# On the Privacy-Utility Tradeoff in Peer-Review Data Analysis

**Wenxin Ding**[1], **Nihar B. Shah**[1, 2], **Weina Wang** [1]
[1] Computer Science Department, [2] Machine Learning Department
Carnegie Mellon University
{wenxind@andrew, nihars@cs, weinaw@cs}.cmu.edu

## Abstract

A major impediment to research on improving peer review is the unavailability of peer-review data, since any release of such data must grapple with the sensitivity of the peer review data in terms of protecting identities of reviewers from authors. We posit the need to develop techniques to release peer-review data in a privacy-preserving manner. Identifying this problem, in this paper we propose a framework for privacy-preserving release of certain conference peer-review data — distributions of ratings, miscalibration, and subjectivity — with an emphasis on the accuracy (or utility) of the released data. The crux of the framework lies in recognizing that a part of the data pertaining to the reviews is already available in public, and we use this information to post-process the data released by any privacy mechanism in a manner that improves the accuracy (utility) of the data while retaining the privacy guarantees. Our framework works with any privacy-preserving mechanism that operates via releasing perturbed data. We present several positive and negative theoretical results, including a polynomial-time algorithm for improving on the privacy-utility tradeoff.

## 1 Introduction

A fair and efficient peer-review process is of utmost importance to the development of scientific research. There are, however, a large number of challenges in peer review, pertaining to its fairness and efficiency. Consequently there is an overwhelming desire to "fix" the "broken" peer review process (Rennie 2016; McCook 2006). And taking heed to this call, there is a growing amount of research on this topic.

Research on improving peer review suffers from a considerable handicap – unavailability of data (Balietti, Goldstone, and Helbing 2016; Tomkins, Zhang, and Heavlin 2017; Squazzoni et al. 2020; Schroter, Loder, and Godlee 2020). Concealing the identities of reviewers from authors of any paper is paramount in most peer review systems. Thus releasing any peer review data is fraught with the risk of compromising on this privacy. As noted by Balietti (2016):

> *"The main reason behind the lack of empirical studies on peer-review is the difficulty in accessing data. In fact, peer-review data is considered very sensitive, and it is very seldom released for scrutiny, even in an anonymous form."*

Although there is a large body of research on the topic of privacy in various domains, not much privacy research directly targets the application of peer review. In an influential recent paper about peer review, Tomkins, Zhang, and Heavlin (2017) highlight the challenges they faced in this respect and their consequent inability to release data:

> *"We would prefer to make available the raw data used in our study, but after some effort we have not been able to devise an anonymization scheme that will simultaneously protect the identities of the parties involved and allow accurate aggregate statistical analysis. We are familiar with the literature around privacy preserving dissemination of data for statistical analysis and feel that releasing our data is not possible using current state-of-the-art techniques. "*

We thus posit the need to develop techniques to help release peer-review data while ensuring that identities of reviewers of any paper are protected. With this motivation, we focus on the privacy-utility tradeoff in releasing certain conference peer-review data. The data to be released comprises distributions of the ratings or miscalibration or subjectivity in the peer-review process. The notion of privacy we consider is quite general – our techniques apply to any notion of privacy which operates by perturbing the data, including differential privacy. We design a framework to improve in this tradeoff by improving the utility (accuracy) while retaining privacy guarantees.

Our work relies on the key observation that a non-trivial part of conference peer-review data is already available in the public domain. We design techniques which use this publicly available information to post-process the data released by any privacy mechanism. Our approach is guided by the following four desiderata for such a post processing:

- D1 Under no circumstances should the accuracy decrease after applying the algorithm.
- D2 Under no circumstances should the privacy guarantee be compromised after applying the algorithm.
- D3 The algorithm should have a computational complexity polynomial in the number of reviewers and papers.[1]

Copyright © 2021, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

[1]In typical conferences, the number of papers per reviewer and

D4 In special cases where an exact answer can be easily obtained from public data, the algorithm should also return the same answer with no error. (This is defined formally in Section 4.3.)

Our technical contributions (detailed in Section 4) towards this problem are as follows. We use the straightforward observation that projecting the (noisy) output of the privacy mechanism on the convex hull of all possible true values is desirable from the perspective of the desiderata. We prove that, however, such a projection is NP-hard (via reducing the $\ell$-partition problem). We then design a polynomial-time computable algorithm which projects the noisy output of the privacy mechanism on a convex set containing all possible true values, and satisfies the four desiderata listed above. As a result of independent interest, we also prove that the more obvious approach of projecting on the set of all true values (instead of a convex set containing them) can, in fact, reduce the accuracy.

Finally, in Appendix C, we conduct synthetic simulations, which reveal that our methods can yield considerable improvements in the privacy-utility tradeoff as compared to standard approaches. The associated code for our algorithm is available at https://github.com/wenxind/privacy-utility-tradeoff-in-peer-review-data.

## 2 Related Work

This work falls in the intersection of two lines of research: peer review and privacy.

**Peer review:** Peer review is the backbone of scientific research. There is an overwhelming desire in many domains of science and engineering for improving peer review, and consequently, there are many past works on the topic of either evaluating the efficacy of peer review or improving the peer review process (Peters and Ceci 1982; Kliewer et al. 2004; Bennett, Jagsi, and Zietman 2018; Mavrogenis, Quaile, and Scarlat 2020; Bernard 2018; Snodgrass 2006; Scott 1974; Lindsey 1988; Douceur 2009; Reinhart 2009). These works, however, largely focus on the journal reviewing setup that is common in non-computer-science fields, whereas our focus is on the conference reviewing setting which is more common in computer science.

The number of submissions to many computer science conferences, particularly to machine learning or artificial intelligence conferences, is growing near-exponentially and is presently in the several thousands. This rapid growth has spurred a considerable amount of recent research on peer review in computer science. These works include those on handling problems related to reviewer-assignment (Goldsmith and Sloan 2007; Charlin and Zemel 2013; Welch 2014; Stelmakh, Shah, and Singh 2018; Kobren, Saha, and McCallum 2019), miscalibration (Roos, Rothe, and Scheuermann 2011; Ge, Welling, and Ghahramani 2013; Wang and Shah 2019), subjectivity (Noothigattu, Shah, and Procaccia 2018), biases (Tomkins, Zhang, and Heavlin 2017; Stelmakh, Shah, and Singh 2019), strategic behavior (Balietti, Goldstone, and Helbing 2016; Xu et al. 2019; Stelmakh, Shah, and Singh 2020) and others (Cabanac and Preuss 2013; Fiez, Shah, and Ratliff 2019; Lawrence and Cortes 2014; Shah et al. 2018; Stelmakh et al. 2020). In particular, as will be detailed later, our work is also useful towards releasing data pertaining to miscalibration and subjectivity, thereby helping in the understanding and mitigation of these problems.

**Privacy:** Privacy-preserving data analytics has been receiving rapidly increasing attention as the big-data regime emerges. There is a large body of research that investigates formal notion of privacy and quantifies the tradeoff between privacy and utility (see, e.g., Dwork et al. 2006b; Dwork 2006; Blum, Ligett, and Roth 2008; Gaboardi et al. 2014; Wang, Ying, and Zhang 2016; Bun, Ullman, and Vadhan 2018). Among these studies, differential privacy (Dwork et al. 2006b; Dwork 2006) has become the de facto standard and has been applied to many areas.

In this paper, we investigate the privacy-utility tradeoff for publishing histograms of peer-review data. Privacy-preserving release of histograms has been a major focus of the literature (Chawla et al. 2005; Dwork et al. 2006a; Hay et al. 2010; Li et al. 2010; Bassily and Smith 2015; Balcer and Vadhan 2019; Abowd et al. 2019). To the best of our knowledge, existing techniques for improving the privacy-utility tradeoff are generally inadequate for the application of peer review since they do not take into account the special structures in peer-review data.

In particular, one special feature of peer-review data is that some specific part of the data such as scores received by papers is already published in its original, non-privacy-preserving form. This provides us an opportunity to utilize the "consistency" with public knowledge. The concept of consistency, with different problem-specific meanings, has been investigated by existing work for privacy-preserving algorithms. Hay et al. 2010 improves accuracy by assuring consistency among answers to different queries. The closest work to ours is the privacy-preserving approach for US Census (Abowd et al. 2019), where consistency with public data is a requirement and it is in the form of a set of linear constraints. In contrast, we are not subject to a strict requirement of consistency, but instead, we exploit consistency as a method to improve accuracy. Additionally, the consistency with public knowledge in our problem is of a more combinatorial nature. As we discuss in the sequel, the idiosyncratic nature of the peer-review setting implies that one can design methods tailored to this application which yield a (considerable) improvement in the privacy-utility tradeoff as compared to standard privacy mechanisms.

**Peer review and privacy:** An exception is the concurrent work (Jecmen et al. 2020) which considers releasing the reviewer-paper similarity matrix and source code for the reviewer assignment (whereas in contrast we consider releasing a function of the scores given by reviewers to papers). Their approach involves modifying and randomizing the reviewer-paper assignment process and their guarantees pertain to plausible deniability (that is, any reviewer may be assigned to any paper with a probability at most a certain value). On the other hand, we do not modify the peer-review process in any way, and instead use any privacy-preserving

the number of reviewers per paper are both constants (Shah et al. 2018).

data-release mechanism coupled with post processing of the data from peer review.

## 3 Background and problem setting

In this section, we provide some background on the peer review setting and privacy, and describe our problem setting in more detail.

### 3.1 Peer review

We consider a conference peer review setting, where there are $n$ reviewers and $m$ papers. We index the papers as $[m] = \{1, 2, \cdots, m\}$ and the reviewers as $[n] = \{1, 2, \cdots, n\}$.[2] For simplicity we assume that the number of papers reviewed by each reviewer is the same for all reviewers – denoted as $\ell$, and that the number of reviewers reviewing each paper is the same for all papers – denoted as $k$.[3] Consequently, we have the relation $n\ell = mk$. All four parameters $(n, m, \ell, k)$ are public knowledge.

Each review comprises a real-valued score. We assume that all papers and all associated reviews (that is, the set of scores received by each paper) are public knowledge (e.g., in conferences such as ICLR and others on the OpenReview.net review platform). The list of all reviewers is also available publicly (such a list is released by many conferences). However, importantly, the identity of which reviewer reviewed which paper is private.

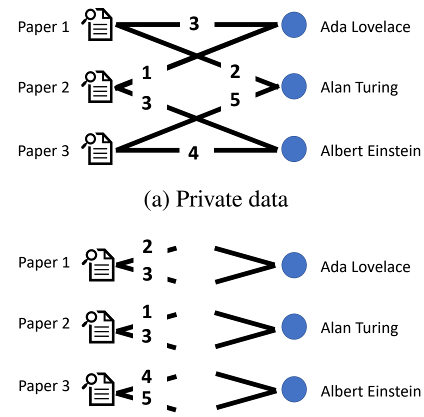
We now introduce notation to describe the score given in any review. If reviewer $j \in [n]$ reviews paper $i \in [m]$, then we use $s_{ij} \in \mathbb{R}$ to denote the score of this review. This score is private in the sense that the identity of the reviewer who gives this score is not publicly available. However, for each paper $i \in [m]$, the multiset $\{s_{ij} |$ reviewer $j \in [n]$ reviews paper $i\}$ is public.

