# Parallel Task Routing for Crowdsourcing

**Jonathan Bragg**
University of Washington (CSE)
Seattle, WA
jbragg@cs.washington.edu

**Andrey Kolobov**
Microsoft Research
Redmond, WA
akolobov@microsoft.com

**Mausam**
Indian Institute of Technology
Delhi, India
mausam@cse.iitd.ac.in

**Daniel S. Weld**
University of Washington (CSE)
Seattle, WA
weld@cs.washington.edu

## Abstract

An ideal crowdsourcing or citizen-science system would route tasks to the most appropriate workers, but the best assignment is unclear because workers have varying skill, tasks have varying difficulty, and assigning several workers to a single task may significantly improve output quality. This paper defines a space of task routing problems, proves that even the simplest is NP-hard, and develops several approximation algorithms for parallel routing problems. We show that an intuitive class of requesters' utility functions is submodular, which lets us provide iterative methods for dynamically allocating batches of tasks that make near-optimal use of available workers in each round. Experiments with live oDesk workers show that our task routing algorithm uses only 48% of the human labor compared to the commonly used round-robin strategy. Further, we provide versions of our task routing algorithm which enable it to scale to large numbers of workers and questions and to handle workers with variable response times while still providing significant benefit over common baselines.

## Introduction

While there are millions of workers present and thousands of tasks available on crowdsourcing platforms today, the problem of effectively matching the two, known as *task routing*, remains a critical open question. Law & von Ahn (2011) describe two modes of solving this problem – the pull and the push modes. In the pull mode, such as on Amazon Mechanical Turk (AMT), workers themselves select tasks based on price, keywords, etc. In contrast, a push-oriented labor market, popular on volunteer crowdsourcing platforms (e.g., Zooniverse (Lintott et al. 2008)), directly allocates appropriate tasks to workers as they arrive. Increasing amounts of historical data about tasks and workers create the potential to greatly improve this assignment process. For example, a crowdsourcing system might give easy tasks to novice workers and route difficult problems to experts.

Unfortunately, this potential is hard to realize. Existing task routing algorithms (e.g., (Donmez, Carbonell, and Schneider 2009; Wauthier and Jordan 2011; Ho and Vaughan 2012; Chen, Lin, and Zhou 2013)) make too restrictive assumptions to be applied to realistic crowdsourcing platforms. For example, they typically assume at least one of the following simplifications: (1) tasks can be allocated purely sequentially, (2) workers are willing to wait patiently for a task to be assigned, or (3) the quality of a worker's output can be evaluated instantaneously. In contrast, an ideal practical task router should be completely unsupervised, since labeling gold data is expensive. Furthermore, it must assign tasks *in parallel to all available workers*, since making engaged workers wait for assignments leads to an inefficient platform and frustrated workers. Finally, the task router should operate in real-time so that workers need not wait. The two latter aspects are especially important in citizen science and other forms of volunteer (non-economically motivated) crowdsourcing.

In this paper we investigate algorithms for task routing that satisfy these desiderata. We study a setting, called JOCR (**Jo**int **Cr**owdsourcing (Kolobov, Mausam, and Weld 2013)), in which the tasks are questions, possibly with varying difficulties. We assume that the router has access to a (possibly changing) set of workers with different levels of skill. The system keeps each available worker busy as long as the worker stays online by assigning him or her questions from the pool. It observes responses from the workers that complete their tasks and can aggregate their votes using majority vote or more sophisticated Bayesian methods in order to estimate its confidence in answers to the assigned questions. The system's objective is to answer all questions from the pool as accurately as possible after a fixed number of allocations.

Even within this setting, various versions of the problem are possible depending upon the amount of information available to the system (Figure 1). In this paper we focus on the case where task difficulties and worker abilities are known a priori. Not only does this setting lay a foundation for the other cases, but it is of immediate practical use. Recently developed community-based aggregation can learn very accurate estimates of worker accuracy from limited interactions (Venanzi et al. 2014). Also, for a wide range of problems, there are domain-specific features that allow a system to roughly predict the difficulty of a task.

