet, can still be satisfied with probability $\frac{1}{2}$ and this gives us the advantage:

$$\begin{aligned} \text{ALG} &= \frac{1}{3}m_u + \frac{1}{2}m_p = \\ &= \frac{1}{3}(m - m_p - m_v) + \frac{1}{2}m_p = \\ &= \frac{1}{3}m + \frac{1}{6}(m_p - 2m_v) = \frac{1}{3}m + \frac{1}{6}w(S, \bar{S}) \quad (12) \end{aligned}$$

- Again the graph's total negative weight is $W^- = 2m$ so the Goemans-Williamson guarantee is:

$$\mathbb{E}(w(S, \bar{S})) = 0.878w(\text{OPT}) - 0.122 \cdot 2m \quad (13)$$

As before, we lower bound the weight $w(\text{OPT})$ by the weight of the *median* cut. Since the labels for the constraints were chosen at random, a simple counting argument implies that in expectation $\frac{3}{4}m$ (i.e., $3c^2(1-c)m + 3(1-c)^2cm$ with $c = \frac{1}{2}$) constraints are postponed by the OPT cut, so $w(\text{OPT}) \geq \frac{3}{4}(1-\epsilon)m - 2 \cdot \frac{3}{4}\epsilon m$ and we get a $(0.402 - 0.329\epsilon)$ -approximation by substituting in equation (13) and then to (12). For an error rate of $\approx 10\%$ we still get ≥ 0.369 -approximation, which is better than $\frac{1}{3}$.

MAS: The goal here is to beat the trivial $\frac{1}{2}$ -approximation achieved by an arbitrary or its reversed (or a random) ordering. We will indeed be able to achieve a 0.642 -approximation:

Theorem 8. *Given m constraints generated according to our stochastic model on n items, MAXCUT satisfies at least $(0.642 - 0.4285\epsilon)m$ on average, where ϵ is the fraction of erroneous comparisons. If moreover $m \geq \Omega(\log n)$, the result holds w.h.p.*

The constraints here are on pairs of labels, e.g., $a < b$. Contrary to BTW and NON-BTW where the constructed graph and cuts were undirected, MAS is *orientated* in the sense that it matters which side of the cut the labels end up at. This introduces the first

challenge since we have to solve approximate MAX-CUT in directed graphs with negative weights. For a query $a < b$ indicating that a should precede b in the ranking, we add two directed edges:

- $+1$ directed from $a \rightarrow b$ and another arc with negative weight -1 directed from $b \rightarrow a$

Here the weight of a directed cut (S, \bar{S}) is the sum of all (positively or negatively) weighted arcs going from S to \bar{S} (and we ignore the arcs going from \bar{S} to S). Here a cut can either satisfy, violate or leave unaffected the status of a query and there are no postponed constraints as they only involve two labels. We describe our steps:

- It is easy to see that the weight of any directed (S, \bar{S}) cut is:

$$w(S, \bar{S}) = m_s(S, \bar{S}) - m_v(S, \bar{S})$$

as satisfied pairs contribute $+1$ and violated pairs contribute -1 .

- Again we can compute the value of ALG (dropping the notation with (S, \bar{S})):

$$\begin{aligned} \text{ALG} &= m_s + \frac{1}{2}m_u = \\ &= m_s + \frac{1}{2}(m - m_s - m_v) = \frac{1}{2}m + \frac{1}{2}(w(S, \bar{S})) \quad (14) \end{aligned}$$

- Again the graph's total negative weight is $W^- = m$. However now that the graph is directed and with negative weights, we cannot use the Goemans-Williamson guarantee. A new ingredient in our proof is an SDP relaxation and rounding scheme that achieves:

$$\mathbb{E}(w(S, \bar{S})) = 0.857w(\text{OPT}) - 0.143 \cdot W^- \quad (15)$$

- Continuing as before, we will lower bound the weight $w(\text{OPT})$ by the weight of the *median* directed cut (as noted in the main body, this cut simply separates the first half of the items in the optimal ordering from the last half). Since the labels for the constraints were chosen at random, a simple counting argument implies that in expectation $\frac{1}{2}m$ (i.e., $2c(1-c)m$ with $c = \frac{1}{2}$) constraints are satisfied by the OPT cut, so $w(\text{OPT}) \geq \frac{1}{2}m(1-\epsilon) - \frac{1}{2}\epsilon m$ due to errors in ϵ fraction of the constraints.

Proof of Theorem 8. Given the above observations, in order to get a $(0.642 - 0.4285\epsilon)$ -approximation, we first substitute in equation (15) the lower bound we got for $w(\text{OPT})$, and then we substitute $w(S, \bar{S})$ to (14). \square

For example, if 10% of the constraints are erroneous we still satisfy $\approx 60\%$ of all constraints, beating the worst-case inapproximability results of Guruswami et al. (2011).