This setting can be described by a bipartite graph, as shown in Figure 1. The bipartite graph has two disjoint sets of vertices, $[m]$ and $[n]$ representing the sets of papers and reviewers, repectively. In private data (Figure 1a), an edge exists between any vertex (paper) $i \in [m]$ and any vertex (reviewer) $j \in [n]$ if reviewer $j \in [n]$ reviews paper $i \in [m]$. We associate each edge $(i, j)$ with the score $s_{ij}$. The edges (and their associated scores) are all private. The private data is accessible to the program chairs of the conference. In public data (Figure 1b), for each vertex (paper) in $[m]$, the weights of the edges connected to it are known publicly. However, the edges of the graph are not known. Note that in both public and private data, identities of papers and reviewers are known.

There are various quantities of interest for release that we consider in this work. An intermediate set of terms towards these quantities is the multiset $\{w_{ij} \mid$ reviewer $j \in [n]$ reviews paper $i \in [m]\}$ discussed below, which we refer to as the set of weights. This multiset can be computed from the scores $\{s_{ij} |$ reviewer $j \in [n]$ reviews paper $i \in [m]\}$.

[2] We follow the standard convention of using $[\beta]$ to represent the set $\{1, 2, \ldots, \beta\}$ for any positive integer $\beta$.

[3] Our work is also applicable to the most general setting in which different reviewers and/or different papers have different loads. We discuss this in Section 4.3.




(a) Private data

(b) Public data

Figure 1: An illustration of the data (a) available privately to the program chairs of the conference, and (b) available to the public under increasingly popular 'open review' paradigms in computer science.

We now discuss three such choices of $\{w_{ij} |$ reviewer $j \in [n]$ reviews paper $i \in [m]\}$, and subsequently describe the data we aim to release.

- **Reviewer ratings.** In this case, the mapping from scores to weights is simply the identity mapping:

$$w_{ij} = s_{ij}. \tag{3.1}$$

- **Miscalibration.** Miscalibration pertains to the problem of strictness or leniency of different reviewers, which can govern the fate of papers (Roos, Rothe, and Scheuermann 2011; Ge, Welling, and Ghahramani 2013; Wang and Shah 2019). In order to understand the amount of miscalibration, it is instructive to see the difference between the scores given by a reviewer and the scores given by other reviewers for the same papers. To this end, we let $w_{ij}$ denote the miscalibration in any individual review (for any paper $i$ by any reviewer $j$):

$$w_{ij} = s_{ij} - \frac{1}{k-1} \sum_{j' \neq j} s_{ij'}. \tag{3.2}$$

- **Subjectivity.** Subjectivity is the problem that different reviewers place different emphasis on the various criteria when making an overall decision for a paper (Lee 2015). Techniques such as that proposed in Noothigattu, Shah, and Procaccia 2018 can be used to normalize each score in a manner that mitigates the subjectivity. Specifically, the technique in Noothigattu, Shah, and Procaccia 2018 uses the public data to transform the score $s_{ij}$ associated to each review into a normalized version, say, $\widetilde{s}_{ij}$. We can then set $w_{ij} = \widetilde{s}_{ij}$ for every review, and the algorithm in this paper will help release statistics of these normalized scores. A second use case we consider is to better understand and investigate the issue of subjectivity, by releasing the amount of subjectivity present in the system, that is, the aggregate difference of reviewers' scores and their normalized scores. Concretely in this case, after obtaining the normalized scores $\{\widetilde{s}_{ij} |$ reviewer $j \in [n]$ reviews paper $i \in [m]\}$, we set $w_{ij} = s_{ij} - \widetilde{s}_{ij}$ for every review.

Analogous to the scores $s_{ij}$'s, the weights $w_{ij}$'s are also associated to public and private components. We can use the same bipartite graphs as in Figure 1 to represent the setting with weights. In particular, the private data continues to include the edges pertaining to which reviewer reviewed which paper. The private data also includes the weight $w_{ij}$ on each edge $(i, j)$ representing the weight that reviewer $i \in [n]$ gives to paper $j \in [m]$. The private data is depicted in Figure 1a where in this interpretation, the values on the edges represent the weights. The public data, as in the case of scores, only includes the multiset of weights received by any paper, that is, the public data comprises the multisets $\{w_{ij} |$reviewer $j \in [n]$ reviews paper $i\}$ for every paper $i \in [m]$. The public data is depicted in Figure 1b where the values on the edges represent the weights.

It is very important to note the following two properties in the transformation of scores $s_{ij}$ to weights $w_{ij}$ for each of the aforementioned choices. First, clearly, given access to all private scores, all weights can be computed. Second, the public weights (that is, the multisets $\{w_{ij} |$reviewer $j \in [n]$ reviews paper $i\}$ for every paper $i \in [m]$) can be computed using only the publicly available score data (that is, the multisets $\{s_{ij} |$reviewer $j \in [n]$ reviews paper $i\}$ for every paper $i \in [m]$). This relation between the public (or private) weights and public (or private) scores allows us to interchange them in the graphs in Figure 1.

For each reviewer $j \in [n]$, let $\mathcal{Y}_j$ be the set of all papers reviewed by reviewer $j$, that is, $\mathcal{Y}_j = \{i \in [m] \mid$ reviewer $j$ reviews paper $i\}$. Let $y_j$ denote the mean weight of reviewer $j$:

$$y_j = \frac{1}{\ell} \sum_{i \in \mathcal{Y}_j} w_{ij}. \tag{3.3}$$

Note that since the identity of the reviewer in any review is private, the values of $\mathcal{Y}_j$ and $y_j$ in general cannot be computed from the public data.

**Quantity to be released:** The quantity of interest is the histogram of the mean weights per reviewer, represented by the *sorted* version of the mean-weight vector $(y_1, y_2, ..., y_n)$, which we denote by $\boldsymbol{\theta}^* = (\theta_1^*, \theta_2^*, ..., \theta_n^*)$. Then $\theta_1^* \leq ... \leq \theta_n^*$ and the multiset $(\theta_1^*, \theta_2^*, ..., \theta_n^*)$ equals the multiset $(y_1, y_2, ..., y_n)$. We call $\boldsymbol{\theta}^*$ the true sorted mean-weight vector. According to the applications discussed above, the vector $\boldsymbol{\theta}^*$ can either represent the mean scores per reviewer or capture the amount of miscalibration, or subjectivity in the reviews.

Our goal is to release $\boldsymbol{\theta}^*$, while ensuring privacy of reviewer identities. When the underlying weights are equal to the scores, the sorted mean-weight vector (that is, the histogram of scores) is commonly released by various conferences (Shah et al. 2018). These are however usually released without any privacy considerations, and our work addresses privacy-preserving release with high accuracy. Addressing the issues of miscalibration and subjectivity is extremely important for fair and high-quality peer review (Ge, Welling, and Ghahramani 2013; Wang and Shah 2019; Roos, Rothe, and Scheuermann 2011; Lee 2015; Noothigattu, Shah, and Procaccia 2018; Siegelman 1991; Kerr, Tolliver, and Petree 1977), and releasing the statistics pertaining to the amount of miscaliabraiton or subjectivity can considerably help both research and policy-design regarding these issues.

Publishing histograms of datasets in a privacy-preserving manner has been a central objective in the literature of privacy research (Chawla et al. 2005; Dwork et al. 2006a; Hay et al. 2010; Li et al. 2010; Bassily and Smith 2015; Balcer and Vadhan 2019). Histograms are typically the most frequently used statistics in official reports, and more importantly, they form the basis for more complicated statistical analysis. To the best of our knowledge, existing techniques for improving the privacy-utility tradeoff (for histograms or otherwise) are generally inadequate for the application of peer review since they do not take into account the special structures in peer-review data. Our goal is to use the specific type of publicly available data in peer review in order to improve the privacy-utility tradeoff.

Finally we note that the high-level ideas behind our proposed algorithm are more general and may also be used to improve the utility of the released data for settings beyond the sorted mean-weight vector. We revisit this point later in the paper.

## 3.2 Privacy

To protect the privacy of reviewers, we consider privacy-preserving mechanisms that (randomly) perturb the quantities of interest. By virtue of the random perturbation, the privacy mechanism makes it hard to infer each individual reviewer's scores given to papers from the noisy data. Specifically, we consider any privacy mechanism that releases a vector $\mathbf{r} = (r_1, r_2, ..., r_n) \in \mathbb{R}^n$ obtained by randomly perturbing the sorted mean-weight vector $(\theta_1^*, \theta_2^*, ..., \theta_n^*) \in \mathbb{R}^n$. An example of a privacy mechanism is the Laplace mechanism, which satisfies the popular notion of differential privacy (Dwork et al. 2016; Dwork and Roth 2014) in which $(r_1, r_2, ..., r_n) = (\theta_1^*, \theta_2^*, ..., \theta_n^*) + (\eta_1, \eta_2, ..., \eta_n)$. Here $\eta_1, \eta_2, \ldots, \eta_n$ are i.i.d. random variables drawn from a zero-mean Laplace distribution.

## 3.3 Utility (Accuracy)

Let $\boldsymbol{t} = (t_1, \cdots, t_n)$ denote the final output (after post-processing) that is released. We measure the utility or accuracy of the output in terms of its *mean squared error* with respect to the true value of the vector $\boldsymbol{\theta}^*$, that is, $\mathbb{E}\left[\sum_{i=1}^{n} (\theta_i^* - t_i)^2\right]$. We say that an (possibly random) output $\boldsymbol{t} = (t_1, ..., t_n)$ is more accurate than another output $\boldsymbol{t}' = (t_1', ..., t_n')$ with respect to $\boldsymbol{\theta}^*$ if

$$\mathbb{E}\left[\sum_{i=1}^{n} (\theta_i^* - t_i)^2\right] < \mathbb{E}\left[\sum_{i=1}^{n} (\theta_i^* - t_i')^2\right]. \tag{3.4}$$

## 3.4 Goal

Our goal is to design algorithms to process the data output by the privacy-preserving mechanism, $\mathbf{r}$, before its actual release in a manner that improves the privacy-utility tradeoff. Specifically, we aim to satisfy the four desiderata D1–D4 listed in Section 1.

