Accurate inference of crowdsourcing properties when using efficient allocation strategies

Abigail Hotaling1,2 & James Bagrow1,2

Allocation strategies improve the efficiency of crowdsourcing by decreasing the work needed to complete individual tasks accurately. However, these algorithms introduce bias by preferentially allocating workers onto easy tasks, leading to sets of completed tasks that are no longer representative of all tasks. This bias challenges inference of problem-wide properties such as typical task difficulty or crowd properties such as worker completion times, important information that goes beyond the crowd responses themselves. Here we study inference about problem properties when using an allocation algorithm to improve crowd efficiency. We introduce Decision-Explicit Probability Sampling (DEPS), a novel method to perform inference of problem properties while accounting for the potential bias introduced by an allocation strategy. Experiments on real and synthetic crowdsourcing data show that DEPS outperforms baseline inference methods while still leveraging the efficiency gains of the allocation method. The ability to perform accurate inference of general properties when using non-representative data allows crowdsourcers to extract more knowledge out of a given crowdsourced dataset.

Crowdsourcing has become a valuable source of information for a wide variety of applications1–3, particularly in areas where humans can perform work not possible or not well suited to computational methods. Common crowdsourcing tasks include generating labeled training data for machine learning algorithms4–6, performing text recognition or natural language tasks4, completing surveys7, or generating novel questions or other creative inputs8–12. However, using human volunteers or paid crowd workers brings with it a host of new problems3,13.

Human workers are often more costly than computational methods, receiving responses from workers can be relatively slow, and one must worry about worker motivation14,15 and reliability3,16. Even with reliable workers, challenging tasks often require a consensus approach: multiple workers are given a task and their set of (possibly noisy) responses are aggregated to generate the final response for the crowdsourcer17.

Algorithmic crowdsourcing focuses on computational approaches to dealing with issues introduced when using a crowdsourcing solution18–21. These algorithms improve the crowd's efficiency by enabling more accurate information gain, and often reduce the numbers of workers needed. Some methods learn the reliability of workers and minimize or eliminate the contributions of unreliable workers16,18,22. Allocation methods, on the other hand, focus on the tasks being crowdsourced, and attempt to allocate workers towards those tasks that can be completed most quickly and accurately19,21. Quickly identifying those tasks where workers tend to give the same response can prevent redundant worker responses and allows the crowdsourcer to decide upon the aggregate response more efficiently.

The focus of crowd allocation algorithms has been on maximizing accuracy when aggregating responses from the crowd while minimizing the number of responses needed in order to meet budget constraints. However, other aspects of a crowdsourcing problem are important to understand besides the final responses to the set of tasks comprising that problem. For example, a crowdsourcer may wish to identify experts within the crowd or understand how many tasks are difficult for workers to complete compared with tasks that are easy for workers. Or a crowdsourcer concerned about latency may want to understand how much time it takes workers to perform a task12,23. Worker completion times are an example of the more general behavioral traces of workers24, and a crowdsourcer may be interested in understanding the pattern of worker dynamics as they perform work. Yet in all these cases, an allocation algorithm focused on completing tasks accurately can change the set of tasks being

1Department of Mathematics and Statistics, University of Vermont, Burlington, VT, USA. 2Vermont Complex Systems Center, University of Vermont, Burlington, VT, USA. *email: james.bagrow@uvm.edu
shown to workers, introducing a bias where easily completed tasks are more likely to receive responses from workers and leaving the crowdsourcer with a dataset that may be unrepresentative of the typical tasks or typical worker behaviour within the crowdsourcing problem.

In this work, we ask if it is possible to reason about properties of crowdsourcing problems while also employing an efficient allocation algorithm. Of course, one can provide an unbiased view of tasks and workers by assigning problems at random, but doing so sacrifices the efficiency gains an allocation strategy can provide; it may simply not be worth the added cost to learn these problem properties, especially if developing accurate responses is the main goal. We introduce Decision-Explicit Probability Sampling (DEPS), a strategy to perform inference on problem properties that places the decision process of an efficient allocation algorithm within the inference model. DEPS can provide inference for a variety of problem properties, although our focus here is on inferring the distribution of problem difficulty, information a crowdsourcer often desires. Experiments with real crowdsourcing data and synthetic models show that DEPS provides accurate information about the crowdsourcing problem while using an allocation algorithm, allowing a crowdsourcer to leverage the efficiency gains of an allocation algorithm while retaining more of the problem information that can be lost when using an allocation algorithm.

The rest of this paper is organized as follows. In “Background” section we describe a model for a crowdsourcing problem with a fixed budget, discuss the properties of crowdsourcing problems, and detail allocations algorithms that have been introduced to improve crowdsourcing efficiency. “Allocation methods lead to non-representative sets of tasks” section illustrates the challenge of property inference when using efficient allocation strategies. “Decision-explicit probability sampling (DEPS)” section introduces Decision-Explicit Probability Sampling (DEPS), a strategy for integrating the results of an allocation algorithm into property inference. Then, in “Materials and methods” and “Experiments” sections we describe and present experiments using DEPS on real and synthetic crowdsourcing data. We discuss our results and future work in “Discussion” section and conclude in “Conclusion” section.

**Background**

Here we describe the model of a crowdsourcing problem along with various problem properties that are meaningful to study for such a problem. We also provide background on efficient allocation algorithms a crowdsourcer can use to maximize the information gained from the crowd. Crowdsourcing problem models can be used to generate synthetic crowdsourced data (as a generative model), but the primary use of the model is to build inference procedures to be used when processing real crowdsourced data (as a statistical model).

**Crowdsourcing model.** We employ the Dawid-Skene model for a crowdsourcing problem\(^2\). A crowdsourcing problem consists of \(N\) tasks, each of which is considered a binary labeling task. Binary tasks can represent image classification tasks, survey questions, and other problems. While relatively simplistic, binary labeling forms the basis of most crowdsourcing models. In our results, we analyze three such tasks: recognizing whether one written statement entails another statement, classifying a photograph of a bird as containing one species or another, and identifying whether a given web page would be relevant to a given topic (see “Datasets” section). The binary task model can also generalize to categorical tasks, but such tasks can always be binarized by treating the most common individual response as a ‘1’ and all other responses as ‘0’. Let \(z_i \in \{0, 1\}\) be the true (unknown) label for task \(i\). A total of \(M\) workers are given one or more of these tasks and respond by providing a label for the given task. Let \(y_{ij} \in \{0, 1\}\) be the response of worker \(j \in [1, M]\) to task \(i \in [1, N]\), \(n_i\) the total number of 1-labels for task \(i\), \(b_i\) the total number of 0-labels, and \(n_i = a_i + b_i\) the total number of responses to \(i\). Individual worker responses are taken as iid for a given task. Not all workers necessarily respond to every task, so define \(I_i\) as the set of workers who responded to task \(i\), and \(|\bigcup_{i=1}^N I_i| = M\) (we assume every worker responds to at least one task and no worker responds to the same task more than once). Likewise, let \(I_j\) be the set of tasks that worker \(j\) responded to, with \(|\bigcup_{j=1}^M I_j| = N\). The crowdsourcer can then aggregate the individual responses to reach a consensus response. The goal is to infer best estimates \(\hat{z}_i \approx z_i\) given worker responses \(y_{ij}\). Often the crowdsourcer must approach this goal under budget constraints, as costs are incurred when employing a crowd. Let \(T\) be a crowdsourcer’s total budget, meaning the crowdsourcer can request at most \(T\) individual worker responses. This constraint leads to \(\sum_{i=1}^N n_i \leq T\). In seminal work, Dawid and Skene use an EM-algorithm to infer \(z_i\) that outperforms the basic majority-vote strategy\(^3\) and much subsequent work has studied and generalized this approach\(^4\).

