The Importance of Being Earnest in Crowdsourcing Systems

Alberto Tarable, † Alessandro Nordio, † Emilio Leonardi, * † Marco Ajmone Marsan * ‡ †
† CNR-IEIIT, Torino, Italy, * Politecnico di Torino, Italy, ‡ IMDEA Networks Institute, Madrid, Spain

***Abstract*— This paper presents the first systematic investigation of the potential performance gains for crowdsourcing systems, deriving from available information at the requester about individual worker earnestness (reputation). In particular, we first formalize the optimal task assignment problem when workers' reputation estimates are available, as the maximization of a monotone (submodular) function subject to Matroid constraints. Then, being the optimal problem NP-hard, we propose a simple but efficient greedy heuristic task allocation algorithm. We also propose a simple "maximum a-posteriori" decision rule. Finally, we test and compare different solutions, showing that system performance can greatly benefit from information about workers' reputation. Our main findings are that: i) even largely inaccurate estimates of workers' reputation can be effectively exploited in the task assignment to greatly improve system performance; ii) the performance of the maximum a-posteriori decision rule quickly degrades as worker reputation estimates become inaccurate; iii) when workers' reputation estimates are significantly inaccurate, the best performance can be obtained by combining our proposed task assignment algorithm with the LRA decision rule introduced in the literature.**

I. Introduction

Crowdsourcing is a term often adopted to identify networked 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]. Alternative terms, each one carrying its own specific nuance, to identify similar types of systems are: collective intelligence, human computation, master-worker computing, volunteer computing, serious games, voting problems, peer production, citizen science (and others). The key characteristic of these systems is that a *requester* structures his problem in a set of *tasks*, and then assigns tasks to *workers* that provide *answers*, which are then used to determine the correct task *solution* through a *decision* rule. Well-known examples of such systems are SETI@home, which exploits unused computer resources to search for extra-terrestrial intelligence, and the Amazon Mechanical Turk, which allows the employment of large numbers of micro-paid workers for tasks requiring human intelligence (HIT – Human Intelligence Tasks). Examples of HIT are image classification, annotation, rating and recommendation, speech labeling, proofreading, etc. In the Amazon Mechanical Turk, the workload submitted by the requester is partitioned into several small atomic tasks, with a simple and strictly specified structure. Tasks, which require small amount of work, are then assigned to (human) workers. 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, workers are not 100 % reliable (earnest), in the sense that they may provide incorrect answers. Hence, the same task is normally assigned in parallel (replicated) to several workers, and then a majority decision rule is applied to their answers. A natural trade-off between the reliability of the decision and cost arises; indeed, increasing the replication factor of every task, we can increase the reliability degree 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). Although the pool of workers in crowdsourcing systems is normally large, it can be abstracted as a finite set of shared resources, so that the allocation of tasks to workers (or, equivalently, of workers to tasks) is of key relevance to the system performance.

Some believe that crowdsourcing systems will provide a significant new type of work organization paradigm, and will employ large numbers of workers in the future, provided that the main challenges in this new type of organizations are correctly solved. In [2] the authors identify a dozen such challenges, including i) workflow definition and hierarchy, ii) task assignment, iii) real-time response, iv) quality control and reputation. Task assignment and reputation are central to this paper, where we discuss optimal task assignment with approximate information about the quality of answers generated by workers (with the term worker reputation we generally mean the worker earnestness, i.e., the credibility of a worker's answer for a given task, which we will quantify with an error probability). Our optimization aims at minimizing the probability of an incorrect task solution for a maximum number of tasks assigned to workers, thus providing an upper bound to delay and a lower bound on throughput. A dual version of our optimization is possible, by maximizing throughput (or minimizing delay) under an error probability constraint. Like in most analyses of crowdsourcing systems, we assume no interdependence among tasks, but the definition of workflows and hierarchies is an obvious next step. Both these issues (the dual problem and the interdependence among tasks) are left for further work.

The performance of crowdsourcing systems is not yet explored in detail, and the only cases which have been extensively studied in the literature assume that the quality of the answers provided by each worker (the worker reputation) are not known at the time of task assignment. This assumption is motivated by the fact that the implementation of reputation-tracing mechanisms for workers is challenging, because the workers' pool is typically large and highly dynamical. Furthermore, in some cases the anonymity of workers must be preserved. Nevertheless, we believe that a clear understanding of the potential impact on the system performance of even

[0]This article has been partially supported by the Madrid Regional Government through the TIGRE5-CM program (S2013/ICE-2919)

approximate information about the workers' reputation in the task assignment phase is extremely important, and can properly assess the relevance of algorithms that trace the reputation of workers. Examples of algorithms that incorporate auditing processes in a sequence of task assignments for the worker reputation assessment can be found in [3]–[9].

Several algorithms were recently proposed in the technical literature to improve the performance of crowdsourcing systems without a-priori information about worker reputation [10]–[14]. In particular, [13] proposed an adaptive simple online algorithm to assign an appropriate number of workers to every task, so as to meet a prefixed constraint on problem solution reliability. In [10]–[12], [14], instead, it was shown that the reliability degree of the final problem solution can be significantly improved by replacing the simple majority decision rule with smarter decision rules that differently weigh answers provided by different workers. Essentially the same decision strategy was independently proposed in [10], [11] and [14] for the case in which every task admits a binary answer, and then recently extended in [12] to the more general case. The proposed approach exploits existing redundancy and correlation in the pattern of answers returned from workers to infer an a-posteriori reliability estimate for every worker. The derived estimates are then used to properly weigh workers' answers.

The goal of this paper is to provide the first systematic analysis of the potential benefits deriving from some form of a-priori knowledge about the reputation of workers. With this goal in mind, first we define and analyze the task assignment problem when workers' reputation estimates are available. We show that in some cases, the task assignment problem can be formalized as the maximization of a monotone submodular function subject to Matroid constraints. A greedy algorithm with performance guarantees is then devised. In addition, we propose a simple "maximum a-posteriori" (MAP) decision rule, which is well known to be optimal when perfect estimates of workers' reputation are available. Finally, our proposed approach is tested in several scenarios, and compared to previous proposals.

Our main findings are:

- even largely inaccurate estimates of workers' reputation can be effectively exploited in the task assignment to greatly improve system performance;
- the performance of the maximum a-posteriori decision rule quickly degrades as worker reputation estimates become inaccurate;
- when workers' reputation estimates are significantly inaccurate, the best performance can be obtained by combining our proposed task assignment algorithm with the decision rule introduced in [10], [14].

The rest of this paper is organized as follows. Section II presents and formalizes the system assumptions used in this paper. Section III contains the formulation of the problem of the optimal allocation of tasks to workers, with different possible performance objectives. Section IV proposes a greedy allocation algorithm, to be coupled with the MAP decision rule described in Section V. Section VI presents and discusses the performance of our proposed approach in several scenarios, and compares it to those of previous proposals. Finally, Section VII concludes the paper and discusses possible extensions.

II. System Assumptions

We consider T binary tasks $\theta_1, \theta_2, \ldots, \theta_T$, whose outcomes can be represented by i.i.d. uniform random variables (RV's) $\tau_1, \tau_2, \ldots, \tau_T$ over $\{\pm 1\}$, i.e., $\mathbb{P}\{\tau_t = \pm 1\} = \frac{1}{2}$, $t = 1, \ldots, T$. In order to obtain a reliable estimate of task outcomes, a requester assigns tasks to workers selected from a given population of size W, by querying each worker ω_w, $w = 1, \ldots, W$ a subset of tasks.