A.3.1 Directed MaxCut with negative weights

Here we proceed by proving an important ingredient in our proof relating to finding directed cuts in graphs with negative weights.

In the seminal paper by [Goemans and Williamson \(1995\)](#), they show how directed MAXCUT can be solved approximately on directed graphs with non-negative weights. They used the following semidefinite programming relaxation where A denotes the arcs of the graph and V the vertices ($|V| = n$):

$$\begin{aligned} & \text{maximize } \frac{1}{4} \sum_{(i,j) \in A} w_{ij}(1 + v_0 v_i - v_0 v_j - v_i v_j) \\ & \text{subject to: } \|v_i\|^2 = 1, v_i \in \mathbb{R}^{n+1}, \forall i \in V \cup 0 \end{aligned}$$

Notice the special role of the vector v_0 , which is used to break the symmetry indicating that we want to maximize edges going from left to right where left is the side in which v_0 belongs to. Observe that in an integral $\{\pm 1\}$ solution if vertex i is on the same side with v_0 and j is on the other side then $(1 + v_0 v_i - v_0 v_j - v_i v_j) = 4$ that's why we chose the coefficient $\frac{1}{4}$ in front of the summation. Also note that due to the symmetry if instead of v_j we set $-v_j$ the relaxation won't change so we can instead think of:

$$\text{maximize } \frac{1}{4} \sum_{(i,j) \in A} w_{ij}(1 + v_0 v_i + v_0 v_j + v_i v_j)$$

This will just simplify some trigonometric expressions later.

In this subsection we will prove a bound on the weight of the cut for directed graphs with positive and negative edge weights. The bound we will be able to show is:

$$\mathbb{E}(w(S, \bar{S})) = 0.857w(\text{OPT}) - 0.143 \cdot W^- \quad (16)$$

where W^- denotes the total weight in absolute value of all negative edges. Notice that if no negative weights are present ($W^- = 0$) then we almost recover the Goemans-Williamson 0.878 coefficient. The above bound follows from the following theorem by rearranging terms:

Theorem 9. *Let $W^- = \sum_{(i,j) \in A} |w_{ij}^-|$ where $x^- = \min(0, x)$. Then we can efficiently find a cut (L, R) such that:*

$$\mathbb{E}(w(L, R)) + W^- \geq 0.857 (w(\text{OPT}) + W^-)$$

where OPT denotes the optimum directed cut in the graph.

Proof. Let SDP denote the optimal SDP value which is larger than $w(\text{OPT})$ since we relaxed the problem. We

will show the above bound where $w(\text{OPT})$ is replaced by SDP . We need to rewrite the SDP relaxation to incorporate the W^- term and then we need to compute the probabilities an edge $(i, j) \in A$ participates or does not participate in the cut and how it compares to the contribution in the SDP relaxation. The probability an edge does not participate in the cut is needed here because negatively weighted edges exist, which could potentially decrease the value of the cut. Separating the positive and negative weights ($A = A^+ \cup A^-$) and rewriting the SDP (θ_{ij} denotes the angle between v_i, v_j):

$$\begin{aligned} & \frac{1}{4} \sum_{(i,j) \in A} w_{ij}(1 + v_0 v_i + v_0 v_j + v_i v_j) + W^- = \\ & = \frac{1}{4} \sum_{(i,j) \in A^+} w_{ij}(1 + v_0 v_i + v_0 v_j + v_i v_j) + \\ & + \frac{1}{4} \sum_{(i,j) \in A^-} |w_{ij}| (4 - (1 + v_0 v_i + v_0 v_j + v_i v_j)) = \\ & = \frac{1}{4} \sum_{(i,j) \in A^+} w_{ij}(1 + \cos \theta_{0i} + \cos \theta_{0j} + \cos \theta_{ij}) + \\ & + \frac{1}{4} \sum_{(i,j) \in A^-} |w_{ij}| (3 - \cos \theta_{0i} - \cos \theta_{0j} - \cos \theta_{ij}) \end{aligned}$$

For the rounding algorithm we can use the standard Goemans Williamson rounding although this will only guarantee a sub-optimal coefficient of 0.796 instead of 0.857 in Equation (15). We will show later how a non-standard but better rounding scheme by [Feige and Goemans \(1995\)](#) gives us the desired 0.857 factor.