# 4 Main Results

## 4.1 Approach

We first derive a representation of the set of all possible values in the sorted mean-weight vector $\boldsymbol{\theta}^*$ based on the public data. For any paper $i \in [m]$, we use $x_{i1}, x_{i2}, \cdots, x_{ik}$ to denote the $k$ weights on edges connected to that vertex (paper) in the public data, listed in an arbitrary order. Note that the second subscript of $x_{ij}$ does not correspond to a reviewer identity. The multiset $\{x_{11}, \cdots, x_{mk}\}$ is available publicly, and for each $i \in [m]$, the multiset $\{x_{i1}, \cdots, x_{ik}\}$ is identical to the multiset $\{w_{ij} | \text{reviewer } j \in [n] \text{ reviews paper } i\}$. Let $\mathcal{G}$ be a set of weighted bipartite graphs comprising all valid reviewer-paper weights based on the public data, that is, each member of $\mathcal{G}$ satisfies:

- It is a bipartite graph, with the vertices in the two parts corresponding to papers $[m]$ and reviewers $[n]$.
- All vertices in $[m]$ are $k$ regular and all vertices in $[n]$ are $\ell$ regular.
- The $k$ edges incident on any vertex $i \in [m]$ have weights $x_{i1}, x_{i2}, \ldots, x_{ik}$.

Furthermore, for any graph $g \in \mathcal{G}$, any paper $i$, and any reviewer $j$, we define $w_{ij}(g)$ equal to the weight of edge between $i$ and $j$ if this edge exists, and $w_{ij}(g) = 0$ otherwise. Then the set $\boldsymbol{\Theta}$, that comprises all possible values of the sorted mean-weight vector based on public data, is given by

$$\boldsymbol{\Theta} = \Big\{\boldsymbol{\theta} \in \mathbb{R}^n \mid \theta_1 \leq ... \leq \theta_n, \; \exists g \in \mathcal{G} \text{ such that } \theta_j = \frac{1}{\ell}\sum_{i=1}^{m} w_{ij}(g) \text{ for all } j \in [n]\Big\}. \quad (4.1)$$

Note that the true paper-reviewer graph is also a member of $\mathcal{G}$ and the true sorted weight vector $\boldsymbol{\theta}^* \in \boldsymbol{\Theta}$. Throughout this section, we consider algorithms based on projecting the noisy data on certain sets. To this end, for any set $\mathcal{C} \subseteq \mathbb{R}^n$, we define the projection of vector $\mathbf{r}$ on the set $\mathcal{C}$ as

$$\underset{\boldsymbol{\theta} \in \mathcal{C}}{\operatorname{argmin}} \sum_{i=1}^{n} (\theta_i - r_i)^2. \quad (4.2)$$

When the privacy-preserving algorithm perturbs the true sorted mean-weight vector $\boldsymbol{\theta}^*$, the resulting noisy mean-weight vector $\mathbf{r}$ may not lie in the set $\boldsymbol{\Theta}$. It is thus intuitive to instead replace the resulting vector with the vector in $\boldsymbol{\Theta}$ closest to it, that is, to instead output the projection (4.2) of the vector $\mathbf{r}$ with the choice $\mathcal{C} = \boldsymbol{\Theta}$.

The following result shows that this intuitive approach can actually *increase* the error. We state and prove this result concretely in the case of additive Laplace noise, but as seen in the proof, the result is much more general.

**Proposition 4.1.** *There exists a review setting such that the true sorted mean-weight vector $\boldsymbol{\theta}^*$, the noisy mean-weight vector $\boldsymbol{r}$ obtained by adding Laplace noise with zero mean and a fixed, non-zero variance to $\boldsymbol{\theta}^*$, and the output $\mathbf{t}$ of the projection* (4.2) *of $\boldsymbol{r}$ on the set $\mathcal{C} = \boldsymbol{\Theta}$, are related as*

$$\mathbb{E}\left[\sum_{i=1}^{n}(t_i - \theta_i^*)^2\right] > \mathbb{E}\left[\sum_{i=1}^{n}(r_i - \theta_i^*)^2\right], \quad (4.3)$$

*where the expectation is taken with respect to the noise distribution.*

The proposition implies that using the closest valid vector violates desideratum D1 of not reducing the accuracy. The proof of Proposition 4.1 is given in Appendix D.1.

Consequently, in order to ensure desideratum D1 of not reducing the accuracy is met, we project the noisy data onto a convex set that contains $\boldsymbol{\Theta}$. The following proposition (proved in Appendix D.2) indicates that such projection can never harm the accuracy, and is a straightforward application of the fact that projection on to convex sets is non-expansive. Note that projection methods have been used in the literature (Hay et al. 2010; Abowd et al. 2019) and it is known that projection does not decrease accuracy. For completeness, we include Proposition 4.2 here as the specific form of such results for our problem.

**Proposition 4.2.** *Consider any true sorted mean-weight vector $\boldsymbol{\theta}^*$, and any arbitrary (noisy mean-weight) vector $\boldsymbol{r}$. Let $\mathcal{C}$ be any closed convex set such that $\boldsymbol{\Theta} \subseteq \mathcal{C}$. Let $\boldsymbol{t} = (t_1, ..., t_n)$ be the projection of $\boldsymbol{r}$ on to set $\mathcal{C}$ as in* (4.2)*. Then it must be that*

$$\sum_{i=1}^{n}(t_i - \theta_i^*)^2 \leq \sum_{i=1}^{n}(r_i - \theta_i^*)^2. \quad (4.4)$$

Since proposition 4.2 holds for all $\mathbf{r}$, it follows that if $\mathbf{r}$ is obtained by randomly perturbing $\boldsymbol{\theta}^*$, then

$$\mathbb{E}\left[\sum_{i=1}^{n}(t_i - \theta_i^*)^2\right] \leq \mathbb{E}\left[\sum_{i=1}^{n}(r_i - \theta_i^*)^2\right]. \quad (4.5)$$

Moreover, for two closed convex sets $\mathcal{C}_1 \subseteq \mathcal{C}_2$ both containing $\boldsymbol{\Theta}$, if we have a projection on $\mathcal{C}_2$, then further projecting it on $\mathcal{C}_1$ can never increase the error and can possibly decrease the error. **Our goal thus is to project the noisy data on to a (small) convex set that contains all possible true values.**

## 4.2 NP-hardness of Projection onto Convex Hull

The smallest convex set that contains $\boldsymbol{\Theta}$ is the convex hull of $\boldsymbol{\Theta}$. Observe that if we could project on to the convex hull, then it can also be used to improve upon the projection on any other convex set. Specifically, if $\boldsymbol{t}$ is the projection of the perturbed data $\mathbf{r}$ on some convex set that contains $\boldsymbol{\Theta}$, and if $\boldsymbol{t}'$ is the projection of $\boldsymbol{t}$ on convex-hull($\boldsymbol{\Theta}$), then with an argument identical to that in Proposition 4.2 we have that $\sum_{i=1}^{n}(t_i' - \theta_i^*)^2 \leq \sum_{i=1}^{n}(r_i - \theta_i^*)^2$.

Consequently, in this section we consider the goal of projecting the noisy data onto the convex hull of $\boldsymbol{\Theta}$. In this case, the final result we will output can be represented as choosing $\mathcal{C} = \text{convex-hull}(\boldsymbol{\Theta})$ in Equation (4.2). Unfortunately, as we show below, projection onto convex-hull($\boldsymbol{\Theta}$) is NP-hard.

**Theorem 4.3.** *When $k = \ell > 2$, $m = n$ and $n$ is a multiple of $\ell$, the problem of projecting noisy data onto convex-hull($\boldsymbol{\Theta}$) is NP-hard.*

We prove this result via reducing the $\ell$-Partition problem to the projection problem. Given any instance of an $\ell$-Partition problem, which is a multiset of integers, we can construct a conference where each paper has a weight from the multiset. We can construct a vector such that the projection result can directly answer the $\ell$-Partition problem. The complete proof of Theorem 4.3 is provided in Appendix D.3.

### 4.3 An Efficient Algorithm

In this section, we present an algorithm that meets the four desiderata D1–D4 listed in Section 1. Since we cannot efficiently project on to the convex hull of $\boldsymbol{\Theta}$, we must make do with a larger convex set that contains $\boldsymbol{\Theta}$. We use desideratum D4 for guidance on what constitutes a reasonably small set and associated projection.

**Axioms defining desideratum D4** Recall that desideratum D4 says that the algorithm should automatically recover the ground truth when the structure of the public data is simple enough. More concretely, we benchmark any algorithm using the following axiomatic properties:

- A1 When all weights are identical, the projection should result in a vector whose entries are all the same as the weight. Formally, if $x_{ij} = z \ \forall i \in [m], j \in [k]$ for some $z$, then the output $\boldsymbol{t}$ of the algorithm must be $t_1 = t_2 = \cdots = t_n = z$.
- A2 When $\ell = 1$ (that is, each reviewer reviews 1 paper), the projection of any noisy data should result in a sorted vector of all weights. Formally, if $\ell = 1$ then the output $\boldsymbol{t}$ of the algorithm must be $(t_1, t_2, \cdots, t_n) =$ sorted$(x_{11}, x_{21}, \cdots, x_{n1})$.
- A3 When all but one papers have all zero weights, the projection of any noisy data should result in a sorted vector with $(n - k)$ zero entries and the remaining entries equal to $\frac{1}{\ell}$ of the weights for the paper that does not have all-zero weights. Formally, if $x_{ij} = 0 \ \forall i \in \{2, \ldots, m\}, j \in [k]$, then the output $\boldsymbol{t}$ of the algorithm must be $(t_1, t_2, \cdots, t_n) =$ sorted$(\frac{x_{11}}{\ell}, \frac{x_{12}}{\ell}, \cdots, \frac{x_{1k}}{\ell}, 0, \cdots, 0)$.

**High-level idea behind the algorithm** The main idea behind our algorithm comprises the following three steps:

- I. From the public data, take all tuples of size $\ell$ containing weights from different papers into consideration.
- II. Use them to construct lower and upper bounds on every entry of (the unknown vector) $\boldsymbol{\theta}^*$.
- III. Project the noisy data $\mathbf{r}$ on the set specified by the aforementioned lower and upper bounds, along with any other problem-specific (convex) constraints.

As one can intuitively see, these three steps imply a projection of the noisy data on a convex set which includes all valid values of the true data, and hence from Proposition 4.2 it will not reduce the utility (desideratum D1). Moreover, the entire algorithm uses only the public data along with the vector $\mathbf{r}$ released by the privacy mechanism, and hence does not compromise privacy (desideratum D2). The idea is general enough to be applied to many forms of the noisy data, and in what follows, we apply it to release the histogram of the true sorted mean-weight vector. Of course, the devil lies in the details of how these steps are executed, which will determine whether the designed algorithm meets desiderata D3 and D4.