Each worker is modeled as a binary symmetric channel (BSC) [15, p. 8]. This means that worker ω_w, if queried about task θ_t, provides a wrong answer with probability p_{tw} and a correct answer with probability $1 - p_{tw}$. Note that we assume that the error probabilities p_{tw} depend on both the worker and the task, but they are taken to be time-invariant, and generally unknown to the requester. The fact that the error probability may depend, in general, both on the worker and the task reflects the realistic consideration that tasks may have different levels of difficulty, that workers may have different levels of accuracy, and may be more skilled in some tasks than in others.

Unlike the model in [10], [11], we assume in this paper that, thanks to a-priori information, the requester can group workers into classes, each one composed of workers with similar accuracy and skills. In practical crowdsourcing systems, where workers are identified through authentication, such a-priori information can be obtained by observing the results of previous task assignments. More precisely, we suppose that each worker belongs to one of K classes, $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_K$, and that each class is characterized, for each task, by a different *average* error probability, known to the requester. Let π_{tk} be the average error probability for class $\mathcal{C}_k$ and task θ_t, $k = 1, \ldots, K$, $t = 1, \ldots, T$. We emphasize that π_{tk} does not necessarily precisely characterize the reliability degree of individual workers within class k while accomplishing task θ_t; this for the effect of possible errors/inaccuracies in the reconstruction of user profiles. Workers with significantly different degree of reliability can, indeed, coexist within class k. In particular our class characterization encompasses two extreme scenarios:

- full knowledge about the reliability of workers, i.e., each worker belonging to class $\mathcal{C}_k$ has error probability for task θ_t deterministically equal to π_{tk}, and
- a hammer-spammer (HS) model [10], in which perfectly reliable and completely unreliable users coexists within the same class. A fraction $2\pi_{tk}$ of workers in class $\mathcal{C}_k$, when queried about task θ_t, has error probability equal to $\frac{1}{2}$ (the spammers), while the remaining workers have error probability equal to zero (the hammers).

Observe that the above two scenarios represent two extremal cases in which the error probability associated to users within a class is either deterministic (former case), or a random variable with given average and maximum possible variance (latter case).

Suppose that class $\mathcal{C}_k$ contains a total of W_k workers, with $W = \sum_{k=1}^{K} W_k$. The first duty the requester has to carry out is the assignment of tasks to workers. We impose the following two constraints on possible assignments:

- a given task θ_t can be assigned at most once to a given worker ω_w, and
- no more than r_w tasks can be assigned to worker ω_w.

Notice that the second constraint arises from practical considerations on the amount of load a single worker can tolerate. We also suppose that each single assignment of a task to a worker has a *cost*, which is independent of the worker's class. In practical systems, such cost represents the (small) wages per task the requester pays the worker, in order to obtain answers to his queries. Alternatively, in voluntary computing systems, the cost can describe the time necessary to perform the computation. The reader may be surprised by the fact that we assume worker cost to be independent from the worker class, while it would appear more natural to differentiate wages among workers, favoring the most reliable, so as to incentivize workers to properly behave [6], [7]. Our choice, however, is mainly driven by the following two considerations: i) while it would be natural to differentiate wages according to the individual reputation of workers, when the latter information is sufficiently accurate, it is much more questionable to differentiate them according to only an average collective reputation index, such as π_{tk}, especially when workers with significantly different reputation coexist within the same class; ii) since in this paper our main goal is to analyze the impact on system performance of a-priori available information about the reputation of workers, we need to compare the performance of such systems against those of systems where the requester is completely unaware of the worker reputation, under the same cost model. Finally, we wish to remark that both our problem formulation and proposed algorithms naturally extend to the case in which costs are class-dependent.

Let an *allocation* be a set of assignments of tasks to workers. More formally, we can represents a generic allocation with a set $\mathcal{G}$ of pairs (t, w) with $t \in \{1, \cdots, T\}$ and $w \in \{1, \cdots, W\}$, where every element $(t, w) \in \mathcal{G}$ corresponds to an individual task-worker assignment. Let $\mathcal{O}$ be the complete allocation set, comprising every possible individual task-worker assignment (in other words $\mathcal{O}$ is the set composed of all the possible $T \cdot W$ pairs (t, w)). Of course, by construction, for any possible allocation $\mathcal{G}$, we have that $\mathcal{G} \subseteq \mathcal{O}$. Hence, the set of all possible allocations corresponds to the power set of $\mathcal{O}$, denoted as $2^{\mathcal{O}}$.

The set $\mathcal{G}$ can also be seen as the edge set of a bipartite graph where the two node subsets represent tasks and workers, and there is an edge connecting task node t and worker node w if and only if $(t, w) \in \mathcal{G}$. It will be sometimes useful in the following to identify the allocation with the biadjacency matrix of such graph. Such binary matrix of size $T \times W$ will be denoted $\mathbf{G} = \{g_{tw}\}$ and referred to as the *allocation matrix*. In the following we will interchangeably use the different representations, according to convenience. This because for the analysis of some of the properties of the scheduling problem, such as sub-modularity of the objective function, will be natural to adopt the set notation, while for others will be much more convenient to adopt the the matrix notation.

In this work, we suppose that the allocation is non-adaptive, in the sense that all assignments are made before any decision is attempted. With this hypothesis, the requester must decide the allocation only on the basis of the a-priori knowledge on worker classes. Adaptive allocation strategies can be devised as well, in which, after a partial allocation, a decision stage is performed, and gives, as a subproduct, refined a-posteriori information both on tasks and on workers' accuracy. This information can then be used to optimize further assignments. However, in [11] it was shown that non-adaptive allocations are order optimal in a single-class scenario.

When all the workers' answers are collected, the requester starts deciding, using the received information. Let $\mathbf{A} = \{a_{tw}\}$ be a $T \times W$ random matrix containing the workers' answers and having the same sparsity pattern as $\mathbf{G}$. Precisely, a_{tw} is nonzero if and only if g_{tw} is nonzero, in which case $a_{tw} = \tau_t$ with probability $1 - p_{tw}$ and $a_{tw} = -\tau_t$ with probability p_{tw}. For every instance of the matrix $\mathbf{A}$ the output of the decision phase is an estimate $\hat{\tau}_1, \hat{\tau}_2, \ldots, \hat{\tau}_T$ for task values.

III. Problem Formulation

In this section, we formulate the problem of the optimal allocation of tasks to workers, with different possible performance objectives. We formalize such problem under the assumption that each worker in class $\mathcal{C}_k$ has error probability for task θ_t deterministically equal to π_{tk}.

If the individual error probability of the workers within one class is not known to the scheduler, it becomes irrelevant which worker in a given class is assigned the task. What only matters is actually how many workers of each class is assigned each task. By sorting the columns (workers) of the allocation matrix $\mathbf{G}$, we can partition it as $\mathbf{G} = [\mathbf{G}_1, \mathbf{G}_2, \ldots, \mathbf{G}_K]$ where $\mathbf{G}_k$ is a binary matrix of size $T \times W_k$ representing the allocation of tasks to class-k workers. Define $W^{(k)} = \sum_{i=1}^{k} W_i$ and $W^{(0)} = 0$. Define also d_{tk} as the weight (number of ones) in the t-th row of matrix $\mathbf{G}_k$, which also represents the degree of the t-th task node in the subgraph containing only worker nodes from the k-th class.