We develop algorithms for both offline and adaptive JOCR task routing problems. For the offline setting, we first prove that the problem is NP hard. Our approximation algorithm, $\text{JUGGLER}_{\text{OFF}}$, is based on the realization that a natural objective function for this problem, expected informa-

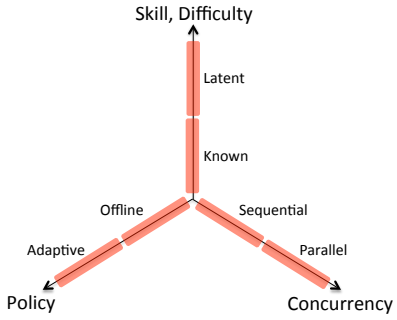



Figure 1: The space of allocation problems.

tion gain, is submodular. This allows JUGGLER$_{\mathrm{OFF}}$ to efficiently obtain a near-optimal allocation. We use similar ideas to devise an algorithm called JUGGLER$_{\mathrm{AD}}$ for the adaptive problem, which achieves even better empirical performance. JUGGLER$_{\mathrm{BIN}}$ is an approximate version of JUGGLER$_{\mathrm{AD}}$, which is able to scale to large numbers of workers and tasks in an efficient manner. Finally, we present JUGGLER$_{\mathrm{RT}}$, which handles workers who take different amounts of time to complete tasks.

We test our algorithms both in simulation and using real data. Experiments with live oDesk workers on a natural language named entity linking task show that JUGGLER$_{\mathrm{AD}}$'s allocation approach achieves the same accuracy as the commonly used round-robin strategy, using only 48% of the labor. Additional experiments on simulated data show that (1) our adaptive method significantly outperforms offline routing, (2) the adaptivity's savings grow when worker skills and task difficulties are more varied, (3) our binned (approximate) adaptive algorithm provides scalability without compromising performance, and (4) our approach yields significant savings even when worker response times vary.

## Problem Definition

In our JOCR model, we posit a set of workers $\mathcal{W}$ and a set of questions $\mathcal{Q}$. Each question $q \in \mathcal{Q}$ has a (possibly latent) *difficulty* $d_q \in [0, 1]$, and each worker $w \in \mathcal{W}$ has a (possibly latent) *skill* parameter $\gamma_w \in (0, \infty)$. While more complex models are possible, ours has the advantage that it is learnable even with small amounts of data. We assume that the probability of a worker $w$ providing a correct answer to a question $q$ is a monotonically increasing function $P(d_q, \gamma_w)$ of worker skill and a monotonically decreasing function of question difficulty. For example, one possible such function, adapted from (Dai, Mausam, and Weld 2010), is

$$P(d_q, \gamma_w) = \frac{1}{2}(1 + (1 - d_q)^{\frac{1}{\gamma_w}}). \quad (1)$$

We assume that JOCR algorithms will have access to a pool of workers, and will be asking workers from the pool to provide answers to the set of questions submitted by a requester. This pool is a subset of $\mathcal{W}$ consisting of workers available at a given time. The pool need not be static — workers may come and go as they please. To formalize the problem, however, we assume that workers disappear or reappear at regularly spaced points in time, separated by a duration equal to the time it takes a worker to answer a question. We call each such a decision point a *round*. At the beginning of every round $t$, our algorithms will assign questions to the pool $P_t \subseteq \mathcal{W}$ of workers available for that round, and collect their responses at the round's end.

The algorithms will attempt to assign questions to workers so as to maximize some utility over a fixed number of rounds, a time *horizon*, denoted as $T$. Since we focus on volunteer crowdsourcing platforms, in our model asking a worker a question incurs no monetary cost. To keep workers engaged and get the most questions answered, our techniques assign a question to every worker in pool $P_t$ in every round $t = 1, \ldots, T$. An alternative model, useful on a *shared* citizen-science platform would allow the routing algorithm a fixed budget of $n$ worker requests over an arbitrary horizon.

