

Selecting the top-quality item through crowd scoring

Alessandro Nordio, Alberto Tarable
CNR-IEIIT, Torino, Italy

Emilio Leonardi, Marco Ajmone Marsan
Politecnico di Torino and CNR-IEIIT, Italy

Abstract—We investigate crowdsourcing algorithms for finding the top-quality item within a large collection of objects with unknown intrinsic quality values. This is an important problem with many relevant applications, for example in networked recommendation systems. The core of the algorithms is that objects are distributed to crowd workers, who return a noisy evaluation. All received evaluations are then combined, to identify the top-quality object. We first present a simple probabilistic model for the system under investigation. Then, we devise and study a class of efficient adaptive algorithms to assign in an effective way objects to workers. We compare the performance of several algorithms, which correspond to different choices of the design parameters/metrics. We finally compare our approach based on scoring object qualities against traditional proposals based on comparisons and tournaments.

I. INTRODUCTION

Crowdsourcing is a term often adopted to identify distributed systems that can be used for the solution of a wide range of complex problems by integrating a large number of human and/or computer efforts [1].

The key elements of a crowdsourcing system are: i) the availability of a large pool of individuals or machines (called *workers* in crowdsourcing jargon) that can offer their (small) contribution to the problem solution by executing a *task*; ii) an algorithm for the partition of the problem at hand into tasks; iii) an algorithm for the selection of workers and the distribution of tasks to the selected workers; iv) an algorithm for the combination of workers' *answers* into the final *solution* of the problem, v) a *requester* (a.k.a. employer), who uses the three algorithms above to structure his problem into a set of tasks, assign tasks to selected workers, and combine workers' answers to obtain the problem solution.

Since on the one hand answers may be subjective, and on the other task execution is typically tedious and the economic reward for workers is pretty small (if any), workers are not 100% reliable, in the sense that they may provide incorrect answers. In addition, workers may be biased for different reasons. Hence, the same task is normally assigned in parallel (replicated) to several workers, and then a decision rule is applied to their answers. A natural trade-off between the accuracy of the decision and cost arises; indeed, by increasing the replication factor of every task, we can increase the accuracy of the final decision about the task solution, but we necessarily incur higher costs (or, for a given fixed cost, we obtain a lower task throughput).

A number of sophisticated software platforms have been recently developed for the exploitation of the crowdsourcing paradigm on a scale that could not be possible without a

networked infrastructure. Some relevant application scenarios taken from the domains of recommendation and evaluation, are the development of hotel and restaurant rating systems, the implementation of recommendation systems for movies, the management of the review process of large conferences.

Given the scale of the current applications of crowdsourcing systems, the relevance of high-performance and scalable algorithms is enormous (in some cases, it can have huge economical impact).

An abstract view of the examples above reveals that they are very similar in nature: their essence is determining the best, or the k best, elements in a group of objects in which each object has an intrinsic (unknown) quality metric. This is a very fundamental algorithmic problem, which has already been investigated by several researchers in the context of crowdsourcing.

Most of the previous literature considers workers only able to directly compare items in groups comprising two or more objects, expressing a preference. The proposed algorithms consist of comparisons arranged in rounds, forming a *tournament*, and the investigation concentrates on the trade-offs that appear in this context (e.g., cost, accuracy, latency) [2]–[7]. In this paper, instead, we assume that workers are able to evaluate (in absolute terms) the quality of an object, providing a noisy score. A similar path was followed in the recent (still unpublished) work by Khan and Garcia Molina [8], which studies algorithms to find the maximum element in a group of objects, and discusses approaches based on comparisons, on ratings, as well as on a mix of the two possibilities. The main difference between their work and this one is in the quantization of workers' scores. Indeed, [8] assumes that workers' answers are coarsely quantized over few levels (typically three or five), and this makes objects with similar quality indistinguishable, so that direct comparisons and tournaments become necessary to break ties.

On the contrary, we first work with unquantized workers' answers, so as to maximize the amount of information provided by the workers. We show that in this context the scoring approach is superior (in some cases by far) to the approach based on direct comparisons. This should not be surprising, since quantization and comparisons entail a partial loss of information. Then, we show that by adopting smart quantization techniques with a sufficiently large number of quantization levels (in the order of few tens) we can closely approach the performance of systems operating on unquantized scores.

Another significant difference with respect to [8] is in the scope of the works. We aim at the definition of smart

multiround *adaptive* algorithms that effectively distribute the resources (workers) among the objects, at every round, making online decisions whether to distribute further resources, based on past collected answers. The paper [8], instead, focuses on non-adaptive algorithms distributing resources to objects according to a fixed, pre-established scheme.

Our main findings are:

- algorithms operating on unquantized scores are intrinsically more efficient than algorithms based on direct comparisons of objects;
- resources (workers) must be allocated in a careful manner, concentrating more resources on the top-quality objects; this can be done only if algorithms are adaptive and, at every round, exploit currently available information about objects’ quality to decide how to distribute further resources;
- if unquantized scores are available, tournament-based approaches, that partition objects into subgroups and move winners in each sub-group to the next round, may become extremely inefficient;
- more practical quantized schemes perform very close to their ideal unquantized counterparts, provided that a reasonable number of quantization levels is properly assigned to workers’ answers.

The rest of this paper is organized as follows. Section II presents our system assumptions. Section III describes the main characteristics of the class of algorithms that is investigated in this paper, and Section IV discusses the parameters for the algorithm design. Section V shows that, with our system assumptions, the approach based on quantitative evaluations of object qualities is superior to the one based on comparisons between objects. Section VI faces the problem of effective answer quantization. Section VII presents and discusses numerical results for the class of algorithms considered in this paper, comparing them to previous proposals. Section VIII briefly comments on the computational complexity of the most promising algorithm according to the numerical results of Section VII, and discusses parameter setting. Finally, Section IX contains our concluding remarks and presents the next steps of our work.

II. SYSTEM ASSUMPTIONS

We consider a set of N objects, each of which is endowed with an intrinsic quality, whose evaluation requires human capabilities. Let $\mathbf{x} = [x_1, \dots, x_N]$ be the vector of all quality values, which are supposed to be instances of the i.i.d. random variables $\mathbf{q} = [q_1, \dots, q_N]$ having common pdf f_q over \mathbb{R} .

Our goal is to use a crowdsourcing approach to identify the “best” object, that is, the object with the largest quality value, denoted by x_{i^*} , where

$$i^* = \arg \max_i x_i.$$

The crowd is composed of statistically identical workers, whose evaluation of the object quality is prone to errors. We assume, for the moment, that workers provide absolute

unquantized estimates (scores) of the intrinsic quality of individual objects, and we model as additive Gaussian noise the error made in such evaluation process. More precisely, if the i -th object is sent for evaluation, say, for the j -th time, to a worker in the crowd, the worker’s answer will be given by:

$$a_{ij} = x_i + n_{ij}$$

where n_{ij} , $i = 1, \dots, N$, $j = 1, 2, \dots$, are i.i.d. Gaussian random variables with zero mean and variance σ^2 . Errors may be caused either by subjective factors influencing the assessment of the intrinsic object quality, or by the fact that workers avoid devoting the effort that is needed for an accurate assessment. Observe that our model encompasses the case in which different workers may suffer from possibly different biases. However we make the conservative assumption that workers’ bias cannot be estimated. This assumption is justified by the fact that, since evaluations of objects are usually carried out by many workers, each one performing few evaluations, estimating the worker bias is very challenging in practical crowdsourcing systems.

