# Fast Adaptive Algorithm for Robust Evaluation of Quality of Experience

Qianqian Xu, Ming Yan, and Yuan Yao

***Abstract*—Outlier detection is an integral part of robust evaluation for crowdsourceable Quality of Experience (QoE) and has attracted much attention in recent years. In QoE for multimedia, outliers happen because of different test conditions, human errors, abnormal variations in context, etc. In this paper, we propose a simple yet effective algorithm for outlier detection and robust QoE evaluation named iterative Least Trimmed Squares (iLTS). The algorithm assigns binary weights to samples, i.e., 0 or 1 indicating if a sample is an outlier, then the outlier-trimmed subset least squares solutions give robust ranking scores. An iterative optimization is carried alternatively between updating weights and ranking scores which converges to a local optimizer in finite steps. In our test setting, iLTS is up to 190 times faster than LASSO-based methods with a comparable performance. Moreover, a varied version of this method shows adaptation in outlier detection, which provides an automatic detection to determine whether a data sample is an outlier without *a priori* knowledge about the amount of the outliers. The effectiveness and efficiency of iLTS are demonstrated on both simulated examples and real-world applications. A Matlab package is provided to researchers exploiting crowdsourcing paired comparison data for robust ranking.**

***Index Terms*—Quality of Experience (QoE); Crowdsourcing; Paired Comparison; Outlier Detection; Iterative Least Trimmed Squares; HodgeRank; Adaptive Outlier Pursuit**

## I. INTRODUCTION

In recent years, the quality of experience (QoE) notion [1], [2] has become a major research theme within the multimedia community, which can be described as the assessment of a user's subjective expectation, feeling, perception, and satisfaction with respect to multimedia content. There are two main quality assessment methodologies, namely subjective and objective assessment. Measuring and ensuring good QoE of multimedia content is highly subjective in nature. The most commonly used subjective method for quality measurement is the mean opinion score (MOS). MOS is standardized in the ITU-T recommendations [3], and it is defined as a numeric value going from 1 to 5 (i.e., bad to excellent). Although the MOS rating method has a long history of pervasive use, it suffers from three fundamental problems: (i) Unable to concretely define the concept of scale; (ii) Dissimilar interpretations of the scale among users; (iii) Difficult to verify whether a participant gives false ratings either intentionally or carelessly [4].

Q. Xu is with BICMR, Peking University, Beijing 100871, China, (email: xuqianqian@math.pku.edu.cn).

M. Yan is with Department of Mathematics, University of California, Los Angeles, CA 90095, USA, (email: yanm@math.ucla.edu).

Y. Yao is with LMAM-LMP-LMEQF, School of Mathematical Sciences, Peking University, Beijing 100871, China, (email: yuany@math.pku.edu.cn).

This work is supported in part by National Basic Research Program of China (973 Program 2012CB825501), NSFC Grant 61071157 and 61402019, NSF Grants DMS-1349855, DMS-1317602, and ARO MURI Grant W911NF-09-1-0383.

Therefore, to address the problems above, we turn to an alternative approach by leveraging the pairwise preference information (i.e., pairwise comparison) obtained from raters. Pairwise comparison has a long history, dating back to the $18^{th}$ century. It also has many nice properties. For example, pairwise comparison is a relative measure which is easier to conduct than absolute rating scores and it helps reduce bias from the rating scale. In Netflix dataset, the rating matrix is 99% incomplete, whereas the paired comparison matrix is only 0.22% incomplete and most entries are supported by many comparisons [5]. In some cases such as tennis tournaments, even only pairwise comparison is possible. However, since the number of pairs $\binom{n}{2}$ grows quadratically with the number of alternatives under investigation, this approach may be an expensive and time-consuming process in a laboratory setting.

To meet this challenge, with the advent of ubiquitous Internet access, the crowdsourcing strategy arises to be a promising alternative approach [6]. It provides an easy and relatively inexpensive way to accomplish small and simple tasks, such as Human Intelligence Tasks (HITs), and to effectively utilize the wisdom of the commons to solve complicated projects. Typically, in a crowdsourcing scenario, each individual contributor is asked to solve a part of a big problem, and a computational algorithm is then developed to combine the partial solutions into an integrated one. Because of the considerable size of the Internet crowd, crowdsourcing could provide us efficient and reliable QoE assessments taking advantage of the power of the mass [2].

Methods for rating/ranking via pairwise comparison in QoE evaluation in crowdsourcing scenario must address a number of inherent difficulties including: (i) incomplete and imbalanced data; (ii) streaming and online data; (iii) outlier detection. To meet the first challenge, the work in [7]–[9] propose randomized paired comparison methods which accommodate incomplete and imbalanced data, a general framework called *HodgeRank on random graphs* (HRRG). It not only can deal with incomplete and imbalanced data collected from crowdsourcing studies but also derives the constraints on sampling complexity in crowdsourcing experiment that the random selection must adhere to. Furthermore, a recent extension of HRRG is introduced in [10], [11] to deal with streaming and online data in crowdsourcing scenario in the second challenge, providing the possibility of making assessment procedure significantly faster than [8], [9] without deteriorating the accuracy.

The third challenge of crowdsourcing QoE evaluations is the fact that not every Internet user is trustworthy. In other words, due to the lack of supervision when subjects perform experiments in crowdsourcing, they may provide erroneous responses perfunctorily, carelessly, or dishonestly [4]. Such random decisions are useless and may deviate significantly from other raters' decisions. Such outliers have to be identified to achieve a robust QoE evaluation. In [4], Transitivity Satisfaction Rate (TSR), which checks all the intransitive triangles, e.g., $A \succ B \succ C \succ A$, is proposed for outlier detection. TSR is defined as the number of judgment triplets (e.g., the three preference relations among A, B, and C) satisfying transitivity divided by the total number of triplets where transitivity may apply; thus, the value of TSR is always between 0 and 1. If a participant's judgments are consistent throughout all the rounds of an experiment, TSR will be 1; otherwise it will be smaller than 1. In this way, we can identify and discard noisy data provided by unreliable assessors. However, TSR can only be applied for complete and balanced paired comparison data. When the paired data are incomplete and imblanced, i.e., having missing edges, the question of how to detect the noisy pairs remains open. The work in [12] attacks this problem and formulates the outlier detection as a LASSO problem based on sparse approximations of cyclic ranking projection of paired comparison data in Hodge decomposition. Regularization paths of the LASSO problem could provide an order on samples tending to be outliers. However, as every sample contributes an outlier indicator variable, solving such a large scale LASSO is expensive, not mentioning the additional cost on model selection via cross-validation, AIC (Akaike Information Criterion), or BIC (Bayesian Information Criterion) which may not even work well in outlier detection [13].

In this paper, we propose a simple yet effective algorithm for outlier detection and robust ranking via iterative Least Trimmed Squares (iLTS). This new method is fast, about 190 times faster than LASSO in our test, and adaptive, which could purify data automatically without *a priori* knowledge on the amount of outliers. In our experimental studies on both simulated and real-world data, the method provide comparable results to LASSO in both outlier detection and robust evaluation scores. Therefore it is a promising tool for crowdsourcing robust QoE evaluation.

The remainder of this paper is organized as follows. Section II contains a review of related work. Then we describe the proposed framework in Section III, which establishes some fast and adaptive algorithms based on iterative least trimmed squares. Detailed experiments are presented in Section IV, followed by the conclusions in Section V.

## II. Related Work

### A. QoE Evaluation

QoE of multimedia content can be divided into two categories: subjective assessment and objective assessment. In subjective viewing tests, stimuli are shown to a group of viewers, and then their opinions are recorded and averaged to evaluate the quality of the stimuli. This process is labor-intensive and time-consuming. On the contrary, objective assessment predicts the perceived quality automatically and intelligently by building objective quality models (see [14], a survey paper, and its references). Objective methods are indeed convenient to use, whereas it can not capture the *true feelings* of users' experiences. Therefore, to obtain factual QoE evaluation results, subjective methods are still required, even though the cost is higher.

A variety of approaches can be employed to conducting subjective tests, among which mean opinion score (MOS) [3] and paired comparison are the two most popular ones. In the MOS test, individuals are asked to specify a rating from "Bad" to "Excellent" (e.g., Bad-1, Poor-2, Fair-3, Good-4, and Excellent-5) to grade the quality of a stimulus; while in paired comparison approach, raters are only asked to make intuitive comparative judgements instead of mapping their perception on a categorical or numerical scale. Among these there may be tradeoffs in the amount of information the preference label contains and the bias associated with obtaining the label. For example, while a graded relevance judgment on a five-point scale may contain more information than a binary judgment, raters may also make more errors due to the complexity of assigning finer-grained judgments. For this reason, the paired comparison method is currently gaining growing attention, which promises assessments that are easier and faster to obtain, less demanding task for raters, and yields more reliable data with less personal scale bias in practice. A shortcoming of paired comparison is that it has more expensive sampling complexity than the MOS test. Therefore, how to make paired comparison method efficient and applicable in reality becomes a hot topic in recent years.

### B. Crowdsourcing

Crowdsourcing can be considered as a further development of the outsourcing principle, where tasks are submitted to an undefined and large group of people or community (a "crowd") in the form of an open call, instead of a designated employee or subcontractor [6]. Most employers submitting tasks to an anonymous crowd use mediators which maintain the crowd and manage the employers campaigns. These mediators are called crowdsourcing platforms. Among various crowdsourcing platforms, Amazon Mechanical Turk (MTurk) is probably the most popular one, which provides a marketplace for a variety of tasks, and anyone who wishes to seek help from the Internet crowd can post their task requests on the website. Besides, InnoCentive, CrowdFlower, CrowdRank, and AllOurIdeas also bring the crowdsourcing revolution to various application fields.

With the help of these platforms, researchers can seek help from the Internet crowd to conduct user studies on document relevance [15], document evaluation [16], image annotation [17], [18], music emotion recognition [19], affection mining in computer games [20], together with some studies on QoE evaluation [4], [8]–[10], [21], [22], etc. However, a major challenge of crowdsourcing QoE evaluation is that not every Internet user is trustworthy. That is, some raters try to maximize their received payment while minimizing their own effort and therefore submit low quality work to obtain such a goal. Therefore, it is necessary to detect unreliable inputs and filter them out since they may cause inaccuracy in the estimation of QoE scores. For example, with complete and balanced data, TSR is proposed in [4] to measure the reliability of the participants' judgments. In contrast, the outlier detection method proposed in this paper is a general and simple one which could deal with not only complete and balanced paired comparison data, but also incomplete and imbalanced data.