## Optimization Criteria

The JOCR framework allows the use of various utility functions $U(S)$ to optimize for, where $S \in 2^{\mathcal{Q}\times\mathcal{W}}$ denotes an assignment of questions to workers. One natural utility function choice is the expected *information gain* from observing a set of worker responses. Specifically, let $\mathcal{A} = \{A_1, A_2, \ldots, A_{|\mathcal{Q}|}\}$ denote the set of random variables over correct answers to questions, and let $\mathcal{X} = \{X_{q,w} \mid q \in \mathcal{Q} \wedge w \in \mathcal{W}\}$ be the set of random variables corresponding to possible worker responses. Further, let $\mathcal{X}_S$ denote the subset of $\mathcal{X}$ corresponding to assignment $S$. We can quantify the uncertainty in our predictions for $\mathcal{A}$ using the joint entropy

$$H(\mathcal{A}) = -\sum_{\mathbf{a}\in \mathrm{dom}\mathcal{A}} P(\mathbf{a}) \log P(\mathbf{a}),$$

where the domain of $\mathcal{A}$ consists of all possible assignments to the variables in $\mathcal{A}$, and $\mathbf{a}$ denotes a vector representing one assignment. The conditional entropy

$$H(\mathcal{A} \mid \mathcal{X}_S) = -\sum_{\substack{\mathbf{a}\ \in\ \mathrm{dom}\mathcal{A},\\ \mathbf{x}\ \in\ \mathrm{dom}\mathcal{X}_S}} P(\mathbf{a}, \mathbf{x}) \log P(\mathbf{a} \mid \mathbf{x})$$

represents the expected uncertainty in the predictions after observing worker votes corresponding to the variables in $\mathcal{X}_S$. Following earlier work (Whitehill et al. 2009), we assume that questions are independent (assuming known parameters) and that worker votes are independent given the true answer to a question, i.e.,

$$P(\mathbf{a}, \mathbf{x}) = \prod_{q\in\mathcal{Q}} P(a_q) \prod_{w\in\mathcal{W} \text{ who answered } q} P(x_{q,w} \mid a_q),$$

where $P(x_{q,w} \mid a_q)$ depends on worker skill and question difficulty as in Equation 1. We can now define the value of a task-to-worker assignment as the expected reduction in entropy

$$U_{IG}(S) = H(\mathcal{A}) - H(\mathcal{A} \mid \mathcal{X}_S). \quad (2)$$

## Problem Dimensions

Given the information gain maximization objective, we consider the problem of finding such an assignment under combinations of assumptions lying along three dimensions (Figure 1):

- **Offline vs. online (adaptive) assignment construction.** In the offline (or *static*) mode, a complete assignment of $T$ questions to each worker is chosen once, before the beginning of the first round, and is not changed as workers provide their responses. Offline assignment construction is the only available option when workers' responses cannot be collected in real time. While we provide an algorithm for the offline case, JUGGLER$_{\text{OFF}}$, it is not our main focus. Indeed, worker responses typically *are* available immediately after workers provide them, and taking them into account allows the router to resolve worker disagreements on the fly. Our experimental results confirm this, with adaptive JUGGLER$_{\text{AD}}$ outperforming the offline algorithm as well as our baselines.
- **Sequential vs. parallel hiring of workers.** A central question in online allocation is the number of workers to hire in a given round. If one could only hire $n$ workers in total, an optimal strategy would ask workers sequentially, one worker per round, because this allows the maximum use of information when building the rest of the assignment. However, this strategy is impractical in most contexts, especially in citizen science — using just a single worker wastes the time of the majority and demotivates them. Therefore, in this paper we consider the harder case of parallel allocation of questions to a *set* of workers. In our case, this set comprises all workers available at the moment of allocation.
- **Known vs. latent worker skill and question difficulty.** Worker skill and question difficulty are critical for matching tasks with workers. Intuitively, the optimal way to use skillful workers is to give them the most difficult tasks, leaving easier ones to the less proficient members of the pool. In this paper, we assume these quantities are known — a reasonable assumption in practice. Since workers and requesters often develop a long-term relationships, EM-based learning (Whitehill et al. 2009; Welinder et al. 2010) can estimate worker accuracy from a surprisingly short record using community-based aggregation (Venanzi et al. 2014).

