Noname manuscript No.
(will be inserted by the editor)

A workload-dependent task assignment policy for crowdsourcing

Ilio Catallo · Stefano Coniglio · Piero Fraternali · Davide Martinenghi

Received: date / Accepted: date

Abstract Crowdsourcing marketplaces have emerged as an effective tool for high-speed, low-cost labeling of massive data sets. Since the labeling accuracy can greatly vary from worker to worker, we are faced with the problem of assigning labeling tasks to workers so as to maximize the accuracy associated with their answers. In this work, we study the problem of assigning workers to tasks under the assumption that workers' reliability could change depending on their workload, as a result of, e.g., fatigue and learning. We offer empirical evidence of the existence of a workload-dependent accuracy variation among workers, and propose solution procedures for our *Crowdsourced Labeling Task Assignment Problem*, which we validate on both synthetic and real data sets.

I. Catallo
Politecnico di Milano, Italy
Tel.: +39 02 2399 7384
Fax: +39 02 2399 3411
E-mail: ilio.catallo@polimi.it

S. Coniglio
University of Southampton, UK
Tel.: +44 023 8059 4546
E-mail: s.coniglio@soton.ac.uk

P. Fraternali
Politecnico di Milano, Italy
Tel.: +39 02 2399 7329
Fax: +39 02 2399 3411
E-mail: piero.fraternali@polimi.it

D. Martinenghi
Politecnico di Milano, Italy
Tel.: +39 02 2399 3669
Fax: +39 02 2399 3411
E-mail: davide.martinenghi@polimi.it

1 Introduction

Many problems in the field of artificial intelligence, engineering, and medicine require the availability of very large data sets consisting of hand-labeled data. While unlabeled data are typically readily available, the subsequent labeling process is extremely time consuming and often represents an important bottleneck that severely limits the applicability of many automatic algorithms.

In recent years, crowdsourcing marketplaces, among which Amazon Mechanical Turk [1], emerged as an effective tool for high-speed, low-cost labeling of massive data sets. Namely, such services allow requesters to publish their own tasks, which are then solved by the workers on the platform in return of few cents of dollar per completed task. The possibility of inexpensively harnessing such a large workforce has therefore enabled the creation of extensive hand-labeled data sets.

In this work, we focus on tasks for which the workers are asked to provide a binary label. Binary classification tasks are prevalent in many application domains, as it is confirmed, for instance, for the case of Amazon Mechanical Turk in [16]. Since the labeling accuracy can greatly vary from worker to worker, requesters often collect different labels for the same instance and, consequently, rely on some aggregation technique in order to increase the quality of the inferred labels [23] and, thus, reduce the classification error.

In this work, we adopt the perspective of the *owner of a crowdsourcing platform*, who provides *task requesters* with a crowdsourcing labeling service for automatically assigning a set of human *annotators* to the requesters' *tasks*.

Similarly to previous works in the literature ([17], [12], [29]), this work is applicable in all contexts involving binary labels, such as entity matching and recognition, copyright detection, and image classification. However, differently from previous works, in this paper we study the problem of assigning annotators to tasks under the assumption that the annotators' reliability could change depending on their workload, as a result of, e.g., fatigue and learning. In this regard, a crowdsourcing platform may use all historic data and trends from previous sessions about its workers to provide a reliable estimate of the typical accuracy variations experienced by each annotator, as successfully done by several works in the literature [28,8,18]. These estimates allow us to devise task assignment policies whose attention to changes in annotators' accuracies proves extremely beneficial in terms of reduction of the classification error, as we will better illustrate in the remainder of the paper.

The main contributions of our work are as follows:

- We introduce the *Crowdsourced Labeling Task Assignment Problem* (CLTAP) for binary classification tasks under the assumption of workload-dependent accuracy variations among workers.
- We offer empirical evidence of the existence of such a workload-dependent accuracy variation phenomenon.
- We propose a measure of correctness for task labels, the maximization of which arguably leads to highly accurate inferred labels.

- We introduce a modified version of our measure of correctness (called *linear correctness*), which allows us to cast CL-TAP as a Mixed-Integer Linear Programming (MILP) problem.
- Due to the poor scaling of state-of-the-art methods to solve our MILP formulation, we propose an efficient heuristic for CL-TAP: Correctness-Based Policy (`CBP`). To validate the quality of the solutions it provides, we also introduce a certification mechanism based on a *column generation* algorithm.
- We extensively validate the solution procedure that we propose on both synthetic and real data, showing that our algorithm attains close-to-optimal quality and always outperforms the best methods available in the literature.

2 Preliminaries

Consider a set $\mathcal{T} = \{1, \ldots, t\}$ of tasks of the same kind, where each task $i \in \mathcal{T}$ is associated with an unknown *true label* $y_i \in \{-1, 1\}$, representing the presence or absence of a target binary property, which we model as a binary random variable. Consider also a set of human annotators $\mathcal{A} = \{1, \ldots, a\}$. Annotators in $\mathcal{A}$ can provide a (noisy) label for each of the t tasks in $\mathcal{T}$, so that each such label is associated with a measure of accuracy, corresponding to the probability of the label being correct. We assume this measure of accuracy to be a function of both the annotator and her workload. In other words, we assume the accuracy of each annotator to depend on the number of labels that the annotator has already provided. Indeed, as it is empirically confirmed by the experiments that we report in Section 6.1, in scenarios involving extensive labeling, the annotators' performance might either decrease due to, e.g., fatigue or increase as a result of, e.g., learning. Thus, for each annotator $j \in \mathcal{A}$, we henceforth denote by p_{jk} her accuracy at her k-th labeling iteration. We denote the set of iterations by $\mathcal{I} = \{1, \ldots, t\}$ (each annotator $j \in \mathcal{A}$ can label at most every task $i \in \mathcal{T}$). Clearly, to be able to assign annotator $j \in \mathcal{A}$ to some task in her k-th iteration, she must have been assigned to some other $k-1$ tasks in the past.

We assume p_{jk} to be known, as inferrable, e.g., from historical data or from the worker's performance on ad-hoc verification tasks, as, e.g., in [28,8]. Indeed, if we consider the perspective of a platform owner, it is reasonable to assume the availability of worker statistics for each task type supported by the platform, as done in [18]. The availability of such data would therefore allow for the estimation of each worker's accuracy. Such an estimation can then be refined each time the worker participates in new labeling sessions. Since each p_{jk} is known, we can assume it to be at least 0.5. For any $p_{jk} < 0.5$, it would indeed be sufficient to consider as annotator's accuracy the quantity $(1 - p_{jk})$, while systematically inverting the provided label. We also assume that, when facing tasks of the same kind, the accuracy p_{jk} for each annotator $j \in \mathcal{A}$

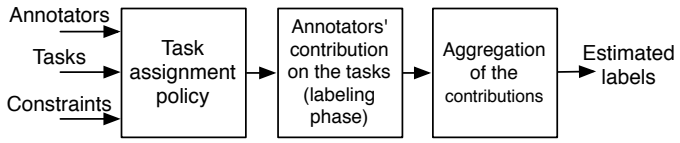


Fig. 1: Execution of a task assignment policy.

does not depend on the specific task at hand, i.e., that annotator j is equally capable of providing the correct label for each task $i \in \mathcal{T}$.

Let $z_{ijk} \in \{-1, 1\}$ denote the *noisy label* that annotator $j \in \mathcal{A}$ would assign to task $i \in \mathcal{T}$ if asked to provide a label for it in her k-th iteration. For each $j \in \mathcal{A}$, $k \in \mathcal{I}$, we assume z_{ijk} to be a random variable over $\{-1, 1\}$, such that:

$$z_{ijk} = \begin{cases} y_i \text{ with probability } p_{jk} \\ -y_i \text{ with probability } (1 - p_{jk}). \end{cases} \quad (1)$$

This corresponds to assuming that each annotator $j \in \mathcal{A}$ provides in her k-th iteration a label z_{ijk} that coincides with the true label y_i with probability p_{jk} or the wrong one, $-y_i$, with probability $1 - p_{jk}$. It is common to assume ([22, 23, 27]) that, for each task $i \in \mathcal{T}$, given the true label y_i, z_{ijk} and $z_{ij'k'}$ are independent for any $j, j' \in \mathcal{A}$, $j \neq j'$, $k, k' \in \mathcal{I}$.

3 Task assignment policy design

As mentioned in the introduction, in this work we assume the perspective of the owner of a crowdsourcing platform who provides the automated assignment of annotators to the requesters' tasks so as to obtain the best possible inferred labels. The natural choice for this kind of objective would be the minimization of the classification error. However, as we will discuss in the following, the error does not lend itself well to such an optimization, due to its combinatorial nature. We will therefore propose an alternative approach based on an approximation of the expected classification accuracy associated with the inferred labels, which we call *correctness*. We show how the corresponding problem of assigning tasks to human annotators to achieve the largest correctness can be described as a combinatorial optimization problem that allows a suitable linearization that we can solve efficiently. After collecting different labels for the same instance, we can rely on some aggregation technique to determine the estimated labels that will be returned to the task requesters. The overall execution pipeline of our technique is reported in Figure 1.

For each task $i \in \mathcal{T}$, we define the *workplan* $\mathcal{W}_i$ as the set of annotator-iteration pairs assigned to task $i \in \mathcal{T}$. We denote by q_i and μ_i the (unknown) prior and, respectively, posterior probabilities of the random event $y_i = 1$, defined as follows: $q_i = \Pr(y_i = 1)$ and $\mu_i = \Pr(y_i = 1 \mid \{z_{ijk}\}_{(j,k) \in \mathcal{W}_i})$. Thus, q_i and μ_i represent our belief on the value of the true label before and after

Notation	Description
$\mathcal{T}$	set of tasks (with $\lvert\mathcal{T}\rvert = t$)
$\mathcal{A}$	set of annotators (with $\lvert\mathcal{A}\rvert = a$)
$\mathcal{I}$	set of annotators' iterations (with $\lvert\mathcal{I}\rvert = t$)
$\mathcal{W}_i$	workplan (set of annotator-iteration pairs that labeled task i)
y_i	true label for task i
z_{ijk}	annotator j's estimated label for task i at her iteration k
p_{jk}	prob. for annotator j to give the right answer at iteration k
ℓ_{jk}	logit of p_{jk}
q_i	prior probability of the event $y_i = 1$ for task i
μ_i	posterior probability of the event $y_i = 1$ for task i
b	budget (max. number of labels that can be obtained)
ϵ_i	classification error associated with workplan $\mathcal{W}_i$
$\hat{p}_i$	classification accuracy associated with workplan $\mathcal{W}_i$

Table 1: Summary of our notation.

collecting the annotators' contributions. The posterior μ_i can be interpreted as a soft (i.e., continuous) estimation of the true label y_i. A hard (i.e., binary) estimation $\hat{y}_i$ of y_i can by obtained by applying a threshold ω ($\omega = 0.5$ in this paper) to μ_i, thus letting $\hat{y}_i = 1$ if $\mu_i \geq \omega$ and $\hat{y}_i = -1$ otherwise.

A summary of our notation is reported in Table 1 (some symbols will be introduced later in the paper).