## C. Statistical Ranking

Statistical preference aggregation, in particular ranking or rating from pairwise comparisons, is a classical problem which can be traced back to the $18^{th}$ century. This subject area has been widely studied in various fields including the social choice or voting theory in Economics [23], [24], Psychology [25], [26], Statistics [27], [28], Computer Vision [29]–[31], Information Retrieval [32], [33], Machine Learning [34], [35], and others [36]–[38].

In particular, learning to rank trains a statistical model for ranking tasks. Popular approaches for learning to rank with pairwise comparisons include Active Ranking [39], [40], IRSVM [41], RankNet [42], and LambdaRank [43]. However, since learning to rank requires a feature vector representation of the items to be ranked, they can not be directly applied to crowdsourced QoE evaluation.

In crowdsourced QoE evaluation, the purpose is not to predict the ranking based on features, but to aggregate a global ranking from the crowdsourcing pairwise preferences. Various methods have been proposed for crowdsourceable pairwise comparison ranking. In [44], it proposes a Bayesian framework to actively select pairwise comparison queries, and effectively combine the pairwise comparisons acquired by crowdsourcing to form a single ranking list. In [45], it infers the preferences from crowdsourced pairwise comparison with matrix completion and compares it to collaborative filtering. In [46], it develops an iterative ranking aggregation algorithm from pairwise comparisons using Bradley-Terry model. Besides, there are two famous frameworks for QoE evaluation in Crowdsourcing: the *Qudrant of Euphoria* by [47] and *QualityCrowd* by [48].

## D. HodgeRank and Random Graphs

HodgeRank, as an application of combinatorial Hodge theory to the preference or rank aggregation problem from pairwise comparison data, was first introduced in [5], inspiring a series of studies in statistical ranking [49]–[51] and game theory [52], in addition to traditional applications in fluid mechanics [53] and computer vision [31], [54], etc.

It is a general framework to decompose paired comparison data on graphs, possibly imbalanced (where different video pairs may receive different number of comparisons) and incomplete (where every participant may only give partial comparisons), into three orthogonal components. In these components HodgeRank not only provides us a mean to determine a global ranking from paired comparison data under various statistical models (e.g., Uniform, Thurstone-Mosteller, Bradley-Terry, and Angular Transform), but also measures the inconsistency of the global ranking obtained. The inconsistency shows the validity of the ranking obtained and can be further studied in terms of its geometric scale, namely whether the inconsistency in the ranking data arises locally or globally. Local inconsistency can be fully characterized by triangular cycles, while global inconsistency involves cycles consisting nodes more than three, which may arise due to data incompleteness and once presented with a large component indicates some serious conflicts in ranking data. However through random graphs, we can efficiently control global inconsistency.

Random graph is a graph generated by some random process. It starts with a set of *n* vertices and adds edges between them at random. Different random graph models produce different probability distributions on graphs. Among various random graphs (i.e., the Erdös-Rényi random graph [55], random regular graph [56], preferential attachment random graph [57], small world random graph [58], and geometric random graph [59]), the most commonly studied one is the Erdös-Rényi random graph [55]. It can be viewed as a random sampling process of pairs or edges independently and identically distributed (I.I.D.), and thus is well suited to crowdsourcing scenario where raters enter the test system in a dynamic and random way. In [8], [9], a random design principle based on the Erdös-Rényi random graph theory is investigated to conduct crowdsourcing tests. It shows that for a large Erdös-Rényi random graph $G(n,p)$ with $n$ nodes and every edge sampled with probability $p$, $p \gg n^{-1}\log n$ is necessary to ensure the graph is connected and the inference of a global ranking is thus possible. To avoid global inconsistency from Hodge decomposition, it suffices to have larger sampling rates at $p \gg n^{-1/2}$. In this paper, we also focus on this simple yet powerful random graph model particularly in the scenarios where outliers are present.

## E. Outlier Detection

Outliers are typically defined to be data samples that have unusual deviation from the remaining data. Hawkins formally defined in [60] the concept of an outlier as follows: "An outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism." Outliers are rare events, but once they have occurred, they may lead to a large instability of models estimated from the data. Statistical approaches were the earliest algorithms used for outlier detection, such as distribution-based, depth-based, distance-based, density-based, and clustering method [61]. More recently, this problem has been studied quite extensively by the computer science community. In subjective quality evaluation in multimedia, there are several reasons why some user ratings are not reliable and need to be filtered out in order to avoid false QoE results [62]: the test subjects may not understand the test and the test instructions properly; wrong test conditions may occur due to errors in the web-based test application or due to incompatibilities of the test application with the subject's hard- and software; or the subjects do the test in a hurry resulting into sloppy work and unreliable results. Numerous efforts have been made in order to detect outliers and improve the quality of the results. In [12], it formulates the outlier detection as a LASSO problem based on sparse approximations of cyclic ranking projection of paired comparison data. Then regularization paths of the LASSO problem could provide us an order on samples tending to be outliers. Such an approach is inspired by Huber's celebrated work on robust regression [63]. On the other hand, recently [64] proposed a fast algorithm called *adaptive outlier pursuit* (AOP) for random-valued impulse noise removal, which has been applied to many applications in image and signal processing such as robust 1-bit compressive sensing [65], robust binary fused compressive sensing [66], and robust low rank matrix completion [67]. Such a work is based on iterative least trimmed squares. In this paper, we develop applications of AOP in the scenario of robust QoE evaluation.

## III. Iterative Least Trimmed Squares

In this section, we propose a method for automatic outlier detection without any priori information about the number of outliers. It adaptively detects outliers and obtains robust QoE evaluation with the outlier removal. Brief introductions on robust ranking are provided before the algorithm is described.

### A. The Problem of Robust Ranking

Assume that there are $m$ participants and $n$ items to be ranked. Let $Y_{ij}^{\alpha}$ denote the degree that participant $\alpha$ prefers item $i$ to item $j$. Without loss of generality, one assumes that $Y_{ij}^{\alpha} > 0$ if $\alpha$ prefers $i$ to $j$ and $Y_{ij}^{\alpha} < 0$ otherwise. In addition, we assume that the paired comparison data is *skew-symmetric* for each $\alpha$, i.e., $Y_{ij}^{\alpha} = -Y_{ji}^{\alpha}$. The strategy used in QoE evaluation can be dichotomous choice or a $k$-point Likert scale with $k \geq 3$. In this paper, we shall focus on the dichotomous choice, in which $Y_{ij}^{\alpha}$ can be taken as $\{\pm 1\}$ only. However, the theory can be applied to more general cases with $k$-point Likert scales.

In subjective multimedia assessment, it is natural to assume

$$Y_{ij}^{\alpha} = \text{sign}(s_i^* - s_j^* + Z_{ij}^{\alpha}), \tag{1}$$

where $\text{sign}(\cdot) = \pm 1$ measures the sign of the value, $\mathbf{s}^* = \{s_1^*, \cdots, s_n^*\} \in \mathbb{R}^n$ is the true scaling score on $n$ items and $Z_{ij}^{\alpha}$ is the noise. In practice the global rating score $\mathbf{s} = \{s_1, \cdots, s_n\}$ can be obtained by solving the following optimization problem

$$\underset{\mathbf{s}\in\mathbb{R}^n}{\text{minimize}} \sum_{i\neq j,\alpha} W_{ij}^{\alpha} \mathcal{L}(s_i - s_j, Y_{ij}^{\alpha}), \tag{2}$$

where $\mathcal{L}(x, y) : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is a loss function depending on the distribution of the noise, $W_{ij}^{\alpha}$ denotes the importance weights (e.g., number of paired comparisons) on $\{i, j\}$ made by rater $\alpha$, and $s_i$ (or $s_j$) represents the global ranking score of item *i* (or *j*). A geometric interpretation of (2) is to look for some potential function $\mathbf{s} : [n] \to \mathbb{R}$ whose *gradient* captures main variations in paired comparison data $Y$.

If the noise is independent and identically distributed (i.i.d.), the Gauss-Markov theorem tells us that the unbiased estimator with minimal variance is obtained by the choice of square loss $\mathcal{L}(x, y) = (x - y)^2$. In this case the global rating score $\mathbf{s}$ satisfies the normal equation:

$$\mathbf{L}\mathbf{s} = \mathbf{b}, \tag{3}$$

where $\mathbf{L} = \mathbf{D} - \mathbf{A}$ is the unnormalized graph Laplacian defined by $A_{ij} = \sum_{\alpha} W_{ij}^{\alpha}$ and $\mathbf{D}$ is the diagonal matrix with $D_{ii} = \sum_{j,\alpha} W_{ij}^{\alpha}$, $\mathbf{b}$ is the divergence flow defined by $b_i = \sum_{j,\alpha} W_{ij}^{\alpha} Y_{ij}^{\alpha}$. Such an algorithm has been used in [8]–[10] to derive scaling scores in subjective multimedia assessment. Via combinatorial Hodge decomposition [5], [9], the residue of the least squares solution $r_{ij}^{\alpha} = Y_{ij}^{\alpha} - s_i - s_j$ can be interpreted as *cyclic rankings* on $n$ items.

However, not all comparisons are trustworthy and there may be sparse outliers due to different test conditions, human errors, or abnormal variations in context. Putting in a mathematical way, here we consider

$$Z_{ij}^{\alpha} = E_{ij}^{\alpha} + N_{ij}^{\alpha}, \tag{4}$$

where outlier $E_{ij}^{\alpha}$ has a much larger magnitude than $N_{ij}^{\alpha}$ and is sparse as zero with probability $p \in (0, 1]$. When sparse outliers exist, (2) becomes unstable and may give bad estimation. If the outliers can be detected and removed, then the solution from least squares on the remaining comparisons is more accurate and gives a better estimation.

In [12], the famous Huber's loss [63] is chosen for robust ranking as $\mathcal{L}(s_i - x_j - Y_{ij}^{\alpha}) = \rho_{\lambda}(s_i - s_j - Y_{ij}^{\alpha})$ where

$$\rho_{\lambda}(x) = \begin{cases} x^2/2, & \text{if } |x| \leq \lambda \\ \lambda|x| - \lambda^2/2, & \text{if } |x| > \lambda. \end{cases}$$

When $|s_i - s_j - Y_{ij}^{\alpha}| < \lambda$, the comparison is regarded as a "good" one with Gaussian noise and L2-norm penalty is used on the residual. Otherwise, it is regarded as a "bad" one contaminated by outliers and one uses L1-norm penalty which is less sensitive to the amount of deviation. Assume that the importance weights are the same ($W_{ij}^{\alpha} = 1$). In this case, (2) is equivalent to the following LASSO problem, often called Huber-LASSO,

$$\underset{\mathbf{s}\in\mathbb{R}^n,\mathbf{E}}{\text{minimize}} \sum_{i,j,\alpha} \frac{1}{2}(s_i - s_j - Y_{ij}^{\alpha} + E_{ij}^{\alpha})^2 + \lambda\|\mathbf{E}\|_1. \tag{5}$$

A simple geometric interpretation from Hodge decomposition [12] is that the outlier $\mathbf{E}$ is a sparse approximation of *cyclic* ranking projection which summarizes the conflicts of interests among voters.