  Similarly, domain-specific features often allow a system to estimate a task's difficulty. For example, the linguistic problem of determining whether two nouns corefer is typically easy in the case of apposition, but harder when a pronoun is distant from the target (Stoyanov et al. 2009; Raghunathan et al. 2010). In the citizen science domain, detecting planet transits (the focus of the Zooniverse project Planet Hunters) is more difficult for planets that are fast-moving, small, or in front of large stars (Mao et al. 2013).

  If skill or difficulty is *not* known, the router faces a challenging exploration/exploitation tradeoff. For example, it gains information about a worker's skill by asking a question for which it is confident of the answer, but of course this precludes asking a question whose answer it does not know.

## Offline Allocation

We first address the simplest form of the allocation problem, since its solution (and complexity) forms a basis for subsequent, more powerful algorithms. Suppose that an agent is able to assign tasks to a pool of workers, but unable to observe worker responses until they have all been received and hence has no way to reallocate questions based on worker disagreement. We assume that the agent has at its disposal an estimate of question difficulties and worker skills, as discussed above.

**Theorem 1.** *Finding an assignment of workers to questions that maximizes an arbitrary utility function $U(S)$ is NP-hard.*

*Proof sketch.* We prove the result via a reduction from the Partition Problem to offline JOCR. An instance of this problem is specified by a set $X$ containing $n$ elements and a function $s : X \to \mathbb{Z}^+$. The Partition Problem asks whether we can divide $X$ into two subsets $X_1$ and $X_2$ s.t. $\sum_{x \in X_1} s(x) = \sum_{x \in X_2} s(x)$.

We construct an instance of offline JOCR to solve an instance of the Partition Problem as follows. For each element $x_i \in X$, we define a corresponding worker $w_i \in W$ with skill $\gamma_i = s(x_i)/\max_{x_i \in X} s(x_i)$, and let the horizon equal 1. We also define two questions, $q_1$ and $q_2$, with the same difficulty $d$. Let $S_{q,w}$ be an indicator variable that takes a value of 1 iff worker $w$ has been assigned question $q$. Let $f(S, q) = \log\left[\sum_w S_{q,w}\gamma_w + 1\right]$ and define the utility function as $U(S) = \sum_q f(S, q)$.

The solution to a Partition Problem instance is *true* iff $f(S, q_1) = f(S, q_2)$ for an optimal solution $S$ to the corresponding JOCR problem constructed above.

$\Rightarrow$: Suppose that there exist subsets of $X$, $X_1$ and $X_2$, such that the sum of elements in the two sets are equal. Further, suppose for contradiction that the optimal solution $S$ from JOCR assigns sets of workers with *different* sums of skills to work on the two questions. Then we can improve $U(S)$ by making sum of skills equal (by monotonicity and submodularity of logarithm), implying that $S$ was suboptimal, leading to a contradiction.

$\Leftarrow$: Suppose that $S$ is an optimal solution to the JOCR problem s.t. $f(S, q_1) = f(S, q_2)$. Then sums of worker skills (multiplied by the maximum worker skill) for each question are equal and the solution to the Partition Problem is *true*. □

Given the intractability of this problem, we next devise approximation methods. Before proceeding further, we review the combinatorial concept of submodularity. A function $f : 2^{\mathcal{N}} \to R$ is *submodular* if for every $A \subseteq B \subseteq \mathcal{N}$ and $e \in \mathcal{N}$: $f(A \cup \{e\}) - f(A) \geq f(B \cup \{e\}) - f(B)$. Further, $f$ is *monotone* if for every $A \subseteq B \subseteq \mathcal{N}$: $f(A) \leq f(B)$. Intuitively, this is a diminishing returns property; a function is submodular if the marginal value of adding a particular

element never increases as other elements are added to a set.