3.1 Expected classification error and expected classification accuracy

Based on the standard notion of logarithm of the "odds ratio", commonly referred to as *logit*, we now consider the logit of the posterior probability μ_i, i.e., $\text{logit}(\mu_i) = \log\left(\frac{\mu_i}{1-\mu_i}\right)$. It can be shown that $\text{logit}(\mu_i)$ can be expressed as a function of the prior probability q_i, the accuracies $\{p_{jk}\}_{(j,k)\in\mathcal{W}_i}$, and the noisy labels $\{z_{ijk}\}_{(j,k)\in\mathcal{W}_i}$:

Proposition 1 *Let ℓ_{jk} denote* $\text{logit}(p_{jk})$. *Then:*

$$\text{logit}(\mu_i) = \text{logit}(q_i) + \sum_{(j,k)\in\mathcal{W}_i} \ell_{jk} z_{ijk}. \tag{2}$$

Proof In order to ease the notation, let us assume, w.l.o.g., that $y_i, z_{ijk} \in \{0, 1\}$. We have:

$$\begin{aligned}\frac{\mu_i}{1-\mu_i} =& \frac{\Pr(y_i = 1 \mid \{z_{ijk}\}_{(j,k)\in\mathcal{W}_i})}{\Pr(y_i = 0 \mid \{z_{ijk}\}_{(j,k)\in\mathcal{W}_i})} = \\ =& \frac{\Pr(\{z_{ijk}\}_{(j,k)\in\mathcal{W}_i} \mid y_i = 1)\Pr(y_i = 1)}{\Pr(\{z_{ijk}\}_{(j,k)\in\mathcal{W}_i} \mid y_i = 0)\Pr(y_i = 0)} = \\ =& \frac{\Pr(y_i = 1)}{\Pr(y_i = 0)} \prod_{(j,k)\in\mathcal{W}_i} \frac{\Pr(z_{ijk} \mid y_i = 1)}{\Pr(z_{ijk} \mid y_i = 0)} = \\ =& \frac{q_i}{1-q_i} \prod_{(j,k)\in\mathcal{W}_i} \frac{\Pr(z_{ijk} \mid y_i = 1)}{\Pr(z_{ijk} \mid y_i = 0)}.\end{aligned}$$

By applying the logarithm on the left and right-hand sides and adopting the notation $\log(\frac{\mu_i}{1-\mu_i}) = \text{logit}(\mu_i)$ and $\log(\frac{q_i}{1-q_i}) = \text{logit}(q_i)$, we obtain:

$$\text{logit}(\mu_i) = \text{logit}(q_i) + \sum_{(j,k)\in\mathcal{W}_i} \log\left(\frac{\Pr(z_{ijk}|y_i = 1)}{\Pr(z_{ijk}|y_i = 0)}\right).$$

We can rewrite the last term as:

$$\begin{aligned}&\sum_{(j,k)\in\mathcal{W}_i} \log\left((p_{jk})^{z_{ijk}}(1-p_{jk})^{(1-z_{ijk})}\right) + \\ &- \sum_{(j,k)\in\mathcal{W}_i} \log\left((p_{jk})^{(1-z_{ijk})}(1-p_{jk})^{z_{ijk}}\right) = \\ &\sum_{(j,k)\in\mathcal{W}_i} z_{ijk}\log(p_{jk}) + \sum_{(j,k)\in\mathcal{W}_i} (1-z_{ijk})\log(1-p_{jk}) + \\ &- \sum_{(j,k)\in\mathcal{W}_i} (1-z_{ijk})\log(p_{jk}) - \sum_{(j,k)\in\mathcal{W}_i} z_{ijk}\log(1-p_{jk}).\end{aligned}$$

After collecting, first, $\log(p_{jk})$ and $\log(1-p_{jk})$ and, then, $(2z_{ijk}-1)$, the last term becomes:

$$\begin{aligned}&\sum_{(j,k)\in\mathcal{W}_i} (2z_{ijk}-1)\left(\log(p_{jk}) - \log(1-p_{jk})\right) = \\ &\sum_{(j,k)\in\mathcal{W}_i} \text{logit}(p_{jk})(2z_{ijk}-1).\end{aligned}$$

The claim is obtained after redefining $y_i, z_{ijk} \in \{-1, +1\}$, which yields:

$$\text{logit}(\mu_i) = \text{logit}(q_i) + \sum_{(j,k)\in\mathcal{W}_i} \text{logit}(p_{jk})z_{ijk}.$$

□

Equation (2) shows that $\text{logit}(\mu_i)$ is the sum of two independent parts: *i)* the logit of the prior probability q_i and *ii)* a linear combination of the labels provided by the annotator-iteration pairs in $\mathcal{W}_i$, each weighted by the logit of the corresponding accuracy p_{jk}. Also note that, by the definition of logit, we have that $\ell_{jk} = \text{logit}(p_{jk})$ equals 0 when $p_{jk} = 0.5$. This implies that contributions coming from *spammers* (users that provide random labels with an accuracy $p_{jk} = 0.5$) are automatically discarded.

Note that, in this paper, we assume that we are not provided with any knowledge on the prior probability of the event $y_i = 1$ (respectively, $y_i = -1$). Thus, we will henceforth assume that $q_i = 0.5$ for each $i \in \mathcal{T}$, implying $\text{logit}(q_i) = 0$. This assumption suitably represents the very common situation in which a set of tasks to be labeled is submitted for the first time, in the lack of other information.

The expected classification error ϵ_i incurred by a workplan $\mathcal{W}_i$ can be expressed as follows:

$$\mathbb{E}[\epsilon_i] = 1 - \mathbb{E}[\hat{p}_i], \tag{3}$$

where $\mathbb{E}[\hat{p}_i]$ denotes the expected classification accuracy associated with workplan $\mathcal{W}_i$, i.e., the probability that the hard estimation $\hat{y}_i$ of y_i coincides with y_i. The term $\mathbb{E}[\hat{p}_i]$ can be expressed as specified in Proposition 2, below. To this end, let $\theta(\cdot)$ indicate a step function whose value is zero for a non-positive argument and one otherwise, i.e.:

$$\theta(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ 1 & \text{otherwise.} \end{cases} \tag{4}$$

Proposition 2 *The expected classification accuracy $\mathbb{E}[\hat{p}_i]$ associated with workplan $\mathcal{W}_i$ can be expressed as follows:*

$$\mathbb{E}[\hat{p}_i] = \mathbb{E}\left[\theta\Big(\sum_{(j,k)\in\mathcal{W}_i} \ell_{jk} z_{ijk}\Big)\,\Big|\, y_i = 1\right]. \tag{5}$$

Proof The term $\mathbb{E}[\hat{p}_i]$ can be computed as follows:

$$\begin{aligned}\mathbb{E}[\hat{p}_i] = &\Pr(\mu_i \geq 0.5 \mid y_i = 1) \cdot \Pr(y_i = 1) + \\ &\Pr(\mu_i < 0.5 \mid y_i = -1) \cdot \Pr(y_i = -1).\end{aligned}$$

By the definition of logit, we obtain:

$$\Pr(\mu_i \geq 0.5 \mid y_i = 1) = \Pr(\text{logit}(\mu_i) \geq 0 \mid y_i = 1).$$

By replacing $\text{logit}(\mu_i)$ with the right-hand side of Equation (2) and using the fact that $q_i = 0.5$, we obtain the following expression:

$$\Pr(\mu_i \geq 0.5 \mid y_i = 1) = \Pr\left(\sum_{(j,k)\in\mathcal{W}_i} \ell_{jk} z_{ijk} \geq 0 \mid y_i = 1\right).$$

In a similar fashion, we obtain the same derivation for $\Pr(\mu_i < 0.5 \mid y_i = -1)$, i.e.:

$$\Pr(\mu_i < 0.5 \mid y_i = -1) = \Pr(\mu_i \geq 0.5 \mid y_i = 1).$$

Therefore, the expected classification accuracy $\mathbb{E}[\hat{p}_i]$ can be expressed as:

$$\begin{aligned}\mathbb{E}[\hat{p}_i] &= \Pr(\mu_i \geq 0.5 \mid y_i = 1) \cdot \big(\Pr(y = 1) + \Pr(y = -1)\big) \\ &= \Pr(\mu_i \geq 0.5 \mid y_i = 1) \\ &= \Pr\left(\sum_{(j,k)\in\mathcal{W}_i} \ell_{jk} z_{ijk} \geq 0 \mid y_i = 1\right).\end{aligned}$$

We introduce the function:

$$I_A(\{z_{ijk}\}) = \theta\Big(\sum_{(j,k)\in\mathcal{W}_i} \ell_{jk} z_{ijk}\Big),$$

which can be regarded as the indicator function of the subset A of the probability space of the random variables $\{z_{ijk}\}$ for which $\sum_{(j,k)\in\mathcal{W}_i} \ell_{jk} z_{ijk} \geq 0$. By the basic properties of indicator functions, we have:

$$\Pr(A) = \mathbb{E}[I_A],$$

and, therefore,

$$\Pr\left(\sum_{(j,k)\in\mathcal{W}_i} \ell_{jk} z_{ijk} \geq 0 \mid y_i = 1\right) = \mathbb{E}\left[\theta\Big(\sum_{(j,k)\in\mathcal{W}_i} \ell_{jk} z_{ijk}\Big)\,\middle|\, y_i = 1\right],$$

from which Equation (5) follows by transitivity. □

Note that, from the right-hand side of Equation (5), it follows that it is sufficient to focus on the case where $y_i = 1$ in order to compute the expected classification accuracy $\mathbb{E}[\hat{p}_i]$.

In order to derive a closed-form expression of $\mathbb{E}[\hat{p}_i]$, we rewrite Equation (5) as the sum of the probabilities of all the disjoint cases (i.e., possible choices of z_{ijk}). To this end, let $S^+ \subseteq \mathcal{W}_i$ and let $S^- = \mathcal{W}_i \setminus S^+$ denote its complement. We obtain:

$$\mathbb{E}[\hat{p}_i] = \sum_{S^+\subseteq\mathcal{W}_i} \left(\theta\Big(\sum_{(j,k)\in S^+} \ell_{jk} - \sum_{(j,k)\in S^-} \ell_{jk}\Big) \prod_{(j,k)\in S^+} p_{jk} \prod_{(j,k)\in S^-} (1 - p_{jk})\right). \tag{6}$$

Note that, in Equation (6), each $S^+ \subseteq \mathcal{W}_i$ contributes towards $\mathbb{E}[\hat{p}_i]$ by the probability that $z_{ijk} = y_i = 1$ for every $(j,k) \in S^+$ and $z_{ijk} \neq y_i = 1$ for every $(j,k) \in S^-$, provided that $\sum_{(j,k)\in S^+} \ell_{jk} - \sum_{(j,k)\in S^-} \ell_{jk} \geq 0$.

Unfortunately, as can be seen in Equation (6), computing the expected classification accuracy associated with a workplan $\mathcal{W}_i$ (and, thus, the associated

expected classification error as of Equation (3)) requires to explicitly enumerate all the sets of values that can be taken by the random variables $\{z_{ijk}\}$. From a computational viewpoint, the number of elementary operations needed to compute it is exponential in the cardinality of $\mathcal{W}_i$. Hence, an optimization problem where this function is maximized is intractable, and most likely too arduous for any interesting instance with a nontrivial number of tasks and annotators. Therefore, in the following, we turn our attention to an approximation of the expected classification accuracy that can be computed efficiently.

3.2 A tractable approximation of classification accuracy

Our first step in this direction is to consider an approximation of the expected classification accuracy that can be linearized so as to obtain a useful lower bound for optimization purposes. The linearization we use is based on the applicability of Jensen's inequality to an expectation of a convex function. As such, this linearization would not be possible for the expected classification accuracy, which, as indicated in Equation (5), can be expressed as the expectation of a function (the step function $\theta(\cdot)$) that is not convex.

A convex approximation of the argument of the expectation of Equation (5) can be obtained by replacing the step function $\theta(\cdot)$ with the *positive norm* $||\cdot||_+$, whose value is zero for a negative argument, while it is equal to the argument itself for a positive one, i.e.:

$$||x||_+ = \max\{0, x\}. \tag{7}$$

The obtained expression will be referred to as *correctness* function $c(\cdot)$, defined as follows:

$$c(\mathcal{W}_i) = \mathbb{E}\left[\left|\left| \sum_{(j,k)\in\mathcal{W}_i} \ell_{jk} z_{ijk} \right|\right|_+ \mid y_i = 1\right]. \tag{8}$$

Similarly to $\mathbb{E}[\hat{p}_i]$, $c(\mathcal{W}_i)$ can be expressed as follows:

$$c[\mathcal{W}_i] = \sum_{S^+\subseteq\mathcal{W}_i} \left(\left|\left| \sum_{(j,k)\in S^+} \ell_{jk} - \sum_{(j,k)\in S^-} \ell_{jk} \right|\right|_+ \prod_{(j,k)\in S^+} p_{jk} \prod_{(j,k)\in S^-} (1-p_{jk}) \right). \tag{9}$$

Intuitively, $c(\mathcal{W}_i)$ can be seen as a weighted version of $\mathbb{E}[\hat{p}_i]$ where each $S^+ \subseteq \mathcal{W}_i$ such that $\sum_{(j,k)\in S^+} \ell_{jk} - \sum_{(j,k)\in S^-} \ell_{jk} \geq 0$ contributes towards $c(\mathcal{W}_i)$ weighted by a factor $\sum_{(j,k)\in S^+} \ell_{jk} - \sum_{(j,k)\in S^-} \ell_{jk}$ rather than by 1.

As with Equation (5), computing the correctness function for some workplan $\mathcal{W}_i$ still requires considering an exponential number of terms.

In spite of this, $c(\mathcal{W}_i)$ allows us to derive a function that does not suffer from this drawback. Indeed, as a result of applying Jensen's inequality to

Equation (8), we can introduce what we call *linear correctness* function $c'(\cdot)$ (see the proof of Proposition 3 below for the full derivation):

$$c'(\mathcal{W}_i) = \sum_{(j,k)\in\mathcal{W}_i} \ell_{jk}(2p_{jk} - 1). \tag{10}$$

Linear correctness $c'(\cdot)$ is much more tractable than correctness $c(\cdot)$, as, for a given $\mathcal{W}_i$, $c'(\mathcal{W}_i)$ can be computed in linear time. Most importantly, $c'(\mathcal{W}_i)$ is a lower bound on $c(\mathcal{W}_i)$:

Proposition 3 *For every* $i \in \mathcal{T}$, $\quad c(\mathcal{W}_i) \geq c'(\mathcal{W}_i)$.