Let r be a vector drawn uniformly from the unit sphere. Let's evaluate the contribution of a positive arc $(i, j) \in A^+$ to the quantity $\mathbb{E}(w(L, R)) + W^-$:

$$\begin{aligned} & \frac{1}{4} w_{ij} (4 \cdot \Pr[\text{sgn}(v_i r) = \text{sgn}(v_j r) = \text{sgn}(v_0 r)]) = \\ & = w_{ij} \Pr[\text{sgn}(v_i r) = \text{sgn}(v_j r) = \text{sgn}(v_0 r)] \end{aligned}$$

For a negative arc $(i, j) \in A^-$, the contribution to the quantity $\mathbb{E}(w(L, R)) + W^-$ is:

$$\begin{aligned} & -\frac{1}{4} |w_{ij}| (4 \Pr[\text{sgn}(v_i r) = \text{sgn}(v_j r) = \text{sgn}(v_0 r)]) + |w_{ij}| = \\ & = |w_{ij}| (1 - \Pr[\text{sgn}(v_i r) = \text{sgn}(v_j r) = \text{sgn}(v_0 r)]) \end{aligned}$$

Finally, if we can manage to lower bound $\Pr[\text{sgn}(v_i r) = \text{sgn}(v_j r) = \text{sgn}(v_0 r)]$ by $(1 + \cos \theta_{0i} + \cos \theta_{0j} + \cos \theta_{ij})$ and simultaneously lower bound $(1 - \Pr[\text{sgn}(v_i r) = \text{sgn}(v_j r) = \text{sgn}(v_0 r)])$ by $(3 - \cos \theta_{0i} - \cos \theta_{0j} - \cos \theta_{ij})$ we will have finished as the final result will follow by linearity of expectations. This can indeed be done using some trigonometric facts and the symmetry of spherical geometry:

Fact 3. Let r be chosen uniformly at random from the unit sphere. Then for any three vectors v_i, v_j, v_0 in the unit sphere:

$$\begin{aligned} \Pr[\text{sgn}(v_i r) = \text{sgn}(v_j r) = \text{sgn}(v_0 r)] &= \\ &= 1 - \frac{1}{2\pi}(\theta_{ij} + \theta_{j0} + \theta_{i0}) \geq \\ &\geq 0.796 \cdot \frac{1}{4}(1 + \cos \theta_{0i} + \cos \theta_{0j} + \cos \theta_{ij}) \end{aligned}$$

and also:

$$\begin{aligned} 1 - \Pr[\text{sgn}(v_i r) = \text{sgn}(v_j r) = \text{sgn}(v_0 r)] &= \\ &= \frac{1}{2\pi}(\theta_{ij} + \theta_{j0} + \theta_{i0}) \geq \\ &\geq 0.878 \cdot \frac{1}{4}(3 - \cos \theta_{0i} - \cos \theta_{0j} - \cos \theta_{ij}) \end{aligned}$$

Putting it all together and using linearity of expectations we have shown:

$$\begin{aligned} \mathbb{E}(w(L, R)) + W^- &\geq 0.796 (\text{SDP} + W^-) \geq \\ &\geq 0.796 (w(\text{OPT}) + W^-) \end{aligned}$$

As we shall see next the first inequality is the one that determines the approximation coefficient. The above proves so far that 0.796 is possible. However there exists a more complicated rounding scheme which does not choose r uniformly at random. It was developed in the context of MAX-2-SAT problem by Feige and Goemans and their main idea behind their improvement is to take advantage of the special role of v_0 . They crucially use v_0 : they map each v_i to another vector w_i that depends both on v_i and on v_0 , and only then they proceed with the Goemans-Williamson rounding algorithm. Specifically, w_i is coplanar with v_0 , on the same side of v_0 as v_i is, and forms an angle with v_0 equal to $f(\theta_{i0})$. By choosing the function f to be:

$$f_{1/2}(\theta) = \frac{1}{2}\theta + \frac{1}{2}\left(\frac{\pi}{2}(1 - \cos \theta)\right)$$

they report that they get a coefficient 0.857 for the first inequality above (instead of 0.796) and simultaneously a coefficient 0.9249 for the second inequality (instead of 0.878). Using again linearity of expectation, this implies our theorem:

$$\mathbb{E}(w(L, R)) + W^- \geq 0.857 (w(\text{OPT}) + W^-)$$

□

A.4 Correlation Clustering from Noisy Constraints

The last of the proofs for the positive results will be for the Correlation Clustering problem, following the same ideas as in the proofs above. In correlation clustering, the information comes as MUST-LINK (ab) or CANNOT-LINK ($a|b$) constraints indicating if two labels should be in the same or in different

parts of an optimal partition. The current best algorithm is a 0.7666-approximation by Swamy (2004) and here we improve under our stochastic model for the input constraints. We achieve a $(0.8226 - 0.775\epsilon)$ -approximation.

$$\begin{aligned} \text{ALG} &= m_s + 0.766m_u = m_s + 0.766(m - m_s - m_v) = \\ &= 0.766m + 0.234(m_s - 3.2735m_v) \end{aligned}$$

We construct an undirected graph where for every CANNOT-LINK constraint ab we add a +1 edge between a, b and for every MUST-LINK constraint ab we add an edge a, b now with negative weight -3.2735 .