**Theorem 2.** *The utility function $U_{IG}(S)$ based on value of information defined in the previous section is monotone submodular in S, provided that worker skill and question difficulty are known.*

*Proof sketch.* Given worker skill, question difficulty, and the true answers, worker ballots are independent. We can use the methods of (Krause and Guestrin 2005) to show that expected information gain $U_{IG}(S)$ is submodular and nondecreasing. □

Since the utility function $U_{IG}(S)$ is monotone submodular, we naturally turn to approximation algorithms. Our optimization problem is constrained by the horizon $T$, meaning that we may not assign more than $T$ questions to any worker. We encode this constraint as a *partition matroid*. Matroids capture the notion of linear independence in vector spaces and are specified as $M = (\mathcal{N}, \mathcal{I})$, where $\mathcal{N}$ is the *ground set* and $\mathcal{I} \in 2^{\mathcal{N}}$ are the subsets of elements in $\mathcal{N}$ that are independent in the matroid. A partition matroid is specified with an additional constraint. If we partition $\mathcal{N} = \mathcal{Q} \times \mathcal{W}$ into disjoint sets $B_w = \{w\} \times \mathcal{Q}$ corresponding to the set of possible assignments for each worker $w$, we can construct the desired partition matroid by defining an independent set to include no more than $T$ elements from each of these sets.

JUGGLER$_{\text{OFF}}$ uses the greedy selection process shown in Algorithm 1, which is guaranteed to provide a 1/2-approximation for monotone submodular optimization with a matroid constraint (Nemhauser, Wolsey, and Fisher 1978), yielding the following theorem:

**Theorem 3.** *Let $S^*$ be a question-to-worker assignment with the highest information gain $U_{IG}$, and let $S$ be the assignment found by maximizing $U_{IG}$ greedily. Then $U_{IG}(S) \geq \frac{1}{2} U_{IG}(S^*)$.*

JUGGLER$_{\text{OFF}}$ improves efficiency by making assignments separately for each worker (exploiting the fact that Equation 1 is monotonically increasing in worker skill) and by using lazy submodular evaluation (Krause and Golovin 2014). Interestingly, while the greedy algorithm dictates selecting workers in order of descending skill in Algorithm 1, we observe drastic improvement in our empirical results from selecting workers in reverse order; our implementation sorts by increasing worker skill, which prioritizes assigning easier questions to the less-skilled workers. We have simplified the utility computation of $\Delta X$ by observing that entropy decomposes by question in our model. We note that more involved algorithms can improve the theoretical guarantee to $(1 - 1/e)$ (Vondrak 2008; Filmus and Ward 2012; Badanidiyuru and Vondrak 2014), but given the large positive effect of simply reversing the order of workers, it is unlikely that these methods will provide significant benefit.

Since we are motivated by the citizen science scenario, we have not included cost in the agent's utility function. However, we note that our utility function remains submodular (but not longer monotone) with the addition of a linear cost term. Thus, one can formulate the optimization problem as non-monotone submodular optimization, for which algorithms exist.

**Algorithm 1** The JUGGLER$_{\text{OFF}}$ algorithm

**Input:** Workers $\mathcal{W}$, prior $P(\mathbf{a})$ over answers, unobserved votes $\mathcal{X}_R$, horizon $T$, initial assignment $S$
**Output:** Final assignment $\mathcal{S}$ of questions to workers
**for** $w$ **in** sorted($\mathcal{W}$, key $= \gamma_w$) **do**
  **for** $i = 1$ **to** $T$ **do**
    $\mathcal{X}_w \leftarrow \{X_{q,w'} \in \mathcal{X}_R \mid w' = w\}$
    **for** $X_{q,w} \in \mathcal{X}_w$ **do**
      $\Delta X \leftarrow H(A_q \mid \mathbf{x}) - H(A_q \mid X_{q,w}, \mathcal{S}, \mathbf{x})$
    **end for**
    $X^* \leftarrow \arg\max\{\Delta X : X \in \mathcal{X}_w\}$
    Set $\mathcal{S} \leftarrow \mathcal{S} \cup \{X^*\}$ and $\mathcal{X}_R \leftarrow \mathcal{X}_R \setminus \{X^*\}$
  **end for**
**end for**