Observe that our model differs from [2]–[7] because we assume that workers can provide absolute estimates (scores) of the intrinsic quality of objects, while [2]–[7] assume workers to be only able to perform noisy comparisons between groups of objects. We will show that, whenever applicable, the scoring approach is significantly more favorable, in the sense that algorithms dealing with absolute estimates of object quality achieve significantly better cost/performance trade-offs. We also wish to remark that the Gaussian model of worker error is in good agreement with Thurstone’s law of comparative judgment [9], according to which comparisons are based on latent quality estimations, whose distribution is Gaussian around the true quality value. In the context of crowdsourcing, the same model has recently been employed also in [8].

III. ALGORITHMIC APPROACH

We investigate a class of adaptive algorithms in which objects are sent out for evaluation through several *rounds*. In each round, each object receives a given number of evaluations by crowd workers (possibly zero, for some objects). Then, on the basis of all collected workers’ answers, the algorithms take decisions about the opportunity of requesting extra evaluations for a subset of the objects in a further round. If no extra evaluations are carried out, the algorithms terminate and a winner is identified.

More formally, denote with $m_i^{(\ell)}$ the number of evaluations the i -th object has received in round ℓ and with $M_i^{(\ell)}$ the total number of evaluations received by object i up to (and including) round ℓ . We define $\mathbf{a}_i^{(\ell)} = [a_{ij}^{(\ell)}]$, $j = 1, \dots, M_i^{(\ell)}$ as the vector¹ of random variables representing the answers about object i collected up to round ℓ , and $\mathbf{A}^{(\ell)} = [\mathbf{a}_1^{(\ell)}, \dots, \mathbf{a}_N^{(\ell)}]$.

At the beginning of round ℓ , with $\ell \geq 1$, a fitness index $\phi_i^{(\ell-1)}$ quantifies the current chances for object i to be the

¹In order to avoid cumbersome notation we sometimes indicate the vector $\mathbf{v} = [v_1, \dots, v_n]$, as $\mathbf{v} = [v_i]$, $i = 1, \dots, n$

winner at the end of the algorithm, as a result of processing of previous evaluations. Let the vector of all fitness indices be

$$\boldsymbol{\phi}^{(\ell-1)} = [\phi_1^{(\ell-1)}, \dots, \phi_N^{(\ell-1)}].$$

For $\ell = 1$, i.e., when no evaluations are available yet, fitness indices are equal for all objects, since the object qualities are assumed to be instances of i.i.d random variables.

In round ℓ , some of the objects may have a fitness index equal to $-\infty$. These objects are not assigned any further evaluation and are out of the contest. Define the contestant set $\mathcal{C}^{(\ell)}$ at round ℓ as the set of objects for which the fitness index is currently larger than $-\infty$, i.e.,

$$\mathcal{C}^{(\ell)} = \left\{ i \in \{1, \dots, N\} : \phi_i^{(\ell-1)} > -\infty \right\}.$$

We remark that, according to our algorithms, the contestant set at round $\ell + 1$ is always a (possibly improper) subset of the contestant set at round ℓ , i.e., $\mathcal{C}^{(\ell+1)} \subseteq \mathcal{C}^{(\ell)}$.

On the basis of $\boldsymbol{\phi}^{(\ell-1)}$ and, possibly, of the total number of past assignments $M^{(\ell-1)} = \sum_{i=1}^N M_i^{(\ell-1)}$, the algorithm decides, according to a termination rule, whether to stop or to go on with the rounds. If rounds are stopped, the object with the largest fitness index is declared the winner. Otherwise, a budget of new worker evaluations is assigned to objects. Such budget is dimensioned as

$$[m_1^{(\ell)}, \dots, m_N^{(\ell)}] = \mathcal{A}(\boldsymbol{\phi}^{(\ell-1)}, M^{(\ell-1)})$$

where $\mathcal{A}(\cdot)$ is the $\mathbb{R}^N \times \mathbb{N} \rightarrow \mathbb{N}^N$ allocation function. We assume function $\mathcal{A}(\cdot)$ to be increasing with respect to the object quality, i.e., our algorithm tends to allocate more workers to objects with top estimated quality, as formally stated below.

Property 1: If $\phi_i^{(\ell-1)} < \phi_j^{(\ell-1)}$, then $m_i^{(\ell)} \leq m_j^{(\ell)}$, with $m_i^{(\ell)} = 0$ if $\phi_i^{(\ell)} = -\infty$.

After determining the number of suitable evaluations, the objects are sent to crowd workers and answers are collected. Then, such answers, together with the previous ones, are used to update the fitness vector for next round, $\boldsymbol{\phi}^{(\ell)}$ (and, consequently, $\mathcal{C}^{(\ell+1)}$).

Several algorithms can be devised in accordance to the previous scheme, depending on how we select the different metrics and parameters, such as $\mathcal{A}(\cdot)$, the fitness index, and the termination rule.

IV. DESIGN PARAMETERS

A. Preliminary considerations

At round ℓ of the algorithm, on the basis of the collected answers we can compute an a-posteriori distribution $f_i^{(\ell)}(x|\mathbf{y}_i^{(\ell)})$ for the quality q_i , where $\mathbf{y}_i^{(\ell)}$ represents a realization of the random vector $\mathbf{a}_i^{(\ell)}$ (assumed unquantized for the moment). Thanks to Bayes' rule, such distribution can be written as:

$$f_i^{(\ell)}(x|\mathbf{y}_i^{(\ell)}) = \kappa \prod_{j=1}^{M_i^{(\ell)}} \exp\left(-\frac{(y_{ij}-x)^2}{2\sigma^2}\right) f_q(x) \quad (1)$$

where κ is such that $\int_{-\infty}^{+\infty} f_i^{(\ell)}(x|\mathbf{y}_i^{(\ell)}) dx = 1$. When the a-priori pdf f_q is Gaussian with mean μ_a and variance σ_a^2 , by substituting such prior into the above expression, we easily obtain that $f_i^{(\ell)}$ is also Gaussian, namely

$$f_i^{(\ell)}(x|\mathbf{y}_i^{(\ell)}) = \left(2\pi\rho_i^{2(\ell)}\right)^{-\frac{1}{2}} e^{-\frac{(x-\hat{x}_i^{(\ell)})^2}{2\rho_i^{2(\ell)}}}, \quad (2)$$

where

$$\hat{x}_i^{(\ell)} = \frac{\sum_{j=1}^{M_i^{(\ell)}} y_{ij} + \beta\mu_a}{M_i^{(\ell)} + \beta}, \quad \rho_i^{2(\ell)} = \frac{\sigma^2}{M_i^{(\ell)} + \beta},$$

and $\beta = \sigma^2/\sigma_a^2$. This is a consequence of the fact that the Gaussian distribution is self-conjugate with respect to the Gaussian likelihood function. It is also worth noting that $\hat{x}_i^{(\ell)}$ is the Minimum-Mean-Square-Error estimate of x_i given that $\mathbf{y}_i^{(\ell)}$ is the realization of $\mathbf{a}_i^{(\ell)}$. Observe that (2) also holds for unknown prior distribution. In such a case we set $\sigma_a = \infty$, i.e., $\beta = 0$. Moreover we remark that (2) can also be employed to obtain a simplified approximate a-posteriori distribution in the case of non-Gaussian priors with variance σ_a^2 .