A. Optimal allocation

We formulate the problem of optimal allocation of tasks to workers as a combinatorial optimization problem for a maximum overall cost. Namely, we fix the maximum number of assignments (or, equivalently, the maximum number of ones in matrix $\mathbf{G}$) to a value C, and we seek the best allocation in terms of degree set $\mathcal{D} = \{d_{tk}, t = 1, 2, \ldots, T, k = 1, 2, \ldots, K\}$. Let $P(\mathcal{D})$ be a given performance parameter to be maximized. Then, the problem can be formalized as follows.

$$
\begin{aligned}
\mathcal{D}^{\text{opt}} \quad = \quad & \arg\max_{\mathcal{D}} \; P(\mathcal{D}) \\
\text{s.t.} \quad & d_{tk} \text{ integer}, \;\; 0 \le d_{tk} \le W_k, \;\; t = 1, 2, \ldots, T, \\
& \qquad\qquad\qquad\qquad\qquad\qquad k = 1, 2, \ldots, K \\
& \sum_{t=1}^{T} d_{tk} \le \sum_{w=W^{(k-1)}+1}^{W^{(k)}} r_w, \;\; k = 1, 2, \ldots, K, \\
& \sum_{t=1}^{T} \sum_{k=1}^{K} d_{tk} \le C
\end{aligned}
\tag{1}
$$

where the first constraint expresses the fact that d_{tk} is the number of ones in the t-th row of $\mathbf{G}_k$, the second constraint derives from the maximum number of tasks a given worker can be assigned, and the third constraint fixes the maximum overall cost.

By adopting the set notation for allocations, we can denote with $\boldsymbol{\mathcal{F}}$ the family of all feasible allocations (i.e. the collection of all the allocations respecting the constraints on the total cost and the worker loads). Observe that by construction $\boldsymbol{\mathcal{F}} \subseteq 2^{\mathcal{O}}$ is composed of all the allocations $\mathcal{G}$ satisfying: i) $|\mathcal{G}| \leq C$, and ii) $|\mathcal{L}(w,\mathcal{G})| \leq r_w \ \forall w$, where $\mathcal{L}(w,\mathcal{G})$ represents the set of individual assignments in $\mathcal{G}$ associated to w. The advantage of the set notation is that we can characterize the structure of the family $\boldsymbol{\mathcal{F}}$ on which the performance optimization must be carried out; in particular, we can prove that:

Proposition 3.1: The family $\boldsymbol{\mathcal{F}}$ forms a Matroid [16]. Furthermore, $\boldsymbol{\mathcal{F}}$ satisfies the following property. Let $\boldsymbol{\mathcal{B}} \in \boldsymbol{\mathcal{F}}$ be the family of maximal sets in $\boldsymbol{\mathcal{F}}$, then

$$q = \frac{\max_{\mathcal{G}\in\boldsymbol{\mathcal{B}}} |\mathcal{G}|}{\min_{\mathcal{G}\in\boldsymbol{\mathcal{B}}} |\mathcal{G}|} = 1\,.$$

The proof is reported in [17].

1) Computational complexity: the complexity of the above optimal allocation problem heavily depends on the structure of the objective function $P(\mathcal{D})$ (which is rewritten as $P(\mathcal{G})$ when we adopt the set notation). As a general property, observe that necessarily $P(\mathcal{G})$ is monotonic, in the sense that $P(\mathcal{G}_1) \leq P(\mathcal{G}_2)$ whenever $\mathcal{G}_1 \subset \mathcal{G}_2$. However, in general, we cannot assume that $P(\mathcal{G})$ satisfies any other specific property (some possible definitions for $P(\mathcal{G})$ are given next). For a general monotonic objective function, the optimal allocation of tasks to workers can be shown to be NP-hard, since it includes as a special case the well-known problem of the maximization of a monotonic submodular function, subject to a uniform Matroid constraint (see [16])[1]. When $P(\mathcal{G})$ is submodular, the optimal allocation problem falls in the well-known class of problems related to the maximization of a monotonic submodular function subject to Matroid constraints. For such problems, it has been proved that a greedy algorithm yields a 1/(1+q)-approximation [16] (where q is defined as in Proposition 3.1).

In the next subsections, we consider different choices for the performance parameter $P(\mathcal{D})$.

B. Average task error probability

A possible objective of the optimization, which is most closely related to typical performance measures in practical crowdsourcing systems, is the average task error probability, which is defined as:

$$P_1(\mathcal{D}) = -\frac{1}{T}\sum_{t=1}^{T} P_{e,t} \tag{2}$$

with $P_{e,t} = \mathbb{P}\{\hat{\tau}_t \neq \tau_t\} = \mathbb{P}\{\hat{\tau}_t \neq 1 | \tau_t = 1\}$ where the second equality follows from symmetry. Of course, $P_{e,t}$ can be exactly computed only when the true workers' error probabilities p_{tw} are available; furthermore it heavily depends on the adopted decoding scheme. As a consequence, in general, $P_{e,t}$ can only be approximately estimated by the requester by confusing the actual worker error probability p_{tw} (which is unknown) with the corresponding average class error probability π_{tk}. Assuming a maximum-a-posteriori (MAP) decoding scheme, namely, $\hat{\tau}_t(\boldsymbol{\alpha}) = \arg\max_{\tau_t \in \pm 1} \mathbb{P}\{\tau_t | \mathbf{a}_t = \boldsymbol{\alpha}\}$, where $\mathbf{a}_t$ is the t-th row of $\mathbf{A}$ and $\boldsymbol{\alpha}$ is its observed value, we have

$$P_{e,t} = \sum_{\boldsymbol{\alpha}:\mathbb{P}\{\tau_t=1|\mathbf{a}_t=\boldsymbol{\alpha}\}<1/2} \mathbb{P}\{\mathbf{a}_t = \boldsymbol{\alpha} | \tau_t = 1\}. \tag{3}$$

It is easy to verify that the exact computation of the previous average task error probability estimate requires a number of operations growing exponentially with the number of classes K. Thus, when the number of classes K is large, the evaluation of (3) can become critical.

To overcome this problem, we can compare the performance of different allocations on the basis of a simple pessimistic estimate of the error probability, obtained by applying the Chernoff bound to the random variable that is driving the maximum-a-posteriori (MAP) decoding (details on a MAP decoding scheme are provided in the next section). We have:

$$P_{e,t} \leq \hat{P}_{e,t} = \exp\left(-\frac{\sum_k d_{tk}(1-2\pi_{tk})z_{tk}}{\sum_j (d_{tk} z_{tk})^2}\right)$$

where $z_{tk} = \log(\frac{1-\pi_{tk}}{\pi_{tk}})$. Thus, the performance metric associated with an allocation becomes:

$$P_2(\mathcal{D}) = -\frac{1}{T}\sum_{t=1}^{T} \hat{P}_{e,t}$$

The computation of $P_2(\mathcal{D})$ requires a number of operations that scales linearly with the product $T \cdot K$. At last, we would like to remark that in practical cases we expect the number of classes to be sufficiently small (order of few units), in such cases the evaluation of (3) is not really an issue.