There are a couple of issues in such a Huber-LASSO approach [12]: 1) the LASSO estimator is well-known to be biased; 2) the computational cost of Huber-LASSO path is expensive as every sample is associated with an outlier indicator variable $E_{ij}^{\alpha}$. To solve (1), one typically exploits Huber-LASSO in outlier detection, followed by a subset least squares with only non-outlier samples. This is often called Least Trimmed Squares (LTS) in robust statistics [68]. In the remaining of this section, we will see some iterative versions of LTS leads to fast algorithms for robust ranking which automatically finds the number of outliers in practice.

## B. Least Trimmed Squares

Given $K$ as the number of outliers, the least trimmed squares model can be written as

$$\left\{\begin{array}{ll}\underset{\mathbf{s}\in\mathbb{R}^n,\Lambda}{\text{minimize}} & \sum\limits_{i,j,\alpha}\Lambda_{ij}^{\alpha}(s_i-s_j-Y_{ij}^{\alpha})^2,\\ \text{subject to} & \sum\limits_{i,j,\alpha}(1-\Lambda_{ij}^{\alpha})\le K, \Lambda_{ij}^{\alpha}\in\{0,1\},\end{array}\right. \tag{6}$$

where $\Lambda_{ij}^{\alpha}$ is used to denote the outlier as follows:

$$\Lambda_{ij}^{\alpha}=\left\{\begin{array}{ll}0, & \text{if } Y_{ij}^{\alpha} \text{ is a outlier},\\ 1, & \text{otherwise}.\end{array}\right. \tag{7}$$

*Remark 1:* When the importance weights are not the same, we can modify the problem into

$$\left\{\begin{array}{ll}\underset{\mathbf{s}\in\mathbb{R}^n,\Lambda}{\text{minimize}} & \sum\limits_{i,j,\alpha}\Lambda_{ij}^{\alpha}W_{ij}^{\alpha}(s_i-s_j-Y_{ij}^{\alpha})^2,\\ \text{subject to} & \sum\limits_{i,j,\alpha}(1-\Lambda_{ij}^{\alpha})W_{ij}^{\alpha}\le K, \Lambda_{ij}^{\alpha}\in\{0,1\}.\end{array}\right.$$

Let

$$F(\mathbf{s},\Lambda)=\sum_{i,j,\alpha}\Lambda_{ij}^{\alpha}(s_i-s_j-Y_{ij}^{\alpha})^2 + \iota_{\{\Lambda:\sum_{i,j,\alpha}(1-\Lambda_{ij}^{\alpha})\le K,\Lambda_{ij}^{\alpha}\in\{0,1\}\}}$$

where $\iota_{\{\Lambda:\sum_{i,j,\alpha}(1-\Lambda_{ij}^{\alpha})\le K,\Lambda_{ij}^{\alpha}\in\{0,1\}\}}$ is the indicator function which equals to zero when both $\sum_{i,j,\alpha}(1-\Lambda_{ij}^{\alpha})\le K$ and $\Lambda_{ij}^{\alpha}\in\{0,1\}$ are satisfied and $+\infty$ otherwise, then problem (6) is equivalent to

$$\underset{\mathbf{s}\in\mathbb{R}^n,\Lambda}{\text{minimize}}\, F(\mathbf{s},\Lambda). \tag{8}$$

This is a nonconvex optimization problem. However one can split the minimization over $\Lambda$ and s into two steps. For solving the problem in s with $\Lambda$ fixed, it is a convex least squares problem, and the problem of finding $\Lambda$ with s fixed can be solved in one step. These two subproblems are:

1) Fix $\Lambda$ and update s. We need to solve a least squares problem with the comparisons that are detected to be outliers removed.

2) Fix s and update $\Lambda$. This time we are solving

$$\left\{\begin{array}{ll}\underset{\Lambda}{\text{minimize}} & \sum\limits_{i,j,\alpha}\Lambda_{ij}^{\alpha}(s_i-s_j-Y_{ij}^{\alpha})^2,\\ \text{subject to} & \sum\limits_{i,j,\alpha}(1-\Lambda_{ij}^{\alpha})\le K, \Lambda_{ij}^{\alpha}\in\{0,1\}.\end{array}\right. \tag{9}$$

This problem is to choose $K$ elements with largest summation from the set $\{(s_i-s_j-Y_{ij}^{\alpha})^2\}$. Denoting $\tau$ as the value of the $K$th largest term in that set, $\Lambda$ can then be calculated by

$$\Lambda_{ij}^{\alpha}=\left\{\begin{array}{ll}1, & \text{if } (s_i-s_j-Y_{ij}^{\alpha})^2<\tau,\\ 0, & \text{otherwise}.\end{array}\right. \tag{10}$$

If the $K$th and $(K+1)$th largest terms have the same value, then we can choose any $\Lambda$ such that $\sum\limits_{i,j,\alpha}(1-\Lambda_{ij}^{\alpha})\le K$ and

$$\min_{i,j,\alpha,\Lambda_{ij}^{\alpha}=0}(s_i-s_j-Y_{ij}^{\alpha})^2\ge\max_{i,j,\alpha,\Lambda_{ij}^{\alpha}=1}(s_i-s_j-Y_{ij}^{\alpha})^2. \tag{11}$$

Such a procedure is described precisely in the following algorithm.

**Algorithm 1** Iterative Least Trimmed Squares with $K$

```
Input: $\{Y_{ij}^{\alpha}\}$, $K \ge 0$.
Initialization: $k = 0$, $\Lambda_{ij}^{\alpha} = 1$.
for $k = 1, 2, \cdots$ do
    Update $\mathbf{s}^k$ by solving the least squares problem (2) using only the comparisons with $\Lambda_{ij}^{\alpha} = 1$.
    Update $\Lambda^k$ from (10) or (11) with one different from previous ones.
end for
return s.
```

Let $E(\mathbf{s})=\min_{\Lambda}F(\mathbf{s},\Lambda)$ and we have the following theorem about the convergence of Algorithm 1.

*Theorem 1:* Algorithm 1 will converge in finite steps and the output s is a local minimum point of $E(\mathbf{s})$.

*Proof:* From the algorithm, we have

$$F(\mathbf{s}^k,\Lambda^k)\ge F(\mathbf{s}^{k+1},\Lambda^k)\ge F(\mathbf{s}^{k+1},\Lambda^{k+1}). \tag{12}$$

Additionally there are only finite number of $\Lambda$'s. Therefore the algorithm will stop in finite steps if the s-subproblem is solved exactly. Assume that we have $F(\mathbf{s}^k,\Lambda^k)=F(\mathbf{s}^{k+1},\Lambda^{k+1})$. Thus

$$F(\mathbf{s}^k,\Lambda^k)=F(\mathbf{s}^{k+1},\Lambda^k)=\min_{\mathbf{s}}F(\mathbf{s},\Lambda^k),$$
$$F(\mathbf{s}^k,\Lambda^k)=\min_{\Lambda}F(\mathbf{s}^k,\Lambda)=E(\mathbf{s}^k),$$

which means that $(\mathbf{s}^k,\Lambda^k)$ is a coordinatewise minimum point of $F(\mathbf{s},\Lambda)$. We will show that $\mathbf{s}^k$ is a local minimum point of $E(\mathbf{s})$.

Let $\tau^k$ be the $K$th largest term of $\{(s_i^k-s_j^k-Y_{ij}^{\alpha})^2\}$, and define $\Lambda_+=\{(i,j,\alpha):(s_i^k-s_j^k-Y_{ij}^{\alpha})^2>\tau^k\}$ and $\Lambda_-=\{(i,j,\alpha):(s_i^k-s_j^k-Y_{ij}^{\alpha})^2<\tau^k\}$. Then we can find $\epsilon>0$ such that when $\|\mathbf{s}-\mathbf{s}^k\|_2<\epsilon$, we have $(s_i-s_j-Y_{ij}^{\alpha})^2>\tau$ for all $(i,j,\alpha)\in\Lambda_+$ and $(s_i-s_j-Y_{ij}^{\alpha})^2<\tau$ for all $(i,j,\alpha)\in\Lambda_-$, where $\tau$ is the $K$th largest term of $\{(s_i-s_j-Y_{ij}^{\alpha})^2\}$. Notice that $E(\mathbf{s})=\min_{\Lambda}F(\mathbf{s},\Lambda)$, then there is $\bar{\Lambda}$ such that $E(\mathbf{s})=F(\mathbf{s},\bar{\Lambda})$, with $\bar{\Lambda}_{ij}^{\alpha}>0$ when $(i,j,\alpha)\in\Lambda_+$ and $\bar{\Lambda}_{ij}^{\alpha}<0$ when $(i,j,\alpha)\in\Lambda_-$. Thus $E(\mathbf{s}^k)=F(\mathbf{s}^k,\bar{\Lambda})$. In addition, we have $F(\mathbf{s}^k,\bar{\Lambda})\le$

$F(\mathbf{s},\bar{\Lambda})$ because all $\Lambda$'s satisfying (11) for $\mathbf{s}^k$ are chosen before the algorithm stops. Hence, $E(\mathbf{s}^k) = F(\mathbf{s}^k,\bar{\Lambda}) \leq F(\mathbf{s},\bar{\Lambda}) = E(\mathbf{s})$, and $\mathbf{s}^k$ is a local minimizer of $E(\mathbf{s})$. ■

## C. *Adaptive Least Trimmed Squares*

If the number of outlier $K$ is given, Algorithm 1 can be used to detect the outliers and improve the performance of least squares. However, in practice, the exact number of outliers $K$ may be unknown. If $K$ is underestimated, some remaining outliers will still damage the performance. On the other hand, if $K$ is overestimated, too many outliers are removed, and the resulting data is not enough or too biased for QoE evaluation. For a few applications such as impulse noise removal, the number of outliers can be estimated accurately, while it is difficult for many applications including crowdsourceable QoE evaluation. Therefore, a outlier detection method which can automatically estimate the number of outliers is strongly needed.