**Problem properties.** Learning labels accurately is the goal of the crowdsourced labeling problem described above. However, there are potentially many other properties of such problems that are of interest to researchers. For example, what constitutes a typical task within the problem? Do tasks tend to be easy for workers to complete or are tasks difficult? Are there distinguishing features of tasks, such as groups or categories of tasks that are similar in some way? Likewise, a crowdsourcer may wish to learn properties of the crowd workers. Are there experts among the workers? Are workers reliable or not? Will workers tend to finish an assigned task quickly or will they require a lot of time to perform the work? Can we learn about worker behavioral traces\(^5\), the patterns of activities the workers undertake as they respond to a task?

In general, learning both task properties and crowd properties can be of significant benefit to a crowdsourcer: tasks can be better designed accounting for features of the tasks and how workers interact with them, costs can be better forecast and planned, and more reliable data can be generated by the crowd. Of course, the details and importance of these properties will be heavily dependent on the type of crowdsourcing problem, but learning properties can potentially reveal significant information for the crowdsourcer, beyond that of the task labels themselves.
Efficient allocation methods. Various allocation strategies have been developed with the goal of assigning tasks to workers in order to maximize information gained about labels while minimizing costs\textsuperscript{1,25}. Here we provide background on how these algorithms assign tasks.

Requallo\textsuperscript{21} is a flexible allocation framework that addresses the challenge of efficient crowdsourcing for labeling tasks. With Requallo, crowdsourcers (1) define completion requirements that tasks must satisfy to be deemed completed, (2) apply a task allocation strategy to maximize the number of completed tasks within a given budget $T$ (number of worker responses). When a task $i$ is completed the crowdsourcer will cease seeking new responses for $i$ and be able to decide the final estimated label $\hat{z}_i$. There are many ways to define a completion requirement. For example, a ratio requirement for a binary labeling task $i$ would determine that $i$ is complete when one label is assigned by workers sufficiently more often than the other label: a ratio requirement of $c$ determines task $i$ to be complete if either $a_i/b_i > c$ or $b_i/a_i > c$, where $a_i$ and $b_i$ are the number of 1-labels and 0-labels given by workers for task $i$, respectively. (In practice, the Requallo authors use smoothed counts $a_i+1$ and $b_i+1$ for their requirement.) Using the requirement, let $C(t) \subseteq \{1, \ldots, N\}$ be the set of completed tasks after receiving $t \leq T$ total responses from workers. The Requallo requirement implements an early stopping rule while ensuring there is sufficient information to decide whether $z_i = 0$ or $z_i = 1$ with some degree of accuracy, with uncompleted tasks being undecided.

With a completeness requirement in place, Requallo then allocates tasks to workers using a Markov Decision Process (MDP) designed to quickly identify and distribute to workers those tasks which can be completed and to identify and avoid distributing those tasks which are unlikely to be completed. Due to computational complexity, the Requallo framework determines a policy for allocating tasks using a one-step look-ahead greedy method; see Li et al.\textsuperscript{21} for full details of the Requallo MDP reward function and optimization method. Requallo's completeness requirement combined with its allocation strategy gives significant efficiency gains, with more tasks completed accurately using a fixed budget of worker responses than other allocation methods.

However, the completeness and allocation components of Requallo also introduce bias in the collected data. Crucially, Requallo stops allocating tasks that reach completeness and the greedy MDP optimization can lead to “hard” tasks (those unlikely to reach completeness) being ignored in favor of “easy” tasks (those likely to reach completeness). (Note that a hard task in terms of completeness does not guarantee it is difficult or time-consuming for a particular worker to submit their response, only that the set of workers given that task will tend to disagree with one another. However, it is plausible that the same aspects of the task that lead to disagreement among workers may also affect the difficulty of the task or the time needed to submit a response.) These properties together greatly contribute to the bias introduced by Requallo, which may be especially harmful when a crowdsourcer wants to study properties of tasks and/or workers beyond the resulting labels themselves.

Not all allocation methods explicitly distinguish decided and undecided tasks. Opt-KG\textsuperscript{28}, for example, works to optimize accuracy among the set of tasks by allocating to the appropriate tasks in order to optimize the accuracy across all tasks. Once the algorithm has determined a sufficiently accurate label for easier tasks, it directs the budget towards tasks that are less accurate (hard tasks). This equates to a similar bias as the Requallo framework, where less information is obtained about easy tasks, even without an explicit decision process for those tasks. Of course, any such allocation method can always be augmented with a decision process by including a completion requirement, and deciding if tasks are completed based on that requirement.

Allocation methods lead to non-representative sets of tasks

Before introducing our method for inferring problem-wide properties of a crowdsourcing problem, we first illustrate the challenge introduced when a crowdsourcer employs an efficient allocation algorithm. Of course, this challenge can be eliminated by forgoing use of an allocation algorithm, but doing so may be too cost prohibitive to consider in practice.

We applied the Requallo allocation algorithm to a crowdsourcing problem of $N = 1000$ tasks averaged over 100 independent simulations defined as follows. Each synthetic task $i$ is represented by a Bernoulli RV parameterized by $p_i$, the probability of a 1-label. Here, tasks closer to $p_i = 1/2$ are more difficult than tasks far from 1/2, because a task with $p \approx 1/2$ will take many worker responses before we can accurately conclude whether $z_i = 1$ or 0. We took the prior distribution $Pr(p_i)$ of $p_i$ to be uniform, $p_i \sim U[0, 1]$. Knowing the true distribution of task difficulty $Pr(p)$ in this situation allows us to compare with the observed distribution developed by employing the Requallo Algorithm with completeness ratio requirement $c = 4$ (“Efficient allocation methods” section).

Figure 1 compares the results a crowdsourcer using the Requallo allocation algorithm would receive with the underlying distribution of the problem. In Fig. 1a we show the difficulties of tasks deemed by Requallo to be completed versus all tasks: $Pr(p_i \mid i \in C)$ versus $Pr(p)$. There is a significant bias in these distributions with Requallo-completed tasks containing far more easy tasks and far fewer difficult tasks than the overall distribution would imply. Of course, this is an example of Requallo working by design: a net consequence of efficient allocation (and in particular the early stopping rule introduced by the completeness requirement) will always be more easy tasks than hard tasks completed. Yet it is evident that we are unable to capture the underlying uniform distribution with this method.