C. Overall mutual information

An alternative information-theoretic choice for $P(\mathcal{D})$ is the mutual information between the vector of RVs associated with tasks $\boldsymbol{\tau} = (\tau_1, \tau_2, \ldots, \tau_T)$ and the answer matrix $\mathbf{A}$, i.e.,

$$P_3(\mathcal{D}) = I(\mathbf{A}; \boldsymbol{\tau}) = \sum_{t=1}^{T} I(\mathbf{a}_t; \tau_t)\,. \tag{4}$$

It is well known that a tight relation exists between the mutual information and the achievable error probability, so that a maximization of the former corresponds to a minimization of the latter. We remark, however, that, contrary to error probability, mutual information is independent from the adopted decoding scheme, because it refers to an optimal decoding scheme. This property makes the adoption of the mutual information as the objective function for the task assignment quite attractive, since it permits to abstract from the decoding scheme. The second equality in (4) comes from the fact that tasks are independent and workers are modeled as BSCs with known error probabilities, so that answers to a given task do not give

[1] A set function $f : 2^{\mathcal{O}} \to \mathbb{R}^+$ is said to be submodular if: $\forall \mathcal{A}, \mathcal{B} \in: 2^{\mathcal{O}}$ we have $f(\mathcal{A}\cup\mathcal{B}) + f(\mathcal{A}\cap\mathcal{B}) \leq f(\mathcal{A}) + f(\mathcal{B})$. The problem of the maximization of a monotonic submodular function subject to a uniform Matroid constraint corresponds to: $\{\max_{|\mathcal{A}|\leq K} f(\mathcal{A})$ for $K < |\mathcal{O}|$ with $f(.)$ submodular.$\}$

any information about other tasks. By definition

$$I(\mathbf{a}_t;\tau_t) = H(\mathbf{a}_t) - H(\mathbf{a}_t|\tau_t) = H(\tau_t) - H(\tau_t|\mathbf{a}_t) \quad (5)$$

where $H(a)$ denotes the entropy of the RV a, given by

$$H(a) = -\mathbb{E}_a[\log \mathbb{P}(a)]$$

and for any two random variables a, b, $H(a|b)$ is the conditional entropy defined as

$$H(a|b) = -\mathbb{E}_b \mathbb{E}_{a|b}[\log \mathbb{P}(a|b)].$$

In what follows, we assume perfect knowledge of worker reliabilities, i.e., we assume that each class-k worker has error probability with respect to task τ_t exactly equal to π_{tk}, remarking than in the more general case, the quantities we obtain by substituting p_{tw} with the corresponding class average π_{tk}, can be regarded as computable approximations for the true uncomputable mutual information.

Since we have modeled all workers as BSCs, each single answer is independent of everything else given the task value, so that

$$H(\mathbf{a}_t|\tau_t) = \sum_{a_{tw}\neq 0} H(a_{tw}|\tau_t) = \sum_{k=1}^{K} d_{tk} H_b(\pi_{tk}). \quad (6)$$

where

$$H_b(p) = -p\log p - (1-p)\log(1-p).$$

For the second equality in (5), $H(\tau_t) = 1$ because τ_t is a uniform binary RV, and

$$\begin{aligned} H(\tau_t|\mathbf{a}_t) &= \sum_{\boldsymbol{\alpha}} \mathbb{P}\{\mathbf{a}_t = \boldsymbol{\alpha}\} H(\tau_t|\mathbf{a}_t = \boldsymbol{\alpha}) \\ &= \sum_{\boldsymbol{\alpha}} \mathbb{P}\{\mathbf{a}_t = \boldsymbol{\alpha}\} H_b(\mathbb{P}\{\tau_t = 1|\mathbf{a}_t = \boldsymbol{\alpha}\}) \quad (7) \end{aligned}$$

where $\boldsymbol{\alpha}$ runs over all possible values of $\mathbf{a}_t$.

By symmetry, for every $\boldsymbol{\alpha}$ such that $\mathbb{P}\{\tau_t = 1|\mathbf{a}_t = \boldsymbol{\alpha}\} < \frac{1}{2}$, there is $\boldsymbol{\alpha}'$ such that $\mathbb{P}\{\mathbf{a}_t = \boldsymbol{\alpha}'\} = \mathbb{P}\{\mathbf{a}_t = \boldsymbol{\alpha}\}$ and $\mathbb{P}\{\tau_t = 1|\mathbf{a}_t = \boldsymbol{\alpha}'\} = 1 - \mathbb{P}\{\tau_t = 1|\mathbf{a}_t = \boldsymbol{\alpha}\}$. As a consequence, we can write

$$\begin{aligned} H(\tau_t|\mathbf{a}_t) &= 2 \sum_{\boldsymbol{\alpha}:\mathbb{P}\{\tau_t=1|\mathbf{a}_t=\boldsymbol{\alpha}\}<1/2} \mathbb{P}\{\mathbf{a}_t = \boldsymbol{\alpha}\} H_b(\mathbb{P}\{\tau_t = 1|\mathbf{a}_t = \boldsymbol{\alpha}\}) \\ &= \sum_{\boldsymbol{\alpha}:\mathbb{P}\{\tau_t=1|\mathbf{a}_t=\boldsymbol{\alpha}\}<1/2} \left(\mathbb{P}\{\mathbf{a}_t = \boldsymbol{\alpha}|\tau_t = 1\} + \mathbb{P}\{\mathbf{a}_t = \boldsymbol{\alpha}|\tau_t = -1\}\right) \cdot \\ &\quad H_b(\mathbb{P}\{\tau_t = 1|\mathbf{a}_t = \boldsymbol{\alpha}\}) \quad (8) \end{aligned}$$

Notice the relationship of the above expression with (3). If in (8) we substitute $H_b(\mathbb{P}\{\tau_t = 1|\mathbf{a}_t = \boldsymbol{\alpha}\})$ with $\mathbb{P}\{\tau_t = 1|\mathbf{a}_t = \boldsymbol{\alpha}\}$, thanks to Bayes' rule, we obtain (3).

An explicit computation of $I(\mathbf{A};\boldsymbol{\tau})$ can be found in Appendix A. As for the task error probability, the number of elementary operations required to compute $I(\mathbf{A};\boldsymbol{\tau})$ grows exponentially with the number of classes K.

An important property that mutual information satisfies is submodularity. This property provides some guarantees about the performance of the greedy allocation algorithm described in Section IV-A.

Proposition 3.2 (Submodularity of the mutual information): Let $\mathcal{G}$ be a generic allocation for task θ. Then, the mutual information $I(\mathcal{G};\theta)$ is a submodular function.

Proof: The proof is given in [17]. ■

D. Max-min performance parameters

The previous optimization objectives represent a sensible choice whenever the target is to optimize the *average* task performance. However, in a number of cases it can be more appropriate to optimize the worst performance among all tasks, thus adopting a max-min optimization approach.

Along the same lines used in the definition of the previous optimization objectives, we can obtain three other possible choices of performance parameters to be used in the optimization problem defined in (1), namely, the maximum task error probability,

$$P_4(\mathcal{D}) = -\max_{t=1,\ldots,T} P_{e,t} \quad (9)$$

the Chernoff bound on the maximum task error probability,

$$P_5(\mathcal{D}) = -\max_{t=1,\ldots,T} \hat{P}_{e,t} \quad (10)$$

and the minimum mutual information,

$$P_6(\mathcal{D}) = \min_{t=1,2,\ldots,T} I(\mathbf{a}_t;\tau_t). \quad (11)$$

IV. Allocation strategies

As we observed in Section III, the optimization problem stated in (1) is NP-hard, but the submodularity of the mutual information objective function over a Matroid, coupled with a greedy algorithm yields a 1/2-approximation [16] (see Proposition 3.1). We thus define in this section a greedy task assignment algorithm, to be coupled with the MAP decision rule which is discussed in the next section.