## Adaptive (Online) Allocation

JUGGLER$_{\text{OFF}}$ performs a complete static allocation and does not adapt based on worker response. However, in most crowdsourcing platforms, we have access to intermediate output after each worker completes her attempt on a question; it makes sense to consider all previous responses during each round, when assigning tasks. Intuitively, the router may choose to allocate more workers to a question, even if easy, that has generated disagreement. In this section we describe our algorithms for the adaptive setting. We believe that these are the first adaptive task allocation algorithms for crowdsourcing to engage all available workers at each time step.

Unlike in the offline setting, the objective of an optimal algorithm for the online setting is to compute *an adaptive way of constructing assignment*, not any fixed assignment per se. Formally, this problem can be seen as a Partially Observable Markov Decision Process (POMDP) with a single (hidden) state representing the set of answers to all questions $q \in \mathcal{Q}$. In each round $t$, the available actions correspond to possible allocations $S_t \in 2^{\mathcal{Q} \times P_t}$ of tasks to all workers in the pool $P_t \subseteq \mathcal{W}$ s.t. each worker gets assigned only one task, the observations are worker responses $\mathcal{X}_{S_t}$, and the total planning horizon is $T$. The optimal solution to this POMDP is a policy $\pi^*$ that satisfies, for each round $t$ and the set of observations $\cup_{i=1}^{t-1} \mathbf{x}_i$ received in all rounds up to $t$,

$$\pi^*(\bigcup_{i=1}^{t} \mathbf{x}_i) = \text{argmax}_\pi \mathbb{E}\left[U(\bigcup_{j=t}^{H}\{\mathcal{S}_j \sim \pi(\bigcup_{i=1}^{t-1}\mathbf{x}_i, \bigcup_{i=t}^{j-1}\mathcal{X}_i))\}\right]$$

Existing methods for solving POMDPs apply almost exclusively to linear additive utility functions $U$ and are intractable even in those cases, because the number of actions (allocations) in each round is exponential. Besides, since workers can come and go, the POMDP's action set keeps changing, and standard POMDP approaches do not handle this complication.

**Algorithm 2** The JUGGLER$_{\mathrm{AD}}$ algorithm

**Input:** Workers $\mathcal{W}$, prior $P(\mathbf{a})$ over answers, unobserved votes $\mathcal{X}_R$, horizon $T$
$\mathbf{x} \leftarrow \emptyset$
**for** $t = 1$ **to** $T$ **do**
  $S_t \leftarrow$ call JUGGLER$_{\mathrm{OFF}}(T = 1, S = \emptyset)$
  Observe $\mathbf{x}_{S_t}$ and set $\mathbf{x} \leftarrow \mathbf{x} \cup \mathbf{x}_{S_t}$
  $\mathcal{X}_R \leftarrow \mathcal{X}_R \setminus S_t$
  Update prior as $P(\mathbf{a} \mid \mathbf{x})$
**end for**

Instead, we use a replanning approach, JUGGLER$_{\mathrm{AD}}$ (Algorithm 2), that uses the information gain utility $U_{IG}$ (Equation 2). JUGGLER$_{\mathrm{AD}}$ allocates tasks by maximizing this utility in each round separately given the observations so far, modifies its beliefs based on worker responses, and repeats the process for subsequent rounds. In this case, the allocation for a single round is simply the offline problem from the previous section with a horizon of 1. Thus, JUGGLER$_{\mathrm{AD}}$ uses JUGGLER$_{\mathrm{OFF}}$ as a subcomponent at each allocation step.