Similarly, in Fig. 1b we examine the distributions of estimates of problem difficulty $\hat{p}$ for both completed and uncompleted tasks. Examining $Pr(\hat{p})$ alongside $Pr(p)$ is important as a crowdsourcer in practice will not have access to the true parameters $p_i$ and must instead infer them from the worker responses. Further, a crowdsourcer must often deal with small-data issues as $n_i$ may be small for a given task $i$. An example of this is when $a_i = 0$ or $a_i = n_i$, leading to $\hat{p} = 0$ or $\hat{p} = 1$; many inference procedures can fail when no counts for a given category are observed. A common solution to this problem is to use a smoothed estimate of $\hat{p}$, where $a_i = b_i = n_i + 2\epsilon$, respectively (we use $\epsilon = 1$ as did the Requallo authors\textsuperscript{21}). Examining the distributions in Fig. 1b, we see a significant bias in the smoothed $Pr(\hat{p})$, with a trimodal sample despite the true underlying
hard tasks are underexplored, but easy tasks are also underexplored. Well in terms of avoiding redundant worker responses, but also emphasizes the bias introduced by Requallo: 

incorporating the decision process of the efficient algorithm into the inferential model.

number of responses to binary labeling task

meanwhile, tend to pile up at because Requallo deemed them too difficult and instead allocated workers to other tasks. Those other tasks, p

is either more or less extreme than the estimate . These phenomena show how Requallo is actually working well in terms of avoiding redundant worker responses, but also emphasizes the bias introduced by Requallo: hard tasks are underexplored, but easy tasks are also underexplored.

Decision-explicit probability sampling (DEPS)

Suppose we wish to learn about the properties of a crowdsourcing problem, such as the distribution of problem difficulty . Accurately estimating the distribution of may be helpful for understanding the types of labeling tasks we ask workers to perform, for categorizing tasks into different groups, and for forecasting budgetary expenses such as total cost to perform typical tasks or total time we expect to wait for workers to complete their assignments. Given sufficient data, the distribution of can be estimated relatively quickly. Let be the number of responses to binary labeling task and be the number of -labels given by workers. Then is the MLE (point) estimate for . Aggregating many estimates can then be used to infer the distribution .

Unfortunately, this estimation strategy will be data-intensive in practice, leading to costly crowdsourcing simply to infer the overall distribution of . Efficient algorithms are commonly introduced to “speed up” crowdsourcing by allowing more accurate labels to be determined with fewer worker responses. However, as we saw in “Allocation methods lead to non-representative sets of tasks” section and Fig. 1, these methods work at the expense of information about : by focusing crowd resources on the easiest-to-complete tasks, the crowdsourcer is left with incomplete and unrepresentative data about the overall distribution of, in this case, . We ask in this work if it is possible to overcome the bias introduced by an efficient allocation algorithm, allowing one to both generate labels efficiently and still perform accurate inference of problem-wide features.

We propose Decision-Explicit Probability Sampling (DEPS), a method to perform inference of the problem distribution for a property when using crowdsourced data gathered by an allocation method. The allocation method will introduce some form of bias into the estimates , so our main focus is introducing a means to reason about the unbiased distribution of given the observed, biased distribution of . DEPS works by explicitly incorporating the decision process of the efficient algorithm into the inferential model.

DEPS proceeds in two phases. The first is developing an inferential model for that incorporates the allocation method’s decisions and the resulting bias, leading to a corrected variable . Next, a probability model is fit to to infer the distribution . To begin, notice that a completed task may be decided to have label or label , depending on the , otherwise a task is undecided. Let the decision indicator be the crowdsourcer’s decision for task after receiving a total of responses (across all tasks):

\[
d_i(t) = \begin{cases} 
-1 & \text{if } i \text{ decided } z_i = 0, \\
0 & \text{if } i \text{ undecided,} \\
1 & \text{if } i \text{ decided } z_i = 1,
\end{cases}
\]

with the set of completed tasks . DEPS incorporates the decision variable into our inference model as follows. For each , we determine a posterior distribution using
MLE values for parameters of (in this case) a beta distribution given the data are, or, as an alternative, method of moments estimators. For maximum likelihood, the joint log-likelihood for the features of property $x$ in how they incorporate the efficient algorithm’s decision into the inference model. For example, a beta prior each decision status. In other words, we use the prior distributions $x_i$ affected by the allocation algorithm. Next, to develop a single probability distribution for $x$ which creates truncation in the distribution of tasks and information about $x$ is lacking. If the efficient budget allocation algorithm is performing well, we can reason that the tasks that are considered complete could have a different distribution for $x$, estimated from the truncated crowd data, tells us. The priors $D_i$ can reflect our understanding of how $x$ may be affected by the allocation algorithm. Next, to develop a single probability distribution for $x$, DEPS continues by generating a sample $\hat{x}$ to serve as a debiased $x$ (as opposed to, for example, working with a mixture of $N$ per-task distributions): for each task $i \in [1, \ldots, N]$, sample $\hat{x}_i$ from the distribution $P(x_i | \{y_{ij}\}, d_i)$. Inference on these aggregated $\{\hat{x}\}$ is then performed using standard techniques such as maximum likelihood estimation (MLE) or the method of moments. For MLE, for example, parameters $\theta$ for $P(\hat{x})$ are determined by maximizing the joint log-likelihood $\ln L(\theta | \{\hat{x}\})$ of the parameters given the data.

As a concrete application of DEPS, we now focus on the property of problem difficulty discussed above (where now $x = p$); we discuss other properties in "Generalizations and other applications of DEPS" section. An allocation algorithm will introduce bias into the data, making the point estimates of $\hat{p}_i = a_i/n_i$ unreliable (Fig. 1b). For problem difficulty $p$, the beta distribution is the natural choice of prior: $p_i | d_i \sim \text{Beta}(\alpha_{d_i}, \beta_{d_i})$. Worker responses are modeled $y_{ij} \sim \text{Bernoulli}(p_i)$, then the posterior for $p_i$ (Eq. 2) will also follow a beta distribution with parameters $\alpha_i \equiv \alpha_{d_i} + a_i$, $\beta_i \equiv \beta_{d_i} + b_i$, where $a_i$ and $b_i$ are the number of 1 and 0 labels, respectively, for task $i$. The choice of hyperparameters $\alpha_i$ and $\beta_i$ gives the researcher flexibility in how they incorporate the efficient algorithm's decision into the inference model. For example, a beta prior that is skewed towards $p = 0$ could be used for tasks that have $d_i = 1$ with another beta skewed towards $p = 1$ used for $d_i = 0$. The amount of skewness of the $d \neq 0$ priors can then reflect how confident the researcher is in the allocation algorithm’s decision. And tasks that are undecided will have priors that provide more weight near $p = 1/2$. We illustrate two choices of decision priors in Fig. 2 and we discuss a data-driven approach to calibrating these priors in “Calibrating decision priors” section.