**Full algorithm description** We now describe our algorithm in full detail. (We also provide an illustrative example in Appendix B.) Recall that we use $x_{i1}, x_{i2}, \cdots, x_{ik}$ to represent the $k$ weights on edges connected to any vertex (paper) $i \in [m]$ in the public data. Note that since reviewer identities are not available publicly, the second subscript "$j$" in "$x_{ij}$" has no particular meaning other than capturing the fact that each paper has $k$ weights. We use matrix $X$ to display all the weights in the public data where row $i$ column $j$ of $X$ has value $x_{ij}$.

**I. Valid weight tuples** We define a weight tuple as a multiset of $\ell$ real values. We say that a tuple is a *valid weight tuple* if it consists of $\ell$ weights from distinct papers. In other words, a valid weight tuple contains $\ell$ entries of matrix $X$ where no two entries are from the same row in $X$. We compute $\Omega'$ as the list of all valid weight tuples. In other words, $\Omega'$ contains all the possible weight tuples given by a reviewer. We sort the list $\Omega'$ based on the mean weight of the weight tuples (breaking ties arbitrarily), and henceforth use the notation $\Omega$ for this sorted list.

**II. Lower and upper bounds** We now compute lower and upper bounds on each entry of $\boldsymbol{\theta}^*$ based only on the public data. We create a graph $G$ which has all weight tuples in $\Omega$ as its vertices. Since each weight tuple in $\Omega$ corresponds to a vertex in graph $G$, we use the terms "weight tuples in $\Omega$" and "vertices in $G$" interchangeably. There is an edge between two vertices if the two weight tuples do not contain weights corresponding to the same entry in $X$. Then for each vertex, we define its left chain and right chain as follows. Recall that $\Omega$ is a sorted list and all of its entries, which are weight tuples, are totally ordered. We use the indices of the tuples (vertices) in this ordering for the following definitions.

**Definition 4.4.** For any vertex $\nu$ in $G$, a left chain (resp. right chain) of $\nu$ is a simple path in $G$ from $\nu$ to another vertex such that the indices of the vertices in this path decrease (resp. increase) starting from $\nu$.

We also define the *length of a chain* to be the number of vertices in the chain. For each vertex, we compute the length of its longest left chain and the length of its longest right chain using dynamic programming. To compute the length of the longest left chain of a vertex $\nu$ in $G$, we check the length of the longest left chain of all its neighbors at lower indices. Then the length of the longest left chain of $\nu$ is the maximum of these neighbors' longest left chain lengths plus one. Similarly, to compute the length of the longest right chain of $\nu$, we check the length of the longest right chain of all its neighbors at higher indices. The length of the longest right chain of $\nu$ is the maximum of its neighbors' longest right chain lengths plus one. We store the length of the longest left and right chain of each vertex for subsequent use in the algorithm.

The algorithm to compute a lower bound on $\theta^*_i$ for each

**Algorithm 1:** Computation of lower bounds

**Input:** matrix $X$ of weights, sorted list of weight tuples $\Omega$
**Initialize** $i = 1$, set $w \in \mathbb{R}^\ell$ as the first tuple in $\Omega$, and all entries of $X$ are unmarked.
**while** $i \leq n$ **do**
  for each weight in the tuple $w$, find its corresponding entry in matrix $X$ and mark the entry
  **if** length of tuple $w$'s longest left chain $\geq i$ **and** number of unmarked entries on each row of $X \leq n - i$ **then**
    lower bound on $\theta_i^*$ = mean of all entries of tuple $w$
    $i += 1$
  **end if**
  set $w$ as the next tuple in $\Omega$
**end while**

$i \in [n]$ is presented in Algorithm 1. In more detail, the algorithm uses two criteria to determine if mean of a weight tuple is a lower bound on $\theta_i^*$. The criteria are

C1 The longest left chain of the tuple has length at least $i$.
C2 In $X$, after we mark the $\ell$ weights from each tuple considered so far, each row has at most $n - i$ unmarked entries.

The intuition is as follows. We call the $n$ weight tuples that compute $\boldsymbol{\theta}^*$ the true weight tuples. The true weight tuple with mean $\theta_i^*$ must have $i-1$ weight tuples with smaller or equal mean to $\theta_i^*$. No two reviewers give the same weight so no two true weight tuples contain weights from the same entry in $X$. Thus, criterion C1 is a necessary condition for a weight tuple to be the true weight tuple that computes $\theta_i^*$. In addition, there are $n - i$ entries in $\boldsymbol{\theta}^*$ whose values are no smaller than $\theta_i^*$. Since no two true weight tuples contain weights from the same paper, each paper can have at most $n-i$ unused weights. Therefore, each row of $X$ cannot have more than $n-i$ unmarked entries. Thus, criterion C2 is necessary for all weights to be assigned among the reviewers. For each entry $i \in [n]$, we choose the valid weight tuple with the smallest mean that satisfies criteria C1 and C2 as the lower bound on $\theta_i^*$. Hence, it is a valid lower bound.

The computation of the upper bounds is analogous to that of lower bounds, and is presented in Appendix A in detail.

**III. Projection** Let $L_i$ denote the lower bound we compute on $\theta_i^*$ and $U_i$ denote the upper bound we compute on $\theta_i^*$ in part II above. The final output of our algorithm is the solution to the following optimization problem:

$$\underset{\boldsymbol{t} \in \mathbb{R}^n}{\operatorname{argmin}} \sum_{i=1}^{n} (\eta_i - t_i)^2 \text{ such that } L_i \leq t_i \leq U_i \forall i \in [n], \quad \sum_{i=1}^{n} t_i = \frac{1}{\ell} \sum_{i=1}^{m} \sum_{j=1}^{k} x_{ij}, t_1 \leq t_2 \leq \cdots \leq t_n. \tag{4.6}$$

This convex optimization problem with a quadratic objective and $2n$ linear constraints can be solved efficiently.

*Remark* 4.5 (Extension to non-uniform paper and reviewer loads). Our algorithm easily extends to the setting of non-uniform reviewer and paper loads. First, if the papers are reviewed by different number of reviewers, the algorithm above continues to work. Now if the reviewers review different numbers of papers, then we make the following modification to the algorithm. Let $\mathcal{L} \subset [m]$ denote the set of all paper loads on the reviewers, that is, $\ell \in \mathcal{L} \iff$ there is a reviewer who reviews exactly $\ell$ papers. Then the set $\Omega'$ computed in the first step of the algorithm includes all weight tuples of size $\ell$ for every $\ell \in \mathcal{L}$. The remainder of the algorithm remains identical to that described above. The proof of correctness in these settings follows from the same arguments (given in Appendix D.5) as those for the setting of uniform reviewer and paper loads. The algorithm continues to have a computational complexity that is polynomial in $n$ and $m$ (where we continue to assume that the maximum reviewer and paper loads are constants (Shah et al. 2018)).

**Guarantees of Our Algorithm** In this section, we evaluate our algorithm with respect to the four desiderata listed in Section 1. We first prove the correctness of the algorithm in terms of projection on to an appropriate set.

**Theorem 4.6.** *The algorithm projects noisy data onto a convex set that contains all true values.*

The proof of this theorem is given in Appendix D.5. This result, combined with Theorem 4.2, guarantees that our algorithm does not increase the error. Thus, our algorithm satisfies desideratum D1. In addition, since our algorithm uses only the public data for post processing, it satisfies desideratum D2. We now discuss the computational complexity of our algorithm.

**Theorem 4.7.** *The algorithm has polynomial time complexity in the number of reviewers and the number of papers.*

Our algorithm thus satisfies desideratum D3. The proof of this theorem is given in Appendix D.6. To be clear, while the algorithm is polynomial time in $n$ and $m$, it is exponential in $\ell$. In practice $\ell$ is usually a small constant (Shah et al. 2018).

We finally visit desideratum D4 – of returning an exact answer when it can easily be deduced from public data.

**Theorem 4.8.** *The algorithm satisfies the axiomatic properties A1, A2 and A3 defined in Section 4.3.*

The proof of this theorem is given in Appendix D.7. We have thus shown that our proposed algorithm meets all four desiderata D1–D4.

# 5 Conclusion

We take the first steps towards designing methods for privacy-preserving release of peer-review data, and posit the need for much more research on this topic to address the important challenge of improving peer review. While we addressed a certain type of peer-review data, it is of theoretical and practical interest to enable privacy-preserving release of more peer-review data such as properties of the reviewer graph, reviewer bids, and other functions of the scores. Moreover, it is of interest to design methods that can utilize data from multiple conferences, while preserving the privacy in each conference, for improving the peer-review process in any subsequent conference.

# References


Abowd, J.; Kifer, D.; Moran, B.; Ashmead, R.; Leclerc, P.; Sexton, W.; Garfinkel, S.; and Machanavajjhala, A. 2019. Census TopDown: Differentially Private Data, Incremental Schemas, and Consistency with Public Knowledge. US Census Bureau. https://github.com/uscensusbureau/census2020-das-e2e/blob/master/doc/20190711_0945_Consistency_for_Large_Scale_Differentially_Private_Histograms.pdf.

Babel, L.; Kellerer, H.; and Kotov, V. 1998. The k-partitioning problem. *Mathematical Methods of Operations Research* 47(1): 59–82.

Balcer, V.; and Vadhan, S. 2019. Differential Privacy on Finite Computers. *Journal of Privacy and Confidentiality* 9(2). doi:10.29012/jpc.679.

Balietti, S. 2016. Science is suffering because of peer review's big problems. *The Conversation* .

Balietti, S.; Goldstone, R. L.; and Helbing, D. 2016. Peer review and competition in the Art Exhibition Game. *Proceedings of the National Academy of Sciences* 113(30): 8414–8419.

Bassily, R.; and Smith, A. 2015. Local, Private, Efficient Protocols for Succinct Histograms. In *Proc. Ann. ACM Symp. Theory of Computing (STOC)*, 127–135. Portland, OR.

Bauschke, H. H.; Combettes, P. L.; et al. 2011. *Convex analysis and monotone operator theory in Hilbert spaces*, volume 408. Springer.

Bennett, K. E.; Jagsi, R.; and Zietman, A. 2018. Radiation oncology authors and reviewers prefer double-blind peer review. *Proceedings of the National Academy of Sciences* 115(9): E1940–E1940.

Bernard, C. 2018. Gender Bias in Publishing: Double-Blind Reviewing as a Solution? *Eneuro* 5(3).

Blum, A.; Ligett, K.; and Roth, A. 2008. A learning theory approach to non-interactive database privacy. In *Proc. Ann. ACM Symp. Theory of Computing (STOC)*, 609–618. Victoria, Canada.

Bun, M.; Ullman, J.; and Vadhan, S. 2018. Fingerprinting Codes and the Price of Approximate Differential Privacy. *SIAM Journal on Computing* 47(5): 1888–1938. doi:10.1137/15M1033587. URL https://doi.org/10.1137/15M1033587.