$$w(S, \bar{S}) = m_s(S, \bar{S}) - 3.2735m_v(S, \bar{S})$$

Hence:

$$\text{ALG} = 0.766m + 0.234w(S, \bar{S})$$

Assuming that the largest cluster in the optimum partition has size at most $\frac{n}{2}$, our stochastic model will generate at least $\frac{m}{2}$ CANNOT-LINK constraints by a simple counting argument. This is in expectation, but of course using a standard large deviation Chernoff bound, all our claims in this paper can be made to hold with high probability. This also implies that the total number of MUST-LINK constraints is at most $\frac{m}{2}$. Thus, once again using MAXCUT for the first split:

$$\mathbb{E}(w(S, \bar{S})) = 0.878w(\text{OPT}) - 0.122 \cdot 3.2735 \cdot \frac{m}{2}$$

An easy lower bound for the value of the OPT cut is: $w(\text{OPT}) \geq \frac{m}{2}$ hence we obtain a 0.8226-approximation.

B Hardness via Ordering CSPs

In this part of the Appendix, we present our hardness of approximation results for the constraint satisfaction problems on trees, extending in some cases the inapproximability results of Guruswami et al. (2011); Austrin et al. (2013) from linear orderings to trees.

B.1 Hardness for Rooted Triplets Consistency

We prove that under the UGC, it is hard to approximate the Desired Triplets Consistency problem better than a factor of $\frac{2}{3}$, even in the unweighted case. Notice that the current best approximation is $\frac{1}{3}$ achieved by a random tree (or a simple greedy algorithm). In fact our result is slightly stronger: it is hard to distinguish between two instances one of which is almost perfect (e.g., 99% of constraints are consistent) and the other is far from perfect (e.g., 67% of constraints are consistent). We base our hardness result on the following theorem by Austrin et al. (2013) about the *Non-Betweenness* problem and its $\frac{2}{3}$ -inapproximability:

Fact 4. Let K be the total number of triplets constraints in an instance of NON-BTW. For any $\epsilon > 0$, it is NP-hard to distinguish between NON-BTW instances of the following two cases:

YES: $val(\pi^*) \geq (1 - \epsilon)K$, i.e. the optimal permutation satisfies almost all constraints.

NO: $val(\pi^*) \leq (\frac{2}{3} + \epsilon)K$, i.e. the optimal permutation does not satisfy more than $2/3$ fraction of the constraints.

Given the above fact, we prove our $\frac{2}{3}$ -inapproximability result for Triplets Consistency:

Theorem 10. Let K be the total number of the triplet constraints in an instance of Desired Triplets Consistency. For any $\delta > 0$, it is NP-hard to distinguish between instances of the following two cases:

YES: $val(T^*) \geq (\frac{1}{2} - \delta)K$, i.e. the optimal tree satisfies almost half of all the triplet constraints.

NO: $val(T^*) \leq (\frac{1}{3} + \delta)K$, i.e. the optimal tree does not satisfy more than $\frac{1}{3}$ fraction of the triplet constraints.

Then, our $\frac{2}{3}$ -inapproximability result follows directly from the gap of these instances: $\frac{1/2}{1/3} = \frac{2}{3}$.

Proof. Start with a YES instance of the NON-BTW problem with optimal permutation π^* and $val(\pi^*) \geq (1 - \epsilon)K$. Viewing each NON-BTW constraint as a desired triplet, we show how to construct a tree T such that $val(T) \geq (\frac{1}{2} - \delta(\epsilon))K$. In fact, the construction is straightforward: simply assign the n labels, either in the order they appear in π^* or reversed, as the leaves of a caterpillar tree (every internal node has at least one child that is a leaf). Observe that this tree satisfies:

$$val(T) \geq (1 - \epsilon)K/2$$

This is because if a NON-BTW constraint $ab|c$ was obeyed by π^* , it will also be obeyed by one of the two caterpillar trees above: if c appears first in the permutation then the former caterpillar will obey $ab|c$ as c gets separated first, otherwise if c appears last, then the reversed caterpillar tree will obey $ab|c$. Here the $\frac{1}{2}$ factor is tight, since for example, the two NON-BTW constraints $ab|c$ and $bc|a$ are both satisfied by the ordering abc , but when viewed as desired triplets, they cannot both be satisfied by a tree.

The NO instance is slightly more challenging. Start with a NO instance of the NON-BTW problem with optimal π^* of value $val(\pi^*) \leq (\frac{2}{3} + \epsilon)K$. Viewing the NON-BTW constraints as desired triplets, we show that the optimum tree T^* cannot achieve better than $> (1/3 + 2\epsilon)K$, because this would imply that $val(\pi^*) > (\frac{2}{3} + \epsilon)K$, which is a contradiction.

For this, assume that some tree T scored a value $val(T) > (1/3 + 2\epsilon)K$. We will construct a permutation π from the tree T with value $val(\pi) > (2/3 + \epsilon)K$.

Observe that directly projecting the leaves of T onto a line (just outputting the n leaves from left to right as they appear in the tree) would already satisfy $> (1/3 + 2\epsilon)K$, since every desired triplet $ab|c$ obeyed by the tree, will also be obeyed (as a NON-BTW constraint) by π as c will either be first or last among the three labels a, b, c .