B. Possible performance parameters

In order to properly choose the metrics of the crowdsourcing algorithm, it is important to identify the performance parameters we may want to optimize. Several options can be devised. In the following, we will omit the round index ℓ for ease of notation. Consider an answer realization $\mathbf{Y} = [\mathbf{y}_1, \dots, \mathbf{y}_N]$ and define the corresponding estimate $\hat{i}^*(\mathbf{Y})$ of i^* , in the following simply denoted by \hat{i}^* .

- A first possible performance parameter (to be minimized) is the *distortion* $D(\mathbf{Y}) = \mathbb{E} (q_{\hat{i}^*} - q_{i^*})^2$, which is averaged with respect to the current a-posteriori distribution of \mathbf{q} given $\mathbf{A} = \mathbf{Y}$. Unfortunately the computation of the distortion is in general too complex, even for moderate values of N .
- A generalization of the previous metric is the *order- k distortion* $D^{(k)}(\mathbf{Y}) = \mathbb{E} |q_{\hat{i}^*} - q_{i^*}|^k$, where $k \in \mathbb{N}$.
- A different performance parameter to be minimized can be the *error probability* $p_e(\mathbf{Y}) = \mathbb{P} \left\{ \hat{i}^* \neq i^* \right\}$. With such a choice, a maximum-a-posteriori (MAP) rule turns out to be optimal. Precisely, let $\pi_i(\mathbf{Y}) = \mathbb{P} \{ i^* = i | \mathbf{A} = \mathbf{Y} \}$ be the probability of i to be the top-quality object, given the answers \mathbf{Y} , for $i \in \mathcal{C}$. We can compute the value of $\pi_i(\mathbf{Y})$ as follows.

$$\begin{aligned} \pi_i(\mathbf{Y}) &= \mathbb{P} \left\{ \bigcap_{j \in \mathcal{C} \setminus i} \{q_j < q_i\} | \mathbf{A} = \mathbf{Y} \right\} \\ &= \int_{\mathbb{R}} \mathbb{P} \left\{ \bigcap_{j \in \mathcal{C} \setminus i} \{q_j < q_i\} | \mathbf{A} = \mathbf{Y}, q_i = x \right\} f_i(x|\mathbf{y}_i) dx \\ &= \int_{\mathbb{R}} f_i(x|\mathbf{y}_i) \prod_{j \in \mathcal{C} \setminus i} F_j(x|\mathbf{y}_j) dx \end{aligned} \quad (3)$$

where F_i is the a-posteriori cdf of q_i given \mathbf{a}_i . It is easy to see that $p_e(\mathbf{Y})$ is minimized when $\hat{i}^* = \arg \max_i \pi_i(\mathbf{Y})$. In the case of Gaussian or unknown priors, (3) holds with

$$F_i(x|\mathbf{y}_i) = \frac{1}{2} \left[1 + \operatorname{erf} \left(\frac{x - \hat{x}_i(\mathbf{y}_i)}{\sqrt{2}\rho_i} \right) \right]$$

Since the evaluation of $\pi_i(\mathbf{Y})$, for $i = 1, \dots, N$, entails a computational complexity growing linearly with N , we propose the following approximation:

$$\begin{aligned} \tilde{\pi}_i(\mathbf{Y}) &= \int_{-\infty}^{\infty} F_{c(i)}(x|\mathbf{y}_{c(i)}) f_i(x|\mathbf{y}_i) dx \\ &= \frac{1}{2} \left(1 + \operatorname{erf} \left(\frac{\hat{x}_i - \hat{x}_{c(i)}}{\sqrt{2(\rho_i^2 + \rho_{c(i)}^2)}} \right) \right) \end{aligned} \quad (4)$$

where $c(i) = \arg \max_{j \neq i} \hat{x}_j$ corresponds to the object with maximum current estimated quality except i . In practice, (4) restricts the comparison to only two objects, the running candidate i and its strongest competitor $c(i)$, and uses the current probability $\tilde{\pi}_i(\mathbf{Y})$ that object i is better than $c(i)$ as an approximation for $\pi_i(\mathbf{Y})$.

In this work, because of complexity considerations, we choose the *error probability* as the performance parameter.

C. Fitness indices

As in the case of performance parameters, different choices for fitness indices are possible.

- **Exact max probability:** With this choice, we identify the fitness index of objects with their estimated probability of being the top-quality object:

$$\phi_i = \pi_i(\mathbf{Y})$$

- **Approximate max probability:** In this case:

$$\phi_i = \tilde{\pi}_i(\mathbf{Y})$$

- **Exact max probability with elimination:** As stated in the previous section, the contestant set, \mathcal{C} , initially set to $\{1, \dots, N\}$, may be shortened along rounds. We have considered a strategy where, at each round, those objects whose π_i is lower than a threshold $\pi_{\text{th,E}}$ are eliminated. For this strategy, the fitness index is given by:

$$\phi_i = \begin{cases} \pi_i(\mathbf{Y}), & \pi_i(\mathbf{Y}) > \pi_{\text{th,E}} \text{ and } i \in \mathcal{C} \\ -\infty, & \pi_i(\mathbf{Y}) \leq \pi_{\text{th,E}} \text{ or } i \notin \mathcal{C} \end{cases} \quad (5)$$

- **Approximate max probability with elimination:** Analogously, we can consider a strategy where objects are eliminated if $\tilde{\pi}_i$ falls below a threshold $\pi_{\text{th,E}}$. The corresponding fitness index is given by (5) where π_i is replaced by $\tilde{\pi}_i$.

All these choices of fitness index have been tested numerically.

D. Allocation function

As stated in previous sections, given the current fitness index, the allocation function $\mathcal{A}(\cdot)$ determines the number

of further evaluations needed by each object in round ℓ . Furthermore, $\mathcal{A}(\cdot)$ is a non-decreasing function of the fitness index (Property 1).

For simplicity, we are particularly interested in the case where $\mathcal{A}(\cdot)$ returns values in $\{0, 1\}^N$, i.e., where the number of workers assigned to every object within a round is either 0 or 1. In such a case, in round ℓ , the $B^{(\ell)} \leq N$ top-quality objects will receive an extra worker, while all other objects will not receive any extra worker.

Two possible choices are considered in this paper, according to whether the total evaluation budget is either fixed or not.

- **Unbounded budget:** If there is no maximum number of requested evaluations, \mathcal{A} only depends on the fitness index, in the following way:

$$m_i^{(\ell)} = \begin{cases} 1, & \phi_i^{(\ell)} > \pi_{\text{th,A}} \\ 0, & \phi_i^{(\ell)} \leq \pi_{\text{th,A}} \end{cases}$$

where $\pi_{\text{th,A}}$ is a suitable accuracy threshold, generally different from $\pi_{\text{th,E}}$ defined in the previous subsection. However, for consistency, we need to have $\pi_{\text{th,E}} \leq \pi_{\text{th,A}}$.