Proof Given a random variable X and a convex function $\phi : X \to \mathbb{R}$, Jensen's inequality reads:

$$\mathbb{E}[\phi(X)] \geq \phi(\mathbb{E}[X]).$$

For $\phi = ||\cdot||_+$ and $X = \sum_{(j,k)\in\mathcal{W}_i} \ell_{jk} z_{ijk}$, we have:

$$\mathbb{E}\left[\left|\left|\sum_{(j,k)\in\mathcal{W}_i} \ell_{jk} z_{ijk}\right|\right|_+ \mid y_i = 1\right] \geq \left|\left|\mathbb{E}\left[\sum_{(j,k)\in\mathcal{W}_i} \ell_{jk} z_{ijk} \mid y_i = 1\right]\right|\right|_+.$$

Due to the linearity of the expectation, we can rewrite the inequality as:

$$\mathbb{E}\left[\left|\left|\sum_{(j,k)\in\mathcal{W}_i} \ell_{jk} z_{ijk}\right|\right|_+ \mid y_i = 1\right] \geq \left|\left|\sum_{(j,k)\in\mathcal{W}_i} \ell_{jk} \mathbb{E}\left[z_{ijk} \mid y_i = 1\right]\right|\right|_+.$$

We have:

$$\mathbb{E}\left[z_{ijk} \mid y_i = 1\right] = (p_{jk} - (1 - p_{jk})) = 2p_{jk} - 1.$$

Hence:

$$\mathbb{E}\left[\left|\left|\sum_{(j,k)\in\mathcal{W}_i} \ell_{jk} z_{ijk}\right|\right|_+ \mid y_i = 1\right] \geq \left|\left|\sum_{(j,k)\in\mathcal{W}_i} \ell_{jk}(2p_{jk} - 1)\right|\right|_+ = \sum_{(j,k)\in\mathcal{W}_i} \ell_{jk}(2p_{jk} - 1).$$

□

Due to Proposition 3, for any optimization problem maximizing $c'(\mathcal{W}_i)$, we are guaranteed to obtain conservative solutions yielding a value of $c(\mathcal{W}_i)$ at least as large as $c'(\mathcal{W}_i)$, i.e., the value that they maximize.

3.3 Problem statement

We are now ready to formally introduce our problem. In order to produce solutions whose workplans are not too unbalanced in terms of their linear correctness value, we adopt a *bottleneck* approach. Such an approach finds its justification in the fact that solutions that are so unbalanced as to leave some tasks without any annotation in favor of other tasks having multiple annotations are never convenient in terms of expected classification error. This leads to the following problem formulation:

Crowdsourced Labeling Task Assignment Problem (CL-TAP)*: Given a set of tasks $\mathcal{T}$, a set of annotators $\mathcal{A}$, a set of iterations $\mathcal{I}$, a set of accuracies $\{p_{jk}\}_{(j,k)\in\mathcal{A}\times\mathcal{I}}$, and a budget b, find an assignment of the annotators to the tasks that maximizes the minimum linear correctness $c'(\mathcal{W}_i)$ achieved by the different tasks $i \in \mathcal{T}$ so that:*

i) the number of used annotator-iteration pairs is at most b;
ii) if an annotator $j \in \mathcal{A}$ is assigned to a task $i \in \mathcal{T}$ in her iteration $k \in \mathcal{I}$, then she is assigned to $k-1$ distinct tasks in $\mathcal{T} \setminus \{i\}$ for all her iterations $\{1, \ldots, k-1\} \subset \mathcal{I}$;
iii) no annotator labels the same task twice.

4 Formalization and solution

In this section, we first provide a mixed-integer linear programming (MILP) formulation for CL-TAP. Then, we present different solution approaches based on the notion of (linear) correctness, as defined in Equation (10). Throughout the section, we adopt mathematical programming and combinatorial optimization concepts. We refer the reader to [20] for an introductory treatment of the subjects.

4.1 Problem formulation

Let the binary variable x_{ijk} be 1 if annotator $j \in \mathcal{A}$ is assigned to task $i \in \mathcal{T}$ at her iteration $k \in \mathcal{I}$, 0 otherwise. We obtain the following MILP formulation of CL-TAP:

$$\max \quad \eta \tag{11}$$

$$\text{s.t.} \quad \eta \leq \overbrace{\sum_{(j,k)\in\mathcal{A}\times\mathcal{I}} \ell_{jk}(2p_{jk}-1)x_{ijk}}^{c'(\mathcal{W}_i)} \qquad \forall i \in \mathcal{T} \tag{12}$$

$$\sum_{i\in\mathcal{T},j\in\mathcal{A},k\in\mathcal{I}} x_{ijk} \leq b \tag{13}$$

$$\sum_{k\in\mathcal{I}} x_{ijk} \leq 1 \qquad \forall j \in \mathcal{A}, i \in \mathcal{T} \tag{14}$$

$$\sum_{i\in\mathcal{T}} x_{ijk} \leq 1 \qquad \forall j \in \mathcal{A}, k \in \mathcal{I} \tag{15}$$

$$\sum_{i\in\mathcal{T}} x_{ijk} \leq \sum_{i\in\mathcal{T}} x_{ij,k-1} \quad \forall j \in \mathcal{A}, k \in \mathcal{I} \setminus \{1\} \tag{16}$$

$$x_{ijk} \in \{0,1\} \qquad \forall i \in \mathcal{T}, j \in \mathcal{A}, k \in \mathcal{I}. \tag{17}$$

Constraints (12) allow us to cast the problem as a bottleneck problem by imposing that η be a lower bound on the linear correctness of each task, which is then maximized in the Objective Function (11). Constraint (13) guarantees that the budget b is not exceeded. Constraints (14) impose that each annotator be assigned to a task in, at most, one of her iterations. Similarly, Constraints (15) establish that each annotator be assigned to at most a task in each of her iterations. Finally, Constraints (16) guarantee that, if an annotator $j \in \mathcal{A}$ is used in her iteration k for some task, then she must be used for some other task in her iteration $k-1$.

A *task assignment* is a sequence $\langle \mathcal{W}_1, \ldots, \mathcal{W}_t \rangle$ of workplans, one for each task $i \in \mathcal{T}$. The binary variables x_{ijk} collectively correspond to a task assignment, where each workplan $\mathcal{W}_i$ amounts to $\{(j,k) \in \mathcal{A} \times \mathcal{I} : x_{ijk} = 1\}$. The task assignment is said to be *feasible* if the x_{ijk}s satisfy Constraints (13)–(17), and *optimal* if the x_{ijk}s are a solution to Problem (11)–(17).

As we will see in Section 6, solving Problem (11)–(17) to optimality with state-of-the-art MILP solvers is very challenging even for medium-size instances. This motivates us to design some faster, albeit not exact, heuristic algorithms to tackle CL-TAP, described next.

4.2 A correctness-based policy

We now present an effective (linear) correctness-based policy (`CBP`) for solving CL-TAP. Such a policy, described in Algorithm 1, addresses the case of decreasing as well as increasing annotators' accuracies, and could, in fact, be used even in the case where the accuracies are non-monotonic.

In the main loop, `CBP` looks for the annotator j^* that reaches the highest *average* accuracy, and indicates as h^* the horizon of available iterations in

which this happens. In order to comply with the available budget, h^* may not exceed b. The search for j^* and h^* can be efficiently performed as follows: for each annotator, we consider the set of annotator-iteration pairs starting from her current iteration until the furthest iteration such that the accuracies are non-decreasing (indeed: adding a lower accuracy would decrease the average). Then, h^* equals the size of the set with the highest average accuracy, and j^* is the corresponding annotator. We then ensure that annotator j^* is used in all the iterations up to the h^*-th by assigning, for $1 \leq h \leq h^*$, the annotator-iteration pair (j^*, h) to the workplan with the h-th highest linear correctness value (among the h^* most uncertain workplans that did not receive an annotation from j^*), and decreasing the available budget correspondingly.

We observe that, in the case of decreasing accuracies, `CBP` always sets $h^* = 1$ and, thus, boils down to assigning, each time, the single most promising available annotator-iteration pair to the most needy task (i.e., the one with the lowest linear correctness). Conversely, in the case of increasing accuracies, `CBP` always sets $h^* = t$ and thus, in each **repeat** cycle, it assigns all the t annotator-iteration pairs available from the best annotator (j^*) to all the t tasks, by taking care of providing the most needy tasks with the annotator-iteration pairs with the highest accuracies. For the latter scenario, we note that, basing the choice of annotator j^* on the average of her accuracies provides us with a much more robust performance than using a single accuracy value (e.g., the initial accuracy value).

`CBP` has a complexity of $O(b(at + t \log t))$. Indeed, the main loop is executed up to b times. Each time, up to $a \cdot t$ (a annotators along t iterations) available accuracies are explored to select the best annotator, which is assigned to the h^* most uncertain tasks (with $h^* \leq t$), the update of which requires $\log t$ time per task.

Algorithm 1: Correctness-based policy (`CBP`).

Input: Annotator-iteration set ($\mathcal{A} \times \mathcal{I}$), task set $\mathcal{T}$, budget b.
Output: A feasible task assignment $\langle \mathcal{W}_1, \ldots, \mathcal{W}_t \rangle$.

$\langle \mathcal{W}_1, \ldots, \mathcal{W}_t \rangle \leftarrow \langle \emptyset, \ldots, \emptyset \rangle$
repeat
  $\langle j^*, h^* \rangle \leftarrow \langle j, h' \rangle : avg_{1 \leq h \leq h'}(p_{jh})$ is maximized for $1 \leq h' \leq \min\{t, b\}$ and $j \in \mathcal{A}$
  $\mathcal{K} \leftarrow h^*$ indexes of workplans in $1, ..., t$ not annotated by j^* with smallest $c'(\cdot)$
  if $avg_{1 \leq h \leq h^*}(p_{jh}) = \frac{1}{2}$ `break`
  foreach h *in* $1 \ldots h^*$ **do**
    $i^* \leftarrow$ index of workplan with the largest $c'(\cdot)$ in $\mathcal{K}$
    $\mathcal{W}_{i^*} \leftarrow \mathcal{W}_{i^*} \cup \{(j^*, k)\}$
    $\mathcal{K} \leftarrow \mathcal{K} \setminus \{i^*\}$
  end
  $b \leftarrow b - h^*$; $\mathcal{A} \leftarrow \mathcal{A} \setminus \{j^*\}$
until $b = 0$

Example 1 Let $\mathcal{A} = \{1, 2\}$, $\mathcal{T} = \{1, 2, 3\}$, and $b = 5$. If we represent each workplan as a binary matrix $\mathcal{W}$ of size $a \times t$ (so that $\mathcal{W}_{jk} = 1$ if the annotator-

iteration (j, k) is present in the workplan, 0 otherwise) the set of feasible workplans, neglecting the empty one, is as follows:

$$\begin{bmatrix}1&0&0\\0&0&0\end{bmatrix}, \begin{bmatrix}0&1&0\\0&0&0\end{bmatrix}, \begin{bmatrix}0&0&1\\0&0&0\end{bmatrix}, \begin{bmatrix}0&0&0\\1&0&0\end{bmatrix}, \begin{bmatrix}0&0&0\\0&1&0\end{bmatrix}, \begin{bmatrix}0&0&0\\0&0&1\end{bmatrix}, \begin{bmatrix}1&0&0\\1&0&0\end{bmatrix},$$
$$\begin{bmatrix}0&1&0\\1&0&0\end{bmatrix}, \begin{bmatrix}0&0&1\\1&0&0\end{bmatrix}, \begin{bmatrix}1&0&0\\0&1&0\end{bmatrix}, \begin{bmatrix}0&1&0\\0&1&0\end{bmatrix}, \begin{bmatrix}0&0&1\\0&1&0\end{bmatrix}, \begin{bmatrix}1&0&0\\0&0&1\end{bmatrix}, \begin{bmatrix}0&1&0\\0&0&1\end{bmatrix}.$$

Note that $\left[\begin{smallmatrix}0&0&1\\0&0&1\end{smallmatrix}\right]$ is not feasible for any task, as a budget $b \geq 6$ would be needed to have both annotators at iteration 3.