Having generated a sample $\{\hat{p}\}$, inference is performed using either standard maximum likelihood estimators or, as an alternative, method of moments estimators. For maximum likelihood, the joint log-likelihood for the parameters of (in this case) a beta distribution given the data are

$$
\ln L(\alpha, \beta | \{\hat{p}\}) = \sum_{i=1}^{N} \ln L_i(\alpha, \beta | \hat{p}_i)
$$

$$
= (\alpha - 1) \sum_{i=1}^{N} \ln \hat{p}_i + (\beta - 1) \sum_{i=1}^{N} \ln(1 - \hat{p}_i) - N \ln B(\alpha, \beta).
$$

Equation (3)

MLE values for $\alpha$ and $\beta$ are then found numerically by solving:

$$
\frac{\partial \ln L(\alpha, \beta | \{\hat{p}\})}{\partial \alpha} = \sum_{i=1}^{N} \ln \hat{p}_i - N(\psi(\alpha) - \psi(\alpha + \beta)) = 0
$$

Equation (4)
\[
\frac{\partial \ln \mathcal{L}(\alpha, \beta \mid \{ \hat{p}_i \})}{\partial \beta} = \sum_{i=1}^{N} \ln(1 - \hat{p}_i) - N(\psi(\beta) - \psi(\alpha + \beta)) = 0 \tag{5}
\]

where \(\psi\) is the digamma function. The method of moments estimators, meanwhile, are:

\[
\hat{\alpha} = \bar{p} \left( \frac{\bar{p}(1 - \bar{p})}{\hat{\sigma}^2} - 1 \right), \quad \hat{\beta} = (1 - \bar{p}) \left( \frac{\bar{p}(1 - \bar{p})}{\hat{\sigma}^2} - 1 \right), \tag{6}
\]

where \(\bar{p}\) and \(\hat{\sigma}^2\) are the sample mean and variance, respectively, of \(\hat{p}\).

**Calibrating decision priors.** Our approach to inferring the difficulty distribution introduces decision priors \(D_d\) to integrate the crowdsourcing allocation algorithm into the inference. The exact choice of these priors is up to the needs of the research; this freedom provides the researcher a framework for expressing their confidence in the algorithm. For example, a researcher studying problem difficulty \(p\) who is very confident in the algorithm’s decision \(d_i = 1 (d_i = -1)\) would select priors that are sharply peaked at \(p_i = 1\) \((p_i = 0)\). Likewise, a researcher with less confidence in the decisions would choose flatter priors over \(0 \leq p_i \leq 1\).

Depending on their problem, researchers may have *a priori* reasons for choosing their priors. But, sometimes, they may wish to calibrate their decision priors from data. Here we describe one calibration method. Suppose the researcher has access to a small number of gold standard (GS) crowdsourcing tasks, where the true labels \(z_j\) for all workers, the researcher can sample from each distribution and aggregate the samples, \(\tilde{p}_j\) and \(\hat{S}^2_j\), for these items are known. We propose to apply the crowdsourcing algorithm to these gold standard items in order to estimate its accuracy, then calibrate the decision priors accordingly.

Suppose there are \(n_0\) GS tasks with true label \(z = 0\) and \(n_1\) GS tasks with true label \(z = 1\). Let \(m_{00}\) be the number of \(n_0\) tasks decided by the algorithm to be label zero \((d = -1)\) and let \(m_{01}\) be the number of \(n_0\) tasks decided to be label one \((d = 1)\) (with \(m_{00} + m_{01} = n_0\)). Likewise, define \(m_{11}\) and \(m_{10}\) for the \(n_1\) tasks (with \(m_{11} + m_{10} = n_1\)). Now, consider the \(d = -1\) prior. We argue that the prior probability for \(p < 1/2\) should be \(m_{00}/n_0\). (In practice, these counts \(m\) will likely need to be smoothed to avoid zero counts; see below.) Likewise, for \(d = 1\), the prior probability for \(p > 1/2\) should be \(m_{11}/n_1\). For a beta \(d = -1\) decision prior, we can thus choose the Beta parameters to satisfy

\[
\frac{m_{00}}{n_0} = \int_0^{1/2} D_{-1}(p \mid a_{-1}, b_{-1}) \, dp, \tag{7}
\]

where \(D_{-1}\) is the beta prior for \(d = -1\) with parameters \(a_{-1}\) and \(b_{-1}\). This integral is the regularized incomplete Beta function \(I_{1/2}(a_{-1}, b_{-1})\). Choosing \(a_{-1} = 1\) reduces this to \(I_{1/2}(1, b_{-1}) = 1 - (1/2)^{b_{-1}}\), giving a calibrated \(b_{-1}\) of

\[
b_{-1} = \log_2 \left( \frac{n_0}{m_{01}} \right). \tag{8}
\]

Likewise, to calibrate the \(d = 1\) decision prior \(D_1 = \text{Beta}(a_1, b_1)\), choose parameters \(a_1\) and \(b_1\) that satisfy

\[
\frac{m_{11}}{n_1} = 1 - \int_0^{1/2} D_1(p \mid a_1, b_1) \, dp. \tag{9}
\]

Again, choosing \(b_1 = 1\) for simplicity gives a calibrated \(a_1 = \log_2 (n_1/m_{10})\).

These choices of prior parameters \((1, b_{-1})\) and \((a_1, 1)\) now serve to calibrate the decision priors given the performance on the GS tasks. Of course, not all problems have corresponding GS tasks (indeed, we do not use GS tasks in this work), but for those problems where such tasks are available, this method, which is applicable beyond the case of \(x = p\) that we focus on, provides a simple means to inform the prior before continuing on to the main set of tasks.

**Generalizations and other applications of DEPS.** DEPS is a general-purpose approach to inference about problems when using efficient allocation strategies. Although we focus our study on inferring the distribution of task difficulties, here we briefly discuss other applications of DEPS.

One application of DEPS is to infer worker completion times\(^{26-28}\) the time a worker needs to finish a given task. If workers are consistently given easy tasks only, then we may have a biased representation of how quickly they can complete work. We can use DEPS to infer completion times as follows. Let \(S_{ij}\) be the completion times for tasks \(i = 1, \ldots, N\) and workers \(j = 1, \ldots, M\), e.g., the number of seconds they took to complete tasks. Distributions of human interevent times are often modeled with a log-normal, \(\ln(S_{ij}) \sim \mathcal{N}(\mu_d, \tau_j)\)^{29}. We establish conjugate priors, \(\mu_j \mid \tau_j, d \sim \mathcal{N}(\mu_d, \tau_j)\) and \(\tau_j \mid d \sim \text{Gamma}(\alpha_d, \beta_d)\), with \(\alpha_d, \beta_d > 0\). The posterior for \(\mu_j\) and \(\tau_j\) is

\[
P(\mu_j, \tau_j \mid d, \{S_{ij}\}) \propto \prod_{i \in \tilde{i}_j} P(S_{ij} \mid \mu_j, \tau_j, d_i) P(\mu_j \mid \tau_j, d_i) P(\tau_j \mid d_i). \tag{10}
\]

This model yields an estimated distribution for completion time for each worker. To understand the distribution for all workers, the researcher can sample from each distribution and aggregate the samples, \(\tilde{\mu}_j\) and \(\tilde{\tau}_j\). The distribution \(P(\tilde{\mu}, \tilde{\tau}) \approx P(\mu, \tau)\), the true, overall distribution of worker completion times. Assuming that completion
times and task difficulty are related, we can reflect the decision \( d_i \) of the algorithm by varying the parameters of the priors for \( \mu_i \) and \( \tau_i \) given \( d_i \). For example, a normal prior with a larger mean may be appropriate for a task that has been undecided \((d_i = 0)\), while decided tasks \((d_i \neq 0)\), should have a prior with a smaller mean.