- **Bounded budget:** If at most M_{max} evaluations can be requested in all rounds, then, in round ℓ , $\mathcal{A}(\cdot)$ must take into account also the number of evaluations already requested, given by $M^{(\ell-1)}$. Let again $\pi_{\text{th,A}}$ be the threshold against which the fitness index is compared, like in the unbounded-budget case, and let $B^{(\ell)}$ be the number of objects that currently pass the threshold. If $B^{(\ell)} \leq M_{\text{max}} - M^{(\ell-1)}$, then the allocation of new evaluations is the same as for unbounded budget, otherwise only the $M_{\text{max}} - M^{(\ell-1)}$ objects with the largest fitness index are allocated a further evaluation.

E. Termination rules

Based on the choice of the fitness index and the allocation function \mathcal{A} , the algorithm termination rule may be different.

- **Maximum budget achieved:** When a maximum of M_{max} evaluations is allowed, reaching this maximum budget will cause rounds to stop.
- **Singleton contestant set:** For algorithms that eliminate objects when their fitness is lower than $\pi_{\text{th,E}}$, the natural termination condition is when the contestant set only contains a single object, i.e., $|\mathcal{C}^{(\ell)}| = 1$.
- **Accuracy:** If only a single object passes the accuracy threshold $\pi_{\text{th,A}}$, while all other objects do not, meaning that there is already a strong candidate winner, the algorithm may terminate rounds.

When applicable, the termination rule can be the combination of all three rules above, i.e., the algorithm may terminate whenever one of the three becomes true.

V. SCORING VERSUS DIRECT COMPARISONS

In this section we want to show that, whenever workers are able to provide (noisy) quantitative estimates of the object quality, algorithms exploiting those estimates are in general more effective than algorithms just resorting to direct comparisons among subsets of objects.

We start by considering a toy case in which only two objects are given, with qualities x_1 and $x_2 = x_1 + \Delta$, and we compare two algorithms requiring the same amount of human effort. The first algorithm resorts to outcomes of direct comparisons between the objects performed by the crowd workers, while the second exploits quantitative estimates of the object qualities provided by the same (or other) crowd workers.

Observe that an algorithm exploiting the outcomes of direct comparisons between objects, and employing a fixed budget of W workers for each comparison, necessarily works as follows. Each of the W enrolled workers returns a binary variable, indicating which object she prefers. Once all answers, collectively denoted \mathbf{Z} , are obtained, a majority rule is applied by the algorithm to choose the “best” object. According to our model, each worker prefers object 1, if she estimates that the quality of object 1 exceeds the quality of object 2 and vice versa. Thus, a worker chooses object 1, i.e., returns an incorrect answer, with probability $p_\Delta = \frac{1}{2}\text{erfc}\left(\frac{\Delta}{2\sigma}\right)$, while she chooses object 2, thus returning a correct answer, with probability $1 - p_\Delta$.

Processing the W collected answers (to simplify the analysis, we assume an odd value for W), the algorithm based on comparisons erroneously selects object 1 whenever the number of answers equal to 1 exceeds $W/2$, i.e.:

$$p_e^{\text{comp}} = \mathbb{P}\left(\text{Bin}(W, p_\Delta) > \frac{W}{2}\right)$$

where $\text{Bin}(W, p)$ denotes a binomial distribution of parameters W and p .

Instead, an algorithm that has access to the quantitative quality estimates \mathbf{Y} provided by the W crowd workers, naturally selects the object with the largest estimated quality $\hat{x}(y_i)$. In this case the error probability is given by:

$$p_e^{\text{est}} = \frac{1}{2}\text{erfc}\left(\frac{\sqrt{W}\Delta}{2\sigma}\right).$$

Not surprisingly, $p_e^{\text{est}} < p_e^{\text{comp}}$, as shown in Fig. 1 for $W = 101$. Indeed, from an information-theoretic perspective, the answers \mathbf{Y} provide much more information on the quality of the two objects than the comparisons \mathbf{Z} . This implies that any algorithm resorting to direct comparisons does not fully exploit the information on object quality that crowd workers are able to provide and, as a consequence, turns out to be suboptimal.

We now analytically compare p_e^{comp} and p_e^{est} to better quantify the advantages of the approach exploiting quantitative estimates of object quality. To this end, we approximate the binomial distribution $\text{Bin}(W, p_\Delta)$ with a Gaussian distribution $\mathcal{N}(Wp_\Delta, \sqrt{Wp_\Delta(1-p_\Delta)})$. By the De Moivre-Laplace theorem, such approximation is asymptotically tight for large W . Following this approach, p_e^{comp} can be approximated as:

$$p_e^{\text{comp}} \approx \frac{1}{2}\text{erfc}\left(\frac{\sqrt{W}(1-2p_\Delta)}{2\sqrt{2p_\Delta(1-p_\Delta)}}\right)$$

To further simplify the expression of p_e^{comp} , we consider the

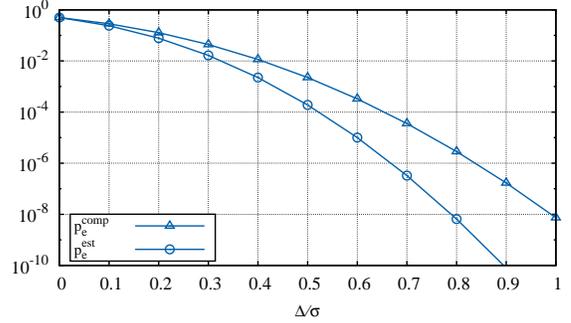


Fig. 1. Comparison between quantitative quality estimations and direct comparisons in terms of error probability in the case of two objects. $W = 101$.

limit for $\frac{\Delta}{\sigma} \rightarrow 0$, in which case $p_\Delta = \frac{1}{2}\left(1 - \frac{2}{\sqrt{\pi}}\frac{\Delta}{2\sigma}\right) + o\left(\frac{\Delta}{\sigma}\right)$, and thus

$$p_e^{\text{comp}} \approx \frac{1}{2}\text{erfc}\left(\sqrt{\frac{2}{\pi}}\frac{\sqrt{W}\Delta}{2\sigma} + o\left(\frac{\Delta}{\sigma}\right)\right)$$

The performance penalty entailed by the approach resorting to direct comparisons of objects is expressed by the factor $\sqrt{2/\pi}$ appearing in the argument of the above erfc function. This factor has a strong impact on the algorithm performance, since, for Δ/σ sufficiently small, as W increases, the ratio $p_e^{\text{est}}/p_e^{\text{comp}}$ tends exponentially fast to zero.

Now, consider a case in which $N > 2$ objects are to be evaluated. For example, let us focus on the case $N = 4$. To declare a winner through direct comparisons of objects, we need to compare at least three pairs of objects. A natural solution is to arrange a *tournament* in which the four objects are first partitioned into two pairs, so that the objects in each pair can be compared in parallel (first round). The two winners of the first round are then compared to identify the globally best object (second round). Observe that the outcomes \mathbf{Z} returned by crowd workers within the first round cannot be exploited in any way at the second round. In other words, at the end of the first round, no useful information is available to rank the two first-round winners. If, instead, the workers return their own quantitative evaluations \mathbf{Y} of the object quality, the evaluations carried out within the first round provide useful information also for the second round.

In light of this discussion, it should not be surprising that the performance gain of algorithms exploiting workers’ quantitative quality evaluations increases as the number of objects increases.