In the following, we propose a method to estimate the number of outliers automatically. At first, when the number of outliers is unknown, we can use least squares to find an estimate of $\mathbf{s}$, then the number of outliers according to this $\mathbf{s}$ can be calculated, i.e., the total number of comparisons with wrong directions ($Y_{ij}^{\alpha}$ has different sign with $s_i - s_j$) denoted as $\widetilde{K}$. Because this $\mathbf{s}$ is not accurate and $\widetilde{K}$ is an overestimation of $K$. We can underestimate the number of outliers as $\underset{\sim}{K} = \beta_1 \widetilde{K}$ ($\beta_1 \in (0,1)$), With this underestimate $\underset{\sim}{K}$, we can solve the least squares problem after the $\underset{\sim}{K}$ comparison that are considered to be outliers removed and obtain an improved $\mathbf{s}$. Then we have to increase the estimation of the number $\underset{\sim}{K}$ by $\beta_2$ ($\beta_2 \in (1,\infty)$), but the number can not be larger than $\widetilde{K}$, the total number of comparisons mismatching the current score, because there are only $\widetilde{K}$ outliers with the current score. Therefore the update of $\underset{\sim}{K}$ is just $\underset{\sim}{K} = \min(\lfloor \beta_2 \underset{\sim}{K} \rfloor, \widetilde{K})$ where $\lfloor x \rfloor$ ($\lceil x \rceil$) is the greatest (smallest) integer no larger (less) than $x \in \mathbb{R}^+$. The weight $\Lambda_{ij}^{\alpha}$ for the new least trimmed squares problem is binary (0 or 1) and determined by $\underset{\sim}{K}$ largest outliers. Iterations go on until a fixed point is met where $\underset{\sim}{K} = \widetilde{K}$ gives the estimated number of outliers. Algorithm 2 describes such a procedure precisely, which is called here *Iterative Least Trimmed Squares without $K$* or simply *Adaptive Least Trimmed Squares*.

*Remark 2:* There are only two parameters to choose and they are easy to set. They are chosen according to following inequalities $\beta_1 < 1 < \beta_2$ ($\beta_1 = 0.75$ and $\beta_2 = 1.03$ are fixed in our numerical experiments). $\beta_1$ has to be small to make sure that the first estimation is underestimated. Then the underestimate $\underset{\sim}{K}$ is increasing geometrically with rate $\beta_2$ and $\beta_2$ can not be too large, because we do not want to increase the underestimate too much.

*Remark 3:* The algorithm is able to detect most of the outliers in our experiments with a maximum iteration number $\mathrm{Miter} = 30$. However, there may be mistakes in the detection, and these mistakes happen mostly between two successive items in the order. Therefore, we can add one step to just compare every pair of two successive items

**Algorithm 2** Adaptive Least Trimmed Squares

```
Input: $\{Y_{ij}^{\alpha}\}$, Miter $> 0$, $\beta_1 < 1$, $\beta_2 > 1$.
Initialization: $k = 0$, $\Lambda_{i,j}^{\alpha}(k) = 1$, $\underset{\sim}{K}_k = 0$.
for $k = 1, \cdots,$ Miter do
    Update $\mathbf{s}^k$ with least squares (2) using only the comparisons with $\Lambda_{ij}^{\alpha}(k-1) = 1$.
    Let $\widetilde{K}_k$ be the total number of comparisons with wrong directions, i.e., $Y_{ij}^{\alpha}$ has different sign with $s_i^k - s_j^k$.
        $$\underset{\sim}{K}_k = \begin{cases} \lfloor \beta_1 \widetilde{K}_k \rfloor, & \text{if } k = 1; \\ \min(\lfloor \beta_2 \underset{\sim}{K}_{k-1} \rfloor, \widetilde{K}_k), & \text{otherwise,} \end{cases} \quad (13)$$
    If $\underset{\sim}{K}_k = \widetilde{K}_k$, break.
    Update $\Lambda(k)$ using (10) or (11) with $K = \underset{\sim}{K}_k$.
end for
Find $\hat{\mathbf{s}}$ with least squares (2) using only the samples with $\Lambda_{ij}^{\alpha}(k) = 1$.
return $\hat{\mathbf{s}}$, $\hat{K} = \widetilde{K}_k$.
```

and make the correction on the detection, i.e., if item $i$ is ranking above $j$ but the number of people choosing item $i$ over $j$ is less than the number of people choosing $j$ over $i$, we can remove those choosing $j$ over $i$ and keep those choosing $i$ over $j$.

The algorithm always stops in finite steps even without a bound on "$\mathrm{Miter}$", due to the following Lemma.

*Lemma 1:* If $\widetilde{K}_k \leq C$ for $k \geq k_0$, Algorithm 2 will stop in no more than $k^*$ steps, where

$$k^* = \left\lceil \frac{\log C - \log \beta_1 \widetilde{K}_1}{\log \beta_2} \right\rceil .$$

*Proof:* It follows from the fact that $\underset{\sim}{K}_k$ is a monotonic increasing sequence for $\beta_2 > 1$ and bounded $\widetilde{K}_k$. ■

However such a result only ensures that the algorithm stops at an overestimate on the correct number of outliers. The following theorem presents a stability condition such that Algorithm 2 returns the correct number of outliers.

*Theorem 2:* Assume that for $k \geq k_0$, every sample subset $\mathrm{supp}(\Lambda(k-1))$ gives an order-consistent least squares estimator $\mathbf{s}^k$, i.e., $\mathbf{s}^k$ induces the same ranking order as the true score $\mathbf{s}^*$, then Algorithm 2 returns the correct number of outliers in $\hat{K}$.

*Proof:* As $\mathbf{s}^k$ is an order-consistent solution of (2), by definition $\widetilde{K}_k$ gives the correct number of outliers, say $K^*$. It actually holds for all $k \geq k_0$, that $\widetilde{K}_k \equiv K^*$. From Lemma 1 the claim follows. ■

Note that Theorem 2 does not require $\mathrm{supp}(\Lambda(k-1))$ to correctly identify the outliers, but just stable estimator $\mathbf{s}^k$ which does not change the order from $\mathbf{s}^*$. In practice, this might not be satisfied easily; but as we shall see in the next section with experiments, Algorithm 2 typically returns stable estimators which slightly deviate in local ranking order.

## IV. EXPERIMENTS

A key question in the outlier detection community is how to evaluate the effectiveness of outlier detection algorithms

TABLE I: ***Precision***s for simulated data via iLTS, 100 times repeat.

| ***Precision*** **(sd)** | **OP=5%** | **OP=10%** | **OP=15%** | **OP=20%** | **OP=25%** | **OP=30%** | **OP=35%** | **OP=40%** | **OP=45%** | **OP=50%** |
|---|---|---|---|---|---|---|---|---|---|---|
| **SN=1000** | 0.997(0.022) | 0.993(0.023) | 0.993(0.015) | 0.978(0.025) | 0.964(0.034) | 0.942(0.037) | 0.893(0.053) | 0.825(0.064) | 0.670(0.078) | 0.505(0.097) |
| **SN=2000** | 1.000(0) | 1.000(0) | 0.998(0.009) | 0.999(0.005) | 0.995(0.010) | 0.976(0.023) | 0.947(0.034) | 0.882(0.051) | 0.751(0.067) | 0.503(0.089) |
| **SN=3000** | 1.000(0) | 1.000(0) | 1.000(0) | 0.999(0.002) | 0.998(0.005) | 0.991(0.013) | 0.970(0.024) | 0.926(0.036) | 0.811(0.060) | 0.502(0.090) |
| **SN=4000** | 1.000(0) | 1.000(0) | 1.000(0) | 1.000(0) | 0.999(0.002) | 0.995(0.010) | 0.988(0.015) | 0.945(0.031) | 0.829(0.059) | 0.498(0.098) |
| **SN=5000** | 1.000(0) | 1.000(0) | 1.000(0) | 1.000(0) | 1.000(0) | 0.998(0.006) | 0.990(0.015) | 0.959(0.027) | 0.847(0.052) | 0.499(0.101) |

TABLE II: ***Precision***s for simulated data via LASSO, 100 times repeat.

| ***Precision*** **(sd)** | **OP=5%** | **OP=10%** | **OP=15%** | **OP=20%** | **OP=25%** | **OP=30%** | **OP=35%** | **OP=40%** | **OP=45%** | **OP=50%** |
|---|---|---|---|---|---|---|---|---|---|---|
| **SN=1000** | 0.972(0.033) | 0.962(0.030) | 0.958(0.025) | 0.931(0.028) | 0.923(0.028) | 0.905(0.032) | 0.863(0.040) | 0.805(0.058) | 0.698(0.067) | 0.513(0.085) |
| **SN=2000** | 0.996(0.011) | 0.990(0.014) | 0.984(0.016) | 0.970(0.022) | 0.960(0.017) | 0.942(0.022) | 0.914(0.031) | 0.860(0.044) | 0.750(0.056) | 0.516(0.084) |
| **SN=3000** | 0.999(0.005) | 0.997(0.008) | 0.992(0.012) | 0.981(0.016) | 0.970(0.016) | 0.957(0.020) | 0.929(0.022) | 0.887(0.031) | 0.796(0.050) | 0.523(0.083) |
| **SN=4000** | 0.999(0.001) | 0.999(0.005) | 0.996(0.009) | 0.990(0.011) | 0.980(0.015) | 0.970(0.016) | 0.946(0.019) | 0.909(0.027) | 0.818(0.048) | 0.518(0.093) |
| **SN=5000** | 0.999(0.002) | 1.000(0) | 0.998(0.006) | 0.992(0.011) | 0.985(0.015) | 0.972(0.016) | 0.955(0.019) | 0.917(0.027) | 0.837(0.038) | 0.525(0.088) |

TABLE III: ***Recall***s for simulated data via iLTS, 100 times repeat.

| ***Recall*** **(sd)** | **OP=5%** | **OP=10%** | **OP=15%** | **OP=20%** | **OP=25%** | **OP=30%** | **OP=35%** | **OP=40%** | **OP=45%** | **OP=50%** |
|---|---|---|---|---|---|---|---|---|---|---|
| **SN=1000** | 1.000(0) | 0.994(0.015) | 0.994(0.010) | 0.981(0.020) | 0.969(0.024) | 0.943(0.036) | 0.885(0.054) | 0.805(0.066) | 0.653(0.080) | 0.438(0.093) |
| **SN=2000** | 1.000(0) | 1.000(0) | 0.999(0.006) | 0.999(0.005) | 0.994(0.011) | 0.978(0.019) | 0.947(0.032) | 0.879(0.052) | 0.727(0.071) | 0.456(0.087) |
| **SN=3000** | 1.000(0) | 1.000(0) | 1.000(0) | 0.999(0.002) | 0.998(0.005) | 0.991(0.012) | 0.970(0.023) | 0.925(0.037) | 0.797(0.062) | 0.464(0.089) |
| **SN=4000** | 1.000(0) | 1.000(0) | 1.000(0) | 1.000(0) | 0.999(0.003) | 0.996(0.007) | 0.988(0.014) | 0.946(0.030) | 0.821(0.060) | 0.466(0.098) |
| **SN=5000** | 1.000(0) | 1.000(0) | 1.000(0) | 1.000(0) | 1.000(0) | 0.998(0.006) | 0.991(0.013) | 0.962(0.025) | 0.842(0.052) | 0.470(0.100) |