Cabanac, G.; and Preuss, T. 2013. Capitalizing on order effects in the bids of peer-reviewed conferences to secure reviews by expert referees. *Journal of the Association for Information Science and Technology* 64(2): 405–415.

Charlin, L.; and Zemel, R. S. 2013. The Toronto Paper Matching System: An automated paper-reviewer assignment system. In *ICML Workshop on Peer Reviewing and Publishing Models*.

Chawla, S.; Dwork, C.; McSherry, F.; and Talwar, K. 2005. On Privacy-Preserving Histograms. In *Conf. Uncertainty in Artificial Intelligence (UAI)*, 120–127.

Douceur, J. R. 2009. Paper rating vs. paper ranking. *ACM SIGOPS Operating Systems Review* 43(2): 117–121.

Dwork, C. 2006. Differential privacy. In *Proc. Int. Conf. Automata, Languages and Programming (ICALP)*, 1–12. Venice, Italy.

Dwork, C.; Kenthapadi, K.; McSherry, F.; Mironov, I.; and Naor, M. 2006a. Our data, ourselves: privacy via distributed noise generation. In *Proc. Annu. Int. Conf. Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, 486–503. St. Petersburg, Russia.

Dwork, C.; McSherry, F.; Nissim, K.; and Smith, A. 2006b. Calibrating noise to sensitivity in private data analysis. In *Proc. Conf. Theory of Cryptography (TCC)*, 265–284. New York, NY.

Dwork, C.; McSherry, F.; Nissim, K.; and Smith, A. 2016. Calibrating noise to sensitivity in private data analysis. *Journal of Privacy and Confidentiality* 7(3): 17–51.

Dwork, C.; and Roth, A. 2014. The Algorithmic Foundations of Differential Privacy. *Found. Trends Theor. Comput. Sci.* 9(3–4): 211–407.

Fiez, T.; Shah, N.; and Ratliff, L. 2019. A SUPER* Algorithm to Optimize Paper Bidding in Peer Review. In *ICML workshop on Real-world Sequential Decision Making: Reinforcement Learning And Beyond*.

Gaboardi, M.; Arias, E. J. G.; Hsu, J.; Roth, A.; and Wu, Z. S. 2014. Dual Query: Practical Private Query Release for High Dimensional Data. In *Int. Conf. Machine Learning (ICML)*. Beijing, China.

Ge, H.; Welling, M.; and Ghahramani, Z. 2013. A Bayesian model for calibrating conference review scores. URL {http://mlg.eng.cam.ac.uk/hong/nipsrevcal.pdf}.

Goldsmith, J.; and Sloan, R. H. 2007. The AI conference paper assignment problem. In *Proc. AAAI Workshop on Preference Handling for Artificial Intelligence, Vancouver*, 53–57.

Hay, M.; Rastogi, V.; Miklau, G.; and Suciu, D. 2010. Boosting the accuracy of differentially private histograms through consistency. *Proceedings of the VLDB Endowment* 3(1-2): 1021–1032.

Jecmen, S.; Zhang, H.; Liu, R.; Shah, N. B.; Conitzer, V.; and Fang, F. 2020. Mitigating Manipulation in Peer Review via Randomized Reviewer Assignments. *arXiv* .

Kerr, S.; Tolliver, J.; and Petree, D. 1977. Manuscript characteristics which influence acceptance for management and social science journals. *Academy of Management Journal* 20(1): 132–141.

Kliewer, M. A.; DeLong, D. M.; Freed, K.; Jenkins, C. B.; Paulson, E. K.; and Provenzale, J. M. 2004. Peer review at the American Journal of Roentgenology: How reviewer and manuscript characteristics affected editorial decisions on 196 major papers. *American Journal of Roentgenology* 183(6): 1545–1550.

Kobren, A.; Saha, B.; and McCallum, A. 2019. Paper Matching with Local Fairness Constraints. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

Lawrence, N.; and Cortes, C. 2014. The NIPS Experiment. http://inverseprobability.com/2014/12/16/the-nips-experiment. [Online; accessed 11-June-2018].

Lee, C. J. 2015. Commensuration bias in peer review. *Philosophy of Science* 82(5): 1272–1283.

Li, C.; Hay, M.; Rastogi, V.; Miklau, G.; and McGregor, A. 2010. Optimizing Linear Counting Queries Under Differential Privacy. In *Symp. Principles Database Systems (PODS)*, 123–134. Indianapolis, IN.

Lindsey, D. 1988. Assessing precision in the manuscript review process: A little better than a dice roll. *Scientometrics* 14(1-2): 75–82.

Mavrogenis, A. F.; Quaile, A.; and Scarlat, M. M. 2020. The good, the bad and the rude peer-review.

McCook, A. 2006. Is peer review broken? Submissions are up, reviewers are overtaxed, and authors are lodging complaint after complaint about the process at top-tier journals. What's wrong with peer review? *The scientist* 20(2): 26–35.

Noothigattu, R.; Shah, N. B.; and Procaccia, A. D. 2018. Loss Functions, Axioms, and Peer Review. *arXiv preprint arXiv:1808.09057* .

Peters, D. P.; and Ceci, S. J. 1982. Peer-review practices of psychological journals: The fate of published articles. *Behavioral and Brain Sciences* 5(2): 187–255.

Reinhart, M. 2009. Peer review of grant applications in biology and medicine. Reliability, fairness, and validity. *Scientometrics* 81(3): 789–809.

Rennie, D. 2016. Make peer review scientific. *Nature* 535(7610): 31–34.

Roos, M.; Rothe, J.; and Scheuermann, B. 2011. How to Calibrate the Scores of Biased Reviewers by Quadratic Programming. In *AAAI Conference on Artificial Intelligence*.

Schroter, S.; Loder, E.; and Godlee, F. 2020. Research on peer review and biomedical publication.

Scott, W. A. 1974. Interreferee agreement on some characteristics of manuscripts submitted to the Journal of Personality and Social Psychology. *American Psychologist* 29(9): 698.

Shah, N. B.; Tabibian, B.; Muandet, K.; Guyon, I.; and Von Luxburg, U. 2018. Design and analysis of the NIPS 2016 review process. *The Journal of Machine Learning Research* 19(1): 1913–1946.

Siegelman, S. S. 1991. Assassins and zealots: variations in peer review. *Radiology* 178(3): 637–642.

Snodgrass, R. 2006. Single- Versus Double-Blind Reviewing: An Analysis of the Literature. *SIGMOD Record* 35: 8–21. doi:10.1145/1168092.1168094.

Squazzoni, F.; Ahrweiler, P.; Barros, T.; Bianchi, F.; Birukou, A.; Blom, H. J.; Bravo, G.; Cowley, S.; Dignum, V.; Dondio, P.; et al. 2020. Unlock ways to share data on peer review.

Stelmakh, I.; Shah, N.; and Singh, A. 2018. PeerReview4All: Fair and Accurate Reviewer Assignment in Peer Review. *arXiv preprint arxiv:1806.06237* .

Stelmakh, I.; Shah, N.; and Singh, A. 2019. On Testing for Biases in Peer Review. In *NeurIPS*.

Stelmakh, I.; Shah, N.; and Singh, A. 2020. Catch Me if I Can: Detecting Strategic Behaviour in Peer Assessment. *arXiv* .

Stelmakh, I.; Shah, N.; Singh, A.; and Daumé III, H. 2020. Prior and Prejudice: The Bias against Resubmissions in Conference Peer Review. *arXiv* .

Tomkins, A.; Zhang, M.; and Heavlin, W. D. 2017. Reviewer bias in single-versus double-blind peer review. *Proceedings of the National Academy of Sciences* 114(48): 12708–12713.

Wang, J.; and Shah, N. B. 2019. Your 2 is My 1, Your 3 is My 9: Handling Arbitrary Miscalibrations in Ratings. In *AAMAS*.

Wang, W.; Ying, L.; and Zhang, J. 2016. On the relation between identifiability, differential privacy, and mutual-information privacy. *IEEE Trans. Inf. Theory* 62(9): 5018–5029.

Welch, I. 2014. Referee recommendations. *The Review of Financial Studies* 27(9): 2773–2804.

Xu, Y.; Zhao, H.; Shi, X.; and Shah, N. 2019. On Strategyproof Conference Review. In *IJCAI*.

# Appendices

## A Algorithm for Upper Bounds

The computation of upper bounds is similar to the methodology in Algorithm 1 from Section 4.3 and is presented in Algorithm 2. The two criteria we use to determine if mean of a tuple is an upper bound on $\theta_i^*$ are

C3 The longest right chain of the tuple has length at least $n - i + 1$.

C4 In $X$, after we mark the $\ell$ weights from each tuple considered so far, each row has at most $i - 1$ unmarked entries.

**Algorithm 2:** Computation of upper bounds

**Input:** matrix $X$ of weights, sorted list of weight tuples $\Omega$
**Initialize** $i = 1$, set $w \in \mathbb{R}^\ell$ as the first tuple in $\Omega$, and all entries of $X$ are unmarked.
**while** $i \geq 1$ **do**
  for each weight in the tuple $w$, find its corresponding entry in matrix $X$ and mark the entry
  **if** length of tuple $w$'s longest right chain $\geq n - i + 1$ **and** number of unmarked entries on each row of $X \leq i - 1$ **then**
    upper bound on $\theta_i^*$ = mean of all entries of tuple $w$
    $i- = 1$
  **end if**
  set $w$ as the previous tuple in $\Omega$
**end while**

## B An Example

In this section, we illustrate our algorithm (described in Section 4.3) by means of an illustrative example.

Consider a case where $n = m = 4$, $\ell = k = 3$. Let 3 papers among the 4 have all 0 weights and the fourth paper has weights 1, 2 and 3. In this example, we can infer that $(\theta_1^*, \theta_2^*, ..., \theta_n^*) = (0, \frac{1}{3}, \frac{2}{3}, 1)$ regardless of assignment. This example reflects axiomatic property A3 presented in Section 4.3. We show that our algorithm for computing bounds indeed results in a convex set that contains only the vector $(0, \frac{1}{3}, \frac{2}{3}, 1)$. And thus the projection of any noisy data onto this convex set results in $(0, \frac{1}{3}, \frac{2}{3}, 1)$.