VI. ANSWER QUANTIZATION

Up to now, we have assumed that workers return unquantized (i.e., infinite-precision) noisy evaluations of object qualities. This assumption is unpractical in many scenarios, where instead workers’ evaluations must belong to a finite alphabet, i.e., they are quantized. In this section, we discuss how quantization can be effectively implemented in order to approach the performance of the proposed unquantized algorithm.

From a system point-of-view, the key parameter of a quantizer is the cardinality L of the alphabet on which answers should be encoded, i.e., the number of levels of the quantizer. Given L , a specific quantization rule is characterized by an $(L + 1)$ -dimensional vector of thresholds $\mathcal{Z} = [z_1, \dots, z_l, \dots, z_{L+1}]$ with $z_1 = -\infty < z_2 < z_3 < \dots, z_L < z_{L+1} = +\infty$ and an L -dimensional vector $\mathcal{W} = [w_1, \dots, w_l, \dots, w_L]$ of representative values, satisfying $w_l \in (z_l, z_{l+1})$. If workers' answers are quantized, then the j -th answer to the evaluation of object i can be modelled as

$$a_{ij}^{(q)} = \mathcal{Q}(a_{ij})$$

where $\mathcal{Q}(x) = w_l$ whenever $x \in (z_l, z_{l+1}]$.² Notice that we consider here a *fixed*, non-adaptive quantizer, that is defined once and for all at the beginning, before any evaluation takes place.

In our context, the problem of optimal quantization can be formulated in terms of the minimization of some distortion index between unquantized answers and their quantized version. The mean square error $\mathbb{E}[(a^{(q)} - a)^2]$ represents a natural candidate for such distortion index, also because the seminal work by Lloyd [10] provides an efficient iterative algorithm for the design of a quantizer that minimizes the mean square error. Now, the nontrivial question to be answered is: *which are the answers whose distortion after quantization should be minimized?* We list in the following a few possible answers to such question.

- Since we have N objects whose quality values are i.i.d., each with pdf f_q , we may want to minimize the distortion on the answer relative to the generic object. With such a choice, the distribution with respect to which the mean square error is to be computed is $f_a^{(I)} = f_q * f_n$, where $*$ is the convolution product and $f_n = \mathcal{N}(0, \sigma)$ is the distribution of the workers' evaluation error.
- From a different perspective, since we are searching for the top-quality object, we should minimize the distortion on the answers associated to that object only, i.e., averaging with respect to $f_a^{(II)} = f_{q_{[1]}} * f_n$, $f_{q_{[1]}}$ being the a-priori distribution of the largest quality value.
- Taking an approach which is in between the previous two, and considering that our target is to discriminate the best object from the others, we could aim at minimizing a weighted combination of the distortions relative to the ordered quality values. This goal can be achieved by using as the answer distribution $f_a^{(III)} = \sum_{i=0}^{N-1} \alpha_i f_{q_{[i]}} * f_n$, where $f_{q_{[i]}}$ is the a-priori distribution for the i -th best object, and α_i is its associated weight, satisfying $\alpha_{i+1} \leq \alpha_i$.

We will see in the next section that the impact on algorithm performance of the different possible choices for f_a is pretty significant. Therefore, the quantizer must be carefully

²Observe, however, that workers can be unaware of representative \mathcal{W} ; every worker is just requested to express a satisfaction level in $\{1, \dots, L\}$, which is obtained by comparing her own unquantized evaluation with thresholds \mathcal{Z} . Therefore, our model perfectly matches the assumptions of [8].

designed. At last, observe that $f_a^{(I)}$ and $f_a^{(II)}$ can be obtained as particular cases of $f_a^{(III)}$ by setting $\alpha_i = 1/N$ for all i in the first case, and $\alpha_1 = 1, \alpha_i = 0$ for $i > 1$ in the second case.

VII. RESULTS

In this section, we compare the performance of several algorithms that are obtained by making different choices for the *fitness index*, the *allocation function*, the *termination rule*, and the *quantizer*.

A. Unquantized answers, unbounded budget

In particular, we first focus on algorithms with unquantized answers and unbounded budget, and define:

- the 'Greedy-Keep-Exact' (GKE) algorithm, employing the *exact max probability* as fitness index, *unbounded budget* as allocation function, and the *accuracy termination rule*;
- the 'Greedy-Keep-Approximate' (GKA) algorithm, employing the *approximate max probability* as fitness index, the *unbounded budget* as allocation function, and the *accuracy termination rule*;
- the 'Greedy-Remove-Approximate' (GRA) algorithm, employing the *approximate max probability with eliminations* as fitness index, the *unbounded budget* as allocation function, and the *singleton contestant set* as termination rule.

To reduce the space of parameters, we have always fixed $\pi_{\text{th},A} = \pi_{\text{th},E} \triangleq \pi_{\text{th}}$. For the sake of comparison, the following algorithms have been also considered.

- We have superimposed a classical tournament scheme to our previously described algorithms, obtaining a family of *Tournament- N_b* (T- N_b) algorithms. Specifically, in T- N_b algorithms, the N objects to be evaluated are first randomly partitioned into subgroups of size N_b (for simplicity, we neglect rounding problems). The GKA algorithm is then run to elect a winner for each of the object groups. Only winners have access to the second stage, in which again objects are partitioned into subgroups of size N_b and winners are elected for each subgroup. The process is iterated until only one winner is left. We remark that our tournament schemes effectively exploit, at every stage, full information about the evaluations of competing objects collected at earlier stages.
- A non-adaptive algorithm, which assigns to every object a fixed number of workers, referred to as *Uniform algorithm* (UA).
- As a reference, we have also considered an unfeasible *Genie-Aided* (GA) algorithm, which has access to the identity of the two best competing objects, after a first initial round of evaluations, where every object receives one score. Therefore, in the following rounds, the GA algorithm equally distributes workers only to the top two objects until the *accuracy termination rule* is met. Observe that, by construction, the performance of the GA algorithm constitutes an upper bound for every feasible

algorithm, since, as discussed in Section V, it implements the optimal policy to find the best between two objects.

We start considering a simple scenario with $N = 16$ objects whose qualities x_i are equally spaced in the interval $[-1, 1]$, so that the smallest difference between quality values is $\Delta = \frac{2}{N-1} = \frac{2}{15}$. The standard deviation of the worker evaluation error has been set to $\sigma = \Delta/2$.

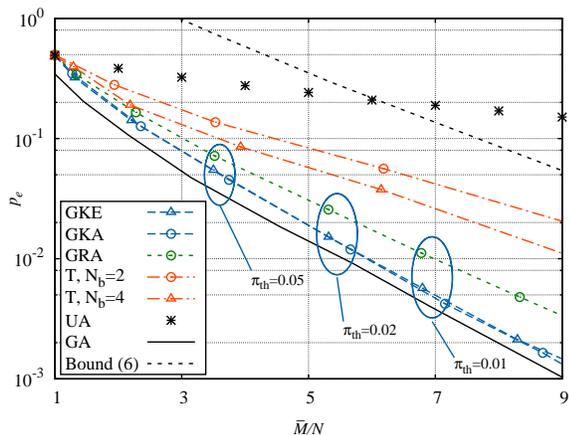


Fig. 2. Performance (p_e vs \overline{M}/N) of different algorithms with unquantized answers and unbounded budget for $N = 16$ equally spaced objects.