Let the accuracies p_{jk} be increasing over the iterations:

$$p_{11} = 0.68,\ p_{12} = 0.75,\ p_{13} = 0.8;$$
$$p_{21} = 0.55,\ p_{22} = 0.7,\ \ p_{23} = 0.9.$$

`CBP` starts from a task assignment only consisting of empty workplans:

$$\langle \mathcal{W}_1, \mathcal{W}_2, \mathcal{W}_3 \rangle = \langle \left[\begin{smallmatrix}0&0&0\\0&0&0\end{smallmatrix}\right], \left[\begin{smallmatrix}0&0&0\\0&0&0\end{smallmatrix}\right], \left[\begin{smallmatrix}0&0&0\\0&0&0\end{smallmatrix}\right] \rangle.$$

Here, annotator 1 has the highest average accuracy in her first $\min\{t, b\} = 3$ iterations and is, thus, the most promising. We have enough budget to let annotator 1 work on every task. Since annotator 1 will improve while working, we reserve her best annotation for the least certain workplan. At this point, all workplans have equal linear correctness (Equation 10), so we break ties lexicographically. As a result, workplan $\mathcal{W}_1$ is assigned annotator-iteration $(1, 3)$, whereas $\mathcal{W}_2$ (the second-most uncertain) and $\mathcal{W}_3$ (the third-most uncertain) will be, resp., assigned annotator-iteration $(1, 2)$ and $(1, 1)$. The current solution reads: $\langle \left[\begin{smallmatrix}0&0&1\\0&0&0\end{smallmatrix}\right], \left[\begin{smallmatrix}0&1&0\\0&0&0\end{smallmatrix}\right], \left[\begin{smallmatrix}1&0&0\\0&0&0\end{smallmatrix}\right] \rangle$. Annotator 2 is now the only possible choice. With a remaining budget of 2, the algorithm assigns annotator-iteration $(2, 2)$ to workplan $\mathcal{W}_3$ (the most uncertain) and annotator-iteration $(2, 1)$ to workplan $\mathcal{W}_2$ (the second-most uncertain). The final solution is: $\langle \left[\begin{smallmatrix}0&0&1\\0&0&0\end{smallmatrix}\right], \left[\begin{smallmatrix}0&1&0\\1&0&0\end{smallmatrix}\right], \left[\begin{smallmatrix}1&0&0\\0&1&0\end{smallmatrix}\right] \rangle$, whose objective value is 0.57, obtained as $\min\{c'(\mathcal{W}_1), c'(\mathcal{W}_2), c'(\mathcal{W}_3)\}$. Although not optimal in general, for this small instance `CBP` identifies an optimal task assignment, since 0.57 is the value found by solving Problem (11)–(17) to optimality.

Example 2 Consider now decreasing accuracies. Let $\mathcal{A} = \{1, 2, 3, 4\}$, $\mathcal{T} = \{1, 2, 3\}$, $b = 5$, and p_{jk} be as follows:

$$p_{11} = 0.90,\ p_{12} = 0.70,\ p_{13} = 0.55;$$
$$p_{21} = 0.80,\ p_{22} = 0.75,\ p_{23} = 0.68;$$
$$p_{31} = 0.79,\ p_{32} = 0.76,\ p_{33} = 0.69;$$
$$p_{41} = 0.89,\ p_{42} = 0.78,\ p_{43} = 0.67.$$

`CBP` starts with an initial task assignment solely composed of empty workplans. The annotator-iteration to be selected next should maximize linear correctness (Equation (10)), and is obtained by identifying the annotator with the highest accuracy at her first iteration, since, with decreasing accuracies, `CBP` will always set the horizon of iterations to consider to $h^* = 1$. As such, the algorithm selects the annotator-iteration $(1, 1)$, which is then assigned to

the least certain workplan. Since all workplans have, initially, the same linear correctness, workplan $\mathcal{W}_1$ is selected by lexicographic order. The current solution thus becomes:

$$\langle \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \rangle.$$

Next, annotator-iteration $(4, 1)$ is selected. Again, ties are broken lexicographically, leading to the selection of $\mathcal{W}_2$. `CBP` proceeds in a similar fashion and assigns $(2, 1)$ to $\mathcal{W}_3$. The current solution then becomes:

$$\langle \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \rangle$$

At this point, the least certain workplan is $\mathcal{W}_3$. Continuing this way, `CBP` will eventually produce the following final solution:

$$\langle \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \rangle$$

Its objective value is 1.63, which, in this case, is the optimal value.

4.3 Improving upon CBP solutions

In the attempt of improving upon suboptimal solutions, we have implemented a *local search* meta-heuristics in order to further explore the set of feasible task assignments. The idea is to iteratively move from an initial solution to a "neighboring" task assignment by applying local changes, until either a timeout expires or a target gain in terms of linear correctness is met. The possible moves, applied only as long as they preserve feasibility (in the sense introduced in Section 4.1), are *i) internal swaps* (swapping two annotations between two different workplans); *ii) external swaps* (swapping an annotation in a workplan with one not used in the task assignment and adjusting iterations as needed); *iii) deletions or additions* (if budget allows) of an annotation. Let Δ indicate the linear correctness gain associated with a move: the move with the highest Δ is chosen; however, if several consecutive moves with negative Δ are applied, the external swap with the highest Δ is forced to take place, as an attempt to escape from local maxima.

5 Certifying solutions

As shown in Section 6, solving MILP (11)–(17) to optimality is very difficult for any instance of interesting size. In spite of this, one could think of using the MILP formulation to construct upper bounds on CL-TAP, to be used to certify the quality of the solutions found via the correctness-based policy. Indeed, it suffices to relax the Integrality Constraints (17) on the variables to obtain a Linear Program (LP). Optimal solutions to the LP, although infeasible, clearly

achieve an objective function value UB which is at least as good as that of any optimal solution to Problem (11)–(17) and, therefore, to CL-TAP. This way, given an upper bound (UB) to CL-TAP, any feasible solution to the problem (yielding a lower bound LB) is guaranteed to be at least $\frac{LB}{UB} \leq 1$ times the value of an optimal solution. In other words, the solution of value LB is a ρ-approximate solution to the problem, with $\rho = \frac{LB}{UB}$, which gives us a guarantee on how close we are to an optimal solution[1].

Since, for large instances, even the LP relaxation of Problem (11)–(17) can be very time consuming to solve, we now develop a faster method to calculate the aforementioned LP upper bound.

5.1 An alternative formulation

We now introduce an alternative formulation which, as we will show in the following, can be used to construct an alternative upper bound to CL-TAP.

We call a workplan *feasible* if it is an element of some feasible task assignment. Let us assume that all the feasible workplans have been precomputed and let $\mathcal{H}$ be the corresponding index set. We denote by $\mathcal{W}_h$ the workplan of index h, for every $h \in \mathcal{H}$. Note the difference w.r.t. the notation $\mathcal{W}_i$, by which we denote a workplan associated with a task $i \in \mathcal{T}$. Let c_h be the linear correctness of workplan $\mathcal{W}_h$. We assume, for now, that it has been precomputed along with $\mathcal{W}_h$. Let the binary parameter w_{hjk} be equal to 1 if workplan $\mathcal{W}_h$ uses annotator $j \in \mathcal{A}$ in her iteration $k \in \mathcal{I}$ and let $w_{hjk} = 0$ otherwise. We introduce the binary variable λ_{ih}, denoting whether workplan $\mathcal{W}_h$ is associated with task $i \in \mathcal{T}$ or not. The following is an alternative formulation of CL-TAP:

$$\max \eta \tag{18}$$

$$\text{s.t. } \eta \leq \sum_{h \in \mathcal{H}} c_h \lambda_{ih} \qquad \forall i \in \mathcal{T} \tag{19}$$

$$\sum_{i \in \mathcal{T}} \sum_{h \in \mathcal{H}} \left(\sum_{j \in \mathcal{A}} \sum_{k \in \mathcal{I}} w_{hjk} \right) \lambda_{ih} \leq b \tag{20}$$

$$\sum_{i \in \mathcal{T}} \sum_{h \in \mathcal{H}} w_{hjk} \lambda_{ih} \leq 1 \qquad \forall j \in \mathcal{A}, k \in \mathcal{I} \tag{21}$$

$$\sum_{h \in \mathcal{H}} \lambda_{ih} \leq 1 \qquad \forall i \in \mathcal{T} \tag{22}$$

$$\sum_{\substack{i \in \mathcal{T} \\ h \in \mathcal{H}}} w_{hjk} \lambda_{ih} \leq \sum_{\substack{i \in \mathcal{T} \\ h \in \mathcal{H}}} w_{hj,k-1} \lambda_{ih} \quad \forall j \in \mathcal{A}, k \in \mathcal{I} \backslash \{1\} \tag{23}$$

$$\lambda_{ih} \in \{0, 1\} \qquad \forall i \in \mathcal{T}, h \in \mathcal{H}. \tag{24}$$

[1] To show this, let OPT be the value of an optimal solution. Let $\rho = \frac{LB}{UB}$. Due to $\frac{LB}{UB} \leq \frac{LB}{OPT}$, we have $LB \geq \rho\, OPT$. Since $LB \leq OPT$, we conclude $\rho OPT \leq LB \leq OPT$, i.e., that the solution of value LB is a ρ-approximate solution to the problem, with $\rho = \frac{LB}{UB}$.

Constraints (19), (20), (21), and (23) correspond, resp., to the original Constraints (12), (13), (15), and (16). Constraint (14) is automatically satisfied when assuming that only *feasible* workplans are considered, while the new Constraint (22) imposes that a workplan be assigned to, at most, a task. Note that, in Constraint (20), the quantity $\sum_{j\in\mathcal{A}}\sum_{k\in\mathcal{I}} w_{hjk}$ corresponds to the number of annotators used in workplan $\mathcal{W}_h$.

5.2 Column generation algorithm

In principle, it is clear that the MILP (18)-(24) cannot be solved directly, as it requires to precompute all the (exponentially many) feasible workplans. Yet, its LP relaxation can be solved quite efficiently with the algorithm that we now describe. The general idea is to solve the LP relaxation of Problem (18)-(24) on an increasing subset $\mathcal{H}'$ of $\mathcal{H}$ while iteratively inspecting the current solution so as to identify, among the workplans not yet considered, a convenient one to add.

This very idea is at the core of a broad family of methods called *column generation* algorithms, which are designed for the solution of large linear programming problems with, usually, exponentially many variables (or *columns*). Given a linear program, those methods solve, alternately, two subproblems called *restricted master problem* (RMP) and *pricing subproblem* (PSP). The former corresponds to a restriction of the original linear program to the subset of variables considered so far. The latter is a new problem, which is solved to identify new promising variables to be added to RMP. For an introduction to this type of methods, see [26].

For the task assignment problem, we relax integrality in Constraints (24) in Problem (18)-(24) by imposing $\lambda_{ih} \in [0,1]$ (this is, intuitively, equivalent to allowing for fractional assignments of workplans to tasks). For all $i \in \mathcal{T}, j \in \mathcal{A}, k \in \mathcal{I}$, let $\alpha_i, \beta, \gamma_{jk}, \delta_i, \epsilon_{jk} \geq 0$ denote the (linear programming) dual variables of, resp., Constraints (19), (20), (21), (22), and (23), corresponding to an optimal solution to RMP. For convenience, let $\epsilon_{j,|\mathcal{I}|+1} = 0$ for all $j \in \mathcal{A}$. The next workplan to be considered (the index of which we omit for readability) is generated by solving the following PSP t times, one per task $i \in \mathcal{T}$, each time with the appropriate value of α_i (see Appendix A for the derivation of the subproblem):

$$\max \alpha_i \overbrace{\sum_{\substack{j\in\mathcal{A}\\k\in\mathcal{I}}} \ell_{jk}(2p_{jk}-1)w_{jk}}^{c'(\mathcal{W}_i)} - \sum_{\substack{j\in\mathcal{A}\\k\in\mathcal{I}}} (\beta+\gamma_{jk}+\epsilon_{jk}-\epsilon_{j,k+1})w_{jk} \tag{25}$$

$$\text{s.t.} \sum_{j\in\mathcal{A}}\sum_{k\in\mathcal{I}} w_{jk} \leq b \tag{26}$$

$$\sum_{k\in\mathcal{I}} w_{jk} \leq 1 \qquad \forall j \in \mathcal{A} \tag{27}$$

$$w_{jk} \in \{0,1\} \qquad \forall j \in \mathcal{A}, k \in \mathcal{I}. \tag{28}$$

Solving Problem (25)–(28) amounts to deciding which w_{jk} should be set to 1, i.e., deciding which annotator-iteration pairs $(j, k) \in (\mathcal{A} \times \mathcal{I})$ to include in the new workplan, based on the current (fractional) solution to Problem (18)-(24) (as conveyed by the dual variables $\{\alpha_i\}$, β, $\{\gamma_{jk}\}$, $\{\delta_i\}$, $\{\epsilon_{jk}\}$). As we show in Appendix A, improving workplans can only be found among those with an objective function value of, at least, δ_i. All the workplans violating this condition are thus discarded. We remark that PSP can be solved in polynomial time. This is because, since its constraints form a totally unimodular matrix (refer to [20] for details on total unimodularity) and its objective function is linear, we can solve it as a linear programming problem, a task which can be carried out in polynomial time.