First, we visualize the matrix $X$ as

$$\begin{array}{llll} \text{paper 1}: & 0_{11} & 0_{12} & 0_{13} \\ \text{paper 2}: & 0_{21} & 0_{22} & 0_{23} \\ \text{paper 3}: & 0_{31} & 0_{32} & 0_{33} \\ \text{paper 4}: & 1_{41} & 2_{42} & 3_{43}. \end{array}$$

The subscripts indicate the entries of the weights in $X$. Some elements in $\Omega'$ are: $(0_{11}, 0_{21}, 0_{31})$, $\cdots, (0_{13}, 0_{23}, 0_{33})$, $(0_{11}, 0_{12}, 1_{41})$, $\cdots, (0_{13}, 0_{23}, 1_{41})$, $(0_{11}, 0_{12}, 2_{42}), \cdots, (0_{11}, 0_{12}, 3_{43})$. After we sort $\Omega'$ based on the mean of the weight tuples, we get $\Omega$ where the first 27 tuples have mean 0, followed by 27 tuples with mean $\frac{1}{3}$, 27 tuples with mean $\frac{2}{3}$, and 27 tuples with mean 1. We construct graph $G$ in which for instance, there is an edge between the tuples $(0_{11}, 0_{21}, 0_{31})$ and $(0_{12}, 0_{22}, 0_{32})$ since all six weights in these two tuples correspond to different entries in $X$. On the other hand, there is no edge between the tuples $(0_{11}, 0_{21}, 1_{41})$ and $(0_{12}, 0_{22}, 1_{41})$ because they both contain weight $1_{41}$.

The first tuple in $\Omega$ meets both the criteria so lower bound on $\theta_1^*$ is mean of the first tuple, which is a tuple with mean 0. Therefore, lower bound on $\theta_1^*$ is 0. Without loss of generality, the first tuple is $(0_{11}, 0_{21}, 0_{31})$ and we mark the corresponding entries in $X$. Now the matrix $X$ can be visualized as follows (where we mark any entry when we need to):

$$\begin{array}{llll} \text{paper 1}: & \cancel{0_{11}} & 0_{12} & 0_{13} \\ \text{paper 2}: & \cancel{0_{21}} & 0_{22} & 0_{23} \\ \text{paper 3}: & \cancel{0_{31}} & 0_{32} & 0_{33} \\ \text{paper 4}: & 1_{41} & 2_{42} & 3_{43}. \end{array}$$

To compute a lower bound on $\theta_2^*$, we start from the second tuple in $\Omega$. Since there are 27 tuples that have mean zero, the second tuple still has mean zero. However, we do not choose any tuple that has mean zero due to criterion C2 from the algorithm. Choosing any $(0, 0, 0)$ tuple leaves all 3 entries of row 4 in $X$ unmarked, and thus will not leave row 4 with at most 2 unmarked entries. Therefore, we will only stop at the first tuple that has mean $\frac{1}{3}$. Without loss of generality, we choose tuple $(0_{11}, 0_{21}, 1_{41})$ and mark the corresponding entries in $X$. This will leave the matrix $X$ as

$$\begin{array}{llll} \text{paper 1}: & \cancel{0_{11}} & \cancel{0_{12}} & \cancel{0_{13}} \\ \text{paper 2}: & \cancel{0_{21}} & \cancel{0_{22}} & \cancel{0_{23}} \\ \text{paper 3}: & \cancel{0_{31}} & \cancel{0_{32}} & \cancel{0_{33}} \\ \text{paper 4}: & \cancel{1_{41}} & 2_{42} & 3_{43}. \end{array}$$

For a similar reason, we do not choose any tuple that has mean $\frac{1}{3}$ to be a lower bound on $\theta_3^*$ as it would not leave row 4 of $X$ with at most 1 unmarked entry. So we choose the first tuple that has mean $\frac{2}{3}$ and a lower bound on $\theta_3^*$ is $\frac{2}{3}$. Lastly, a lower bound on $\theta_4^*$ is computed using the first tuple that has mean 1.

Now we can look at the computation of upper bounds using the proposed algorithm. Upper bound on $\theta_4^*$ is taken as mean of the last tuple, which is a tuple with mean 1. Therefore, an upper bound on $\theta_4^*$ is 1. In addition, we mark two entries of weight 0 and one entry of weight 3. Without loss of generality, we mark entries $0_{11}, 0_{21}, 3_{43}$. Now the matrix $X$ can be visualized as

$$\begin{array}{llll} \text{paper 1}: & \cancel{0_{11}} & 0_{12} & 0_{13} \\ \text{paper 2}: & \cancel{0_{21}} & 0_{22} & 0_{23} \\ \text{paper 3}: & 0_{31} & 0_{32} & 0_{33} \\ \text{paper 4}: & 1_{41} & 2_{42} & \cancel{3_{43}}. \end{array}$$

To compute an upper bound on $\theta_3^*$, we start from the second to last tuple in $\Omega$. Since there are 27 tuples that have mean 1, the second to last tuple still has mean 1. However, we do not choose any tuple with mean 1 due to criterion C3 from the algorithm. Any $(0, 0, 3)$ tuple does not have a right chain of length at most 2 because all tuples with value $(0, 0, 3)$ are not connected due to the uniqueness of the weight 3. Therefore, we will only stop at the first tuple that has mean $\frac{2}{3}$ as it has a right chain of length 2. Since we have encountered all combinations of $(0, 0, 3)$, the matrix $X$ after we choose a tuple $(0, 0, 2)$ becomes

$$
\begin{array}{llll}
\text{paper 1}: & \cancel{0_{11}} & \cancel{0_{12}} & \cancel{0_{13}} \\
\text{paper 2}: & \cancel{0_{21}} & \cancel{0_{22}} & \cancel{0_{23}} \\
\text{paper 3}: & \cancel{0_{31}} & \cancel{0_{32}} & \cancel{0_{33}} \\
\text{paper 4}: & 1_{41} & \cancel{2_{42}} & \cancel{3_{43}}
\end{array}
$$

For a similar reason, we do not choose any tuple that has mean $\frac{2}{3}$ to be an upper bound on $\theta_2^*$ as it would not have a right chain of length at least 3. So we choose the first tuple we encounter that has mean $\frac{1}{3}$ and an upper bound on $\theta_2^*$ is $\frac{1}{3}$. Lastly, an upper bound on $\theta_1^*$ is computed using the first tuple that has mean 0.

Thus, the bounds on $\boldsymbol{\theta}^* = (\theta_1^*, \theta_2^*, ..., \theta_n^*)$ are $0 \le \theta_1^* \le 0$, $\frac{1}{3} \le \theta_2^* \le \frac{1}{3}$, $\frac{2}{3} \le \theta_3^* \le \frac{2}{3}$ and $1 \le \theta_4^* \le 1$. Along with the conditions that $\theta_1^* + \theta_2^* + \theta_3^* + \theta_4^* = 2$ and $\theta_1^* \le \theta_2^* \le \theta_3^* \le \theta_4^*$, the only possible value of $\boldsymbol{\theta}^*$ is $(0, \frac{1}{3}, \frac{2}{3}, 1)$. Thus the output of our algorithm is the singleton set $\{(0, \frac{1}{3}, \frac{2}{3}, 1)\}$. Projection of any data to the convex set $\{(0, \frac{1}{3}, \frac{2}{3}, 1)\}$ results in $(0, \frac{1}{3}, \frac{2}{3}, 1)$, which is consistent with axiomatic property A3.

# C Simulations

In this section, we conduct synthetic simulations to evaluate the performance of our algorithm. We synthetically generate a conference review setting in one of several ways as described below. In each of the settings, the number of reviewers equals the number of papers, and each reviewer reviews 2 papers and each paper is reviewed by two reviewers. The assignment of reviewers to papers is done uniformly at random subject to given load constraints. The weight given by any reviewer to any reviewed paper is drawn from a beta distribution. For preserving privacy, we consider the common method of adding i.i.d. Laplace noise (with mean zero and variance 2) to each component of the true sorted mean-weight vector.

We evaluate the following three methods of releasing the sorted mean-weight vector, which includes our proposed algorithm and two baselines:

- **Noisy** where Laplace noise is added but no post-processing is performed;
- **Baseline projection** where the noisy data is post-processed via projecting onto a convex set which constrains the sum of all entries, the value of each entry in terms of the range of weights (0 to 1), and imposes a monotonicity constraint; We project on the set $\{\boldsymbol{t} \in \mathbb{R}^n | 0 \le t_i \le 1 \forall i \in [n],\ \sum_{i=1}^{n} t_i = \frac{1}{\ell} \sum_{i=1}^{m} \sum_{j=1}^{k} x_{ij},\ t_1 \le t_2 \le \cdots \le t_n\}$.
- **Our algorithm** where the noisy data is post-processed via our algorithm described in Section 4.

The simulations compute the mean squared error between the true sorted mean-weight vector $\boldsymbol{\theta}^*$ and the output from each of these three methods, that is, $\sum_{i=1}^{n} (t_i - \theta_i^*)^2$ where $\boldsymbol{t}$ is the output of any of these algorithms. Note that in the figures, the error bars (standard error of the mean) are plotted but not visible in most cases since they are too small.

We now describe the method for generating the weights in each simulation, and refer the reader to the corresponding plots. Note that the y-axes (representing the mean squared error) on each of the plots is on a logarithmic scale.

- In Figure 2a—2e, the number of reviewers ranges from 10 to 50. The weights are all i.i.d. and are generated from the beta distribution specified in the corresponding subcaption.
- In Figure 2f, the number of reviewers is fixed at 10. On the x-axis, we vary a parameter $a \in \{0.5, 1, \ldots, 10\}$. For each value of $a$, we draw all weights i.i.d. from the beta$(a, a)$ distribution.
- In Figure 2g, we again vary the number of reviewers $n$ on the x-axis. For any paper $i \in [n]$, one weight is generated from beta$(1, i)$ and the other weight is generated from beta$(2, i)$ independently.
- In Figure 2h, whenever any paper $i \in [m]$ is reviewed by reviewer $j \in [n]$, the weight of that review is generated from beta$(i, j)$.

All in all, these simulations reveal that our algorithm can lead to a multi-fold improvement in the utility (accuracy) while not compromising the privacy.

# D Proofs

We present proofs of all the claimed results.

## D.1 Proof of Proposition 4.1

We prove the proposition using a counter example. Assume the true value $\boldsymbol{\theta}^* = 0$ and the set of all possible values $\boldsymbol{\Theta} = \{-4, -2, 0, 2, 4\}$. The noisy data $\mathbf{r} = \boldsymbol{\theta}^* + \boldsymbol{\eta}$ where $\boldsymbol{\eta}$ is a Laplace random variable with probability density function $\eta(x) = 0.5e^{-|x|}$.

Without projection, the expected error incurred by the noise is $\int_{-\infty}^{\infty} 0.5e^{-|x|}x^2 dx = 2$. But if we project the noisy data on the set $\boldsymbol{\Theta}$ and get result $\boldsymbol{t}$, the expected error after the projection is computed as $16\int_{-\infty}^{-3} 0.5e^{-|x|}dx + 4\int_{-3}^{-1} 0.5e^{-|x|}dx + 4\int_{1}^{3} 0.5e^{-|x|}dx + 16\int_{3}^{\infty} 0.5e^{-|x|}dx = 2.06896$, which is greater than the expected error without projection. Thus, projecting on the set that contains all true values could decrease the accuracy of data.