Fig. 2 shows the results obtained with the different proposed algorithms, plotted in terms of error probability p_e versus the average number of performed evaluations per object \overline{M}/N . We highlight that the different trade-offs between p_e and \overline{M}/N correspond to different values of the threshold π_{th} , as explicitly shown in Fig. 2. Observe that the choice of π_{th} has a direct impact on the expected error probability of the algorithm. In particular, for the GKE algorithm a simple analysis yields the following relationship between π_{th} and p_e :

$$p_e = \sum_{i \neq i^*} \pi_i(\mathbf{y}) \leq (N-1)\pi_{th} \quad (6)$$

More in general, for every algorithm, by decreasing π_{th} we achieve a larger accuracy at the cost of employing more resources.

From the results, the following observations can be made.

- i) Every adaptive algorithm performs much better than the uniform algorithm, employing on average the same amount of resources.
- ii) Greedy algorithms without elimination perform better than greedy algorithms with elimination. Observe, however that the latter are preferable in terms of computational complexity, since in such schemes, at round ℓ , the fitness index has to be computed only for objects in the contestant set $\mathcal{C}^{(\ell)}$, while in schemes without elimination it has to be computed for all objects.
- iii) The selection of the approximate max probability as fitness index, in place of the exact max probability, does not lead to any appreciable performance degradation, while having a significant beneficial impact on the computational complexity of the algorithm (this has been

checked also for algorithms with elimination).

- iv) Tournament algorithms perform worse than our adaptive algorithms; furthermore, their performance tends to worsen as N_b is reduced.
- v) The performance of greedy algorithms without elimination is only marginally worse than that of the GA algorithm.

To gather more insight on the algorithm behavior, a further performance comparison for the different schemes is reported in Fig. 3 for the case in which the number of objects is increased to $N = 256$ (object qualities x_i are still equally spaced in the interval $[-1, 1]$, with $\sigma = \Delta/2$). Observe that the relative ranking among the algorithms does not change, but the performance gap between algorithms tends to become more significant. In particular, tournament algorithms perform much worse than GKA and GRA. We remark that our results seem somehow in contrast with findings in [8], where it has been shown that tournament algorithms provide the best performance, for cases in which users are only able to compare objects pairs. To intuitively grasp why they become inefficient in our context as N increases, notice that tournament algorithms waste a significant amount of resources to discriminate among objects with similar quality (accidentally placed in the same group), even when the quality of such objects is much worse than top-quality values. Observe that, also in this case, GKA (whose performance is again practically indistinguishable from GKE, not reported in Fig. 3 for the sake of readability) performs similarly to the GA algorithm. This proves the effectiveness of GKA in the considered scenarios.

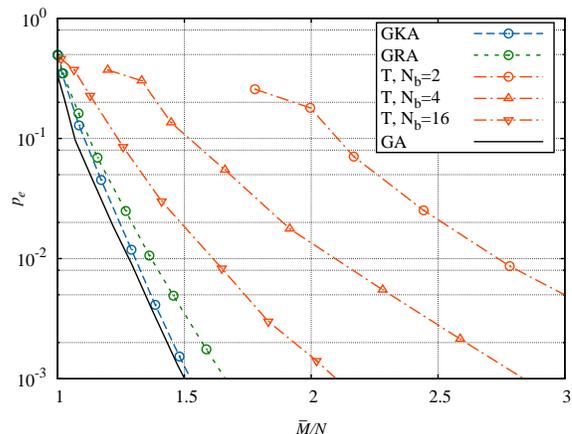


Fig. 3. Performance (p_e vs \overline{M}/N) of different algorithms with unquantized answers and unbounded budget for $N = 256$ equally spaced objects.

To evaluate the impact of human evaluation errors on the overall performance of algorithms, Fig. 4 reports a performance comparison between the GKA and GA algorithms for different values of the parameter Δ/σ . Observe that Δ/σ plays an important role: as the ratio Δ/σ decreases, more and more resources are needed to meet the same error probability. The performance gap between the two algorithms is rather limited in all cases. In particular, uniformly over all cases, the penalty cost in terms of evaluations required to obtain the same error

probability does not exceed 10%. Once again, this confirms the effectiveness of our approach for a broad range of scenarios where evaluation errors have different impact.

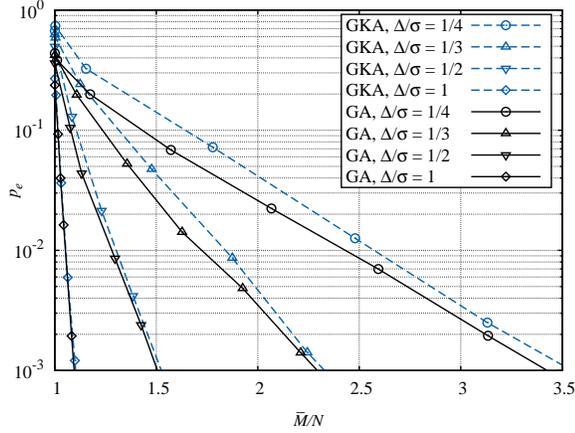


Fig. 4. Performance of GKA vs GA under different values of Δ/σ , $N = 256$ equally spaced objects.

B. Unquantized answers, bounded budget

Now, we move to scenarios in which the budget of allocations is bounded, and we restrict our analysis to:

- the 'bounded-Greedy-Keep-Approximate' (bGKA), employing the *approximate max probability* as fitness index, the *bounded budget* as allocation function, the *maximum budget achieved or accuracy termination rule*;
- the 'bounded-Greedy-Remove-Approximate', (bGRA) employing the *approximate max probability with eliminations* as fitness index, the *bounded budget* as allocation function, the *maximum budget achieved or singleton contestant list* as termination rule.

As a reference, we report also the performance of the bounded version of the GA algorithm (bGA), which again provides an obvious upper bound to performance. Also in this case, to reduce the space of parameters, we fix $\pi_{th} = \pi_{th,A} = \pi_{th,E}$.

Fig. 5 compares the performance of different algorithms for different values of the normalized budget $K = M_{max}/N$. In the same figure, we also report the performance of the unbounded versions of the algorithms. We observe that:

- the error probability for bGKA and bGRA now is not monotonic with respect to \bar{M}/N . In particular, if the average number of evaluations \bar{M} is sufficiently smaller than M_{max} (i.e., for sufficiently large values of π_{th}), the performance of the bounded algorithms does not significantly differ from the respective unbounded version: this happens because the probability that the algorithm terminates for achieving maximum budget is negligible. As we further reduce π_{th} , increasing the required accuracy, the probability that the algorithm terminates because the maximum budget is achieved quickly increases, and the overall performance of the bounded algorithms degrades. These effects can be better understood from Fig 6, which reports the average number of evaluations per

object, \bar{M}/N , and the error probability, as a function of the threshold π_{th} , for the bGKA algorithm (similar considerations hold for bGRA). Now, observe that \bar{M}/N monotonically increases as π_{th} decreases, since, by decreasing π_{th} , the algorithm tends to be more conservative in excluding objects from receiving further evaluations. As a result, we distribute a larger number of allocations to objects with a quality quite distant from the maximum. While the error probability behaves monotonically with respect to π_{th} (decreasing as π_{th} is decreased) in the case of an unbounded budget, in the case of bounded budget, choosing a value of π_{th} too small will lead to an inefficient distribution of the limited resources.

- As a consequence of i), only a limited range of p_e values can be achieved in the bounded budget case.
- Also in this case, bGKA outperforms bGRA.