In addition to worker completion times, DEPS can also consider task completion times. Since the tasks are not randomly shown to the workers the data collected on task completion time will also be biased. For this application of DEPS, we again model the time to complete tasks, \( S_{ij} \), as a log-normal, \( \ln(S_{ij}) \sim \mathcal{N}(\mu_i, \tau_i) \) and establish priors \( \mu_i \ | \ \tau_i, d_i \sim \mathcal{N}(\mu_{d_i}, \tau_i) \) and \( \tau_i \ | \ d_i \sim \text{Gamma}(\alpha_d, \beta_d) \) with \( \alpha_d, \beta_d > 0 \). The posterior for \( \mu_i \) and \( \tau_i \) of task \( i \) is

\[
P(\mu_i, \tau_i \ | \ d_i, (S_{ij})) \propto \prod_{j \in i} P(S_{ij} \ | \ \mu_i, \tau_i, d_i)P(\mu_i \ | \ \tau_i, d_i)P(\tau_i \ | \ d_i).
\]

(11)

The priors are then adjusted to reflect the decisions \( d_i \) in a manner similar to the worker completion time model given above, as is the inference of the overall distribution of task completion times.

Even further generalizations of DEPS are possible. Briefly, suppose a crowdsourcer is interested in understanding the behavioural traces of workers as they complete tasks. One approach is to use observed worker dynamics to determine an “embedding” vector for each worker. These vectors can then be used to predict worker features or activities\(^{39}\). Let \( u_{ij} = x_i + e_j \) be a \( k \)-dimensional data vector representing the behavioural trace data for worker \( j \) responding to task \( i \), where \( x_i \) is the population data for that task and \( e_j \) is the worker-specific deviations from the population. Then, let \( w_j \) be a \( k \)-dimensional unit vector for embedding worker \( j \). This embedding vector and its estimator \( \hat{w}_j \) can capture the various contributions of the behavioural trace data to the worker’s labeling decision: the worker responds with \( y_{ij} = 1 \) if the projection \((u_{ij}, \hat{w}_j) \geq \tau_j\), otherwise \( y_{ij} = 0 \). Inference of this vector can be performed given priors for the parameters (such as \( \tau_j \)) and \( u_{ij} \), parameterized by a multivariate Gaussian distribution, for example. The allocation strategy’s decisions can then be incorporated by defining suitable decision priors for \( u_{ij}, \tau_j \), and so forth, conditioned on \( d_i \).

While only brief outlines, the above applications of DEPS indicate that suitable conditioning of statistical models, even complex ones, using the decision variables, has the potential to improve the information attainable about the crowdsourcing problem at hand.

**Materials and methods**

In this section, we describe the real-world crowdsourcing datasets we analyze (“Datasets” section), the synthetic crowdsourcing we simulate (“Crowdsourcing simulations” section), and how DEPS is applied including the details of the efficient allocation framework we use (“Allocation method” section). We focus on using DEPS to infer the task difficulty distribution \( P(p) \), where \( p \) is the probability a worker response \( y_{ij} = 1 \); tasks with \( p \approx 1/2 \) are difficult in that it takes many worker responses to accurately distinguish if \( z = 1 \) or \( z = 0 \). See “Decision-explicit probability sampling (DEPS)” section for a general specification of DEPS along with examples for inferring properties other than \( p \). We also describe a traditional “baseline” method to compare DEPS (“Baseline method—Wald estimation” section) to, and we describe our quantitative measures of evaluation for these methods (“Evaluating performance” section). The results of our experiments are presented in “Experiments” section.

**Datasets.** We study three crowdsourcing datasets. These data were not generated using an efficient allocation algorithm, and so it has become standard practice to evaluate such algorithms with these data\(^{10,21}\)—since labels were collected independently, one can use an allocation algorithm to choose what order to reveal labels from the full set of labels, essentially “rerunning” the crowdsourcing after the fact. Following Li et al.\(^{21}\), we only utilize at most 50% of the total responses available so that the allocation algorithm does not “run out” of requested labels as it polls the data.

**RTE** Recognizing Textual Entailment (RTE) dataset\(^4\). Pairs of written statements were taken from the PASCAL RTE-1 data challenge\(^{31}\) and shown to Amazon Mechanical Turk workers who responded whether or not one statement entailed the other. These data consist of \( N = 800 \) binary tasks each of which received 10 labels from workers, giving a total of 8,000 responses. RTE is available at https://sites.google.com/site/nlpcommittee/.

**Bluebirds Identifying Bluebirds** dataset\(^{30}\). Each task is a photo of a bluebird, either an Indigo Bunting or a Blue Grosbeak, and the worker is asked if the photo contains an Indigo Bunting. There are \( N = 108 \) binary tasks, and 39 responses for each tasks, therefore, a total of 4,212 responses. Bluebirds is available at https://github.com/welinder/cubam.

**Relevance** Identifying Page Relevance dataset\(^{32}\). Each task displays a webpage and a given topic, the worker is asked to determine if the webpage is relevant to the given topic. This dataset contains \( N = 2,275 \) tasks, with a range of 1–10 responses per task. There is on average 6.04 responses per task. In total there are 13,749 responses. Relevance is available at https://sites.google.com/site/trecrowd/2011 (Task 2 test set).

When applying DEPS to infer the distributions of task difficulty \( p \) for these datasets, we used beta decision priors with parameters \( \Phi_{-1} = (1, 2) \), \( \Phi_0 = (2, 2) \), and \( \Phi_1 = (2, 1) \).

**Allocation method.** We used the Requallo allocation algorithm to choose which task labels are received, either from the real datasets (“Datasets” section) or the simulated crowdsourcing (“Crowdsourcing simulations” section). We chose for completeness a ratio requirement with \( \epsilon = 4 \) (“Efficient allocation methods” section). All other details were as those given by Li et al.\(^{21}\). While other allocation methods and parameter choices are worth
exploring (see also Discussion), here this method and parameter choices were held fixed so that post-allocation inference methods DEPS and the baseline method (see below) are always compared on the same collected data.

**Baseline method—Wald estimation.** To understand the performance of DEPS, we compare to the following baseline method, known as Wald estimation, which is a conventional approach to this inference problem. Wald estimation uses the MLE estimator for $p$: for each task $i$, $\hat{p}_i = a_i / n_i$, where $a_i$ is the number of positive responses to $i$ from workers and $n_i$ is the total number of responses from workers. The $\hat{p}$ are then used directly to approximate the distribution $\Pr(p) \approx \Pr(\hat{p})$ and estimate the beta parameters using Eqs. (3)–(5) or Eq. (6).