A. Greedy task assignment

The task assignment we propose to approximate the optimal performance is a simple greedy algorithm that starts from an empty assignment ($\mathcal{G}^{(0)} = \emptyset$), and at every iteration i adds to $\mathcal{G}^{(i-1)}$ the individual assignment $(t,w)^{(i)}$, so as to maximize the objective function. In other words;

$$(t,w)^{(i)} = \underset{(t,w)\in\mathcal{O}\setminus\mathcal{G}^{(i-1)}, (\mathcal{G}^{(i-1)}\cup\{(t,w)\})\in\boldsymbol{\mathcal{F}}}{\arg\max} P(\mathcal{G}^{(i-1)} \cup \{(t,w)\})$$

The algorithm stops when no assignment can be further added to $\mathcal{G}$ without violating some constraint.

To execute this greedy algorithm, at step i, for every task t, we need to i) find, if any, the best performing worker to which task t can be assigned without violating constraints, and mark the assignment (t,w) as a candidate assignment; ii) evaluate for every candidate assignment the performance index $P(\mathcal{G}^{(i-1)}\cup(t,w))\ \forall t$; iii) choose among all the candidate assignments the one that greedily optimizes performance.

Observe that, as a result, the computational complexity of our algorithm is $O(T^2 \cdot WZ)$ where Z represents the number of operations needed to evaluate $P(\mathcal{G})$.

Note that in light of both Propositions 3.1 and 3.2, the above greedy task assignment algorithm provides a $1/2$-approximation when the objective function $P_3(\mathcal{G})$ is chosen.

Furthermore, we wish to mention that a better $1 - 1/e$-approximation can be obtained by cascading the above greedy algorithm with the special local search optimization algorithm proposed in [16]; unfortunately, the corresponding cost in terms of computational complexity is rather severe, because the number of operations requested to run the local search procedure is $\widetilde{O}((T \cdot W)^8 Z)$ [2].

B. Uniform allocation

Here we briefly recall that [10], [11] proposed a simple task allocation strategy (under the assumption that workers are indistinguishable) according to which a random regular bipartite graph is established between tasks and selected workers. Every selected worker is assigned the same maximal number of tasks, i.e. $r_w = r \ \forall w$, except for rounding effects induced by the constraint on the maximum total number of possible assignments C.

V. Decision rules

A. Majority voting

Majority voting is the simplest possible task-decision rule which is currently implemented in all real-world crowdsourcing systems. For every task θ_t, it simply consists in counting the $\{+1\}$ and the $\{-1\}$ in $\mathbf{a}_t$ and then taking $\hat{\tau}_t(\mathbf{a}_t)$ in accordance to the answer majority. More formally:

$$\hat{\tau}_t(\mathbf{a}_t) = \mathrm{sgn}\left(\sum_w a_{tw}\right). \tag{12}$$

B. MAP decision rule

We investigate the performance of the greedy task assignment algorithm, when coupled with the MAP decision rule for known workers reputation.

Given an observed value of $\mathbf{a}_t$, the posterior log-likelihood ratio (LLR) for task τ_t is

$$\begin{aligned} \mathrm{LLR}_t(\mathbf{a}_t) &= \log \frac{\mathbb{P}\{\tau_t = 1|\mathbf{a}_t\}}{\mathbb{P}\{\tau_t = -1|\mathbf{a}_t\}} \\ &= \log \frac{\mathbb{P}\{\mathbf{a}_t|\tau_t = 1\}}{\mathbb{P}\{\mathbf{a}_t|\tau_t = -1\}} \\ &= \sum_{w:a_{tw}\neq 0} \log \frac{\mathbb{P}\{a_{tw}|\tau_t = 1\}}{\mathbb{P}\{a_{tw}|\tau_t = -1\}} \end{aligned} \tag{13}$$

where the second equality comes from Bayes' rule and the fact that tasks are uniformly distributed over ± 1, and the third equality comes from modelling workers as BSCs. Let m_{tk} be the number of "-1" answers to task t from class-k workers. Then

$$\mathrm{LLR}_t(\mathbf{a}_t) = \sum_{k=1}^{K} (d_{tk} - 2m_{tk}) \log \frac{1 - \pi_{tk}}{\pi_{tk}}. \tag{14}$$

The MAP decision rule outputs the task solution estimate $\hat{\tau}_t = 1$ if $\mathrm{LLR}_t > 0$ and $\hat{\tau}_t = -1$ if $\mathrm{LLR}_t < 0$, that is,

$$\hat{\tau}_t(\mathbf{a}_t) = \mathrm{sgn}\left(\mathrm{LLR}_t(\mathbf{a}_t)\right). \tag{15}$$

[2]The function $f(n)$ is $\widetilde{O}(g(n))$ if $f(n) = O(g(n)\log^b n)$ for any positive constant b.

Observe that the computation of (14) has a complexity growing only linearly with K, and that, according to (15), each task solution is estimated separately. Note also that, whenever worker reputation is *not* known a-priori, the above decision rule is no more optimal, since it neglects the information that answers to other tasks can provide about worker reputation.

C. Low-rank approximation (LRA)

Finally, for the sake of comparison, we briefly recall here the Low-Rank Approximation decision rule proposed in [10], [11], [14] for the case when: i) no a-priori information about the reputation of workers is available, ii) the error probability of every individual worker w is the same for every task, i.e., $p_{tw} = p_w \ \forall t$. The LRA decision rule was shown to provide asymptotically optimal performance under assumptions i) and ii) [11].

Denote with $\mathbf{v}$ the leading right singular vector of $\mathbf{A}$, the LRA decision is taken according to:

$$\hat{\tau}_t(\mathbf{a}_t) = \mathrm{sgn}\left(\mathrm{LRA}(\mathbf{a}_t)\right)$$

where

$$\mathrm{LRA}(\mathbf{a}_t) = \sum_w a_{tw} v_w$$

The idea underlying the LRA decision rule is that each component of the leading singular vector of $\mathbf{A}$, measuring the degree of coherence among the answers provided by the correspondent worker, represents a good estimate of the worker reputation.

VI. Results

In this section, we study the performance of a system where $T = 100$ tasks are assigned to a set of workers which are organized in $K = 3$ classes. Each worker can handle up to 20 tasks, i.e., $r_w = 20$, $w = 1, \ldots, W$.

We compare the performance of the allocation algorithms and decision rules described in Sections IV and V, in terms of achieved average error probability, P_e. More specifically, we study the performance of:

- the "Majority voting" decision rule applied to the "Uniform allocation" strategy, hereinafter referred to as "Majority";
- the "Low rank approximation" decision rule applied to the "Uniform allocation" strategy, in the figures referred to as "LRA uniform";
- the "Low rank approximation" decision rule applied to the "Greedy allocation" strategy, in the figures referred to as "LRA greedy";
- the "MAP" decision rule applied to the "Greedy allocation" strategy, in the following referred to as "MAP greedy".

Specifically, for the greedy allocation algorithm, described in Section IV-A, we employed the overall mutual information $P_3(\mathcal{D})$ as objective function.

The first set of results is reported in Figure 1. There we considered the most classical scenario where tasks are identical.