TABLE IV: ***Recall***s for simulated data via LASSO, 100 times repeat.

| ***Recall*** **(sd)** | **OP=5%** | **OP=10%** | **OP=15%** | **OP=20%** | **OP=25%** | **OP=30%** | **OP=35%** | **OP=40%** | **OP=45%** | **OP=50%** |
|---|---|---|---|---|---|---|---|---|---|---|
| **SN=1000** | 0.972(0.033) | 0.962(0.030) | 0.958(0.025) | 0.931(0.028) | 0.923(0.028) | 0.905(0.032) | 0.863(0.040) | 0.805(0.058) | 0.698(0.067) | 0.513(0.085) |
| **SN=2000** | 0.996(0.011) | 0.990(0.014) | 0.984(0.016) | 0.970(0.022) | 0.960(0.017) | 0.942(0.022) | 0.914(0.031) | 0.860(0.044) | 0.750(0.056) | 0.518(0.084) |
| **SN=3000** | 0.999(0.005) | 0.997(0.008) | 0.992(0.012) | 0.981(0.016) | 0.970(0.016) | 0.957(0.020) | 0.929(0.022) | 0.887(0.031) | 0.796(0.050) | 0.523(0.083) |
| **SN=4000** | 0.999(0.001) | 0.999(0.005) | 0.996(0.009) | 0.990(0.011) | 0.980(0.015) | 0.970(0.016) | 0.946(0.019) | 0.909(0.027) | 0.818(0.048) | 0.518(0.093) |
| **SN=5000** | 0.999(0.002) | 1.000(0) | 0.998(0.006) | 0.992(0.011) | 0.985(0.015) | 0.972(0.016) | 0.955(0.019) | 0.917(0.027) | 0.837(0.038) | 0.525(0.088) |

TABLE V: ***F1*** scores for simulated data via iLTS, 100 times repeat.

| ***F1*** **(sd)** | **OP=5%** | **OP=10%** | **OP=15%** | **OP=20%** | **OP=25%** | **OP=30%** | **OP=35%** | **OP=40%** | **OP=45%** | **OP=50%** |
|---|---|---|---|---|---|---|---|---|---|---|
| **SN=1000** | 0.998(0.012) | 0.994(0.019) | 0.994(0.012) | 0.980(0.022) | 0.966(0.028) | 0.943(0.036) | 0.889(0.053) | 0.815(0.064) | 0.675(0.079) | 0.469(0.095) |
| **SN=2000** | 1.000(0) | 1.000(0) | 0.999(0.007) | 0.999(0.005) | 0.994(0.010) | 0.977(0.021) | 0.947(0.033) | 0.880(0.051) | 0.739(0.069) | 0.478(0.088) |
| **SN=3000** | 1.000(0) | 1.000(0) | 1.000(0) | 0.999(0.002) | 0.998(0.005) | 0.991(0.012) | 0.970(0.023) | 0.925(0.036) | 0.804(0.061) | 0.482(0.089) |
| **SN=4000** | 1.000(0) | 1.000(0) | 1.000(0) | 1.000(0) | 0.999(0.003) | 0.996(0.009) | 0.988(0.014) | 0.946(0.030) | 0.825(0.059) | 0.482(0.098) |
| **SN=5000** | 1.000(0) | 1.000(0) | 1.000(0) | 1.000(0) | 1.000(0) | 0.998(0.006) | 0.990(0.014) | 0.960(0.026) | 0.845(0.052) | 0.484(0.101) |

TABLE VI: ***F1*** scores for simulated data via LASSO, 100 times repeat.

| ***F1*** **(sd)** | **OP=5%** | **OP=10%** | **OP=15%** | **OP=20%** | **OP=25%** | **OP=30%** | **OP=35%** | **OP=40%** | **OP=45%** | **OP=50%** |
|---|---|---|---|---|---|---|---|---|---|---|
| **SN=1000** | 0.972(0.033) | 0.962(0.030) | 0.958(0.025) | 0.931(0.028) | 0.923(0.028) | 0.905(0.032) | 0.863(0.040) | 0.805(0.058) | 0.698(0.067) | 0.513(0.085) |
| **SN=2000** | 0.996(0.011) | 0.990(0.014) | 0.984(0.016) | 0.970(0.022) | 0.960(0.017) | 0.942(0.022) | 0.914(0.031) | 0.860(0.044) | 0.750(0.056) | 0.516(0.084) |
| **SN=3000** | 0.999(0.005) | 0.997(0.008) | 0.992(0.012) | 0.981(0.016) | 0.970(0.016) | 0.957(0.020) | 0.929(0.022) | 0.887(0.031) | 0.796(0.050) | 0.523(0.083) |
| **SN=4000** | 0.999(0.001) | 0.999(0.005) | 0.996(0.009) | 0.990(0.011) | 0.980(0.015) | 0.970(0.016) | 0.946(0.019) | 0.909(0.027) | 0.818(0.048) | 0.518(0.093) |
| **SN=5000** | 0.999(0.002) | 1.000(0) | 0.998(0.006) | 0.992(0.011) | 0.985(0.015) | 0.972(0.016) | 0.955(0.019) | 0.917(0.027) | 0.837(0.038) | 0.525(0.088) |

TABLE VII: Computing time for 100 runs in total on simulated data via iLTS.

| time (second) | OP=5% | OP=10% | OP=15% | OP=20% | OP=25% | OP=30% | OP=35% | OP=40% | OP=45% | OP=50% |
|---|---|---|---|---|---|---|---|---|---|---|
| **SN=1000** | 4.38 | 3.92 | 3.65 | 3.58 | 3.54 | 3.61 | 3.71 | 3.75 | 3.66 | 3.73 |
| **SN=2000** | 6.54 | 5.93 | 5.62 | 5.32 | 5.29 | 5.33 | 5.28 | 5.19 | 5.13 | 5.15 |
| **SN=3000** | 8.86 | 8.14 | 7.62 | 7.31 | 7.11 | 6.98 | 7.05 | 7.07 | 7.15 | 7.02 |
| **SN=4000** | 11.01 | 10.32 | 9.67 | 9.37 | 8.67 | 8.87 | 7.82 | 8.51 | 8.78 | 8.81 |
| **SN=5000** | 13.23 | 12.36 | 12.14 | 11.73 | 12.04 | 11.59 | 11.03 | 10.82 | 10.79 | 10.49 |

TABLE VIII: Computing time for 100 runs in total on simulated data via LASSO.

| time (second) | OP=5% | OP=10% | OP=15% | OP=20% | OP=25% | OP=30% | OP=35% | OP=40% | OP=45% | OP=50% |
|---|---|---|---|---|---|---|---|---|---|---|
| **SN=1000** | 625.14 | 673.75 | 690.31 | 625.85 | 595.65 | 636.35 | 592.65 | 595.15 | 638.65 | 560.71 |
| **SN=2000** | 905.04 | 973.64 | 938.37 | 1017.06 | 791.37 | 887.72 | 855.61 | 825.98 | 818.99 | 806.25 |
| **SN=3000** | 1116.23 | 1167.45 | 1184.89 | 1127.83 | 1118.26 | 1032.88 | 916.89 | 952.67 | 822.35 | 929.75 |
| **SN=4000** | 1158.67 | 1256.82 | 1305.28 | 1227.76 | 1161.78 | 1087.81 | 1016.97 | 1035.82 | 948.75 | 1011.45 |
| **SN=5000** | 1288.02 | 1375.14 | 1368.75 | 1256.89 | 1228.56 | 1104.32 | 992.46 | 976.06 | 1034.12 | 1077.93 |

when the ground-truth outliers are not available. In this section, we show the effectiveness of the proposed method on simulated data with known ground truth outliers and on real-world datasets without ground truth outliers. The codes for the numerical experiments and the real-world datasets can be downloaded from https://code.google.com/p/irls/.

## A. Simulated data

The simulated data is constructed as follows. A random total order on $n$ candidates is created as the ground-truth order. Then we add paired comparison edges $(i, j)$ randomly with preference directions following the ground-truth order. We simulate the outliers by randomly choosing a portion of the comparison edges and reversing them in preference direction. A paired comparison graph with outliers, possibly incomplete and imbalanced, is constructed.

Here we choose $n = 16$, which is consistent with the real-world datasets, and make the following definitions for the experimental parameters. The total number of paired comparisons occurred on this graph is **SN** (Sample Number), and the number of outliers is **ON** (Outlier Number). Then the outlier percentage **OP** can be obtained as **ON/SN**.

Most outlier detection algorithms adopt a tuning parameter ($t$) in order to select different amount of data samples as outliers [12] and the number of outliers detected changes as $t$ changes. If $t$ is picked too restrictively, then the algorithm will miss true outlier points (false negatives). On the other hand, if the algorithm declares too many data samples as outliers, then it will lead to too many false positives. This tradeoff can be measured in terms of ***precision*** and ***recall***, which are commonly used for measuring the effectiveness of outlier detection methods. Specifically, the ***precision*** is defined as the percentage of reported outliers, which truly turn out to be outliers; and the ***recall*** is correspondingly defined as the percentage of ground-truth outliers, which have been reported as outliers.

The proposed method iLTS (Algorithm 2) is compared with LASSO [12] for outlier detection on the simulated data. For the ease of comparison, here we should tell LASSO in advance the exact percentage of outliers exist in the dataset.

The ***precision***s, ***recall***s and ***F1-score***s with standard deviations (**sd**) over 100 runs for these two methods on different choices of **SN** and **ON** are shown in Tables I, II, III, IV, V, and VI. ***F1-score*** is a combined measure that assesses the ***precision/recall*** tradeoff, which reaches its best value at 1 and worst score at 0. It can be defined as follows:

$$\boldsymbol{F1} = 2 \cdot \frac{\boldsymbol{precision} \cdot \boldsymbol{recall}}{\boldsymbol{precision} + \boldsymbol{recall}}. \tag{14}$$

When the number of outliers is not too large (i.e., **OP** $\leq$ 40 %), iLTS could produce better performance (indicated by higher ***precision***s, ***recall***s, and ***F1-score***s) than LASSO. When **OP** = 50%, i.e., half of the edges are reverted by outliers, both of these two methods show a rapid decrease of ***precision***, ***recall***, and ***F1*** to about 0.5, which is the performance of random guess. It is impossible to distinguish the true signal from noise by any method when more than half of the edges are perturbed, thus a phase transition can be observed in the tables. The worse performance of iLTS for high **OP**s is because the number of outliers estimated by iLTS is smaller than the exact number of outliers when the percentage of outliers is too high, which is further confirmed by the ***precision***s and ***recall***s for **OP** = 50%. When **OP** = 50%, the ***recall***s are less than 0.5 (i.e., there are more false negatives than true positives), and ***precision***s are greater than 0.5 (i.e., there are more true positives than false positives). Therefore, the number of true positives and false positives (the estimated number of outliers) is smaller than the number of true positives and false negatives (the exact number of outliers).