Although we can prove theoretical guarantees for the offline problem (and therefore for the problem of making an assignment in each round), it is more difficult to provide guarantees for the $T$-horizon adaptive allocation. A natural analysis approach is to use *adaptive submodularity* (Golovin and Krause 2011), which generalizes performance guarantees of the greedy algorithm for submodular optimization to adaptive planning. Intuitively, a function $f$ is adaptive submodular with respect to probability distribution $p$ if the conditional expected marginal benefit of any fixed item does not increase as more items are selected and their states are observed. Unfortunately, we have the following negative result.

**Theorem 4.** *The utility function $U_{IG}(S)$ based on the value of information is not adaptive submodular even when question difficulties and worker abilities are known.*

*Proof.* In order for $U(S)$ to be adaptive submodular, the conditional expected marginal benefit of a worker response should never decrease upon making more observations. As a counterexample, suppose we have one binary question with difficulty $d = 0.5$ and two workers with skills $\gamma_1 = 1$ and $\gamma_2 \to \infty$. If the prior probability of the answer is $P(A = True) = 0.5$, the expected information gain $H(A) - H(A \mid X_2)$ of asking for a vote from the second worker is initially one bit, since she will always give the correct answer. However, if we observe a vote from the first worker before asking for a vote from the second worker, the expected information gain for the second worker will be less than one bit since the posterior probability $P(A = True)$ has changed. □

While we are unable to prove theoretical bounds on the quality of the adaptive policy in the absence of adaptive submodularity, our experiments in the next section show that JUGGLER$_{\mathrm{AD}}$ uses dramatically less human labor than commonly-used baseline algorithms.

## Scalability

**Assigning at Most One Worker at a Time to a Task** As an adaptive scheduler, JUGGLER$_{\mathrm{AD}}$ must perform allocations quickly for all available workers, so that workers do not abandon the system due to high latency. Computing the incremental value of assigning a worker to a question in a given round is normally very fast, but can become computationally intensive if the system has already assigned many other workers to that question in that round (and has not yet observed their responses). Such a computation requires taking an expectation over a combinatorial number of possible outcomes for this set of votes.

This computational challenge may lead one to wonder whether it is beneficial to restrict the router to assigning each question to *at most one* worker in every round. With this restriction in place, finding a question-to-worker assignment in a given round amounts to constructing a maximum-weight matching in the bipartite graph of questions and available workers (where edge weights give the expected utility gain from the corresponding assignment). This problem looks simpler than what we have discussed above; in fact, theoretically, it is solvable optimally in polynomial time. Moreover, the aforementioned restriction should be moot in practice, because large crowdsourcing systems typically have many more tasks to complete than available workers (i.e., $|\mathcal{W}| \ll |\mathcal{Q}|$). Intuitively, in such scenarios a scheduling algorithm has some flexibility with regard to the exact round in which it schedules any particular task, and hence is rarely forced to allocate a question to several workers in a round.

Experiments presented later in the paper confirm the intuition that the restriction of one worker per question realistically has little impact on the solution quality. We find that assigning multiple workers to a task in a round was beneficial only when the fraction of workers to tasks was at least 1/4. Nonetheless, we have discovered that optimally computing maximum matching is still very intensive computationally. Fortunately, our tests also suggest a cheaper alternative — greedily constructing an approximation of a maximum matching. Doing so (1) does not hurt the solution quality of JUGGLER$_{\mathrm{AD}}$, implying that our greedy algorithm finds a solution that is close to optimal, and (2) despite being in the same theoretical complexity class as the algorithm for finding an exact optimal matching, is much faster in practice, making it more appropriate for the large-scale setting. For these reasons, we do not discuss the exact maximum matching approach further, concentrating instead on the greedy approximate strategy.