The pseudo-code of the column generation (`colgen`) algorithm is reported in Algorithm 2.

Algorithm 2: Column generation algorithm (`colgen`).

```
Input: Annotator-iteration set (A × I), task set T, budget b.
Output: a valid UB to the optimal value of linear correctness
H' ← heuristic_procedure((A × I), T, b)
repeat
    Δ ← 0
    UB, {α_i}, β, {γ_jk}, {δ_i}, {ε_jk} ← solve_rmp(H')
    foreach i ∈ T do
        ⟨W_h, Λ⟩ ← solve_pricing(α_i, β, {γ_jk}, {ε_jk})
        if Λ > δ_i and Δ < (Λ − δ_i) then
            Δ ← Λ − δ_i
            H^l ← heuristic_search(W_h, (A × I), T, b)
            H' ← H' ∪ {h} ∪ H^l
        end
    end
until Δ = 0
return UB
```

First, we find an initial set of indexes of feasible workplans $\mathcal{H}'$ by invoking a heuristic procedure `heuristic_procedure` (a good choice are the workplans of the task assignment found by `CBP`). Then, column generation is applied until no more improving workplans are found. In `colgen`, we substitute the tighter inequality $\sum_{j \in \mathcal{A}} \sum_{k \in \mathcal{I}} k w_{jk} \leq b$ for Constraint (26), which prevents the generation of many useless workplans. The tighter constraint reflects the fact that, if an annotator is selected in her k-th iteration, $k-1$ extra units of budget must be reserved to account for her previous $k-1$ iterations. In spite of forcing us to solve PSP as an MILP (rather than as an LP), overall this leads to a substantially faster method.

We introduce a few other enhancements. First, in order to focus on the most promising workplans, we take a new workplan into consideration only if the corresponding objective function value Λ is at least δ_i and the best one seen so far in the current iteration. At each iteration, we use the new workplan $\mathcal{W}_h$ to bootstrap a heuristic search (`heuristic_search`) for a feasible task assignment. Let $\{h\} \cup \mathcal{H}^l$ be the indexes of its workplans. Then, the indexes $\mathcal{H}^l$

are added to $\mathcal{H}'$. This way, we try to anticipate the construction of promising workplans, so as to achieve faster convergence of the method.

6 Experimental evaluation

In this section, we present the results for the two classes of experimental evaluations we conducted. Namely, we first provide empirical evidence of the existence of a workload-dependent accuracy variation among workers, whether coming from a volunteer or a professional crowd. Then, we evaluate the performance of the proposed task assignment policy on different (synthetic and real) data sets, showing the need of taking such accuracy variations into account.

6.1 Estimating the accuracy model

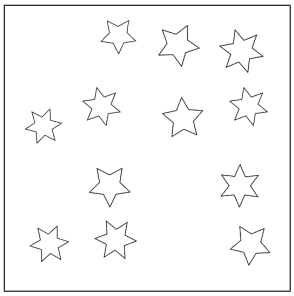
We conducted three experiments involving real crowds with the aim of verifying that the accuracy level of each annotator changes as the number of assignments increases.

6.1.1 Methodology

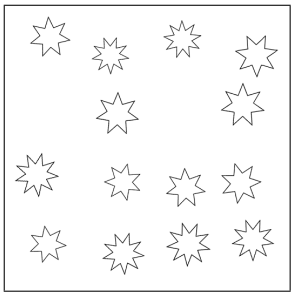
Experiments. The first experiment was conducted on a volunteer crowd of 80 participants, who were required to solve a sequence of CAPTCHAs within a time frame of 30 minutes. Each participant was allowed to leave the session at any time. The contributions were submitted via an in-house web application specifically implemented for the experiment, while CAPTCHAs were provided by the reCAPTCHA service [4]. The contributions for the second experiment were collected via a GWAP for image segmentation [13]. Specifically, each one of the 159 players who took part in the experiment was presented with a sequence of images, where each image depicted one or more individuals wearing fashionable clothes. Participants were asked to draw the contour of a specific garment in the image. Such a contour was then automatically filled so as to obtain a binary mask of the image, effectively segmenting the garment from the rest of the image, as reported in Figure 2a. For each image in the dataset, we compared the the obtained contour with the actual one (available as a "gold standard"), and then computed the accuracy of the worker accordingly. As with the first experiment, players were free to leave at any time. The third experiment was conducted on the MicroTask platform [3], with 30 workers involved. In this case, the participants were asked to solve a sequence of 1000 visual perception tasks within a time frame of 5 hours. Differently from the two previous experiments, in order to reflect more accurately the working conditions of a paid crowd, we mandated participants to complete the entire set of tasks. Given two natural numbers X and Y, for each task, workers had to inspect a picture similar to those reported in Figures 2b and 2c and answer the binary question *"Are the X-pointed stars more numerous than the Y-pointed*

ones?". The sequence of images was obtained by alternating between tasks with $(X = 5, Y = 6)$, as in Figure 2b, and tasks with $(X = 7, Y = 9)$, as in Figure 2c. Finally, in order to make the images more challenging, each star was randomly rotated and scaled down.

(a) Fashion-related image presented to the user and binary mask resulting from user's segmentation.



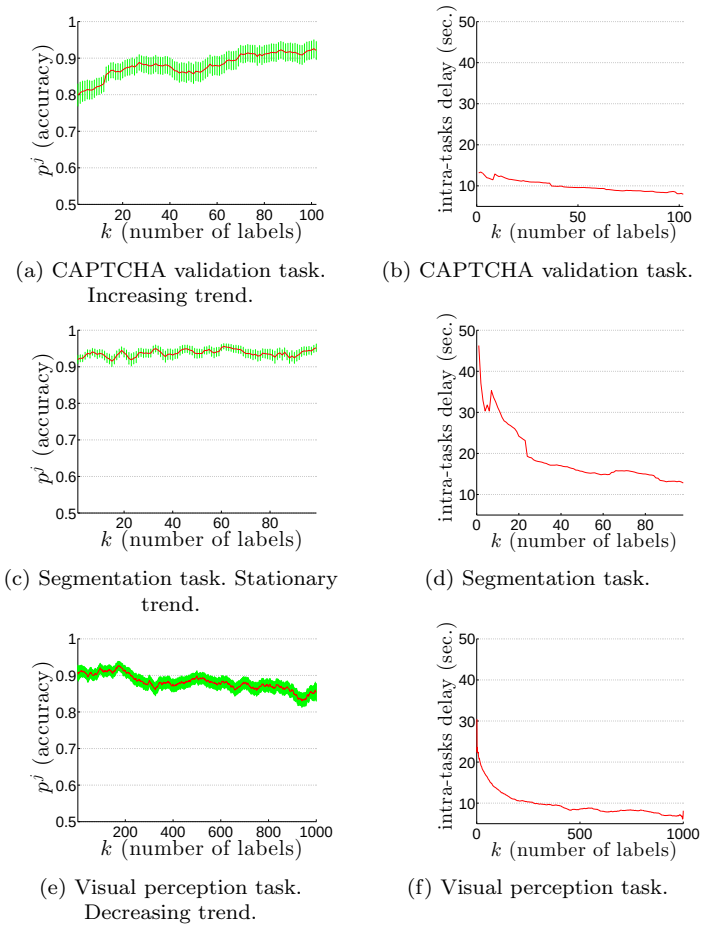
(b) Stars with $X = 5, Y = 6$.



(c) Stars with $X = 7, Y = 9$.

Fig. 2: Segmentation task (a): the participants were asked to draw the contour of a given garment in the image. Visual perception tasks (b)-(c): the participants were asked to answer the binary question *"Are the X-pointed stars more numerous than the Y-pointed ones?"*.

Incentive model. Since, in the first and the second experiments, the contributions were made on a voluntary basis, no incentive model was adopted. Conversely, in the third experiment we adopted a pay-per-performance incentive model. Namely, we fixed 5 USD as a fair payment in return for 950 correct answers. Each worker was then paid proportionally w.r.t. the number of correct answers. If such a number exceeded 950, the payment function was adapted so as to set a wage of 20 USD in return for 1000 correct answers, thus dramatically increasing the payment with the number of correct answers provided. Moreover, as we required workers to annotate the whole data set within a maximum of 5 hours, we set an additional multiplier on the total payment for the first 5 fastest workers to complete the session among those



(a) CAPTCHA validation task. Increasing trend.

(b) CAPTCHA validation task.

(c) Segmentation task. Stationary trend.

(d) Segmentation task.

(e) Visual perception task. Decreasing trend.

(f) Visual perception task.

Fig. 3: Average accuracy variation and task duration.

providing at least 950 correct answers. Specifically, the multiplier was computed as $1.5 - 0.1(R-1)$, where $R = 1, \ldots, 5$ is the worker's rank. We opted for such an incentive model in order to discourage as much as possible malicious behaviors, since a pilot experiment revealed that a high percentage of the total workforce provided labels at random if hired with a flat pay-per-hour incentive model.

Methods and evaluation metrics. We characterize the behavior of both the volunteer and the professional crowd by means of two aspects: *i)* the average accuracy variation w.r.t. the number of already provided labels; *ii)* the delay between two consecutive answers. Since participants in the first two experiments were free to leave at any time, the last tasks in the sequence may receive a low number of labels. In order not to bias the results, the histogram of the contributions provided by the participants was constructed so as to

suppress any negligible tail in the distribution. For instance, even though the single best participant in the CAPTCHA validation experiment provided 378 correct answers, by analyzing the overall distribution, we decided to consider only the first 100 answers. Moreover, in order to better highlight the emerging trends, we smoothed the data by means of a moving average filter.

6.1.2 Results

Average accuracy variation. Figures 3a, 3c, and 3e show the aggregated accuracy variation as a function of the number of provided labels for the CAPTCHA validation experiment, the segmentation experiment, and the visual perception experiment, respectively. As shown, the trend of the accuracy function strictly depends on the specific task to be executed. In particular, an improvement in the accuracies of the annotators emerged in the CAPTCHA validation experiment (Figure 3a). Conversely, annotators exhibited decreasing accuracies, likely due to fatigue, during the visual perception experiment (Figure 3e). Finally, a stationary accuracy trend emerged from the segmentation experiment (Figure 3c). This might be explained if we consider that humans can easily recognize visual objects in images, even if they are partially visible or appear in unusual lighting conditions or view points. Furthermore, possible fatigue phenomena could have been counterbalanced by the engagement factor induced by the game-based interaction of the GWAP. Indeed, the average accuracy value exceeded 0.9 for the entire duration of the experiment.

Task duration. Figures 3b, 3d, and 3f report the average task duration observed during the three different experiments. Although all experiments showed a decreasing task duration, the reported accuracy trends suggest that this could be due to different reasons. Namely, the decreasing trend in Figure 3b could be a consequence of the learning phenomenon reported in Figure 3a, i.e., learning impacted positively not only on accuracy, but also on task completion speed. The same rationale applies for the segmentation experiment (Figure 3d), in which participants maintained the initial high level of accuracy, while reducing progressively the task completion time as they learned how to play. Conversely, the decreasing trend in the visual perception experiment (Figure 3f) coupled to a decrease in accuracy could be considered as a different manifestation of the same fatigue phenomenon emerged in Figure 3e, i.e., as participants got more fatigued, they decided to invest less time to complete each single task, thus worsening the accuracy.

6.2 Evaluating the task assignment policies

Once the task assignment problem is solved and tasks are accordingly assigned to workers and completed by them, each task $i \in \mathcal{T}$ gets associated with a set of noisy labels $\{z_{ijk}\}_{(j,k)\in\mathcal{W}_i}$. At that point, a label aggregation technique can be used in order to obtain an estimation $\hat{y}_i$ of the true label y_i. Namely, for each task $i \in \mathcal{T}$, we first compute its posterior probability μ_i starting

Full name	Param.	Tested value
Size of the task set	t	3, 4, 5, 100, 200, **1K**, 10K, 100K
Size of the annotator set	a	3, 10, 20, 50, **100**, 150, 200
Budget	b	$1.5t$, **2t**, $2.5t$, $3t$
Accuracy variation degree	τ	0.25, 0.5, 1, 10, 100, **mixed population**
Type of crowd	-	**NAIVE**, EXPERT

Table 2: Operating parameters for the task assignment policy evaluation (defaults in bold).

from Equation (2). As anticipated in Section 3, the estimated label $\hat{y}_i$ is then computed by applying a threshold $\omega = 0.5$ to the posterior probability μ_i, thus letting $\hat{y}_i = 1$ if $\mu_i \geq 0.5$ and $\hat{y}_i = -1$ otherwise. Hence, we evaluate the performance of a task assignment policy in terms of the final *classification error* (i.e., the percentage of misclassified tasks):

$$\epsilon = \frac{1}{t} \sum_{1 \leq i \leq t}{}_i \epsilon_i = \frac{1}{t} \sum_{\substack{1 \leq i \leq t \\ \hat{y}_i \neq y_i}}{}_i 1. \tag{29}$$

In order to evaluate the effectiveness of the proposed task assignment policy, we investigate how close the obtained linear correctness is to the optimal value and how the classification error is affected by: *i)* the budget availability; *ii)* different degrees of accuracy variation (either decreasing or increasing); *iii)* the number of tasks; *iv)* the number of available annotators; and *v)* the crowd expertise. The relevant parameters are shown in Table 2, with defaults in bold.