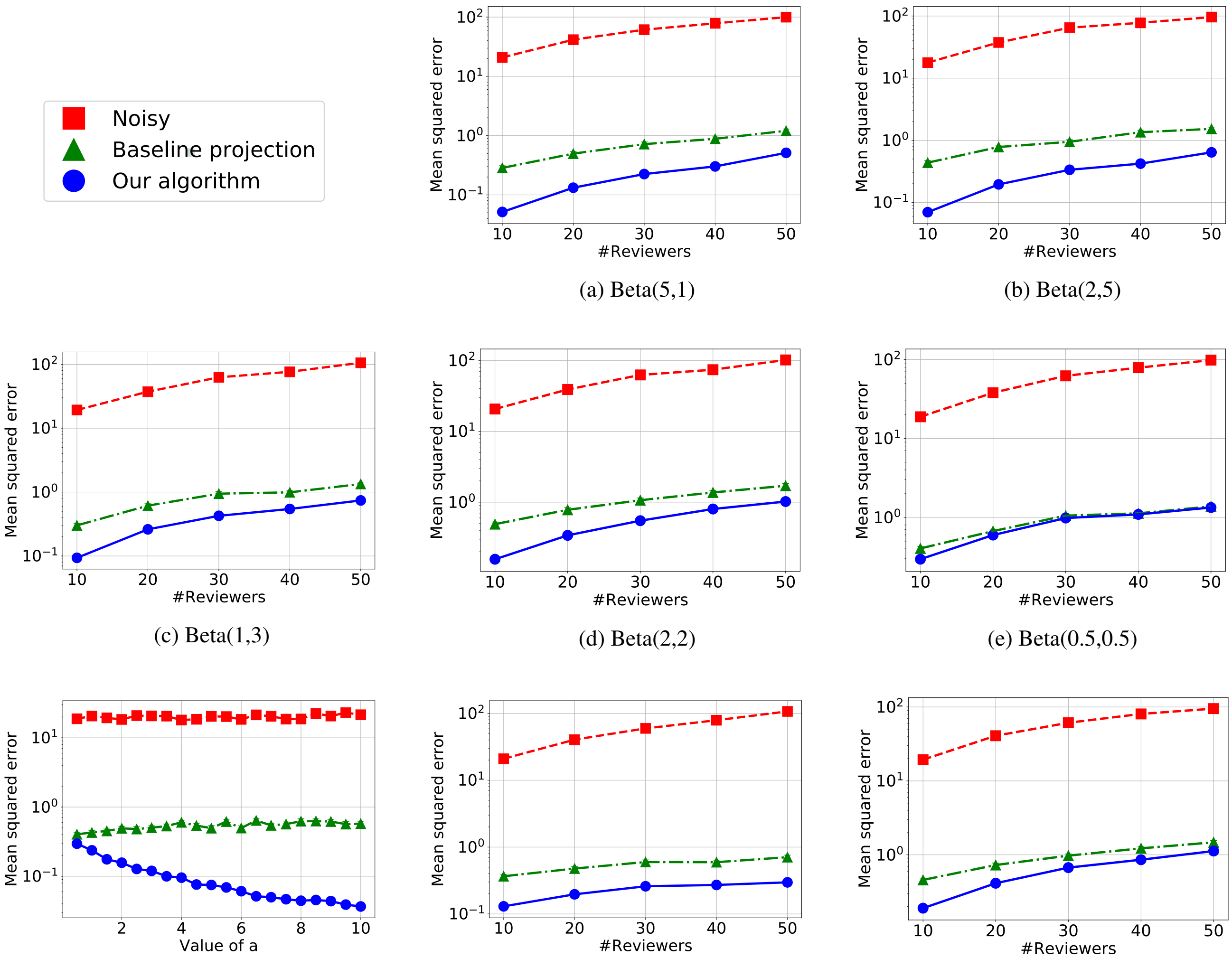
## D.2 Proof of Proposition 4.2

It is known that projection on a closed convex set is non-expansive (Bauschke, Combettes et al. 2011). Since $\boldsymbol{\theta}^*$ results from a valid assignment, it is contained in $\boldsymbol{\Theta}$. Therefore it is contained in any closed convex set that contains $\boldsymbol{\Theta}$. Projection of $\mathbf{r}$ onto any such convex set will not increase its squared error from $\boldsymbol{\theta}^*$. Therefore, proposition 4.2 holds.

## D.3 Proof of Theorem 4.3

We will prove the NP-hardness by reducing the $\ell$-Partition problem, which is NP-hard (Babel, Kellerer, and Kotov 1998), to the problem of projecting noisy data onto convex hull of $\boldsymbol{\Theta}$. The $\ell$-Partition problem where $\ell > 2$ is defined as follows.

**Definition D.1.** $\ell$-Partition problem: Given a multi-set $\mathcal{W} = \{w_1, w_2, ..., w_n\}$ of $n$ non-negative integers where $n$ is a multiple of $\ell$, decide if we can partition $\mathcal{W}$ into $\frac{n}{\ell}$ subsets such that each subset has size $\ell$ and the sums of all subsets are the same.




(a) Beta(5,1)

(b) Beta(2,5)

(c) Beta(1,3)

(d) Beta(2,2)

(e) Beta(0.5,0.5)

(f) Beta(a, a) with 10 reviewers

(g) Beta(a, b) with a ∈ [2] and b ∈ [n]

(h) Beta(a, b) with a, b ∈ [n]

Figure 2: Simulation results. The y-axes of all plots are on a logarithmic scale.

Consider any instance of the $\ell$-Partition problem with $\mathcal{W} = \{w_1, w_2, ..., w_n\}$, where $w_i \geq 0$ and $n$ is a multiple of $\ell$. Now we construct a peer-review dataset where there are $n$ reviewers and $n$ papers, each reviewer reviews $\ell$ papers and each paper receives $\ell$ reviews. Note that the number of reviewers is the same as the number of elements in $\mathcal{W}$. Let each paper $i$ has weight $w_i$ and $\ell - 1$ zero weights, and $a$ denote the average of all elements in $\mathcal{W}$, i.e.,

$$a = \frac{1}{n}\sum_{i=1}^{n} w_i. \tag{D.1}$$

Let $v = (0, ..., 0, a, ..., a)$ be a vector of $n$ entries whose last $\frac{n}{\ell}$ entries all have value $a$. Let $\mathcal{V} = \mathcal{W} \cup \{0, ..., 0\}$ be the multiset containing all values of $\mathcal{W}$ and $n \cdot (\ell - 1)$ zeros. Then the projection problem is to project $v$ onto the convex hull of $\mathbf{\Theta}$ defined for this peer-review dataset.

The reduction from the $\ell$-Partition problem to the projection problem constructed above is as follows. If the solution to the projection problem is $v$ itself, we return True for the $\ell$-Partition problem; otherwise we return False.

We first prove the correctness of the reduction. Suppose $\mathcal{W}$ can be $\ell$-partitioned into $\frac{n}{\ell}$ subsets of equal sums. Then we can partition $\mathcal{V}$ into subsets of size $\ell$ where these subsets are the subsets that give the $\ell$-partition of $\mathcal{W}$ and subsets that consist of $\ell$ zeros. By Lemma D.2 below, this partition of $\mathcal{V}$ gives a valid assignment for the peer-review problem, and thus $v = (0, ..., 0, a, ..., a)$ corresponds to a valid assignment. Therefore, the projection of $v$ is itself. The proof of Lemma D.2 is presented in Section D.4.

**Lemma D.2.** *In the setting described above, any $\ell$-partition of the $n \cdot \ell$ weights in $\mathcal{V}$, i.e., any partition of $\mathcal{V}$ into subsets of size $\ell$, can be interpreted as a valid assignment such that subset $i$ corresponds to the weights from reviewer $i$ given to $\ell$ distinct papers.*

Next, suppose that the projection of $v$ is itself. We show that $\mathcal{W}$ can be $\ell$-partitioned into subsets of equal sums. We first claim that $v$ must correspond to a valid assignment itself. To see this, suppose $v = (0, ..., 0, a, ..., a)$ is a convex combination of some sorted mean weight vectors. Then these vectors must all have value $a$ for their last $\frac{n}{\ell}$ entries since each of these mean weight vector is sorted. Due to the sum constraint, these mean weight vectors have to be $(0, ..., 0, a, ..., a)$. Next we note that in the assignment given by $v$, each reviewer who has an average weight of $a$ must give $\ell$ weights with values from $\mathcal{W}$ due to the pigeonhole principle. Therefore, the $\frac{n}{\ell}$ subsets each of which consists of weights given by one of the last $\frac{n}{\ell}$ reviewers form an $\ell$-partition of $\mathcal{W}$ with equal sums.

Finally, we prove the efficiency of the reduction. Since the construction of $v$ has $\mathcal{O}(n)$ time complexity and the construction of $\mathcal{V}$ has $\mathcal{O}(1)$ time complexity, the reduction has $\mathcal{O}(n)$ time complexity, which is polynomial in the size of the input. Thus, the reduction can be done efficiently, which completes the proof.

## D.4 Proof of Lemma D.2

Fix $\ell$, we will prove the lemma by induction on $n$, the number of reviewers, which is the same as the number of papers.

Base case: when $n = \ell$, every reviewer reviews all papers, so any $\ell$ partition of the weights can be validly assigned to reviewers.

Inductive hypothesis: suppose when there are fewer than $n$ reviewers for an $n > \ell$, every $\ell$ partition of the weights forms a valid assignment for reviewers.

Consider when there are $n$ reviewers and $n$ papers. Without loss of generality, assume all weights in $\mathcal{W}$ are non-zero. Consider an $\ell$ partition of the set $\mathcal{V}$. We will argue in two cases based on whether there is a subset that contains exactly one non-zero weight.

1. Case 1: In the partition, if there is a subset with exactly one non-zero weight.

   Without loss of generality, assume that the subset with exactly one non-zero weight contains $w_1$ and the subset is $\{w_1, 0, ..., 0\}$. We denote the subset $\mathcal{S}_1$. In $\mathcal{S}_1$, there are $\ell - 1$ zero weights and a non-zero weight $w_1$ from $\mathcal{W}$. Since paper 1 receives $\ell$ weights in total, we can remove $\mathcal{S}_1$, paper 1 and reviewer 1.

   Now we are left with $n - 1$ reviewers and papers. The removal does not affect the number of reviews received by the rest of the papers. We still have each paper getting $\ell$ weights. Among the weights, there is one non-zero weight from $\mathcal{W}$ and $\ell - 1$ zero weights. By the inductive hypothesis, the rest of the subsets in the partition form a valid assignment of $\mathcal{V} \setminus \mathcal{S}_1$. We can assign the weights to $n - 1$ reviewers.