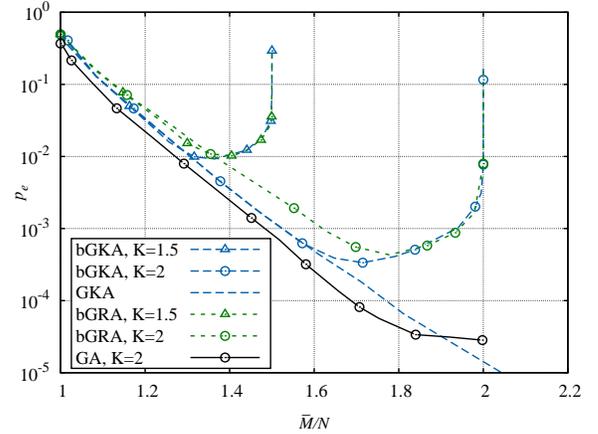


Fig. 5. p_e vs \bar{M}/N for bGKA and bGRA, $N = 256$ equally spaced objects.

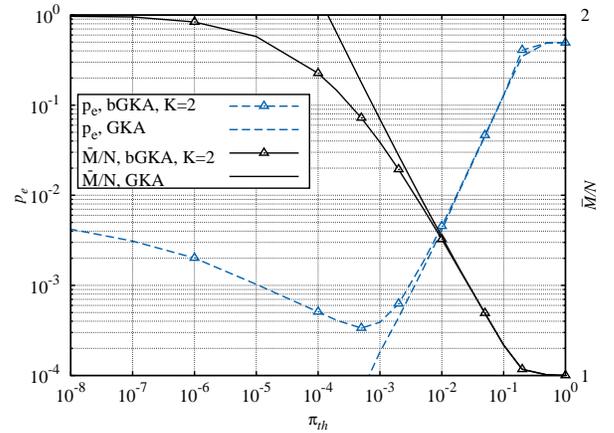


Fig. 6. p_e and \bar{M}/N vs π_{th} for bGKA, $N = 256$ equally spaced objects.

Now, we consider a scenario in which object qualities x_i are randomly drawn from a Gaussian distribution with zero mean and standard deviation σ_a .

Fig. 7 presents the performance of bGKA for the cases in which $\sigma_a/\sigma = 3$ and $\sigma_a/\sigma = 10$. For the sake of comparison, the figure also reports the performance of the uniform algorithm.

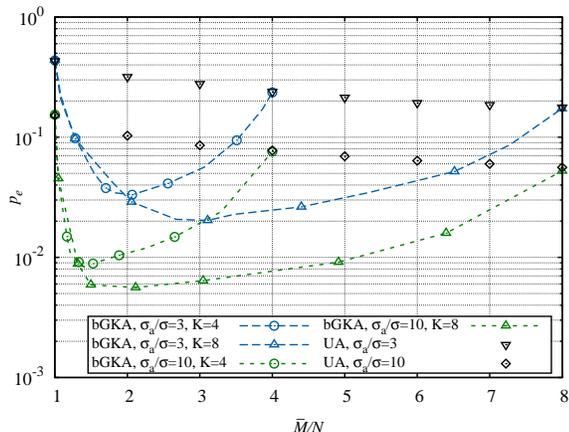


Fig. 7. p_e vs \bar{M}/N for bGKA and Uniform, $N = 256$, object qualities are Gaussian.

We observe that:

- i) also when qualities are drawn from a Gaussian distribution, bGKA improves performance with respect to the uniform allocation algorithm that employs the same average budget of resources;
- ii) from a qualitative point of view, bGKA exhibits a behavior which is pretty similar to the case where objects are equally spaced. By decreasing π_{th} we initially increase accuracy at the cost of increasing also the amount of employed resources. However, beyond a given point, further decrements of π_{th} cause a loss of efficiency, worsening the overall performance of the algorithm;
- iii) the performance of the algorithms heavily depends on the parameter σ_a/σ , which can be regarded as a difficulty index for the problem.

To complement the previous results for the bGKA algorithm, Fig. 8 reports the first-order conditional distortion, i.e., $\mathbb{E}[q_{i^*} - q_{\hat{i}^*} | i^* \neq \hat{i}^*(\mathbf{y})]$, which may be interpreted as the average quality gap between the best object and the object selected by the bGKA algorithm, conditional over the fact that the bGKA algorithm is not choosing the best object. Observe that the values of the distortion are extremely small, proving that, whenever the bGKA algorithm makes an error, it selects an object with a quality very close to the best. In other words, errors occur (almost) only in very difficult problem instances, where a cluster of two or more objects with quality very close to the top value exists. For completeness, Fig. 8 reports also the conditional distortion for the uniform allocation algorithm. Observe that the distortion experienced with the uniform allocation algorithms is more than one order of magnitude larger than under bGKA. This implies that the algorithms we propose in this paper not only offer the benefit of a drastic reduction in error probability, but they also bring a second crucial improvement: a drop in the gap between the quality of the selected object and the actual maximum quality, in the case of errors.

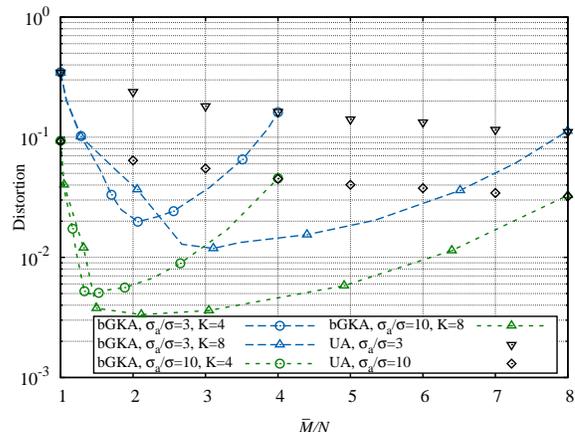


Fig. 8. Distortion vs \bar{M}/N for bGKA, $N = 256$, object qualities are Gaussian.

C. Quantization effects

We finally consider the effect of quantization on the system performance.

In Fig. 9, we test the impact of different quantizers, in terms of p_e versus \bar{M}/N , in the case where $N = 256$ objects are equally spaced in the interval $[-1, 1]$. In the figure, the GKA algorithm with unbounded budget is employed for all curves. Moreover, the standard deviation of the worker evaluation error has been set to $\sigma = \Delta/2$. We recall that, for equally spaced quality values, $\Delta = 2/(N-1)$ is the smallest distance between quality values. The solid line without markers represents the performance of the GKA algorithm without quantization and is used as a benchmark. The line with triangle markers refers to the performance obtained employing a quantizer with $L = 32$ representative values uniformly distributed in $[-1-2\sigma, 1+2\sigma]$. We observe that, despite of the high number of levels, uniform quantization significantly worsen the error probability. Instead, much better performance is achievable when the quantizer design is more accurate. As an example, the solid line with filled square markers refers to the case when $L = 32$, and the quantizer is designed according to the criteria in [10], over the answer distribution $f_a^{(III)} = \sum_{i=0}^{N-1} \alpha_i f_{q_{[i]}} * f_n$, where $\alpha_i = \gamma^i$, and $\gamma = 1/2$. This quantizer, labeled ‘‘Lloyd’’ in the legend, provides performance close to the unquantized case. In general, we observe that the performance is quite insensitive to the design parameter γ except if its value is close to the extreme point $\gamma = 1$. Fig. 9 also shows the performance of Lloyd’s quantizers with $\gamma = 1/2$ and number of levels $L = 4, 8$. We observe that an accurately designed quantizer with only $L = 8$ levels is enough to provide performance close to the unquantized case.