In addition, we compare the computing time required for these two methods to finish all the 100 runs in Tables VII and VIII. All computation is done using MATLAB R2010a on a Lenovo laptop running Windows 7 with 2.40 GHz Intel

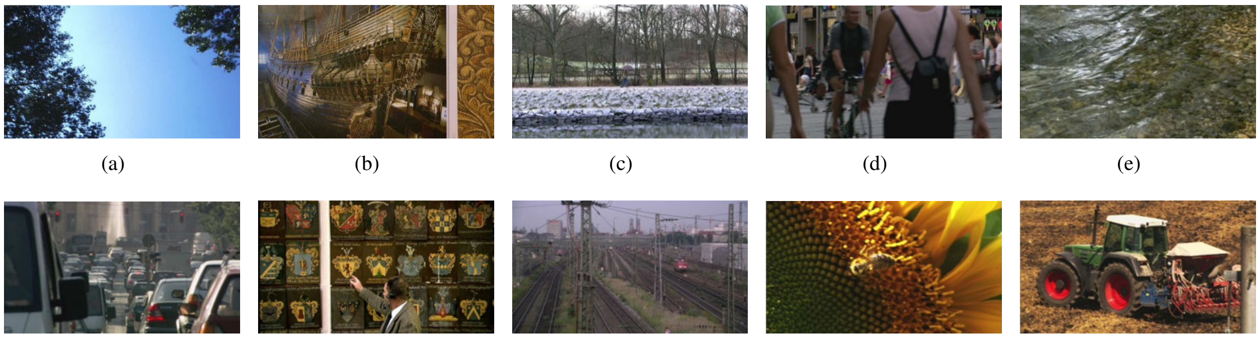

Fig. 1: Reference videos in LIVE database.

Core i7-3630QM and 8 GB 1600 MHz DDR3 memory. It is easy to see that on the simulated dataset, iLTS can achieve up to about 190 times faster than LASSO. Take (SN = 1000, OP = 15%) as an example, LASSO needs 690.31s for 100 runs, while iLTS only needs 3.65s, which is almost 190 times faster than LASSO. Note that, in most cases iLTS could automatically determine the number of outliers exist in the dataset without *a priori* knowledge.

## *B. Real-world Data*

Two real-world datasets are adopted in this subsection. The first dataset PC-VQA, which is collected by [8], contains 38,400 paired comparisons of the LIVE dataset [69] (Figure 1) from 209 random observers. An attractive property of this dataset is that the paired comparison data is complete and balanced. As LIVE includes 10 different reference videos and 15 distorted versions of each reference video (obtained using four different distortion processes: MPEG-2 compression, H.264 compression, lossy transmission of H.264 compressed bitstreams through simulated IP networks, and lossy transmission of H.264 compressed bitstreams through simulated wireless networks), for a total of 160 videos. A round of complete comparisons for this video database requires $10 \times \binom{16}{2} = 1200$ comparisons, and 38,400 comparisons correspond to 32 complete rounds.

There is no ground-truth for outliers in these real-world datasets, and we can not compute ***precision*** and ***recall*** as in the simulated data to evaluate the performance of outlier detection methods. Therefore, we inspect the outliers returned and compare them with the whole data to see if they are reasonably good outliers.

We take reference (a) in the PC-VQA dataset as an illustrative example (other reference videos exhibit similar results). We compare iLTS and LASSO again in the real-world datasets. The number of outliers estimated by iLTS is used for LASSO to choose regularization parameters and select the outliers. Outliers detected by both methods are shown in the paired comparison matrix in Table IX. The paired comparison matrix is constructed as follows (Table X is constructed in the same way). For each video pair $\{i, j\}$, let $n_{ij}$ be the number of comparisons, among which $a_{ij}$ raters agree that the quality of $i$ is better than $j$ ($a_{ji}$ carries the opposite meaning). So $a_{ij} + a_{ji} = n_{ij}$ if no tie occurs, and in the PC-VQA dataset, $n_{ij} \equiv 32$ for all videos. The order of the video ID in this Table is arranged from high to low according to the global ranking score calculated by the least squares method (2). The outliers picked out by both methods are mainly distributed in the lower left corner of this matrix, which implies that the outliers are those preference orders with a large deviation from the global ranking scores by L2. The total number of outliers estimated by iLTS from this reference video is 761, so the outlier percentage (**OP**) = 761/3840 = 18.65%. For comparison, we also inspect the top 18.65% returned by LASSO. It is easy to see that outliers returned by iLTS and LASSO are almost the same except one pair (ID = 3 and ID = 4). In the dataset, 15 raters agree that the quality of ID = 3 is better than that of ID = 4, while 17 raters have the opposite opinion. iLTS treats $a_{3,4} = 15$ as outliers, while LASSO chooses the opposite direction (i.e., treats $a_{4,3} = 17$ as outliers). LASSO tends to choose outliers as the large deviation from the gradients of global ranking scores while iLTS prefers to choose the minority in a paired comparison data. Such a small difference only leads to a local order change of nearby ranked items, ID = 3 and ID = 4. Therefore the ranking algorithms are stable.

The global ranking scores of these three algorithms, namely L2, LASSO, and iLTS, are shown in Table XI(a). Removing the top 18.65% outliers in both LASSO and iLTS changes the orders of some competitive videos. Both LASSO and iLTS think ID = 12 has better performance than ID = 3 and ID = 4. The scores of LASSO and iLTS are quite similar except that the orders and scores of ID = 3 and ID = 4 are exchanged, because LASSO and iLTS choose different preference directions as outliers.

The effectiveness of iLTS is demonstrated on a complete and balanced dataset, and we want to show the effectiveness of iLTS on incomplete and imbalanced datasets. The PC-IQA dataset is taken into consideration. This dataset contains 15 reference images and 15 distorted versions of each reference image, for a total of 240 images, which come from two publicly available datasets: LIVE [69] and IVC [70] (Figure 2). The distorted images in the LIVE dataset [69] are obtained using five different distortion

TABLE IX: Paired comparison matrices of reference (a) in PC-VQA dataset. Red numbers are outliers obtained by both iLTS and LASSO. Open blue circles are those obtained by LASSO but not iLTS, while filled blue circles are obtained by iLTS but not LASSO.

| Video ID | **1** | **9** | **10** | **13** | **7** | **8** | **11** | **14** | **15** | **3** | **12** | **4** | **16** | **5** | **6** | **2** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 0 | 22 | 29 | 30 | 30 | 29 | 29 | 29 | 30 | 28 | 29 | 32 | 32 | 31 | 32 | 31 |
| **9** | 10 | 0 | 22 | 20 | 14 | 23 | 23 | 25 | 29 | 29 | 32 | 30 | 29 | 30 | 29 | 31 |
| **10** | 3 | 10 | 0 | 22 | 11 | 21 | 29 | 23 | 31 | 27 | 31 | 30 | 32 | 30 | 32 | 31 |
| **13** | 2 | 12 | 10 | 0 | 18 | 22 | 23 | 27 | 31 | 28 | 29 | 29 | 29 | 25 | 27 | 28 |
| **7** | 2 | 18 | 21 | 14 | 0 | 21 | 14 | 16 | 28 | 23 | 31 | 25 | 19 | 27 | 26 | 28 |
| **8** | 3 | 9 | 11 | 10 | 11 | 0 | 25 | 14 | 28 | 25 | 29 | 27 | 24 | 25 | 28 | 32 |
| **11** | 3 | 9 | 3 | 9 | 18 | 7 | 0 | 22 | 27 | 26 | 26 | 30 | 30 | 27 | 27 | 31 |
| **14** | 3 | 7 | 9 | 5 | 16 | 18 | 10 | 0 | 28 | 27 | 18 | 29 | 29 | 26 | 28 | 29 |
| **15** | 2 | 3 | 1 | 1 | 4 | 4 | 5 | 4 | 0 | 25 | 20 | 22 | 26 | 25 | 29 | 24 |
| **3** | 4 | 3 | 5 | 4 | 9 | 7 | 6 | 5 | 7 | 0 | 11 | ⓯ | 26 | 24 | 29 | 28 |
| **12** | 3 | 0 | 1 | 3 | 1 | 3 | 6 | 14 | 12 | 21 | 0 | 16 | 20 | 24 | 26 | 26 |
| **4** | 0 | 2 | 2 | 3 | 7 | 5 | 2 | 3 | 10 | ⑰ | 16 | 0 | 15 | 26 | 27 | 30 |
| **16** | 0 | 3 | 0 | 3 | 13 | 8 | 2 | 3 | 6 | 6 | 12 | 17 | 0 | 22 | 24 | 28 |
| **5** | 1 | 2 | 2 | 7 | 5 | 7 | 5 | 6 | 7 | 8 | 8 | 6 | 10 | 0 | 26 | 27 |
| **6** | 0 | 3 | 0 | 5 | 6 | 4 | 5 | 4 | 3 | 3 | 6 | 5 | 8 | 6 | 0 | 21 |
| **2** | 1 | 1 | 1 | 4 | 4 | 0 | 1 | 3 | 8 | 4 | 6 | 2 | 4 | 5 | 11 | 0 |

TABLE X: Paired comparison matrices of reference (c) in PC-IQA dataset. Red numbers, open blue circles, and filled blue circles carry the same meanings with Table IX.