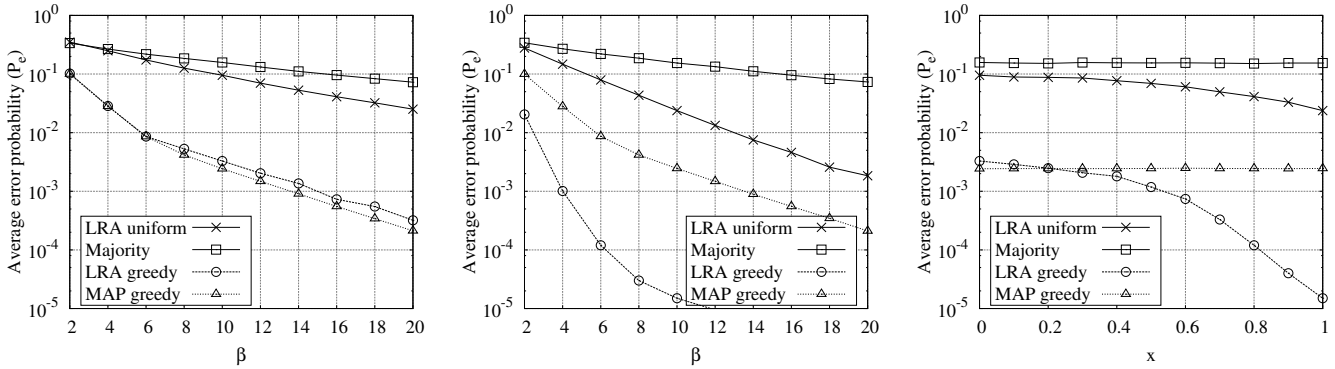

Fig. 1. Average error probability as a function of the average number of workers per task, β, and of the parameter x, for $\pi_{t1} = 0.1, \pi_{t2} = 0.2, \pi_{t3} = 0.5$, $W_1 = 30, W_2 = 120$, and $W_3 = 150$. In figure (a) and (b) we set $x = 0$ and $x = 1$, respectively, while in figure (c) we set $\beta = 10$.

The results depicted in Figure 1(a) assume that all workers belonging to the same class have the same error probability i.e., $p_{tw} = \pi_{tk}$. In particular, we set $\pi_{t1} = 0.1, \pi_{t2} = 0.2, \pi_{t3} = 0.5$ for all t. This means that workers in class 1 are the most reliable, while workers in class 3 are spammers. Moreover, the number of available workers per class is set to $W_1 = 30, W_2 = 120, W_3 = 150$. The figure shows the average error probability achieved on the tasks, plotted versus the average number of workers per task, $\beta = C/T$. As expected, greedy allocation strategies perform better due to the fact that they exploit the knowledge about the workers' reliability (p_{tw}), and thus they assign to tasks the best possible performing workers. These strategies provide quite a significant reduction of the error probability, for a given number of workers per task, or a reduction in the number of assignments required to achieve a fixed target error probability. For example, $P_e = 10^{-2}$ can be achieved by greedy algorithms by assigning only 4 workers per task (on average), while algorithms unaware of workers reliability require more than 20 workers per task (on average). We also observe that the LRA algorithm proposed in [10] performs similarly to the optimal MAP algorithm.

Next, we take into account the case where in each class workers do not behave exactly the same. As already observed, this may reflect both possible inaccuracies/errors in the reconstruction of user profiles, and the fact that the behavior of workers is not fully predictable, since it may vary over time. Specifically, we assume that, in each class, two types of workers coexist, each characterized by a different error probability p_{tw}. More precisely, workers of type 1 have error probability $p_{tw} = (1 - x)\pi_{tk}$, while workers of type 2 have error probability probability $p_{tw} = (1 - x)\pi_{tk} + x/2$, where $0 \leq x \leq 1$ is a parameter. Moreover workers are of type 1 and type 2 with probability $1 - 2\pi_{tk}$ and $2\pi_{tk}$, respectively, so that the average error probability over the workers in class k is π_{tk}. We wish to emphasize that this bimodal worker model, even if it may appear somehow artificial, is attractive for the following two reasons: i) it is simple (it depends on only one scalar parameter x), and ii) it encompasses as particular cases the two extreme cases of full knowledge and hammer-spammer. Indeed, for $x = 0$ all workers in each class behave exactly the same (they all have error probability $p_{tw} = p_{tk}$); this is the case depicted in Figure 1(a), while for $x = 1$ we recover the hammer-spammer scenario. This case is represented in Figure 1(b), where workers are spammers with probability $2\pi_{tk}$ and hammers with probability $1 - 2\pi_{tk}$. Here, the greedy allocation algorithms still outperform the others. However, the MAP decision rule provides performance lower than the "LRA greedy" due to the following two facts: i) MAP adopts a mismatched value of the error probability of individual workers, when $x \neq 0$, ii) MAP does not exploit the extra information on individual worker reliability that is possible to gather by jointly decoding different tasks. In Figure 1(c), for $\beta = 10$, we show the error probability plotted versus the parameter x. We observe that the performance of the "MAP greedy" strategy is independent on the parameter x while the performance of "LRA greedy" improve as x increases. This effect can be explained by observing that the LRA ability of distinguishing good performing workers from bad performing workers within the same class increases as x increases.

Next, we assume that the $T = 100$ tasks are divided into 2 groups of 50 each. Workers processing tasks of group 1 and 2 are characterized by average error probabilities $\pi_{t1} = 0.05, \pi_{t2} = 0.1, \pi_{t3} = 0.5$ and $\pi_{t1} = 0.1, \pi_{t2} = 0.2, \pi_{t3} = 0.5$, respectively. This scenario reflects the case where tasks of group 2 are more difficult to solve than tasks of group 1 (error probabilities are higher). Workers of class $\mathcal{C}_3$ are spammers for both kinds of tasks. The error probabilities provided by the algorithms under study are depicted in Figure 2, as a function of x and for $\beta = 10$.

We observe that all strategies perform similarly, like in the scenario represented by Figure 1(c), meaning that the algorithms are robust enough to deal with changes in the behavior of workers with respect to tasks. We wish to remark that the LRA decoding scheme is fairly well performing also in this scenario, even if it was conceived and proposed only for the simpler scenario of indistinguishable tasks. This should not be too surprising, in light of the fact that, even if the error probability of each user depends on the specific task, the relative ranking among workers remains the same for all tasks.

Finally, in Figure 3 we consider the same scenario as in Figure 2. Here, however, the number of available workers per class is set to $W_1 = 40, W_2 = 120, W_3 = 40$, and the workers error probabilities for the tasks in group 1 and 2 are given

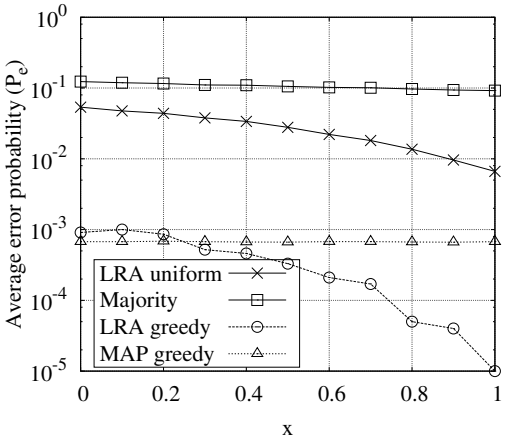

Fig. 2. Average error probability plotted versus x, for $\beta = 10$ and task organized in two groups of different difficulties. For the first group of tasks $\pi_{t1} = 0.05, \pi_{t2} = 0.1, \pi_{t3} = 0.5$, while for the second group $\pi_{t1} = 0.1, \pi_{t2} = 0.2, \pi_{t3} = 0.5$, Moreover $W_1 = 30, W_2 = 120, W_3 = 150$.