6.2.1 Methodology

Data sets. The first data set family consists of synthetically generated tasks and annotators. For each annotator $j \in \mathcal{A}$, her initial accuracy p_{j1} is randomly sampled from a Beta distribution. In particular, for a "naive" crowd, p_{j1} is sampled from a Beta with mean 0.7 (resp., 0.6) for decreasing (resp., increasing) accuracies; the standard deviation is 0.14 – a high value, since the crowd is heterogeneous. Conversely, for an "expert" crowd, p_{j1} is sampled from a Beta with mean 0.9 and standard deviation 0.03, since all annotators are consistently good, both for decreasing and increasing accuracies. Accuracy was modeled as a power law:

$$p_{jk} = \sigma k^{\tau} + \zeta, \tag{30}$$

where τ can be interpreted as an endurance factor (resp., improvement rate), and the parameters σ and ζ are appropriately defined so that the obtained curves cover several different trends, including those of the curves shown in Figure 3. Namely, a random value $\bar{t}$ where accuracy "saturates" (i.e., $p_{j\bar{t}} = 0.5$ for decreasing accuracies, and $p_{j\bar{t}} = 1$ for increasing accuracies) is sampled from a Beta distribution in the interval[2] $[1, 2t]$. The remaining parameters σ

[2] For decreasing accuracies, mean $= 0.15 \cdot 2t$; for increasing accuracies, mean $= 0.85 \cdot 2t$. Standard deviation $= 0.14$.

and ζ are then computed by solving Equation (30) in two cases: *i)* when $k = 1$, and thus $p_{jk} = p_{j1}$, and *ii)* when $k = \bar{t}$, and thus $p_{jk} = p_{j\bar{t}}$, i.e., for the initial and the final accuracy values of annotator j. In order to model annotators with different accuracy trends, the default population is "mixed", and consists of 5 equally sized groups – one per tested value of τ (see Table 2). Task labels y_i are equally balanced between the two possible binary labels.

The second data set family includes three real data sets, comprising tasks and annotators coming from the CAPTCHA validation, the image segmentation, and the visual perception experiment.

Methods and evaluation metrics. We compared the performance of the algorithms of Section 4 (indicated as `CBP` and `local_search`) with the state-of-art CPLEX tool, denoted `optimal`, which solves Problem (11)-(17) to optimality, and four baseline policies: *i)* a policy named `random` (`RND`), which randomly assigns annotators to tasks until budget exhaustion, so that each task gets at least an annotation; *ii)* the task assignment policy proposed in [17], in which annotators are randomly assigned to tasks so that the resulting solution can be represented as a regular bipartite graph, here referred to as `regular` (`RGL`); *iii)* the offline task assignment policy proposed in [14], indicated as `primal_approximation` (`PA`); and *iv)* an offline task assignment policy inspired by the top annotator selection criterion used in iCrowd [12], indicated as `top_annotators` (`TA`).

Note that `PA` has the objective of *minimizing* the budget, while guaranteeing a maximum value $\psi \in [0, 1)$ for a proxy of the classification error. Such a value ψ thus needs to be provided as an input parameter to `PA`, while our policy uses the budget b as input. Fortunately, for every value of ψ a corresponding value of b can be found. Therefore, in the experiments, we first map the value b under test into the smallest corresponding value of ψ, and then execute `PA`. Moreover, since `PA` is agnostic with respect to the presence of different iterations for the same annotator, and thus gives no indication about the order in which an annotator labels the different tasks, we adopted, for `PA`, a random order. For the same reason, `PA` only "sees" one accuracy value per annotator; to improve `PA`'s performance, we provided `PA` with the average accuracy of the annotator during her available iterations. Other choices, such as providing `PA` with the initial or the final accuracy value, would entail a significant reduction in the result quality.

We also point out that `TA` is a particularly fair offline counterpart of iCrowd. First of all, iCrowd spends a relevant part of the budget on a step whose sole purpose is to estimate the accuracies of the annotators, and in particular to assess their ability on kinds of tasks they have not dealt with before. In our context, instead, we deal with tasks that are all of the same kind, and accuracies are already available from the start. Therefore, our baseline `TA`, unlike iCrowd, completely avoids the above step. In addition, in iCrowd each task receives the same amount K of annotations (where K is a fixed odd number). Such a constraint leads to extremely poor quality whenever the available budget b largely differs from $K \cdot t$. Therefore, to alleviate this problem, `TA` chooses each time the most suitable value for K depending on the available budget b.

For all these algorithms we report both linear correctness $c'(\cdot)$ (Table 3) and classification error ϵ (Table 3 and Figures 4–11). Since the time needed to aggregate the annotators' labels is negligible w.r.t. the time needed to find a task assignment, only the latter will be shown when we report execution times (Figure 6). All metrics are averaged over 10 different instances. When evaluating the quality of the `random`, `PA`, and `regular` baselines, metrics are additionally averaged over 10 rounds so as to properly take into account the stochastic component of the methods. All the experiments have been conducted on a computer with an Intel® Core i7 processor operating at 3.4GHz and 16GB of RAM.

6.2.2 Results

Certification of the solution quality. We initially compared the methods for a small problem size ($t = 3, 4, 5$), so as to be able to compute the results of the `optimal` algorithm. Table 3 reports a comparison between `CBP`, `local_search`, and `optimal` in terms of both linear correctness and classification error, as t increases. Tables 3b and 3d also report in parentheses the ratio $\frac{LB}{OPT}$ between the solution found by the algorithm (LB) and the optimal linear correctness value (OPT), when available ($t \leq 5$). When OPT is not available ($t > 5$), we report the ratio $\frac{LB}{UB}$, where UB is the upper bound provided by our column generation method (see Section 5.2), showing that the solution is *at least* $\frac{LB}{UB} \cdot OPT$. As shown in Table 3a–3b (decreasing accuracies) and Table 3c–3d (increasing accuracies), `CBP` is practically as good as the more expensive algorithms in terms of classification error, as shown by the fact that, except for a single case, it attains the same error value for all the test scenarios in Table 3a-3c. As mentioned in Section 5.2, we can use our column generation method `colgen` in order to produce an indication of how far, in terms of linear correctness, a given solution is from an optimal one, even when the latter cannot be computed in reasonable time by `optimal`, such as, e.g., for $t = 100$. Yet, after inspecting the upper bounds computed by `colgen`, we conclude that the linear correctness values found by `CBP` were always guaranteed to be, on average, at least 86.30% of the optimal linear correctness value (93.26% with default parameter values) when decreasing accuracies are considered (83.27% when increasing accuracies are considered). In fact, the actual ratio between a given solution and an optimal solution is unknown in practice, but could be well above such values. As for our experiments, these ratios give us an indication of the fact that, even in the practically more challenging case of annotators suffering from decreasing accuracies (for which the classification error is significantly higher than for increasing accuracies), we are guaranteed to be rather close to an optimal solution. Note, in addition, that `CBP` and `local_search` keep finding solutions of comparable or identical quality even for task set sizes that cannot be dealt with by `optimal`, as shown in Table 3 for $t \geq 100$. Therefore in the following experiments on large-scale data sets we will only focus on `CBP` and contrast it with the `random`, `regular`, `TA` and `PA` baselines.

	optimal	CBP	local_search
$t = 3, a = 3$	0.30	0.30	0.30
$t = 4, a = 3$	0.30	0.30	0.30
$t = 5, a = 3$	0.32	0.32	0.32
$t = 100, a = 10$	-	0.21	0.21
$t = 200, a = 20$	-	0.17	0.17
$t = 1000, a = 100$	-	0.07	0.07

(a) Decreasing accuracies - classification error ϵ

	optimal		CBP		local_search	
$t = 3, a = 3$	0.43	(100%)	0.43	(100%)	0.43	(100%)
$t = 4, a = 3$	0.38	(100%)	0.37	(98.30%)	0.38	(100%)
$t = 5, a = 3$	0.35	(100%)	0.35	(100%)	0.35	(100%)
$t = 100, a = 10$	-	(100%)	0.96	($\geq$86.30%)	0.98	($\geq$88.48%)
$t = 200, a = 20$	-	(100%)	1.41	($\geq$87.81%)	1.42	($\geq$88.70%)
$t = 1000, a = 100$	-	(100%)	2.71	($\geq$93.26%)	2.72	($\geq$93.47%)

(b) Decreasing accuracies - linear correctness

	optimal	CBP	local_search
$t = 3, a = 3$	0.21	0.21	0.21
$t = 4, a = 3$	0.19	0.19	0.19
$t = 5, a = 3$	0.19	0.20	0.19
$t = 100, a = 10$	-	0.13	0.13
$t = 200, a = 20$	-	0.12	0.12
$t = 1000, a = 100$	-	0.04	0.04

(c) Increasing accuracies - classification error ϵ.

	optimal		CBP		local_search	
$t = 3, a = 3$	1.28	(100%)	1.27	(99.25%)	1.28	(100%)
$t = 4, a = 3$	1.33	(100%)	1.32	(99.41%)	1.33	(100%)
$t = 5, a = 3$	1.35	(100%)	1.34	(99.46%)	1.35	(100%)
$t = 100, a = 10$	-	(100%)	1.86	($\geq$83.99%)	1.86	($\geq$83.99%)
$t = 200, a = 20$	-	(100%)	2.04	($\geq$83.27%)	2.04	($\geq$83.27%)
$t = 1000, a = 100$	-	(100%)	2.80	($\geq$86.53%)	2.80	($\geq$86.53%)

(d) Increasing accuracies - linear correctness.

Table 3: (a),(c) classification error ϵ.
(b),(d) linear correctness value (see Eq. (10)); in parentheses, the ratio between the solution found by the algorithm and the optimal value; when the latter is not available ($t > 5$), we replace the optimal value by the upper bound provided by the column generation method (see Section 5.2). Synthetic data sets: task set size t and annotator set size a vary, defaults otherwise.

Budget availability. Figure 4 shows the classification error ϵ as the budget b varies within the values defined in Table 2. Predictably, the classification error ϵ decreases as the budget b increases, and CBP outperforms the baselines in all the execution scenarios. Namely, CBP leads to an error value 29% smaller, on average, than the closest baseline (TA, here) in the decreasing accuracy scenario; the same distance to the closest baseline is attained in the increasing case, although against a different algorithm (PA, here). For poorer baselines, ϵ may exceed 30%.

Size of the task set. Figures 5-6 show the classification error ϵ and the algorithm execution time as the task set size t varies within the values reported in Table 2. In order not to change the population of annotators throughout the

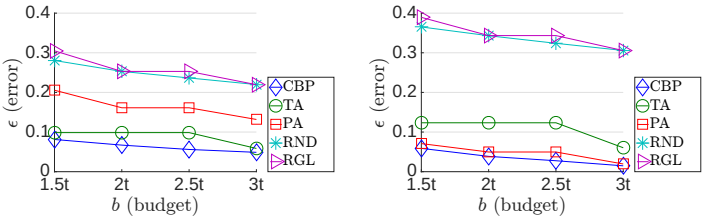
experiment, we set a limit of 1000 iterations per annotator (which equals t in the default case). As shown in Figure 5a-5b, the error increases as t increases. This can be easily explained by noticing that we require a hundred annotators to make, e.g., 20000 annotations overall for $t = 10000$, i.e., on average 200 each. Because of this, it is reasonable to expect the accuracy to be low for some of those annotations. Nevertheless, `CBP` leads to a classification error 32% (22%) smaller for $t = 1000$, and 6% (18%) smaller for $t = 10000$ when compared to the nearest baseline, for decreasing and increasing accuracies, respectively. Note that, since only 1000 iterations are available per annotator, when $t = 100000$ (a size hardly ever needed by a requester) each task receives exactly one annotation, thus causing all the algorithms to perform the same.