   We then add $\mathcal{S}_1$ back to the assignment. Since reviewer 1 $\ell$ weights to paper 1, it is not valid. We can solve this by swapping the zero weights in $\mathcal{S}_1$ with zeros in other subsets. We need to make $\ell - 1$ swaps. We label the rest of the subsets $\mathcal{S}_2, \dots, \mathcal{S}_n$ where $\mathcal{S}_2$ is the subset that contains most non-zero weights and the labels go in decreasing order based on the number of non-zero weights contained in a subset. We look at the rest of the subsets in the order of their labels.

   Since none of the rest of the subsets contain any weight from paper 1, swapping a zero weight from paper 1 into any of these subsets will nor affect the validity of the subset. There are at least $n - 1 - \frac{n-1}{\ell}$ subsets that contain at least a zero weight. Since $n > \ell$ and $\ell > 2$, $n - 1 - \frac{n-1}{\ell} = \frac{(\ell-1)(n-1)}{\ell} \geq \ell - 1$. Thus, we have enough subsets to swap the zero weights from paper 1 in. Then we make sure the zero weights swapped into $\mathcal{S}_1$ will not come from the same paper. We label the zeros in $\mathcal{S}_1$ with index $1, \dots, \ell - 1$. Suppose there are no subsets that do not contain any zero weights. Then when we need to swap out the zero weight at index $i$ in $\mathcal{S}_1$, there are at most $n - i$ non-zero weights in the untouched subsets due to the order we look at the subsets. There are $n - i$ untouched subsets as well. Then there exists an untouched subset that contains $i$ zero weights. Since at this stage $\mathcal{S}_1$ has already completed $i - 1$ swaps, we can find a zero weight from the untouched subset to swap so that the zero weight does not come from the same paper as the zero weights from previous swaps. Note that if we have any subset that does not contain any zero weight or we skip some subsets due to conflict of papers, then the fraction of non-zero weights left and untouched subsets will be even smaller. So we are guaranteed to find a proper zero weight to swap. Thus, we can make $\ell - 1$ swaps of the zero weights to $\mathcal{S}_1$ and make all subsets valid assignments of weights. Such swaps do not affect the values in each subset.

   Therefore, such partition can result in a valid assignment of the $n \cdot \ell$ scores among $n$ reviewers.

2. Case 2: In the partition, if there are no subsets with exactly one non-zero weight.

   Since $\mathcal{W}$ contains $n$ elements and there are $n$ subsets, by pigeon hole principle, there must be a subset $\mathcal{S}_1$ that contains all zero weights.

   Without loss of generality, we find the subset that contains $w_1$ and then swap $w_1$ with a zero weight in $\mathcal{S}_1$. This results in $\mathcal{S}_1' = \{w_1, 0, \dots, 0\}$.

   Now we have a subset that contains exactly 1 weight from $\mathcal{W}$. Like in case 1, we remove the subset, reviewer 1 and paper 1. We can find a valid assignment of the rest of the weights to $n - 1$ reviewers. Then we will put $\mathcal{S}_1'$ back to the assignment. Currently all weights in $\mathcal{S}_1'$ are from paper 1. We identify the subset where $w_1$ comes from, and swap $w_1$ back into the subset with a zero weight there. Since the subset can not contain any weights from paper 1, we can safely put $w_1$ back without having two weights from the same paper.

   After the swap, $\mathcal{S}_1'$ has $\ell - 1$ zero weights from paper 1 and a zero weight from a different paper, say paper 2. We need to make $\ell - 2$ swaps for the zeros in $\mathcal{S}_1'$. We label the rest of the subsets $\mathcal{S}_2, \dots, \mathcal{S}_n$ where $\mathcal{S}_2$ is the subset that contains most non-zero weights and the labels go in decreasing order based on the number of non-zero weights contained in a subset. We look at the rest of the subsets in the order of their labels.

   Since none of the rest of the subsets contain any weight from paper 1, swapping a zero weight from paper 1 into any of these subsets will nor affect the validity of the subset. In the worst case, there exists a subset that contains $w_1$ and there are at most $\frac{n-2}{\ell-1}$ subsets that only contains a zero weight from paper 2 because such tuples cannot contain $w_2$. Then there are at least $n-1-\frac{n-2}{\ell-1}-1$ subsets that we can swap the zero weights in $\mathcal{S}_1'$. Since $n > \ell$ and $\ell > 2$, $n-1-\frac{n-2}{\ell-1}-1 = \frac{(\ell-2)(n-2)}{\ell-1} \geq \ell-2$. Thus, we have enough subsets to swap the zero weights from paper 1 in.

We keep a zero weight from paper 1 in $\mathcal{S}_1$ and label the rest of the zero weights in $\mathcal{S}_1$ with index $1,\ldots,\ell-2$. Suppose there are no subsets that do not contain any zero weights. Then when we need to swap the zero weight at index $i$ in $\mathcal{S}_1$, there are at most $n-i$ non-zero weights in the untouched subsets due to the order we look at the subsets. There are $n-i$ untouched subsets as well. Then there exists a subset that contains $i+1$ zero weights. Since at this stage $\mathcal{S}_1$ has already completed $i-1$ swaps, we can find a zero weight to swap that does not conflict with the weights from previous swaps and not from paper 2 either. Note that if we have any subset that does not contain any zero weight or we skip some subsets due to conflict of papers, then the fraction of non-zero weights left and untouched subsets will be even smaller. So we are guaranteed to find a proper zero weight to swap. Thus, we can make $\ell-2$ swaps of the zero weights to $\mathcal{S}_1'$ and makes all subsets valid assignments of weights. Such swaps do not affect the value in each subset.

Therefore, such partition can result in a valid assignment of the $n \cdot \ell$ weights among $n$ reviewers.

In conclusion, any $\ell$-partition of $\mathcal{V}$ can be interpreted as a valid assignments of weights to $n$ reviewers.

## D.5 Proof of Theorem 4.6

We would like to show that the convex set contains $\boldsymbol{\Theta}$. We will show that the bounds are indeed lower and upper bounds on each entry.

We will first show that the lower bounds computed by the algorithm are correct.

Assume for the sake of contradiction, there exists an assignment such that $\theta_i^*$ is less than the lower bound on $\theta_i^*$ we computed, denoted as $\theta_i$. We use $\nu$ to denote the tuple that results in $\theta_i^*$ and use $\nu'$ to denote the tuple that we choose in the algorithm that has mean $\theta_i$. Since $\nu$ is a valid assignment, it is the sum of $\ell$ weights from $\ell$ distinct papers. Since $\Omega$ contains all such tuples, it contains $\nu$. And since $\theta_i^* < \theta_i$, we encountered $\nu$ before we encounter $\nu'$ in $\Omega$. We did not choose $\nu$ as the tuple for lower bound due to its violation of either criterion C1 or criterion C2.

If $\nu$ violates criterion C1, it does not have a left chain of size at least $i$. There cannot be $i-1$ weight tuples each containing $\ell$ weights from different papers such that they all have mean no larger than $\theta_i^*$. Otherwise they form a left chain of length $i$. So $\nu$ cannot have its mean appear at entry $i$ in $\boldsymbol{\theta}^*$.

If $\nu$ violates criterion C2, there exists a row that has more than $n-i$ unmarked entries in $X$. The weights of the unmarked entries have not been encountered so far, which indicates that any tuple that contains the weights from unmarked entries has mean no less than $\theta_i^*$. Otherwise, we would have encountered the weight before $\nu$ and mark its entry. We know that there are $n-i$ reviewers who has mean weight no less than $\theta_i^*$. In addition, there are more than $n-i$ weights left for at least one paper. By Pigeon Hole Principle, there exists a reviewer gives a weight tuple that contains two weights from the same paper. However, no two weights from the same paper can be in the same tuple since one reviewer cannot give 2 weights to the same paper. So $\nu$ cannot have its mean appear at entry $i$ in $\boldsymbol{\theta}^*$.

Thus, $\theta_i^*$ cannot be a value for entry $i$ in $\boldsymbol{\theta}^*$. The value $\theta_i$ we computed is indeed a lower bound on that entry.

Following a similar argument, we can prove the correctness of the upper bounds from the algorithm.

## D.6 Proof of Theorem 4.7

We will show that the proposed algorithm has polynomial time complexity in the number of reviewers. There are $n \cdot \ell$ weights, so the size of $\Omega'$, denoted $|\Omega'|$, has size at most $\binom{n\cdot\ell}{\ell}$, which is of complexity $\mathcal{O}(n^\ell)$. Sorting $\Omega'$ has $\mathcal{O}(|\Omega'|\log(|\Omega'|))$ time complexity, which is still polynomial in $n$. There are $\binom{|\Omega'|}{2}$ pairs of vertices to examine for edges. Therefore, constructing $G$ is of polynomial time in $n$. To compute the length longest left chain and right chain of a vertex, we can make use of a dynamic programming algorithm that only requires us to loop through $\Omega$ once to compute length of longest left chain of all vertices and loop one more time to compute the length of longest right chain. For each vertex, we examine at most all its neighbors, which is of size polynomial in $n$. Lastly, after all preparation work, for each vertex, we take $\mathcal{O}(1)$ time to check criteria C1 and and C3 at most $\mathcal{O}(m)$ time to check criteria C2 and C4. Since $m \leq n \cdot \ell$, both operations are polynomial in $n$. Thus, the proposed algorithm computes the bounds in time polynomial in $n$.

We will use quadratic programming to project noisy data onto the convex set and there are $2n$ linear constraints. This operation is also polynomial in $n$.

Thus, the proposed algorithm has time complexity that is polynomial in $n$.

## D.7 Proof of Theorem 4.8

Axiomatic property A1: When all weights are the same, all weight tuples have the same mean, which equals the weight. Thus, all lower and upper bounds have the same value as the weight. The convex set contains a single vector and projection of any noisy data on such convex set will result in the vector, whose entries are all the same as the weight.

Axiomatic property A2: When $\ell = 1$, there are exactly $n$ weight tuples, each containing one weight. We will choose the same weight tuple for lower bound and upper bound on $\theta_i^*$. The mean of the chosen weight tuple is the weight of rank $i$ among all $n$ weights. Therefore, the convex set contains exactly one vector, which is the sorted vector of all weights. Projection of any noisy data onto this convex set will result in the vector of sorted weights.

Axiomatic property A3: When all except for one paper receives all zero weights, computation of lower bound on $\theta_i^*$ when $i < n-k$ will choose a tuple whose weights are all zeros. When $i \geq n-k$, computation of lower bound will choose a tuple that contains a nonzero weights due to criterion C2. Similarly, to compute an upper bound on $\theta_i^*$ when $i \geq n-k$, we will choose a tuple with a nonzero weight due to the criterion C3. But when $i < n-k$, the algorithm will choose a tuple with all zero weights. The example we present in Section B illustrates this process. Therefore, the convex set again contains only a vector who has $n-k$ zero entries. Projection of any noisy data will result in this vector.