Unlike DEPS (“Decision-explicit probability sampling (DEPS)” section), this baseline estimation using $\hat{p}$ suffers from small-data problems. Specifically, if either $a_i = 0$ or $a_i = n_i$, then the likelihood used in MLE is undefined. Yet, either situation is likely when only a few labels are collected for task, which can often occur when using an efficient allocation strategy (see Fig. 1b). A common solution to this problem is to use a smoothed estimate of $\hat{p}$, where $a_i, b_i$, and $n_i$ are replaced with $a_i + \varepsilon, b_i + \varepsilon$, and $n_i + 2\varepsilon$, respectively (we use $\varepsilon = 1$). Unfortunately, while smoothing to this degree using “pseudo-labels” is common, such a level of smoothing may overly bias our Wald estimates of $p$ towards $\hat{p} = 1/2$, so we also explore an alternative solution: the values of $\hat{p}$ are transformed using $(\hat{p} N - 1 + 1/2) / N$ such that now $\hat{p} \in (0, 1)$ instead of $\hat{p} \in [0, 1]^{33}$. For these two procedures, when fitting to $\hat{p}$, we found better parameter estimates when using MLE for smoothed Wald and Method of Moments for transformed Wald. DEPS, in contrast, is not affected by this small-data problem.

**Crowdsourcing simulations.** We wish to supplement our results using real crowdsourcing with controlled simulations of crowdsourcing problems. To generate synthetic datasets according to the crowdsourcing model defined in “Background” section, we assume each task $i$ has an intrinsic parameter $p_i$ with each worker response to task $i$ as a Bernoulli variable with parameter $p_i$. This probability governs how difficult a task is in terms of how many responses are necessary to determine its label: the closer $p$ is to 1/2 the more labels are necessary to accurately distinguish $z = 0$ from $z = 1$. We further endow the model with a prior probability distribution on $p_i$, specifically $p \sim \text{Beta}(\alpha, \beta)$, a beta distribution with hyperparameters $\alpha$ and $\beta$. This prior distribution lets us generate $N$ tasks and control the difficulty of each by determining how many tasks are near $p \approx 1/2$ and how many tasks are near $p \approx 0$ or $p \approx 1$. Conversely, statistical inference can be performed to determine the posterior distribution of $p$ given the data $\{y_i\}$. Using this simulation model we can implement efficient budget allocation techniques such as Requallo in order to study the effect of efficient allocation on the distribution $\Pr(p)$. Unlike with the real data, the true distribution of $p$ is known, and we can test inference methods by comparing their estimates to the true distribution.

To apply DEPS to simulated data, we used decision prior parameters $\Phi_{-1} = (1, 5)$, $\Phi_0 = (5, 5)$, and $\Phi_1 = (5, 1)$. These are more confident priors than the ones used with real data (“Datasets” section); see “Discussion” section for further discussion.

**Evaluating performance.** We use an information-theoretic measure to quantitatively compare the distribution $\Pr(p)$ of task difficulty $p$ found under various conditions. For synthetic datasets we have imposed the ground truth distribution of $p$, so we can compare our inferred estimates to this known ground truth. The ground truth distributions are not available to us when examining the real datasets, so instead we compare the estimated $\Pr(p)$ found with a biased portion of the data revealed using Requallo with the estimated $\Pr(p)$ found using all the data, which were collected in an unbiased manner.

The Kullback–Leibler (KL) divergence (or relative entropy) between two random variables $X$ and $Y$ (measured in ‘nats’) is

$$D_{\text{KL}}(X, Y) = \int_{-\infty}^{\infty} f_X(x; \theta) \ln \left( \frac{f_X(x; \theta)}{f_Y(x; \theta)} \right) dx$$

(12)

$$= \ln \left( \frac{B(\alpha', \beta')}{\overline{B(\alpha, \beta)}} \right) + (\alpha - \alpha') \psi(\alpha) + (\beta - \beta') \psi(\beta) + (\alpha' - \alpha + \beta' - \beta) \psi(\alpha + \beta)$$

(13)

where $f_X$ ($f_Y$) is the density for $X$ ($Y$), and the second line holds for the case where $X$ and $Y$ are both beta-distributed: $X \sim \text{Beta}(\alpha, \beta)$ and $Y \sim \text{Beta}(\alpha', \beta')$. $\overline{B}$ is the Beta function, and $\psi$ is the digamma function. We utilize Eq. (13) to measure how well our inference method estimates the underlying prior beta distribution of $p$. If the distributions of $p$ are not Beta, or one is interested in a different property than $p$, one can still use the general Eq. (12), perhaps with an appropriate sample estimator for the KL-divergence$^{44}$. We take $X$ to be the ground truth distribution for $p$ (synthetic data) or the distribution estimated using the full, unbiased data (real data), and $Y$ to be the distribution inferred using DEPS on Requallo-collected data.

**Experiments**

We divide our experiments into those analyzing real-world data (“Real-world data” section, Figs. 3, 4 and 5, and Table 1) and those analyzing synthetic crowdsourcing simulations (“Synthetic data” section and Fig. 6).

**Real-world data.** Figure 3 considers estimation of the difficulty distribution $\Pr(p)$ when limited to 25% of the available budget. For the Wald baseline method, Fig. 3 show how efficient allocation biases the estimated $\hat{p}$ towards 1/2 in the “smoothed” case and toward 0 or 1 in the “transformed” case. In contrast to Wald, which
Figure 3. Inference of problem difficulty distributions using 25% of labels. Tasks are allocated using Requallo, then Wald estimation or DEPS estimation of problem difficulty is performed. These results are compared to the distribution of \( \hat{p} \) estimated using the full dataset (100% of labels), show as histograms. The “smoothed” and “transformed” variants of Wald estimation tend to add too much probability near either the center (\( p = 1/2 \)) or extremes (\( p = 0, 1 \)). DEPS achieves good agreement with the full data distribution.

Figure 4. Convergence of estimated distributions as more crowd data are used. Visually, we see that DEPS and Wald (transformed) both converge relatively quickly, often after using only 10% of the available data.

Figure 5. DEPS generally provides more information (lower KL-divergence) about the distribution estimated using the full, unbiased dataset than Wald, whose performance degrades as more (biased) data are received, with the exception of Bluebirds. Shaded areas denote 95% CI computed over Requallo realizations.
requires either smoothed or transformed point estimates, DEPS shows good qualitative agreement with the full data, and fits the unmodified data, making it more robust than Wald to the bias from efficient allocation.

Expanding upon Fig. 3 we now investigate in Fig. 4 how the estimates of $\Pr(p)$ change as more budget is made available to the crowdsourcer, up to 50% of the total number of labels provided by the data. We observe in both Wald and DEPS that the estimated distributions converge quickly, often with as little as 10% of available data.