Fig. 10 refers to the case of $N = 256$ objects with Gaussian-distributed qualities, the budget is bounded to $K = 3$, and workers’ answers are quantized. The quantizer is designed according to the Lloyd’s algorithm over the weighted distribution $f_a^{(III)} = \sum_{i=0}^{N-1} \alpha_i f_{q_{[i]}} * f_n$, with $\alpha_i = \gamma^i$, and $\gamma = 1/2$. In the figure, the solid line refers to the unquantized case, while the lines with markers refer to the case where quantization is

employed with $L = 4, 8, 16, 32$ levels. Also in this scenario, we observe that $L = 8$ quantization levels are enough to provide performance close to the unquantized case.

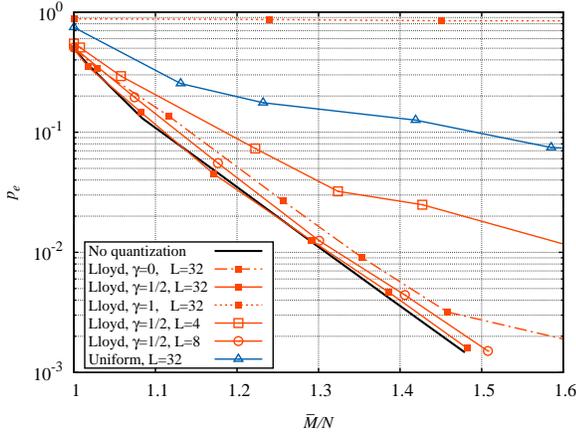


Fig. 9. Performance (p_e vs \bar{M}/N) of the GKA algorithm with unbounded budget and quantized workers' answers, for $N = 256$ objects with equally spaced quality values.

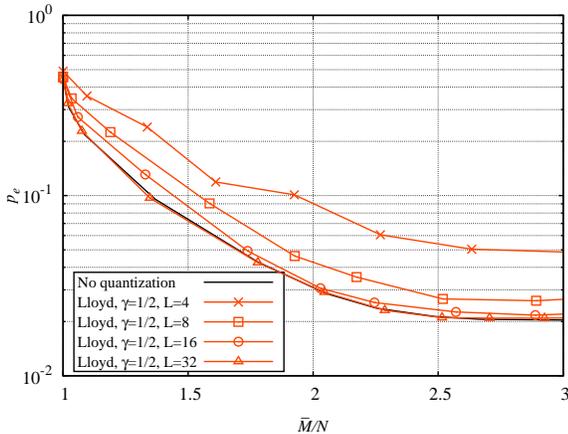


Fig. 10. Performance of the GKA algorithm with bounded budget, quantized workers' answers, for $N = 256$ objects with Gaussian-distributed qualities.

VIII. DESIGN CONSIDERATIONS

In this section we address two important issues: we evaluate the computational complexity of our algorithms, and we discuss how to set algorithm parameters (and in particular π_{th}). For the sake of brevity, we restrict our investigation to bGKA, which turns out to be the best-performing algorithm.

For what concerns computational complexity, at round ℓ , for every object i in the contestant set, bGKA: i) must update $\hat{x}_i^{(\ell)}$ (this requires $O(1)$ operations per object); ii) must compute a fitness index $\phi_i^{(\ell)}$ (this again requires $O(1)$ operations per objects, since the *approximate max probability* can be computed exploiting (4)); iii) must compare $\phi_i^{(\ell)}$ with π_{th} , in order to decide whether to allocate an extra worker to i (again $O(1)$ operations are required). In the final round, if the number of objects that pass the threshold exceeds the residual budget, the execution of an extra task is necessary: objects must be sorted

in order of their fitness index (the cost of sorting is notoriously $O(N \log N)$). Observing that M_{max} is an obvious upper bound to the number of rounds, it turns out that the overall complexity of bGKA is $O(M_{max}N + N \log N) = O(M_{max}N)$ in consideration of the fact that by construction $M_{max} \geq N$.

As regards the setting of the algorithm parameters, which in the case of bGKA are π_{th} and M_{max} , we observe that the choice of the value for M_{max} normally depends on economical or application-oriented considerations, whose discussion is beyond the scope of this paper. Given the value of M_{max} , it is possible to tune the algorithm by selecting the value for π_{th} , which should be set in order to minimize the error probability. For a careful setting of π_{th} , a preliminary parametric analysis of the algorithm performance is necessary to estimate the key system parameters (such as the pdf f_q of object qualities, and the variance σ of the workers' quality estimation errors).

IX. CONCLUDING REMARKS

In this paper, we have studied the problem of finding the top-quality element within a large collection of objects, resorting to human evaluations affected by noise. Differently from previous works, our study started assuming that unquantized scores are returned by the evaluators, and highlights the potential advantages of such approach. Then we have shown how to properly design quantized schemes whose performance is very close to their ideal unquantized counterparts, provided that a reasonable number of quantization levels is assigned to workers' answers. We plan to generalize the approach proposed in this paper to the case of workers with different skills, and to the problem of finding the k top-quality elements within a large collection of objects through crowdsourcing algorithms.

REFERENCES

- [1] M.-C. Yuen, I. King and K.-S. Leung, "A Survey of Crowdsourcing Systems," *IEEE PASSAT-SOCIALCOM*, Boston (USA), Oct. 2011.
- [2] P. Venetis, H. Garcia-Molina, K. Huang and N. Polyzotis, "Max algorithms in crowdsourcing environments," *Intern. Conf. on World Wide Web (WWW '12)*, New York (USA), 989–998, 2012.
- [3] P. Venetis and H. Garcia-Molina, "Dynamic Max Algorithms in Crowdsourcing Environments," *Tech. Rep.*, Stanford InfoLab.
- [4] S. Guo, A. Parameswaran and H. Garcia-Molina, "So who won?: dynamic max discovery with the crowd," *2012 ACM Intern. Conf. on Management of Data*, New York (USA), 385–396, 2012.
- [5] V. Verroios, P. Lofgren and H. Garcia-Molina, "tDIP: An Optimal-Latency Budget Allocation Strategy for Crowdsourced MAXIMUM Operations," *2015 ACM Intern. Conf. on Management of Data*, New York (USA), 1047–1062, 2015.
- [6] S. B. Davidson, S. Khanna, T. Milo and S. Roy, "Using the crowd for top- k and group-by queries," *ICDT '13*, New York (USA), 225–236, 2013.
- [7] U. Feige, P. Raghavan, D. Peleg, and E. Upfal, "Computing with noisy information," *SIAM J. Comput.*, 23(5): 1001–1018, 1994.
- [8] A. R. Khan and H. Garcia-Molina, "Hybrid Strategies for Finding the Max with the Crowd," *Tech. Rep.*, Stanford InfoLab.
- [9] L. L. Thurstone, "A law of comparative judgment," *Psychological Review*, vol. 34, pp. 273–286, 1927.
- [10] S. P. Lloyd, "Least Squares Quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, pp. 129–137, No. 2, March 1982.