by $\pi_{t1} = 0.1, \pi_{t2} = 0.25, \pi_{t3} = 0.5$, and $\pi_{t1} = 0.5, \pi_{t2} = 0.25, \pi_{t3} = 0.1$, respectively. This situation reflects the case where workers are more specialized or interested in solving some kinds of tasks. More specifically, here workers of class 1 (class 3) are reliable when processing tasks of group 1 (group 2), and behave as spammers when processing tasks of group 2 (group 1). Workers of class 2 behave the same for all tasks. In terms of performance, the main difference with respect to previous cases is that the "LRA greedy" algorithm shows severely degraded error probabilities for $\beta \leq 16$. This behavior should not surprise the reader, since our third scenario may be considered as adversarial for the LRA scheme, in light of the fact that the relative ranking among workers heavily depends on the specific task. Nevertheless, it may still appear amazing that "LRA greedy" behaves even worse than the simple majority scheme in several cases. The technical reason for this behavior is related to the fact that, in our example, for $\beta \leq 16$, tasks of group 1 (group 2) are allocated to workers of class 1 (class 3) only, whilst workers of class 2 are not assigned any task. For this reason, the matrix $\mathbf{A}$ turns out to have a block diagonal structure, which conflicts with the basic assumption made by LRA that matrix $\mathbb{E}[\mathbf{A}]$ can be well approximated by a unitary rank matrix. For $\beta > 16$, tasks are also allocated to workers of class 2; in this situation, the matrix $\mathbf{A}$ is not diagonal anymore, and the "LRA greedy" performs quite well. Observe, however, that also in this case even a fairly imprecise characterization of the worker behavior can be effectively exploited by the requester to significantly improve system performance.

Finally, we want to remark that we have tested several versions of greedy algorithms under different objective functions, such as $P_1(\mathcal{D})$, $P_2(\mathcal{D})$, and $P_3(\mathcal{D})$, finding that they provide, in general, comparable performance. The version employing mutual information was often providing slightly better results, especially in the case of LRA greedy. This can be attributed to the following two facts: i) the mutual information was proved to be submodular; ii) being mutual information independent from the adopted decoding scheme, it provides a more reliable metric for comparing the performance of different task allocations under the LRA decoding scheme with respect to the error probability $P_1(\mathcal{D})$ (which, we recall, is computed under the assumption that the decoding scheme is MAP). Unfortunately, due to the lack of space, we cannot include these results in the paper.

VII. Concluding Remarks

In this paper we have presented the first systematic investigation of the impact of information about workers' reputation in the assignment of tasks to workers in crowdsourcing systems, quantifying the potential performance gains in several cases. We have formalized the optimal task assignment problem when workers' reputation estimates are available, as the maximization of a monotone (submodular) function subject to Matroid constraints. Then, being the optimal problem NP-hard, we have proposed a simple but efficient greedy heuristic task allocation algorithm, combined with a simple "maximum a-posteriori" decision rule. We have tested our proposed algorithms, and compared them to different solutions, which can be obtained by extrapolating the proposals for the cases when reputation information is not available, showing that the crowdsourcing system performance can greatly benefit from even largely inaccurate estimates of workers' reputation. Our numerical results have shown that:

- even a significantly imperfect characterization of the workers' earnestness can be extremely useful to improve the system performance;
- the application of advanced joint tasks decoding schemes such as LRA can further improve the overall system performance, especially in the realistic case in which the a-priori information about worker reputation is largely affected by errors;
- the performance of advanced joint tasks decoding schemes such as LRA may become extremely poor in adversarial scenarios.

Appendix

The workers' answers about the tasks $\boldsymbol{\tau}$ are collected in the random $T \times W$ matrix $\mathbf{A}$, defined in Section II. The information that the answers $\mathbf{A}$ provide about the tasks $\boldsymbol{\tau}$ is denoted by

$$\mathcal{I}(\mathbf{A}; \boldsymbol{\tau}) = H(\mathbf{A}) - H(\mathbf{A}|\boldsymbol{\tau})$$

where the entropy $H(a)$ and the conditional entropy $H(a|b)$ have been defined in Section III-C. We first compute $H(\mathbf{A}|\mathbf{t})$ and we observe that, given the tasks $\boldsymbol{\tau}$, the answer $\mathbf{A}$ are independent, i.e., $\mathbb{P}(\mathbf{A}|\boldsymbol{\tau}) = \prod_{k=1}^{K} \prod_{t=1}^{T} \mathbb{P}(\mathbf{a}_{tk}|\tau_t)$, where $\mathbf{a}_{tk}$ is the vector of answers to task θ_t from users of class $\mathcal{C}_k$. Since $\mathbb{P}(\mathbf{A}|\boldsymbol{\tau})$ has a product form, we obtain $H(\mathbf{A}|\boldsymbol{\tau}) = \sum_{k=1}^{K} \sum_{t=1}^{T} H(\mathbf{a}_{tk}|\tau_t)$. Thanks to the fact that workers of the same class are independent and all have error probability π_{tk}, we can write $H(\mathbf{a}_{tk}|\tau_t) = d_{ik} H_b(\pi_k)$ where $H_b(p) = -p \log p - (1-p)\log(1-p)$ and d_{tk} is the number of allocations of task t in class $\mathcal{C}_k$. In conclusion, we get:

$$H(\mathbf{A}|\boldsymbol{\tau}) = \sum_{t=1}^{T} \sum_{k=1}^{K} d_{tk} H_b(\pi_{tk})$$

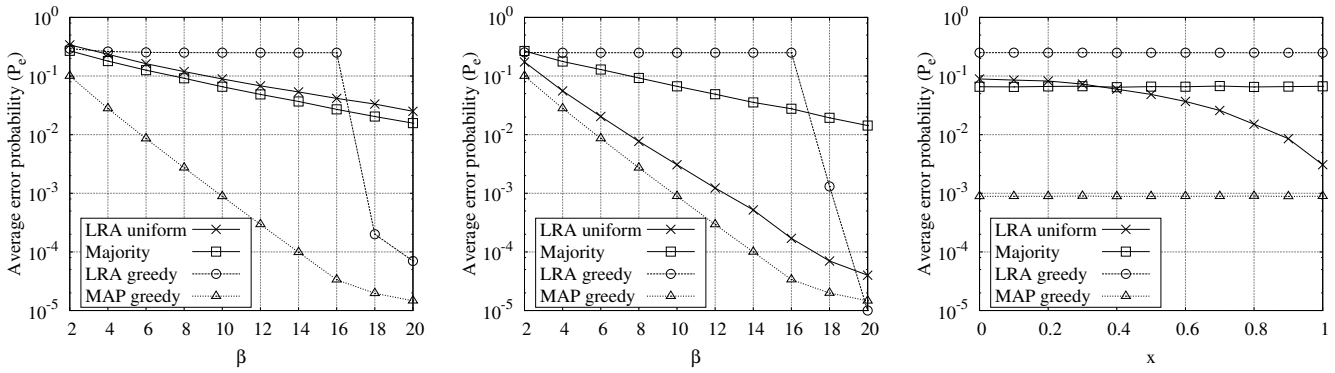

Fig. 3. Average error probability as a function of the average number of workers per task, β, and of the parameter x. Task organized in two groups of different difficulties. For the first group of tasks $\pi_{t1} = 0.1, \pi_{t2} = 0.25, \pi_{t3} = 0.5$, while for the second group $\pi_{t1} = 0.5, \pi_{t2} = 0.25, \pi_{t3} = 0.1$. Moreover $W_1 = 40, W_2 = 120$ and $W_3 = 40$. In figures (a) and (b) we set $x = 0$ and $x = 1$, respectively, while in figure (c) we set $\beta = 10$.