Moreover, there are potentially desired triplet constraints that are disobeyed by the tree T , yet obeyed by the permutation. We know that the number of remaining constraints is $K - (1/3 + 2\epsilon)K = (2/3 - 2\epsilon)K$. By randomly swapping each left and right child in the tree T before we do the projection to the permutation π , will actually lead to an excess of $1/2 \cdot (2/3 - 2\epsilon)K = (1/3 - \epsilon)K$ number of NON-BTW constraints. To see this notice that for every triplet that is disobeyed in the tree, there is a $\frac{1}{2}$ probability that it becomes obeyed in the permutation. Summing up, we get $val(\pi) > (1/3 + 2\epsilon)K + (1/3 - \epsilon)K > (2/3 + \epsilon)K \implies val(\pi^*) \geq val(\pi) > (2/3 + \epsilon)K$, a contradiction. \square

B.2 Hardness for Forbidden Triplets: Random is Optimal

We prove that under the UGC, it is hard to approximate the Forbidden Triplets Consistency problem better than a factor of $\frac{2}{3}$, even in the unweighted case. Notice that the current best approximation is in fact $\frac{2}{3}$ achieved by a random tree (or a simple greedy algorithm), hence we settle the computational complexity of the problem. Our result is slightly stronger: it is hard to distinguish between two instances one of which is almost perfect (e.g., 99% of constraints are consistent) and the other is far from perfect (e.g., 67% of constraints are consistent). We base our hardness result on the following theorem by Guruswami et al. (2011) about the BTW problem and its $\frac{1}{3}$ -inapproximability:

Fact 5. Let K be the total number of triplets constraints in an instance of BTW. For any $\epsilon > 0$, it is UGC-hard to distinguish between BTW instances of the following two cases:

YES: $val(\pi^*) \geq (1 - \epsilon)K$, i.e. the optimal permutation satisfies almost all constraints.

NO: $val(\pi^*) \leq (\frac{1}{3} + \epsilon)K$, i.e. the optimal permutation does not satisfy more than $1/3$ fraction of the constraints.

Given the above fact, we prove our $\frac{2}{3}$ -inapproximability result for Forbidden Triplets Consistency:

Theorem 11. Let K be the total number of the triplet constraints in an instance of Forbidden Triplets Consistency. For any $\delta > 0$, it is UGC-hard to distinguish between instances of the following two cases:

YES: $val(T^*) \geq (1 - \delta)K$, i.e. the optimal tree satisfies almost half of all the triplet constraints.

NO: $val(T^*) \leq (\frac{2}{3} + \delta)K$, i.e. the optimal tree does not satisfy more than $\frac{2}{3}$ fraction of the triplet constraints.

Then, our $\frac{2}{3}$ -inapproximability result follows directly from the gap of these instances: $\frac{2}{3}/1 = \frac{2}{3}$.

Proof. Start with a YES instance of the BTW problem with optimal permutation π^* and $val(\pi^*) \geq (1 - \epsilon)K$. Viewing each BTW constraint $a|b|c$ as a forbidden triplet $ac|b$, we show how to construct a tree T such that $val(T) \geq (\frac{1}{3} - \delta(\epsilon))K$. In fact, the construction is straightforward: simply assign the n labels, in the order they appear in π^* , as the leaves of a caterpillar tree (every internal node has its left child being a leaf). Observe that this caterpillar tree satisfies:

$$val(T) \geq (1 - \epsilon)K$$

This is because if a BTW constraint $a|b|c$ was obeyed by π^* , it will also be avoided (viewed as a forbidden triplet $ac|b$) by the caterpillar tree above: if a appears first in the permutation then the caterpillar will avoid $ac|b$ as a gets separated first, otherwise if c appears first, then again the caterpillar tree will avoid $ac|b$ as c gets separated first.

The NO instance is slightly more challenging. Start with a NO instance of the BTW problem with optimal π^* of value $val(\pi^*) \leq (\frac{1}{3} + \epsilon)K$. Viewing the BTW constraints as forbidden triplets, we show that the optimum tree T^* cannot achieve better than $> (2/3 + 2\epsilon)K$, because this would imply that $val(\pi^*) > (\frac{1}{3} + \epsilon)K$, which is a contradiction.

For this, assume that some tree T scored a value $val(T) > (2/3 + 2\epsilon)K$. We will construct a permutation π from the tree T with value $val(\pi) > (1/3 + \epsilon)K$, a contradiction. Notice that there are forbidden triplets that may be avoided by the tree, yet obeyed by the permutation: for example for a forbidden triplet $t = ac|b$, the tree R that first removes a and then splits b, c will successfully avoid t , however the permutation acb can come from R by projection, however acb do not obey the BTW constraint $a|b|c$.