Expanding on the distribution convergence, Fig. 5 shows how much information about the distribution of $p$ given by the full (unbiased) data is provided by DEPS and Wald as more biased data are made using Requallo. With the exception of the Relevance dataset, where performance is relatively comparable, DEPS provides more information as evidenced by the lower KL-divergence (“Evaluating performance” section). Interestingly, Wald shows performance that degrades slightly with more data in Fig. 5a and b, likely due to the fact that, while more data are available, there is more biased data available, as these responses are gathered in a non-uniform manner due to Requallo. DEPS, however, does not exhibit this degraded performance on these data. We discuss this further in “Discussion” section.

| Dataset    | Method          | Estimates from biased 50% data | Full data |
|------------|-----------------|-------------------------------|-----------|
|            |                 | $\alpha$ [95% CI] | $\beta$ [95% CI] | $\alpha$ | $\beta$ |
| RTE        | Wald (smoothed) | 3.04 [2.83, 3.35] | 2.41 [2.28, 2.63] | 1.24 | 0.92 |
|            | Wald (transformed) | 0.88 [0.78, 1.05] | 0.62 [0.53, 0.73] | 1.24 | 0.92 |
|            | DEPS | **1.43** [1.27, 1.63] | **1.12** [0.97, 1.27] | 1.24 | 0.92 |
| Bluebirds  | Wald (smoothed) | **2.02** [1.62, 2.30] | **3.09** [2.65, 3.69] | 2.25 | 3.66 |
|            | Wald (transformed) | 0.46 [0.31, 0.61] | 0.88 [0.63, 1.19] | 2.25 | 3.66 |
|            | DEPS | 1.00 [0.78, 1.25] | 1.56 [1.28, 2.07] | 2.25 | 3.66 |
| Relevance  | Wald (smoothed) | 4.13 [3.98, 4.26] | 2.79 [2.70, 2.86] | 1.51 | 0.72 |
|            | Wald (transformed) | 0.98 [0.93, 1.03] | 0.49 [0.46, 0.52] | 1.51 | 0.72 |
|            | DEPS | **1.53** [1.53, 1.61] | **1.00** [0.95, 1.06] | 1.51 | 0.72 |

Table 1. Parameter estimates using a biased sample compared to parameters inferred using the full, unbiased dataset. Bold denotes sample estimates that are closest to estimates from the full data.

Figure 6. The KL-divergence (or relative entropy) $D_{KL}$ between the inferred and true distributions of $p$ over a range of parameters for the true distribution. Here we see that DEPS outperforms the Wald baselines for most parameter values, achieving a lower divergence in its estimates of the true distribution. Panel d shows the mean KL-divergence averaged over the values of $\alpha$ and $\beta$ in the matrices shown in panels a–c.
Lastly, to further compare DEPS and Wald estimates on the real-world datasets, Table 1 shows the estimated parameters $\theta = (\alpha, \beta)$ for $\Pr(p)$. We applied both methods to 50% of the data gathered using Requallo (and the bias entailed by Requallo). For comparison, we also report $\theta$ as estimated using the full, unbiased data. Although there remains room for improvement, Table 1 shows that DEPS achieves estimates of $\theta$ closer to estimates from the unbiased data than Wald does in all cases.

**Synthetic data.** Supplementing our results using real datasets, we also explored the performance of DEPS and the Wald baseline method for crowdsourcing problems where the true distribution $\Pr(p)$ is known. Figure 6 shows the KL-divergence $D_{KL}$ between estimated $P_t(p)$ and the true distribution across a range of parameter values $(\alpha, \beta)$. Across most parameters, except for some cases with extremely small values of $\alpha$ or $\beta$, DEPS provides more information (lower $D_{KL}$) than Wald does about the true distribution. (Note the logarithmic color scale for $D_{KL}$ used in Fig. 6a–c.) Taken together, DEPS is able to generate more information about the underlying distribution $\Pr(p)$ than baseline methods even when baseline methods use the same efficient allocation algorithm.

**Discussion**

Efficient crowd allocation methods such as Requallo can minimize the costs of crowdsourcing, but we showed that this efficiency comes at a cost: the data collected are now biased. Indeed, in our results we showed that Requallo completed far more easy tasks and far fewer hard tasks, in terms of reaching a consensus from a few responses as possible. For some tasks this bias may not be a concern, but many tasks, such as studying the behavior of the crowd as they perform the work, may be harmed by this bias. Even the common task of using crowdsourcing to gather machine learning training data may be hindered by an allocation method: images, for example, where the crowd most disagrees about the label are some of the most useful examples when training an image classifier, and so if anything more data should be collected for those images, not less.

Our experiments ("Experiments" section) provide evidence supporting the performance of DEPS, both its accuracy and its efficiency, but more work is warranted in several directions. While we generally found that DEPS provided the best performance, this was not always the case: smoothed Wald performed best on the Bluebirds data and so it is worth exploring which methods are most appropriate to which data and when. Likewise, we focussed our validation procedure around a single inference task, estimating the difficulty distribution for a crowdsourcing problem, and a single allocation method, Requallo. Future work should consider other problem properties, such as better understanding completion times or behavioural traces of works, as we briefly sketched out in "Generalizations and other applications of DEPS" section, as well as better exploring how well DEPS can work with allocation methods besides Requallo and how well DEPS compares with (or can be combined with) other inferential methods.\textsuperscript{35,36} DEPS also works using an informative “decision” prior, and this introduces subjectivity into the property inference. We provided some guidance for calibrating decision priors, but there remains considerable researcher flexibility. Indeed, we found variations in DEPS performance across the parameter space in Fig. 6 that imply care may be needed when matching decision priors to specific problems. A systematic study of choosing and tuning decision priors efficiently is warranted.

It is possible that the performance of DEPS can degrade as more data are collected. As more labels are received overall, more difficult tasks are more likely to reach the completeness requirement, pushing the “horizon” of completeness towards $p = 1/2$. In this case, we may want to flatten our priors accordingly. In other words, we motivated the choice of steep decision priors in “Decision-explicit probability sampling (DEPS)” section as representing the confidence that we have in the algorithm’s decisions, but, in fact, that steepness is also related to the budget available to the crowdsourcer. Incorporating a budget dependency into the design of the decision priors is therefore an important avenue for improvement, particularly for a large-scale deployment of DEPS. This would be especially interesting for crowdsourcing problems designed to distribute the crowd non-uniformly, deploying more workers in some areas of the problem space than others and adapting (perhaps dynamically) the DEPS decision priors accordingly.

**Conclusion**

In this work, we introduced Decision Explicit Probability Sampling (DEPS), a flexible approach to estimating a distribution when given biased data. By explicitly incorporating an allocation method’s decisions into prior distributions, DEPS can adjust for the bias induced by the allocation method. Using Requallo as an exemplar allocation strategy, DEPS estimates a more accurate distribution for the true, unknown property distribution than if estimation were performed using traditional methods. This allows researchers to still collect data from the crowd efficiently, while being able to extract better information about more than just the correct label for a task.