Figures 6a and 6b show that the presence of an accuracy variation phenomenon does not affect the execution time of `CBP`. Moreover, for $t = 100000$, the `regular` baseline dominates the execution time. This is because the algorithm may need several attempts to build a graph that is both random and regular. However, in all cases the time required to compute offline the task assignments under any of the policies is negligible w.r.t. the total time needed for an annotator to label her tasks. Indeed, even in the best experimental scenario reported in Figure 3, an annotator needs on average over 5 seconds to provide a single answer.

Size of the annotator set. Figure 7 shows the classification error ϵ as the annotator set size a varies within the values in Table 2. When considering decreasing accuracies, the classification error ϵ achieved by `CBP` is from 4% ($a = 10$) to over 38% ($a = 200$) smaller than the nearest baseline. Similarly, when considering increasing accuracies, the classification error ϵ ranges from 15% ($a = 10$) to more than 22% ($a = 100$) smaller than the nearest baseline.

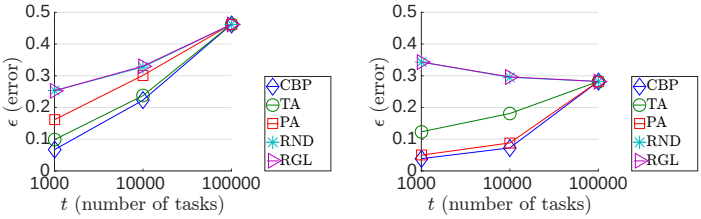
Accuracy variation degree. The default populations of annotators tested so far were mixing groups of annotators with several values of τ (accuracy variation degree). Figure 8 now shows the classification error ϵ as τ varies within the values in Table 2. As shown in Figure 8a, when considering decreasing accuracies, the effectiveness of the algorithms is greatly reduced by smaller values of τ. Indeed, when $\tau = 0.25$, `CBP` leads to a 13% improvement against the nearest baseline. However, the performance of `CBP` significantly improves for more realistic values of τ, e.g., $\tau = 1$ (which best match the trends of the curves in Figure 3), in which `CBP` achieves a classification error ϵ over 26% smaller than the nearest baseline. Figure 8b reveals a similar trend for increasing accuracies. In this case, however, larger values of τ (corresponding to very unlikely situations in practice) lead to a stalled increase in the accuracy of the annotators until the very last iterations. This entails flattening the performance of the different approaches and, therefore, for $\tau = 100$, `CBP` exhibits no observable improvement with respect to the closest baseline, and a modest 2% improvement is achieved for $\tau = 10$.

Type of crowd. Figure 9 shows the classification error ϵ when testing the algorithms against two crowds of different expertise (whose parameters are reported in Section 6.2.1). As shown, the crowd expertise deeply affects the outcome of the `random` and `regular` baseline methods, regardless of the



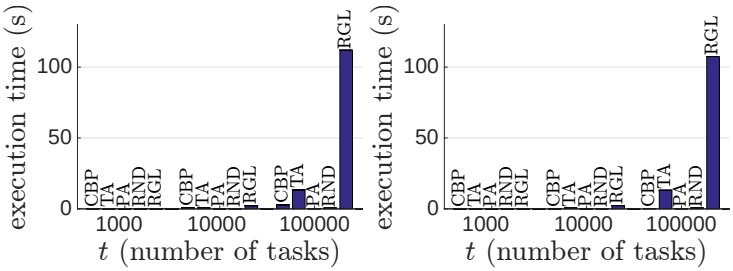
(a) Decreasing accuracies. (b) Increasing accuracies.

Fig. 4: Classification error ϵ on large-scale data sets: budget b varies, defaults otherwise.



(a) Decreasing accuracies. (b) Increasing accuracies.

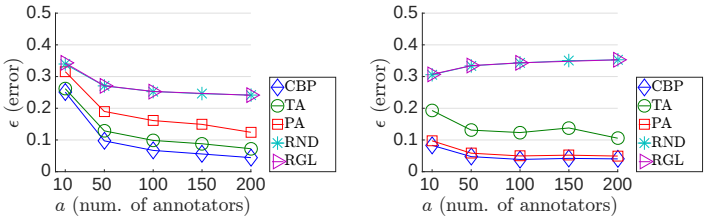
Fig. 5: Classification error ϵ on large-scale data sets: task set size t varies, defaults otherwise.



(a) Decreasing accuracies. (b) Increasing accuracies.

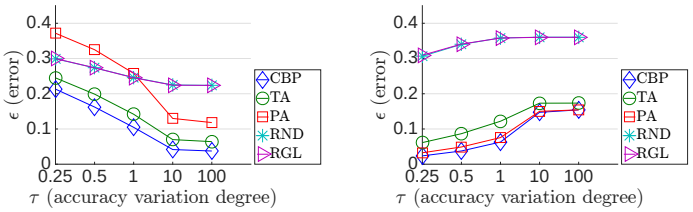
Fig. 6: Execution time on large-scale data sets: task set size t varies, defaults otherwise.

type of accuracy variation. Conversely, CBP shows more robust and consistent performance when moving from a crowd of heterogeneous expertise (such as NAIVE, which has a high variance) to a professional crowd. Moreover, in the NAIVE case, CBP leads to a classification error ϵ 32% smaller for decreasing accuracies (Figure 9a) and more than 22% smaller for increasing accuracies (Figure 9b) w.r.t. the nearest baseline. Similarly, in the EXPERT case, CBP's error is 44% and 25% smaller than the closest baseline in the decreasing and, resp., increasing scenario. This shows that, both in the realistic case of a



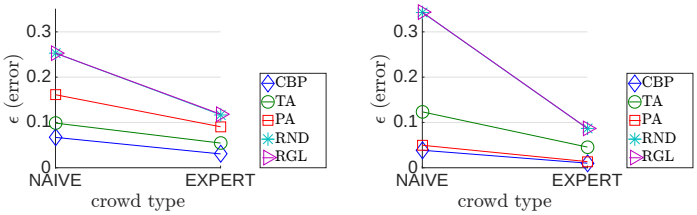
(a) Decreasing accuracies. (b) Increasing accuracies.

Fig. 7: Classification error ϵ on large-scale data sets: annotator set size a varies, defaults otherwise.



(a) Decreasing accuracies. (b) Increasing accuracies.

Fig. 8: Classification error ϵ on large-scale data sets: accuracy variation τ varies, defaults otherwise.



(a) Decreasing accuracies. (b) Increasing accuracies.

Fig. 9: Classification error ϵ on large-scale data sets: crowd type varies, defaults otherwise.

heterogeneous crowd and in the desirable case of an expert crowd, it becomes crucial to explicitly consider the expected linear correctness for each task.

Non-monotonic trends. In order to assess the robustness of CBP to different possible accuracy functions, we tested its performance with annotators featuring non-monotonic accuracy functions. Specifically, we constructed an annotator population so that annotators would first improve up to a given iteration, beyond which they would instead start decreasing their accuracy. These accuracy variations were simply obtained by juxtaposition of one increasing trend and one decreasing trend, defined similarly to what is done in

Section 6.2.1, with the additional indication that the iteration marking the transition between increasing and decreasing trend is sampled from a uniform distribution in the interval $[1, t]$. Figure 10 shows the attained classification error ϵ for all methods under test. When compared with the nearest baseline, `CBP` leads to a classification error 38% and 53% smaller for $b = 1.5t$ and $b = 3t$, respectively.

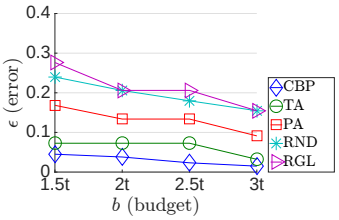


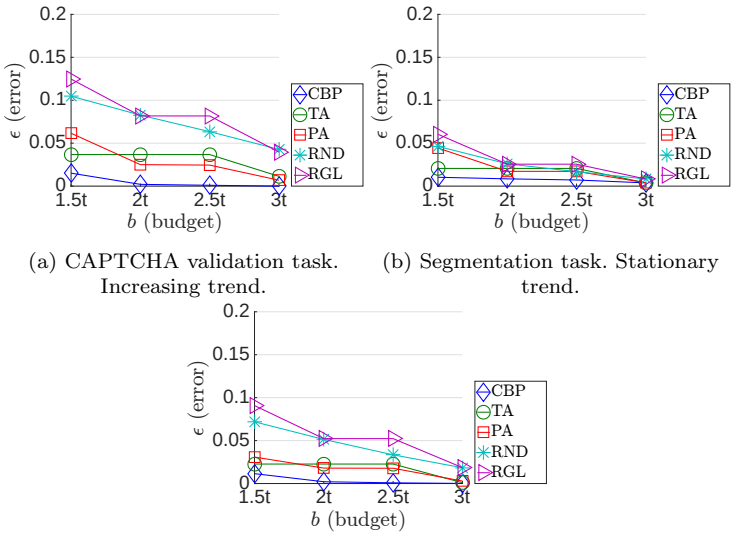
Fig. 10: Classification error ϵ on large-scale data sets: budget b varies, non-monotonic trends.

Real data sets. In the case of real data sets, the budget b is the only parameter that can vary, as all the other parameters are fixed given the specific real crowd in use. Due to the iterative nature of `CBP`, we needed to execute the experiment just once, with the largest budget value $b = 3t$, and measured the outcomes at all the intermediate points indicated in Table 2. Figure 11 shows the classification error ϵ using the real data set as described in Section 6.1. The resulting classification error for `CBP` is, resp., 93%, 46%, and 91% smaller than the closest baseline, see Figures 11a (excluding $b{=}3t$, where `CBP` achieves $\epsilon{=}0$), 11b and 11c.

7 Related work

We proposed an offline approach for the task assignment problem in order to completely determine the task assignment before collecting the contributions coming from the annotators. Alternative formulations of the offline task assignment problem are present in the literature. We have already compared our work to [17] and [14], and we have devised an offline counterpart of the online iCrowd algorithm [12], which we have used as three of our baselines.

In [19], the authors propose an algorithm for the assignment and pricing of tasks under quality and temporal constraints. In [25], the authors focus on the problem of allocating budgets to a set of tasks so that the total allocated budget does not exceed a certain limit, and provide a Probably Approximately Correct (PAC) bound on the classification error. Nevertheless, all the afore-mentioned works do not explicitly take into account that the annotators' accuracy may change.



(a) CAPTCHA validation task. Increasing trend.

(b) Segmentation task. Stationary trend.

(c) Visual perception task. Decreasing trend.

Fig. 11: Classification error ϵ (real data set): budget b varies within the values in Table 2.

Clearly, other solving approaches are possible, and, indeed, a significant number of works ([24], [15], [7], [9], [5], [29], [12]) cast the task assignment problem as an online decision-making problem, thus assuming that each annotator must be assigned to a task in an online fashion. Unlike our work, the accuracy of each annotator is initially unknown, and must be learnt through observation. As a result, such an online policy needs to find a trade-off between exploration (sampling the accuracy of the annotators) and exploitation (assigning the annotators believed to be the most accurate). As with offline approaches, these works do not consider changes in the annotators' accuracy either.

As regards changes in annotators' accuracies, of particular interest are the works [11], [6], [10] and [21], which, similarly to what we presented, try to explicitly take into account temporal-dependent phenomena. Namely, in [11], the authors propose an online task assignment policy that adaptively assigns annotators based on the current, time-dependent, estimation of their accuracy. Unlike our approach, the accuracy of each annotator varies at each time step, regardless of the number of times the annotator is assigned (that is, the annotators' workload is assumed not to be related with the accuracy variations). To wit, the annotator accuracy value at time t is computed as the sum of the accuracy value at time $t-1$ and a zero-mean Gaussian noise. In [6] and [10], the average accuracy and response time variation of two crowd platforms over a certain number of weeks is studied. Moreover, the authors in [6] and [10] propose a crowd platform selection algorithm based on the multi-

armed bandit framework and the supervised learning framework, respectively. On the other hand, instead of focusing on how crowd platforms globally differ in their accuracy trend, we address the problem of leveraging the accuracy variation of each single annotator within the same platform. In [21], the hourly trend in average accuracy and response time (over a one-week period) of an undisclosed crowd platform is reported. Such data are synthetically extended over a period of four weeks, and consequently used to evaluate an online task assignment policy that adaptively selects the task batch size to be submitted to the crowd. When the performance parameter to optimize is the task accuracy, the task assignment policy estimates the size of the next batch based on the accuracy of the previous batch. Thus, we cannot directly compare to this work due to the implicit assumption that the task requester is able to instantly and accurately evaluate the correctness of the tasks. Indeed, in the context of binary tasks, verifying the quality of the answer (i.e., assessing that a given binary property holds for a task) has the same cost as executing the task.