Hence directly projecting the leaves of T onto a line may not satisfy $> (1/3 + 2\epsilon)K$, since every forbidden triplet $ac|b$ avoided by T , can be ordered by this projected permutation in a way that would not obey the corresponding BTW constraint $a|b|c$.

However, just by randomly swapping each left and right child for every internal node in the tree before we do the projection to the permutation, would satisfy $1/2 \cdot (2/3 + 2\epsilon)K = (1/3 + \epsilon)K$ number of constraints. To see this, note that with probability $\frac{1}{2}$ a forbidden $ac|b$ avoided by T will be mapped to the desired abc (and not acb) or cba (and not cab) ordering.

Finally, we get $val(\pi) > (1/3 + \epsilon)K \implies val(\pi^*) \geq val(\pi) > (1/3 + \epsilon)K$, a contradiction that we were given a NO instance. \square

B.3 Hardness for Desired Quartets Consistency

The main result in this section is that for the desired quartets problem, one cannot do better than $\frac{2}{3}$ -approximation. Notice that a random unrooted tree achieve $\frac{1}{3}$ -approximation which is currently the best known algorithm.

To prove our results, we make use of a consequence from the results in Guruswami et al. (2011) for orderings CSPs of arity 4. Specifically, we define the following problem, which we call 4-SEPARATEDNESS.

Definition 1. For an ordering problem, a 4-SEPARATEDNESS constraint $\{ab|cd\}$ specifies that both elements a, b should precede c, d or that both c, d should precede a, b in the output ordering (e.g., $badc$, but not $acbd$). No constraints are placed on the relative ordering between a, b or on the ordering between c, d .

Fact 6. Given 4-SEPARATEDNESS constraints, no polynomial time algorithm can beat the performance of a random permutation, which achieves a $\frac{1}{3}$ -approximation, assuming UGC. In fact, if K is the total number of constraints, for any $\epsilon > 0$, it is UGC-hard to distinguish between the two cases:

YES: $val(\pi^*) \geq (1 - \epsilon)K$, i.e. the optimal permutation satisfies almost all constraints.

NO: $val(\pi^*) \leq (\frac{1}{3} + \epsilon)K$, i.e. the optimal permutation does not satisfy more than $1/3$ fraction of the constraints.

Observe that from the $4! = 24$ permutations on a, b, c, d only 8 of them obey the 4-SEPARATEDNESS constraint, that's why random achieves $\frac{1}{3}$.

Theorem 12. Let K be the total number of the quartet constraints in an instance of Desired Quartets Consistency. For any $\delta > 0$, it is UGC-hard to distinguish between instances of the following two cases:

YES: $val(T^*) \geq (1 - \delta)K$, i.e. the optimal tree satisfies almost all the quartet constraints.

NO: $val(T^*) \leq (\frac{2}{3} + \delta)K$, i.e. the optimal tree does not satisfy more than a $\frac{2}{3}$ fraction of the quartet constraints.

Proof. We will make a reduction from the 4-SEPARATEDNESS problem. Start from a YES instance and consider the optimum permutation π^* . Construct an unrooted caterpillar tree T with leaves the labels of π^* as they appear in the permutation. It is easy to see that if a 4-SEPARATEDNESS constraint $ab|cd$ was obeyed by the permutation, then the corresponding quartet constraint $ab|cd$ was also obeyed in the caterpillar tree T . For that, we can assume w.l.o.g. that the

elements appear with relative order $abcd$ in π^* and observe that the paths $a \rightarrow b$ and $c \rightarrow d$ in T are disjoint, so the quartet is obeyed.

The harder case is the **NO** instance. For that we will show how from a tree T with high value, we can construct a permutation π with high value. Specifically, we will show that if $\text{val}(T) > (\frac{2}{3} + 2\epsilon)K$ then we can find π with $\text{val}(\pi) > \frac{1}{2}(\frac{2}{3} + 2\epsilon)K = (\frac{1}{3} + \epsilon)K$, a contradiction since we started from a **NO** instance.

The tree T is an unrooted tree on $n \geq 4$ leaves, whose internal nodes have degree exactly 3. We can make T rooted by selecting an arbitrary internal node r and making it the root of a binary tree whose internal nodes have exactly 2 children and one parent. The only exception is the root r that has 3 children and no parent. Call this tree T_r . Let A, B, C denote the leftmost, middle and rightmost child of r respectively, which are themselves rooted binary trees. Assume w.l.o.g. that A contains the largest number of leaves among A, B, C , so $|A| \geq 2$, where $|A|$ denotes the number of leaves contained in the subtree rooted at A .