| **Image ID** | **1** | **8** | **16** | **2** | **3** | **11** | **6** | **12** | **9** | **14** | **5** | **13** | **7** | **10** | **15** | **4** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 0 | 13 | 9 | 16 | 19 | 12 | 15 | 13 | 14 | 14 | 14 | 17 | 16 | 17 | 16 | 16 |
| **8** | 6 | 0 | 8 | 7 | 8 | 5 | 13 | 7 | 7 | 8 | 19 | 8 | 15 | 9 | 12 | 15 |
| **16** | 4 | 0 | 0 | 9 | 11 | 9 | 8 | 15 | 3 | 18 | 16 | 17 | 12 | 7 | 21 | 18 |
| **2** | 5 | 5 | 6 | 0 | 8 | 9 | 10 | 11 | 7 | 14 | 13 | 14 | 14 | 13 | 14 | 15 |
| **3** | 3 | 4 | 6 | 7 | 0 | 6 | 11 | 9 | 10 | 16 | 12 | 15 | 14 | 14 | 18 | 13 |
| **11** | 4 | 6 | 3 | 5 | 6 | 0 | ❺ | 3 | 5 | 6 | 21 | 5 | 11 | 7 | 12 | 18 |
| **6** | 0 | 2 | 7 | 4 | 2 | ⑦ | 0 | 12 | 12 | 7 | 22 | 15 | 17 | 13 | 13 | 17 |
| **12** | 3 | 4 | 1 | 4 | 4 | 3 | 1 | 0 | 8 | 15 | 18 | 12 | 9 | 8 | 13 | 17 |
| **9** | 1 | 3 | 3 | 5 | 1 | 3 | 1 | 0 | 0 | 5 | 18 | 10 | 14 | 9 | 7 | 16 |
| **14** | 0 | 0 | 1 | 0 | 0 | 3 | 7 | 2 | 1 | 0 | 14 | 15 | 10 | 8 | 17 | 19 |
| **5** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 14 | 19 | 19 | 15 | 17 |
| **13** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 6 | 0 | 5 | 7 | 17 | 16 |
| **7** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 8 | 9 | 18 |
| **10** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 2 | 0 | ❸ | 11 |
| **15** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ⑤ | 0 | 11 |
| **4** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 6 | 6 | 0 |

Fig. 2: Reference images in the LIVE and IVC datasets. (The first six images are from the LIVE dataset and the remaining nine images are from the IVC dataset.)

processes: JPEG2000, JPEG, White Noise, Gaussian Blur, and Fast Fading Rayleigh, while the distorted images in the IVC dataset [70] are derived from four distortion types — JPEG2000, JPEG, LAR Coding, and Blurring. Totally, 186 observers, each of whom performs a varied number of comparisons via Internet, provide 23,097 paired comparisons for subjective IQA.

Table X shows the comparable experimental results of iLTS vs. LASSO on a randomly selected reference image (image (c) in Figure 2). Similar observations as above can be made and we note that outliers distributed on this dataset are much sparser than PC-VQA, shown by many zeros in the lower left corner of the paired comparison matrix. Outlier percentage (OP) returned by iLTS is 173/1655 = 10.45% in Table X, and it is easy to find that the detection results of LASSO vs. iLTS are different on two pairs: 1) ID = 6 and ID = 11; 2) ID = 10 and ID = 15. Similar to the last experiment, iLTS prefers to choose the minority in paired comparisons, i.e., the 5 in $11 \succ 6$ and the 3 in $10 \succ 15$, while LASSO selects outliers as the large deviation from the gradients of global ranking scores even when the votings are in majority. Such a difference leads to a local order change of involved items which are adjacent in ranking list, exhibiting stability in global rankings, as shown in Table XI(b).

## C. *Discussion*

As we have seen in the numerical experiments, iLTS and LASSO mostly find the same outliers and when they disagree, iLTS tends to choose the minority and LASSO prefers to choose outliers as the large deviation from the gradients of global ranking scores even when the votings are in majority. When outliers consist of minority voting as in simulated experiments, iLTS may perform better. Besides, iLTS tents to choose fewer outliers to make sure that there are no outliers in the remaining comparisons. This can also be explained from the algorithm. We choose a small initial estimation for the number of outliers, and increase this estimation until there is no outliers in the remaining comparisons. The parameter $\beta_2 > 1$ is chosen to be small so we will not overestimate the number of outliers too much.

Finally, we would like to point out that subject-based outlier detection can be a straightforward extension from our proposed iLTS. From the detection results of iLTS, one may evaluate the reliability of one participant based on all the comparisons from the participant, and drop unreliable participants.

## V. CONCLUSIONS

In this paper, we have proposed a fast and adaptive algorithm iLTS for outlier detection and robust ranking in

TABLE XI: Comparison of different rankings. Three ranking methods are compared with the integer representing the ranking position and the number in parentheses representing the global ranking score returned by the corresponding algorithm.

(a) Reference (a) in the PC-VQA dataset

| Video ID | L2 | LASSO | iLTS |
|---|---|---|---|
| 1 | 1 ( 0.7930 ) | 1 ( 0.9123 ) | 1 ( 0.9129 ) |
| 9 | 2 ( 0.5312 ) | 2 ( 0.7537 ) | 2 ( 0.7539 ) |
| 10 | 3 ( 0.4805 ) | 3 ( 0.6317 ) | 3 ( 0.6322 ) |
| 13 | 4 ( 0.3906 ) | 4 ( 0.5522 ) | 4 ( 0.5524 ) |
| 7 | 5 ( 0.2852 ) | 5 ( 0.4533 ) | 5 ( 0.4537 ) |
| 8 | 6 ( 0.2383 ) | 6 ( 0.3159 ) | 6 ( 0.3163 ) |
| 11 | 7 ( 0.2148 ) | 7 ( 0.2113 ) | 7 ( 0.2120 ) |
| 14 | 8 ( 0.1641 ) | 8 ( 0.1099 ) | 8 ( 0.1103 ) |
| 15 | 9 ( -0.1758 ) | 9 ( -0.1024 ) | 9 ( -0.1029 ) |
| 3 | 10 ( -0.2227 ) | 11 ( -0.3195 ) | 12 ( -0.3999 ) |
| 12 | 11 ( -0.2500 ) | 10 ( -0.2149 ) | 10 ( -0.2158 ) |
| 4 | 12 ( -0.2930 ) | 12 ( -0.4054 ) | 11 ( -0.3252 ) |
| 16 | 13 ( -0.3633 ) | 13 ( -0.5311 ) | 13 ( -0.5332 ) |
| 5 | 14 ( -0.4414 ) | 14 ( -0.6573 ) | 14 ( -0.6568 ) |
| 6 | 15 ( -0.6289 ) | 15 ( -0.8054 ) | 15 ( -0.8057 ) |
| 2 | 16 ( -0.7227 ) | 16 ( -0.9046 ) | 16 ( -0.9042 ) |

(b) Reference (c) in the PC-IQA dataset

| Image ID | L2 | LASSO | iLTS |
|---|---|---|---|
| 1 | 1 ( 0.7575 ) | 1 ( 0.9015 ) | 1 ( 0.9022 ) |
| 8 | 2 ( 0.5670 ) | 2 ( 0.7088 ) | 2 ( 0.7129 ) |
| 16 | 3 ( 0.5124 ) | 3 ( 0.6472 ) | 3 ( 0.6504 ) |
| 2 | 4 ( 0.4642 ) | 4 ( 0.5242 ) | 4 ( 0.5248 ) |
| 3 | 5 ( 0.4423 ) | 5 ( 0.4119 ) | 5 ( 0.4148 ) |
| 11 | 6 ( 0.3277 ) | 6 ( 0.2592 ) | 7 ( 0.1763 ) |
| 6 | 7 ( 0.3128 ) | 7 ( 0.2515 ) | 6 ( 0.3124 ) |
| 12 | 8 ( 0.2423 ) | 8 ( 0.1209 ) | 8 ( 0.1261 ) |
| 9 | 9 ( 0.1453 ) | 9 ( 0.0043 ) | 9 ( 0.0069 ) |
| 14 | 10 ( -0.0455 ) | 10 ( -0.1274 ) | 10 ( -0.1243 ) |
| 5 | 11 ( -0.3376 ) | 11 ( -0.3205 ) | 11 ( -0.3214 ) |
| 13 | 12 ( -0.4785 ) | 12 ( -0.4621 ) | 12 ( -0.4560 ) |
| 7 | 13 ( -0.5396 ) | 13 ( -0.5515 ) | 13 ( -0.5494 ) |
| 10 | 14 ( -0.7486 ) | 14 ( -0.7005 ) | 15 ( -0.7485 ) |
| 15 | 15 ( -0.7658 ) | 15 ( -0.7511 ) | 14 ( -0.7106 ) |
| 4 | 16 ( -0.8559 ) | 16 ( -0.9163 ) | 16 ( -0.9166 ) |

QoE evaluation. It achieves up to 190 times faster than LASSO in outlier detection. Moreover, this method can automatically estimate the number of outliers and detect them without any priori information about the number of outliers existing in the dataset. The effectiveness and efficiency of iLTS is demonstrated on both simulated examples and real-world applications. iLTS exhibits comparable accuracy to LASSO in outlier detection. There are small distinctions between them indicating that iLTS prefers to choose minority voting data as outliers, while the LASSO selects large deviations from the gradient of global ranking score as outliers even when they are in majority voting. In both cases, the global rankings obtained are stable. A future direction is to understand under what kind of conditions such an adaptive least trimmed squares algorithm works.

In summary, we expect that the proposed iLTS for QoE evaluations will be a helpful tool for people in the multimedia community exploiting crowdsourceable paired comparison data for robust ranking.

## References

[1] R. Schatz, T. Hoßfeld, L. Janowski, and S. Egger, "From packets to people: Quality of experience as new measurement challenge," in *Data Traffic Monitoring and Analysis: From measurement, classification and anomaly detection to Quality of experience*. Springer's Computer Communications and Networks series, 2012.

[2] C.-C. Wu, K.-T. Chen, Y.-C. Chang, and C.-L. Lei, "Crowdsourcing multimedia QoE evaluation: A trusted framework," *IEEE Transactions on Multimedia*, vol. 15, no. 5, pp. 1121–1137, 2013.

[3] *Methods for subjective determination of transmission quality, ITU-R Recommendation*. Rec. ITU-T-P.800, 1996.

[4] K.-T. Chen, C.-C. Wu, Y.-C. Chang, and C.-L. Lei, "A crowdsourceable QoE evaluation framework for multimedia content." ACM Multimedia, 2009, pp. 491–500.

[5] X. Jiang, L.-H. Lim, Y. Yao, and Y. Ye., "Statistical ranking and combinatorial Hodge theory," *Mathematical Programming*, vol. 127, no. 6, pp. 203–244, 2011.

[6] J. Howe, "The rise of crowdsourcing," *Wired Magazine*, vol. 14, no. 6, pp. 176–183, 2006.

[7] A. Eichhorn, P. Ni, and R. Eg, "Randomised pair comparison: an economic and robust method for audiovisual quality assessment." International Workshop on Network and Operating Systems Support for Digital Audio and Video, 2010, pp. 63–68.

[8] Q. Xu, T. Jiang, Y. Yao, Q. Huang, B. Yan, and W. Lin, "Random partial paired comparison for subjective video quality assessment via HodgeRank." ACM Multimedia, 2011, pp. 393–402.

[9] Q. Xu, Q. Huang, T. Jiang, B. Yan, W. Lin, and Y. Yao, "HodgeRank on random graphs for subjective video quality assessment," *IEEE Transactions on Multimedia*, vol. 14, no. 3, pp. 844–857, 2012.