In summary, DEPS provides a method to account for the bias introduced by efficient allocation algorithms in order to better understand properties of interest for a crowdsourcing problem. The researcher can apply DEPS to problems of interest by adapting the decision priors to reflect how the allocation method introduces bias. The flexibility of DEPS allows for researchers to implement DEPS for a variety of research questions, such as understanding problem difficulty, worker completion times, and behavioral traces. Researchers can implement efficient crowdsourcing while performing accurate inference about the distribution of interest, showing that DEPS can help address one of the key challenges of crowdsourcing: maximizing the information gained from finite, and often costly to gather, data.
References

1. Howe, J. The rise of crowdsourcing. Wired Mag 14, 1–4 (2006).
2. Brabham, D. C. Crowdsourcing as a model for problem solving: An introduction and cases. Convergence 14, 75–90 (2008).
3. Kittur, A. et al. The future of crowd work. In Proceedings of the 2013 Conference on Computer Supported Cooperative Work 1301–1318 (ACM, 2013).
4. Snow, R., O’Connor, B., Jurafsky, D. & Ng, A. Y. Cheap and fast—but is it good?: Evaluating non-expert annotations for natural language tasks. In Proceedings of the Conference on Empirical Methods in Natural Language Processing 254–263 (Association for Computational Linguistics, 2008).
5. Yan, Y., Rosales, R., Fung, G. & Dy, J. I. Active learning from crowds. ICML. 11, 1161–1168 (2011).
6. Kamar, E., Hacker, S. & Horvitz, E. Combining human and machine intelligence in large-scale crowdsourcing. In Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems—Volume 1 467–474 (International Foundation for Autonomous Agents and Multiagent Systems, 2012).
7. Behrend, T. S., Sharek, D. J., Meade, A. W. & Wiebe, E. N. The viability of crowdsourcing for survey research. Behavior Res. Methods 43, 800 (2011).
8. Bongard, J. C., Hines, P. D., Conger, D., Hurd, P. & Lu, Z. Crowdsourcing predictors of behavioral outcomes. IEEE Trans. Syst. Man Cybern. Syst. 43, 176–185 (2013).
9. Salganik, M. J. & Levy, K. E. Wiki surveys: Open and quantifiable social data collection. PLOS ONE 10, e0123483 (2015).
10. McAndrew, T. C., Guseva, E. A. & Bagrow, J. P. Reply & supply: Efficient crowdsourcing when workers do more than answer questions. PLOS ONE 12, e0182662 (2017).
11. Waggy, M. D., Bongard, J. C., Bagrow, J. P. & Hines, P. D. Crowdsourcing predictors of residential electric energy usage. IEEE Syst. J. 1, 1–10 (2017).
12. Berenberg, D. & Bagrow, J. P. Efficient crowd exploration of large networks: The case of causal attribution. Proc. ACM Hum. Comput. Interact. 2, 24 (2018).
13. García-Molina, H., Joglekar, M., Marcus, A., Parameswaran, A. & Verroios, V. Challenges in data crowdsourcing. IEEE Trans. Knowl. Data Eng. 28, 901–911 (2016).
14. Zheng, H., Li, D. & Hou, W. Task design, motivation, and participation in crowdsourcing contests. Int. J. Electron. Commerce 15, 57–88 (2011).
15. Kaufmann, N., Schulze, T. & Veit, D. More than fun and money: worker motivation in crowdsourcing—a study on mechanical turk. In AMCIS Vol. 11, 1–11 (Detroit, Michigan, USA, 2011).
16. Evkoff, C. & de Vries, A. P. Increasing the realism of crowdsourcing tasks. Inform. Retriev. 16, 121–137 (2013).
17. Sheng, V. S., Provost, F. & Ipeirotis, P. G. Get another label! Improving data quality and data mining using multiple, noisy labelers. In Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining 614–622 (ACM, 2008).
18. Karger, D. R., Oh, S. & Shah, D. Iterative learning for reliable crowdsourcing systems. In Advances in Neural Information Processing Systems 1953–1961 (2011).
19. Karger, D. R., Oh, S. & Shah, D. Budget-optimal task allocation for reliable crowdsourcing systems. Oper. Res. 62, 1–24 (2014).
20. Oyama, S., Baba, Y., Sakurai, Y. & Kashima, H. Accurate integration of crowdsourced labels using workers’ self-reported confidence scores. In IJCAI 2554–2560 (2013).
21. Li, Q., Ma, F., Gao, J., Su, L. & Quinn, C. J. Crowdsourcing high quality labels with a tight budget. In Proceedings of the Ninth ACM International Conference on Web Search and Data Mining 237–246 (ACM, 2016).
22. Dawid, A. P. & Skene, A. M. Maximum likelihood estimation of observer error-rates using the EM algorithm. Appl. Stat. 20–28 (1979).
23. Dow, S., Kulkarni, A., Klemmer, S. & Hartmann, B. Shepherding the crowd yields better work. In Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work 1013–1022 (ACM, 2012).
24. Rzeszotarski, J. M. & Kittur, A. Instrumenting the crowd: Using implicit behavioral measures to predict task performance. In Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology 13–22 (ACM, 2011).
25. Chen, X., Lin, Q. & Zhou, D. Optimistic knowledge gradient policy for optimal budget allocation in crowdsourcing. In International Conference on Machine Learning 64–72 (2013).
26. Difallah, D. E., Catasta, M., Demartini, G., Ipeirotis, P. G. & Cudré-Mauroux, P. The dynamics of micro-task crowdsourcing: The case of amazon mturk. In Proceedings of the 24th International Conference on World Wide Web, WWW ’15 238–247. https://doi.org/10.1145/2736277.2741685 (International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 2015).
27. Difallah, D., Checco, A., Demartini, G. & Cudré-Mauroux, P. Deadline-aware fair scheduling for multi-tenant crowd-powered systems. Trans. Soc. Comput. 2, https://doi.org/10.1145/3301003 (2019).
28. Han, L. et al. The impact of task abandonment in crowdsourcing. IEEE Trans. Knowl. Data Eng. 33, 2266–2279. https://doi.org/10.1109/TKDE.2019.2948168 (2021).
29. Iribarren, J. L. & Moro, E. Impact of human activity patterns on the dynamics of information diffusion. Phys. Rev. Lett. 103, 038702 (2009).
30. Welinder, P., Branson, S., Perona, P. & Belongie, S. J. The multidimensional wisdom of crowds. In Advances in Neural Information Processing Systems 2424–2432 (2010).
31. Dagan, I., Glickman, O. & Magnini, B. The PASCAL recognising textual entailment challenge. In Machine learning challenges. Evaluating predictive uncertainty, visual object classification, and recognising textual entailment 177–190 (Springer, 2006).
32. Lease, M. & Kazai, G. Overview of the TREC 2011 crowdsourcing track. In Proceedings of the Text Retrieval Conference (TREC) (2011).
33. Smithson, M. & Verkuilen, J. A better lemon squeezer? Maximum-likelihood regression with beta-distributed dependent variables. Psychol. Methods 11, 54 (2006).
34. Pérez-Cruz, F. Kullback–Leibler divergence estimation of continuous distributions. In 2008 IEEE International Symposium on Information Theory 1666–1670 (IEEE, 2008).
35. Han, T., Sun, H., Song, Y., Wang, Z. & Liu, X. Budgeted task scheduling for crowdsourced knowledge acquisition. In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM ’17 1059–1068. https://doi.org/10.1145/3132872.3133002 (Association for Computing Machinery, New York, NY, USA, 2017).
36. Wang, P., Chen, Z. & Zhang, Z. Suitability-based task assignment in crowdsourcing markets. In 2020 IEEE International Conference on Services Computing (SCC) 361–369. https://doi.org/10.1109/SCC49832.2020.00054 (2020).

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. IIS-1447634.
Author contributions
J.B. formulated the project. A.H. and J.B devised the method. A.H. performed simulations and data analysis and prepared the figures. J.B. helped with figures. A.H. and J.B. wrote the main manuscript text. All authors reviewed the manuscript.

Competing interests
The authors declare no competing interests.

Additional information
Correspondence and requests for materials should be addressed to J.B.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher’s note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

© The Author(s) 2022