As for the entropy $H(\mathbf{A})$, we have:

$$\mathbb{P}(\mathbf{A}) = \mathbb{E}_{\boldsymbol{\tau}}\mathbb{P}(\mathbf{A}|\boldsymbol{\tau}) = \mathbb{E}_{\boldsymbol{\tau}} \prod_{t=1}^{T} \mathbb{P}(\mathbf{a}_t|\tau_t) = \prod_{t=1}^{T} \mathbb{E}_{\tau_t}\mathbb{P}(\mathbf{a}_t|\tau_t)$$

where $\mathbf{a}_t$ is the vector of answers to task θ_t (corresponding to the t-th row of $\mathbf{A}$). Note that $\mathbb{E}_{\tau_t}\mathbb{P}(\mathbf{a}_t|\tau_t) = \mathbb{P}(\mathbf{a}_t)$, hence $\mathbb{P}(\mathbf{A}) = \prod_{t=1}^{T} \mathbb{P}(\mathbf{a}_t)$ and we immediately obtain $H(\mathbf{A}) = \sum_{t=1}^{T} H(\mathbf{a}_t)$. The probabilities $\mathbb{P}(\mathbf{a}_{tk}|\tau_t = 1)$ and $\mathbb{P}(\mathbf{a}_{tk}|\tau_i = -1)$ are easy to compute. Indeed for $\tau_t = -1$ we have

$$\mathbb{P}(\mathbf{a}_{tk}|\tau_t = -1) = \pi_{tk}^{d_{tk}-m_{tk}}(1-\pi_{tk})^{m_{tk}} \tag{16}$$

where m_{tk} is the number of "-1" answers to task θ_t from class-k workers. The above formula derives from the fact that workers of the same class are independent and have the same error probability π_{tk}. Similarly

$$\mathbb{P}(\mathbf{a}_{tk}|\tau_t = +1) = \pi_{tk}^{m_{tk}}(1-\pi_{tk})^{d_{tk}-m_{tk}} \tag{17}$$

The expressions (16) and (17) can compactly written as

$$\mathbb{P}(\mathbf{a}_{tk}|\tau_t) = (1-\pi_{tk})^{d_{tk}} b_{tk}^{(1-\tau_t)d_{tk}/2 - m_{tk}\tau_t} \tag{18}$$

where $b_{tk} = \pi_{tk}/(1-\pi_{tk})$. Since, given τ_t, workers are independent, we obtain

$$\begin{aligned}
\mathbb{P}(\mathbf{a}_t) &= \mathbb{E}_{\tau_t}\mathbb{P}(\mathbf{a}_t|\tau_t) = \gamma_{tk}\mathbb{E}_{\tau_t}\left[\prod_{k=1}^{K} b_{tk}^{(1-\tau_t)d_{tk}/2 - m_{tk}\tau_t}\right] \\
&= \frac{\gamma_{tk}}{2}\left[\prod_{k=1}^{K} b_{tk}^{-m_{tk}} + \prod_{k=1}^{K} b_{tk}^{d_{tk}+m_{tk}}\right] = \frac{\gamma_{tk}}{2} f(\mathbf{m}_t)
\end{aligned}$$

with $\mathbf{m}_t = [m_{t1}, \ldots, m_{tK}]$ and $f(\mathbf{n}) = \prod_{k=1}^{K} b_{tk}^{-n_k} + \prod_{k=1}^{K} b_{tk}^{d_{tk}+n_k}$. Finally, by using the definition of entropy,

$$\begin{aligned}
H(\mathbf{a}_t) &= \mathbb{E}_{\mathbf{a}_t}[-\log \mathbb{P}(\mathbf{a}_t)] \\
&= -\log\frac{\gamma_{tk}}{2} - \mathbb{E}_{\mathbf{a}_t} f(\mathbf{m}_t) \\
&= -\log\frac{\gamma_{tk}}{2} - \frac{\gamma_{tk}}{2}\sum_{\mathbf{n}} f(\mathbf{n}) \log f(\mathbf{n}) \prod_{k=1}^{K} \binom{m_{tk}}{n_k}
\end{aligned}$$

where $\mathbf{n} = [n_1, \ldots, n_K]$ and $n_k = 0, \ldots, m_{tk}$, $k = 1, \ldots, K$.

REFERENCES

[1] M.-C. Yuen, I. King, K.-S. Leung, "A Survey of Crowdsourcing Systems," *IEEE PASSAT-SOCIALCOM*, Boston (USA), Oct. 2011.

[2] A. Kittur, J. V. Nickerson, M. Bernstein, E. Gerber, A. Shaw, J. Zimmerman, M. Lease, J. Horton, "The future of crowd work," *ACM CSCW '13*, New York (USA), 2013.

[3] E. Christoforou, A. Fernandez Anta, C. Georgiou, M. A. Mosteiro, A. Sanchez, "Applying the dynamics of evolution to achieve reliability in master-worker computing," *Concurrency and Computation: Practice and Experience* vol. 25, n. 17, pp. 2363–2380, 2013.

[4] A. Fernandez Anta, C. Georgiou, M. A. Mosteiro, "Algorithmic mechanisms for internet-based master-worker computing with untrusted and selfish workers," *IEEE IPDPS 2010*, Atlanta (USA), 2010.

[5] A. Fernandez Anta, C. Georgiou, L. Lopez, A. Santos, "Reliable internet-based master-worker computing in the presence of malicious workers," *Parallel Processing Letters*, vol. 22, n. 1, 2012.

[6] A. Singla, A. Krause, "Truthful incentives in crowdsourcing tasks using regret minimization mechanisms," *WWW '13*, Rio de Janeiro (Brazil), 2013.

[7] E. Kamar, E. Horvitz, "Incentives for truthful reporting in crowdsourcing," *AAMAS '12*, Valencia (Spain), 2012.

[8] P. Donmez, J. G. Carbonell, J. Schneider, "Efficiently learning the accuracy of labeling sources for selective sampling," *ACM KDD '09*, New York (USA), 2009.

[9] Y. Zheng, S. Scott, K. Deng, "Active learning from multiple noisy labelers with varied costs," *IEEE ICDM 2010*, Sydney (Australia), 2010.

[10] D. R. Karger, S. Oh, D. Shah, "Budget- optimal Crowdsourcing Using Low-rank Matrix Approximations," 2011 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton), pp.284,291, 28-30 Sept. 2011.

[11] D. R. Karger, S. Oh, D. Shah, "Budget-Optimal Task Allocation for Reliable Crowdsourcing Systems," *Operations Research* vol. 62, no. 1, pp. 1–24, 2014.

[12] D. R. Karger, S. Oh, D. Shah, "Efficient crowdsourcing for multi-class labeling," *ACM SIGMETRICS*, Pittsburgh (USA), 2013.

[13] I. Abraham, O. Alonso, V. Kandylas, A. Slivkins, "Adaptive Crowdsourcing Algorithms for the Bandit Survey Problem," arXiv:1302.3268.

[14] A. Ghosh, S. Kale, P. McAfee, "Who moderates the moderators?: crowdsourcing abuse detection in user-generated content," *ACM EC '11*, New York (USA).

[15] T. M. Cover, J. M. Thomas, *Elements of information theory,* 2nd ed., John Wiley, 2005.

[16] G. Calinescu, C. Chekuri, M. Pl, J. Vondrk, "Maximizing a Monotone Submodular Function Subject to a Matroid Constraint," *SIAM Journal on Computing,* vol. 40, n. 6, pp. 1740–1766, 2011.

[17] Technical Report http://arxiv.org/pdf/1411.7960v1.pdf