From this rooted tree T_r , we generate a permutation π by randomly swapping every left and right child on each internal node of T_r and also randomly swapping A, B, C at the root r ; then we simply project the leaves onto a line to get π . We show that each quartet $q_1q_2|q_3q_4$ obeyed by T will be obeyed in π with probability $p \geq \frac{1}{2}$. We have several cases depending on the labels q_1, q_2, q_3, q_4 :

- If $q_1, q_2 \in A$ and $q_3 \in B$ and $q_4 \in C$: Notice that the status of the quartet is decided by the random choices at the root r since after the final projection, labels from A will be consecutive in π and similarly for B and C . Here, π will actually obey the quartet with probability $\frac{2}{3}$, as there are 3 equally likely outcomes ABC, BCA and CAB and the first two ABC and BCA obey the quartet, irrespectively of how labels from A, B, C are ordered.
- If $q_1, q_2 \in A$ and $q_3, q_4 \in B$: This is the easiest case as every quartet of this form will be obeyed in π with probability 1. This follows as labels from A will be consecutive in π and similarly for B .
- If $q_1, q_2, q_3 \in A$ and $q_4 \in B$: The status of this quartet only depends on how the elements q_1, q_2, q_3 are placed. Specifically, depending on the random choices at the root r , q_4 can appear either first (if BA was chosen) or last (if AB was chosen) among the 4 elements in π . If the former is true, then q_3 should appear second and we get $q_4q_3|\cdot\cdot$ otherwise q_3 should appear third and we get $\cdot\cdot|q_3q_4$. We need to compute the probability for each of these events. Notice that the lowest

common ancestor both for q_3, q_1 and for q_3, q_2 is A . Hence, the status of the quartet is determined at A and with probability $\frac{1}{2}$, q_3 is correctly placed on the same side as B (and q_4).

- If $q_1, q_2, q_3, q_4 \in A$: This case essentially reduces to the analyses of the previous two cases. Just find the lowest common ancestor A_1 of all 4 labels q_1, q_2, q_3, q_4 in T_r . If two of the labels belong to one child and the remaining to the other child, then the quartet will be obeyed with probability 1, irrespectively of the random choices at A_1 (similar to the second case above). Moreover, if one child contains three of the 4 elements, then the analysis is the same as the previous case yielding a probability of $\frac{1}{2}$.

The other cases are symmetric for B, C . This proves that if a quartet is obeyed by the tree then with probability $\frac{1}{2}$ will be obeyed in π which means that $\text{val}(\pi) > \frac{1}{2}(\frac{2}{3} + 2\epsilon)K = (\frac{1}{3} + \epsilon)K$ by linearity of expectation. This contradicts the fact that we were given a **NO** instance. \square

B.4 Hardness for Forbidden Quartets Consistency

The proof proceeds in the same way as the previous paragraph, where we now account for the forbidden quartets and we make use of the complement problem to 4-SEPARATEDNESS, which we call 4-NON-SEPARATEDNESS:

Definition 2. For an ordering problem, a 4-NON-SEPARATEDNESS constraint $\{ab|cd\}$ specifies that either a or b should be between c, d or that either c or d should be between a, b in the output ordering (e.g., $adcb$, but not $abcd$). No constraints are placed on the relative ordering between a, b or on the ordering between c, d .

Fact 7. Given 4-NON-SEPARATEDNESS constraints, no polynomial time algorithm can beat the performance of a random permutation, which achieves a $\frac{2}{3}$ -approximation, assuming UGC. In fact, if K is the total number of constraints, for any $\epsilon > 0$, it is UGC-hard to distinguish between the two cases:

YES: $\text{val}(\pi^*) \geq (1 - \epsilon)K$, i.e. the optimal permutation satisfies almost all constraints.

NO: $\text{val}(\pi^*) \leq (\frac{2}{3} + \epsilon)K$, i.e. the optimal permutation does not satisfy more than $2/3$ fraction of the constraints.

Observe that from the $4! = 24$ permutations on a, b, c, d , 16 of them obey the 4-NON-SEPARATEDNESS constraint, that's why random achieves $\frac{2}{3}$.

Theorem 13. Let K be the total number of the quartet constraints in an instance of Forbidden Quartets

Consistency. For any $\delta > 0$, it is UGC-hard to distinguish between instances of the following two cases:

YES: $\text{val}(T^*) \geq (1 - \delta)K$, i.e. the optimal tree satisfies almost all the quartet constraints.

NO: $\text{val}(T^*) \leq (\frac{8}{9} + \delta)K$, i.e. the optimal tree does not satisfy more than a $\frac{8}{9}$ fraction of the quartet constraints.

Proof. We will make a reduction from the 4-NON-SEPARATEDNESS problem. Start from a **YES** instance and consider the optimum permutation π^* . Construct an unrooted caterpillar tree T with leaves the labels of π^* as they appear in the permutation. It is easy to see that if a 4-NON-SEPARATEDNESS constraint $ab|cd$ was disobeyed (hence successfully avoided) by the permutation, then the corresponding quartet constraint $ab|cd$ was also disobeyed (i.e., avoided) in the caterpillar tree T . For that, we can assume w.l.o.g. that the elements appear with relative order $acbd$ in π^* and observe that the paths from $a \rightarrow c$ and from $b \rightarrow d$ in T are disjoint, so the quartet is disobeyed as we wanted.