[10] Q. Xu, Q. Huang, and Y. Yao, "Online crowdsourcing subjective image quality assessment." ACM Multimedia, 2012, pp. 359–368.

[11] Q. Xu, J. Xiong, Q. Huang, and Y. Yao, "Online HodgeRank on random graphs for crowdsourceable QoE evaluation," *IEEE Transactions on Multimedia*, vol. 16, no. 2, pp. 373–386, 2014.

[12] ——, "Robust evaluation for quality of experience in crowdsourcing," in *ACM Multimedia*, 2013, pp. 43–52.

[13] Y. She and A. B. Owen, "Outlier detection using nonconvex penalized regression," *Journal of the American Statistical Association*, vol. 106, no. 494, pp. 626–639, 2011.

[14] W. Lin and C.-C. J. Kuo, "Perceptual visual quality metrics: A survey," *Journal of Visual Communication and Image Representation*, vol. 22, no. 4, pp. 297–312, 2011.

[15] O. Alonso, D. Rose, and B. Stewart, "Crowdsourcing for relevance evaluation," *SIGIR Forum*, vol. 42, no. 2, pp. 9–15, 2008.

[16] A. Kittur, E. Chi, and B. Suh, "Crowdsourcing user studies with Mechanical Turk." SIGCHI conference on Human factors in computing systems, 2008, pp. 453–456.

[17] A. Sorokin and D. Forsyth, "Utility data annotation with Amazon Mechanical Turk." Computer Vision and Pattern Recognition Workshops, June 2008, pp. 1–8.

[18] S. Nowak and S. Ruger, "How reliable are annotations via crowdsourcing: a study about inter-annotator agreement for multi-label image annotation." International conference on Multimedia information retrieval, 2010, pp. 557–566.

[19] M. Soleymani, M. Caro, E. Schmidt, C. Sha, and Y. Yang, "1000 songs for emotional analysis of music." ACM international workshop on Crowdsourcing for multimedia, 2013, pp. 1–6.

[20] G. Tavares, A. Mourao, and J. Magalhaes, "Crowdsourcing for affective-interaction in computer games." ACM international workshop on Crowdsourcing for multimedia, 2013, pp. 7–12.

[21] B. Gardlo, M. Ries, and T. Hoßfeld, "Impact of screening technique on crowdsourcing qoe assessments." International Conference Radioelektronika, Special Session on Quality in multimedia systems, 2012.

[22] C. Keimel, J. Habigt, and K. Diepold, "Challenges in crowd-based video quality assessment," in *International Workshop on Quality of Multimedia Experience*, 2012.

[23] M. de Condorcet, "Éssai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix (essay on the application of analysis to the probability of majority decisions)," *Imprimerie Royale, Paris*, 1785.

[24] K. J. Arrow, *Social Choice and Individual Values, 2nd Ed.* Yale University Press, New Haven, CT, 1963.

[25] L. Thurstone, "A law of comparative judgement," *Psychological Review*, vol. 34, pp. 278–286, 1927.

[26] T. L. Saaty, "A scaling method for priorities in hierarchical structures," *Journal of Mathematical Psychology*, vol. 15, no. 3, pp. 234–281, 1977.

[27] G. Noether, "Remarks about a paired comparison model," *Psychometrika*, vol. 25, pp. 357–367, 1960.

[28] H. David, *The method of paired comparisons*, ser. 2nd Ed., Griffin's Statistical Monographs and Courses, 41. Oxford University Press, New York, NY, 1988.

[29] S. X. Yu, "Angular embedding: from jarring intensity differences to perceived luminance," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 2302–2309.

[30] ——, "Angular embedding: A robust quadratic criterion," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 1, pp. 158–173, 2012.

[31] W. Ma, J. M. Morel, S. Osher, and A. Chien, "An $L_1$-based variational model for retinex theory and its application to medical images," in *Computer Vision Pattern Recognition*, 2011, pp. 153–160.

[32] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," in *International Conference on World Wide Web*, 1998, pp. 107–117.

[33] J. Kleinberg, "Authoritative sources in a hyperlinked environment," *Journal of the ACM*, vol. 46, no. 5, pp. 604–632, 1999.

[34] C. Cortes, M. Mohri, and A. Rastogi, "Magnitude-preserving ranking algorithms," vol. 24, 2007, pp. 169–176.

[35] A. Rajkumar and S. Agarwal, "A statistical convergence perspective of algorithms for rank aggregation from pairwise data," in *International Conference on Machine Learning*, 2014, pp. 118–126.

[36] R. T. Stefani, "Football and basketball predictions using least squares," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 7, pp. 117–121, 1977.

[37] Y. Sismanis, "How I won the "chess ratings – Elo vs. the Rest of the world" competition," *http://arxiv.org/abs/1012.4571v1*, 2010.

[38] B. Osting, J. Darbon, and S. Osher, "Statistical ranking using the $l_1$-norm on graphs." *AIMS Journal on Inverse Problems and Imaging*, vol. 7, no. 3, pp. 907–926, 2013.

[39] N. Ailon, "An active learning algorithm for ranking from pairwise preferences with an almost optimal query complexity," *Journal of Machine Learning Research*, vol. 13, pp. 137–164, 2012.

[40] K. G. Jamieson and R. D. Nowak, "Active ranking using pairwise comparisons," *Annual Conference on Neural Information Processing Systems*, pp. 2240–2248, 2011.

[41] Y. Cao, J. Xu, T.-Y. Liu, H. Li, Y. Huang, and H.-W. Hon, "Adapting ranking SVM to document retrieval," in *ACM Special Interest Group on Information Retrieval*, 2006, pp. 186–193.

[42] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender, "Learning to rank using gradient descent," in *International Conference on Machine Learning*, 2005, pp. 89–96.

[43] C. J. C. Burges, R. Ragno, and Q. V. Le, "Learning to rank with nonsmooth cost functions," in *Annual Conference on Neural Information Processing Systems*, 2006, pp. 193–200.

[44] X. Chen, P. N. Bennett, K. Collins-Thompson, and E. Horvitz, "Pairwise ranking aggregation in a crowdsourced setting," in *International conference on Web search and data mining*, 2013, pp. 193–202.

[45] J. Yi, R. Jin, S. Jain, and A. K. Jain, "Inferring users' preferences from crowdsourced pairwise comparisons: A matrix completion approach," in *AAAI Conference on Human Computation and Crowdsourcing*, 2013, pp. 207–215.

[46] S. Negahban, S. Oh, and D. Shah, "Iterative ranking from pairwise comparisons," in *Annual Conference on Neural Information Processing Systems*, 2012, pp. 2483–2491.

[47] K.-T. Chen, C.-J. Chang, C.-C. Wu, Y.-C. Chang, and C.-L. Lei, "Quadrant of euphoria: a crowdsourcing platform for QoE assessment," *Network, IEEE*, vol. 24, no. 2, pp. 28–35, 2010.

[48] C. Keimel, J. Habigt, and K. Diepold, "Challenges in crowd-based video quality assessment," in *International Workshop on Quality of Multimedia Experience*, 2012, pp. 13–18.

[49] A. N. Hirani, K. Kalyanaraman, and S. Watts, "Least squares ranking on graphs," *arXiv:1011.1716v4*, 2011.

[50] B. Osting, J. Darbon, and S. Osher, "Statistical ranking using the $l_1$-norm on graphs," *Inverse Problems & Imaging*, vol. 7, no. 3, 2013.

[51] B. Osting, C. Brune, and S. Osher, "Enhanced statistical rankings via targeted data collection," in *International Conference on Machine Learning*, 2013, pp. 489–497.

[52] O. Candogan, I. Menache, A. Ozdaglar, and P. A. Parrilo, "Flows and decompositions of games: Harmonic and potential games," *Mathematics of Operations Research*, vol. 36, no. 3, pp. 474–503, 2011.

[53] A. J. Chorin and J. E. Marsden, *A Mathematical Introduction to Fluid Mechanics*, ser. Texts in Applied Mathematics. Springer, 1993.

[54] J. Yuan, G. Steidl, and C. Schnorr, "Convex Hodge decomposition and regularization of image flows," *Journal of Mathematical Imaging and Vision*, vol. 33, no. 2, pp. 169–177, 2009.

[55] P. Erdos and A. Renyi, "On random graphs i," *Publicationes Mathematicae-Debrecen*, vol. 6, pp. 290–297, 1959.

[56] N. Wormald, "Models of random regular graphs." In Surveys in Combinatorics, 1999, pp. 239–298.

[57] A.-L. Barabasi and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999.

[58] D. Watts and S. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, no. 393, pp. 440–442, 1998.

[59] M. Penrose, *Random Geometric Graphs (Oxford Studies in Probability)*. Oxford University Press, 2003.

[60] D. Hawkins, *Identification of outliers*. Springer, 1980, vol. 11.

[61] S. Papadimitriou, H. Kitagawa, P. Gibbons, and C. Faloutsos, "Loci: Fast outlier detection using the local correlation integral," in *IEEE International Conference on Data Engineering*, 2003, pp. 315–326.

[62] T. Hossfeld, C. Keimel, M. Hirth, B. Gardlo, J. Habigt, K. Diepold, and P. Tran-Gia, "Best practices for QoE crowdtesting: QoE assessment with crowdsourcing," *IEEE Transactions on Multimedia*, vol. 16, no. 2, pp. 541–558, 2014.

[63] P. J. Huber, *Robust Statistics*. New York: Wiley, 1981.

[64] M. Yan, "Restoration of images corrupted by impulse noise and mixed gaussian impulse noise using blind inpainting," *SIAM Journal on Imaging Sciences*, vol. 6, no. 3, pp. 1227–1245, 2013.

[65] M. Yan, Y. Yang, and S. Osher, "Robust 1-bit compressive sensing using adaptive outlier pursuit," *IEEE Transactions on Signal Processing*, vol. 60, no. 7, pp. 3868–3875, 2012.

[66] X. Zeng and M. A. T. Figueiredo, "Robust binary fused compressive sensing using adaptive outlier pursuit," *CoRR*, vol. abs/1402.5076, 2014.

[67] M. Yan, Y. Yang, and S. Osher, "Exact low-rank matrix completion from sparsely corrupted entries via adaptive outlier pursuit," *Journal of Scientific Computing*, vol. 56, no. 3, pp. 433–449, 2013.

[68] P. J. Rousseeuw and A. M. Leroy, *Robust Regression and Outlier Detection*. Wiley, 1987.

[69] "LIVE image & video quality assessment database." `http://live.ece.utexas.edu/research/quality/`, 2008.

[70] "Subjective quality assessment irccyn/ivc database." `http://www2.irccyn.ec-nantes.fr/ivcdb/`, 2005.