8 Conclusions and future work

In this work, we studied the problem of assigning human workers to tasks under the assumption that the workers' reliability could change over time, as a result of, e.g., fatigue and learning. In this respect, we provided empirical evidence of the existence of a workload-dependent accuracy variation among workers, both for volunteer and professional crowds.

When attacking the problem, we adopted the perspective of the owner of a crowdsourcing platform, who has all the historic data and trends from previous sessions about its workers. A platform may then offer a service for automatically assigning human annotators to tasks that a requester, endowed with a given budget, needs to label. Thanks to the collected data, the owner could maintain typical trends (as a collection of templates) for the different types of tasks and potential workers. This way, once a new worker enters the system, her performance could be sampled on a small subset of (possibly, test) tasks, such as it is done in, e.g., CrowdFlower [2]. This would allow the owner to decide which template she is most likely to follow, so to be able to rely, in the subsequent task assignment phase, on a forecast of p_{jk} for a sufficiently large sequence of labeling iterations k.

After introducing a suitable measure of the quality of the aggregate annotations provided by a set of workers to each task (correctness) and its linearized counterpart, we cast the task assignment problem as a mixed-integer linear programming problem, with the objective of finding a feasible assignment of the workers (i.e., one compatible with the given budget) that maximizes the minimum of the (linear) correctness that is achieved over the different tasks. We proposed several implementations of our approach, including a (linear) correctness-based policy and a local-search meta-heuristic. The performance of the algorithms was experimentally evaluated on both synthetic and real data sets, and compared with several baselines from the state of the art, which are

always outperformed by our approaches. We also developed a procedure based on column generation that allowed us to conclude that the results found by the (linear) correctness-based policy are of very high quality, guaranteeing solutions at least 86% as good as optimal ones for large problem instances.

According to Section 2, all human annotators provide a (possibly) noisy binary label z_{ijk} with a certain accuracy p_{jk}. We have shown the benefits in terms of error reduction of an offline task-to-worker assignment policy that maximizes the linear correctness while taking into account the worker's reliability and its evolution. However, the proposed approach is not relevant only to crowdsourcing scenarios: the described data labeling capabilities are not a prerogative of human beings, and, in fact, there are other *information sources* that adhere to the aforementioned definition. In many binary classification problems, e.g., detecting the presence of an object in an image, a preprocessing phase can be carried out by means of automatic algorithms, whose outcome can usually be modeled as a binary label. Hence, in future work, the annotator set could be extended so as to comprise both human and automatic annotators. Differently from the human annotators, automatic annotators can be freely assigned without affecting the available budget. As a consequence, it could be convenient to let the automatic annotators express their label for each task before determining the assignment, so that the provided labels contribute to the linear correctness of each task. In the subsequent assignment phase, the linear correctness of each task would serve as a first indication of which tasks are in greater need of additional labels.

The solution proposed in this paper was specifically designed for sets of tasks that, within the same set, are all of the same type. This is a common enough scenario in crowdsourcing applications that deserved a focused study. Accommodating batches of heterogeneous tasks in the proposed model might require a certain level of asymmetry between the considered scenarios of increasing and decreasing accuracies. For instance, a worker's "fatigue" naturally continues across tasks of different type; instead, a worker's "learning" might have specific per-type rates and may not carry from one task type to another. Future research might try to address these issues.

References

1. Amazon Mechanical Turk. `https://www.mturk.com`.
2. CrowdFlower. `https://www.crowdflower.com`.
3. Microtask. `http://microtask.com`.
4. reCAPTCHA. `https://www.google.com/recaptcha`.
5. I. Abraham, O. Alonso, V. Kandylas, and A. Slivkins. Adaptive crowdsourcing algorithms for the bandit survey problem. In *COLT*, pages 882–910, 2013.
6. L. E. Celis, K. Dasgupta, and V. Rajan. Adaptive crowdsourcing for temporal crowds. In *WWW Companion*, pages 1093–1100, 2013.
7. X. Chen, Q. Lin, and D. Zhou. Optimistic knowledge gradient policy for optimal budget allocation in crowdsourcing. In *ICML*, pages 64–72, 2013.
8. E. Ciceri, P. Fraternali, D. Martinenghi, and M. Tagliasacchi. Crowdsourcing for top-k query processing over uncertain data. *Knowledge and Data Engineering, IEEE Transactions on*, 2015.

9. P. Dai, C. H. Lin, Mausam, and D. S. Weld. Pomdp-based control of workflows for crowdsourcing. *A. I.*, 202:52–85, 2013.
10. K. Dasgupta, V. Rajan, S. Karanam, K. Ponnavaikko, C. Balamurugan, and N. M. Piratla. Crowdutility: Know the crowd that works for you. In *CHI*, pages 145–150, 2013.
11. P. Donmez, J. G. Carbonell, and J. G. Schneider. A probabilistic framework to learn from multiple annotators with time-varying accuracy. In *SDM*, pages 826–837, 2010.
12. J. Fan, G. Li, B. C. Ooi, K.-l. Tan, and J. Feng. iCrowd: An Adaptive Crowdsourcing Framework. In *SIGMOD*, pages 1015–1030. ACM, 2015.
13. L. Galli, P. Fraternali, D. Martinenghi, M. Tagliasacchi, and J. Novak. A draw-and-guess game to segment images. In *2012 International Conference on Privacy, Security, Risk and Trust, PASSAT 2012, and 2012 International Conference on Social Computing, SocialCom 2012, Amsterdam, Netherlands, September 3-5, 2012*, pages 914–917, 2012.
14. C.-J. Ho, S. Jabbari, and J. W. Vaughan. Adaptive task assignment for crowdsourced classification. In *ICML*, pages 534–542, 2013.
15. C.-J. Ho and J. W. Vaughan. Online task assignment in crowdsourcing markets. In *AAAI*, pages 45–51, 2012.
16. P. G. Ipeirotis. Analyzing the amazon mechanical turk marketplace. *XRDS*, pages 16–21, 2010.
17. D. R. Karger, S. Oh, and D. Shah. Iterative learning for reliable crowdsourcing systems. In *NIPS*, pages 1953–1961, 2011.
18. A. Kobren, C. H. Tan, P. Ipeirotis, and E. Gabrilovich. Getting more for less: Optimized crowdsourcing with dynamic tasks and goals. In *WWW*, pages 592–602, 2015.
19. P. Minder, S. Seuken, A. Bernstein, and M. Zollinger. Crowdmanager - combinatorial allocation and pricing of crowdsourcing tasks with time constraints. In *ACM-EC 2012 Workshops*, 2012.
20. G. L. Nemhauser and L. A. Wolsey. *Integer and combinatorial optimization*, volume 18. Wiley New York, 1988.
21. V. Rajan, S. Bhattacharya, L. E. Celis, D. Chander, K. Dasgupta, and S. Karanam. Crowdcontrol: An online learning approach for optimal task scheduling in a dynamic crowd platform. In *ICML Workshop: Machine Learning Meets Crowdsourcing*, 2013.
22. V. C. Raykar, S. Yu, L. H. Zhao, G. H. Valadez, C. Florin, L. Bogoni, and L. Moy. Learning from crowds. *J. of Machine Learning Research*, pages 1297–1322, 2010.
23. V. S. Sheng, F. Provost, and P. G. Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *KDD*, pages 614–622, 2008.
24. L. Tran-Thanh, S. Stein, A. Rogers, and N. R. Jennings. Efficient crowdsourcing of unknown experts using bounded multi-armed bandits. *A. I.*, 214:89–111, 2014.
25. L. Tran-Thanh, M. Venanzi, A. Rogers, and N. R. Jennings. Efficient budget allocation with accuracy guarantees for crowdsourcing classification tasks. In *AAMAS*, pages 901–908, 2013.
26. L. A. Wolsey. *Integer programming*, volume 42. Wiley New York, 1998.
27. Y. Yan, R. Rosales, G. Fung, and J. G. Dy. Active learning from crowds. In *ICML*, pages 1161–1168, 2011.
28. C. J. Zhang, L. Chen, H. V. Jagadish, and C. C. Cao. Reducing uncertainty of schema matching via crowdsourcing. *PVLDB*, 6(9):757–768, 2013.
29. Y. Zheng, J. Wang, G. Li, R. Cheng, and J. Feng. QASCA: A Quality-Aware Task Assignment System for Crowdsourcing Applications. In *SIGMOD*, pages 1031–1046, 2015.

A Derivation of the pricing subproblem

Consider the *primal* linear program $\max\{cx : Ax \leq b, x \geq 0\}$, with $c \in \mathbb{R}^n$, $x \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, and $b \in \mathbb{R}^m$. By aggregating the constraints $Ax \leq b$ with a vector $y \in \mathbb{R}^m_+$, we have the valid inequality $yAx \leq yb$. If we choose y such that $yA \geq c$, then $cx \leq yAx \leq yb$. This implies that, for any $y \geq 0$ satisfying $yA \geq c$, we obtain an upper bound of value yb on the value of an optimal solution to the primal problem. The tightest upper bound

is obtained by solving $\min\{yb : yA \geq c, y \geq 0\}$ (the *dual* problem). For each $j = 1, \ldots, n$, the dual constraint of x_j can be obtained by first aggregating all the primal constraints as $\sum_{i=1}^{m} y_i \sum_{j=1}^{n} a_{ij} x_j \leq \sum_{i=1}^{m} y_i b_i$, then collecting x_j on the left-hand side, yielding $\sum_{j=1}^{n} x_j (\sum_{i=1}^{m} y_i a_{ij}) \leq \sum_{i=1}^{m} y_i b_i$, and finally imposing $\sum_{i=1}^{m} y_i a_{ij} \geq c_j$.

We now derive the dual constraint for Problem (14)–(19), corresponding to variable λ_{ih}. We first aggregate the inequalities (15)–(19), each of which multiplied by the corresponding dual variable (the calculation of the right-hand side is omitted):

$$\begin{aligned}
&\sum_{i\in\mathcal{T}} \alpha_i \left(\eta - \sum_{h\in\mathcal{H}} c_h \lambda_{ih} \right) + \\
&+\beta \left(\sum_{i\in\mathcal{T}} \sum_{h\in\mathcal{H}} \left(\sum_{j\in\mathcal{A}} \sum_{k\in\mathcal{I}} w_{hjk} \right) \lambda_{ih} - b \right) + \\
&+ \sum_{j\in\mathcal{A}} \sum_{k\in\mathcal{I}} \gamma_{jk} \left(\sum_{i\in\mathcal{T}} \sum_{h\in\mathcal{H}} w_{hjk} \lambda_{ih} - 1 \right) + \\
&+ \sum_{i\in\mathcal{T}} \delta_i \left(\sum_{h\in\mathcal{H}} \lambda_{ih} - 1 \right) + \\
&+ \sum_{j\in\mathcal{A}} \sum_{k\in\mathcal{I}\setminus\{1\}} \epsilon_{jk} \left(\sum_{i\in\mathcal{T}} \sum_{h\in\mathcal{H}} (w_{hjk} - w_{hj,k-1}) \lambda_{ih} \right) \geq (\cdot). \qquad (27)
\end{aligned}$$

Let $\epsilon_{j,|\mathcal{I}|+1} = 0$. After collecting λ_{ih} (and omitting the coefficient for η and the right-hand side), Inequality (27) becomes:

$$\sum_{\substack{i\in\mathcal{T} \\ h\in\mathcal{H}}} \left(-\alpha_i c_h + \sum_{\substack{j\in A \\ k\in\mathcal{I}}} \left(\beta + \gamma^{jk} + \epsilon^{jk} - \epsilon^{j,k+1} \right) w_{hjk} + \delta_i \right) \lambda_{ih} + (\cdot)\eta \geq (\cdot).$$

For each $i \in \mathcal{T}$ and $h \in \mathcal{H}$, the dual constraint corresponding to λ_{ih} thus reads:

$$-\alpha_i c_h + \sum_{j\in A} \sum_{k\in\mathcal{I}} \left(\beta + \gamma_{jk} + \epsilon_{jk} - \epsilon_{j,k+1} \right) w_{hjk} + \delta_i \geq 0, \qquad (28)$$

where the right-hand side is zero since λ_{ih} does not show up in the objective function of Problem (14)–(19).

By standard linear programming duality, we have that the reduced cost of a column is equal to the slack of the corresponding dual constraint. More precisely, to a primal column with nonnegative reduce costs corresponds a dual constraint which is violated. Hence, the pricing subproblem amounts to finding a feasible workplan that minimizes the left-hand side of (28) (or, equivalently, that maximizes its opposite). Thus, Problem (21)–(24) is obtained. When solving it for index i, any solution of value greater than or equal to δ_i yields a new column with a nonnegative reduced cost which, when added to Problem (14)-(19), might improve its solution.