The harder case is the **NO** instance. For that we will show how from a tree T with high value, we can construct a permutation π with high value. Specifically, we will show that if $\text{val}(T) > (\frac{8}{9} + \frac{4}{3}\epsilon)K$ then we can find π with $\text{val}(\pi) > \frac{3}{4}(\frac{8}{9} + \frac{4}{3}\epsilon)K = (\frac{2}{3} + \epsilon)K$, a contradiction since we started from a **NO** instance.

The tree T is an unrooted tree on $n \geq 4$ leaves, whose internal nodes have degree exactly 3. We follow the same algorithm to generate the rooted T_r and the final permutation π as above. the notation for A, B, C is the same as previously. We show that each quartet $q = q_1q_2|q_3q_4$ disobeyed by T will be disobeyed in π with probability $p \geq \frac{3}{4}$. We have several cases depending on the labels q_1, q_2, q_3, q_4 :

- If $q_1, q_3 \in A$ and $q_2 \in B$ and $q_4 \in C$: First notice that indeed quartet $q = q_1q_2|q_3q_4$ is disobeyed by the unrooted tree since it instead obeys $q_1q_3|q_2q_4$. We show that the probability that π disobeys q is $\frac{5}{6}$. If the random choices at the root r produce ABC or BCA , then with probability 1 the quartet q is disobeyed after the projection. For example, if the realization is ABC notice that either q_3 will be between q_1, q_2 or q_1 will be between q_3, q_4 , thus disobeying the corresponding 4-NON-SEPARATEDNESS constraint. Symmetrically, we handle the scenario where the realization was BCA . However, with probability $\frac{1}{3}$ the realization at the root was CAB and now the status of the quartet is determined by the random choice at the lowest common ancestor of q_1, q_3 . With probability $\frac{1}{2}$ label q_1 precedes q_3 , thus giving the ordering $q_4q_1|q_3q_2$ disobeying q . In total q is avoided with probability $\frac{2}{3} + \frac{1}{6} = \frac{5}{6}$.
- If $q_1, q_3 \in A$ and $q_2, q_4 \in B$: This is the easiest case as every quartet of this form will be disobeyed in π with probability 1. This follows as labels from A will be consecutive in π and similarly for B .
- If $q_1, q_2, q_3 \in A$ and $q_4 \in B$: The status of this quartet only depends on how the elements q_1, q_2, q_3 are placed. Specifically, depending on the random choices at the root r , q_4 can appear either first (if BA was chosen) or last (if AB was chosen) among the 4 elements in π . If the former is true, then q_3 should appear third or fourth and we get $q_4 \cdot |q_3 \cdot$ or $q_4 \cdot | \cdot q_3$, otherwise q_3 should appear first or second and we get $q_3 \cdot | \cdot q_4$ or $\cdot q_3 | \cdot q_4$. We need to compute the probability for each of these events. By the fact that the tree T disobeys q , we can assume w.l.o.g. that label q_1 is the closest to q_3 , otherwise we just rename q_2 as q_1 and vice versa. We get that the lowest common ancestor A_{13} of q_1, q_3 in A is strictly lower than the lowest common ancestor A_{12} of q_1, q_2 in A (in terminology of triplets consistency we have $q_1q_3|q_2$). W.l.o.g. assume that BA was chosen at the root r , so q_4 will appear first. By the random choice in our algorithm, A_{12} placed q_2 at the left child (hence second among the 4 elements) with probability $\frac{1}{2}$ and the quartet q is disobeyed. If instead our algorithm placed q_2 at the right child (and hence fourth in the ordering), there is still $\frac{1}{4}$ probability of placing q_3 at the right child of A_{13} . This means that with probability $\frac{1}{2} + \frac{1}{4} = \frac{3}{4}$, the projected π disobeys q as promised by the theorem.
- If $q_1, q_2, q_3, q_4 \in A$: This case essentially reduces to the analyses of the previous two cases. Just find the lowest common ancestor A_1 of all 4 labels q_1, q_2, q_3, q_4 in T_r . If two of the labels belong to one child and the remaining to the other child, then the quartet will be disobeyed with probability 1, irrespectively of the random choices at A_1 (similar to the second case above). Moreover, if one child contains three of the 4 elements, then the analysis is the same as the previous case yielding a probability of $\frac{3}{4}$.

The other cases are symmetric for B, C . This proves that if a quartet is disobeyed by the tree then with probability $\frac{3}{4}$ will be disobeyed in π which means that $\text{val}(\pi) > \frac{3}{4}(\frac{8}{9} + \frac{4}{3}\epsilon)K = (\frac{2}{3} + \epsilon)K$ by linearity of expectation. This contradicts the fact that we were given a **NO** instance. \square