**Binning Questions for Faster Performance** When scaling to large numbers of workers and questions, even a greedy strategy that never assigns more than one worker to a task at a time can be prohibitively expensive, as it requires making $O(|\mathcal{Q}|)$ utility computations per worker in each round. However, since the utility of an assignment is a function of the current belief about the true answer, as well as of the question difficulty and worker skill, we can reduce the com-

**Algorithm 3** The $\text{JUGGLER}_{\text{BIN}}$ algorithm

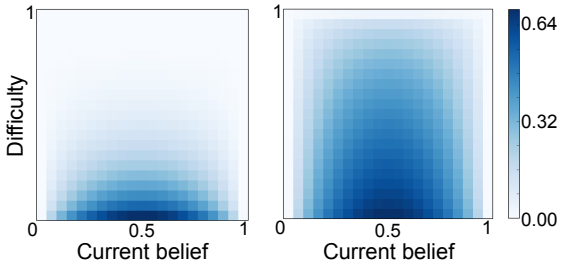
**Input:** Workers $\mathcal{W}$, prior $P(\mathbf{a})$ over answers, unobserved votes $\mathcal{X}_R$, horizon $T$
**function** LOAD($\mathcal{Q}_{in}$)
  **for** $w$ **in** $\mathcal{W}$ **do**
    **for** $q$ **in** $\mathcal{Q}_{\text{in}}$ **do**
      **if** $\mathbf{x}_{q,w} \notin \mathbf{x}$ **then**
        $b \leftarrow$ TO_BIN$(d_q, P(a_q \mid \mathbf{x}))$
        bins$[w][b]$.add$(q)$
      **end if**
    **end for**
  **end for**
**end function**

*# Initialize bins*
bins $\leftarrow \{\}$
**for** $w$ **in** $\mathcal{W}$ **do**
  bins$[w] \leftarrow \{\}$
  **for** $b$ **in** ENUMERATE_BINS() **do**
    bins$[w][b] \leftarrow \emptyset$
    $q' \leftarrow$ temp question with $\overline{d}, \overline{P(a)}$ for bin
    $\Delta\overline{X}_{b,w} \leftarrow H(A_{q'}) - H(A_{q'} \mid X_{q',w})$
  **end for**
**end for**
LOAD($\mathcal{Q}$)

*# Make assignments*
$\mathbf{x} \leftarrow \emptyset$
**for** $t = 1$ **to** $T$ **do**
  $S \leftarrow \emptyset$
  **for** $w$ **in** sorted$(\mathcal{W},$ key $= \gamma_w)$ **do**
    **for** $b$ **in** sorted(bins$[w],$ key $= \Delta\overline{X}_{b,w})$ **do**
      **if** bins$[w][b] \neq \emptyset$ **then**
        $q \leftarrow$ bins$[w][b]$.pop()
        $S \leftarrow S \cup \{X_{q,w}\}$
        **for** $w'$ **in** $\mathcal{W} \setminus \{w\}$ **do**
          bins$[w'][b]$.remove$(q)$
        **end for**
        Break and move to next $w$
      **end if**
    **end for**
  **end for**
  Observe $\mathbf{x}_S$ and set $\mathbf{x} \leftarrow \mathbf{x} \cup \mathbf{x}_S$
  $\mathcal{X}_R \leftarrow \mathcal{X}_R \setminus S$
  Update prior as $P(\mathbf{a} \mid \mathbf{x})$
  LOAD($\{q$ **for** $x_{q,w}$ **in** $\mathbf{x}_S)\}$
**end for**

plexity by partitioning assignments into bins.

$\text{JUGGLER}_{\text{BIN}}$ (Algorithm 3) is an approximate version of $\text{JUGGLER}_{\text{AD}}$, which makes uses of binning in order to reduce the cost of making an assignment to $O(|\mathcal{W}|^2 + |\mathcal{W}|C^2)$, where $C$ is user-specified parameter for the number of partitions. $\text{JUGGLER}_{\text{BIN}}$ uses $C^2$ bins representing the cross product of $C$ question difficulty partitions and $C$ belief partitions (evenly-spaced on the interval from 0 to 1). During initialization, the system sorts bins from highest to lowest utility for each worker (computed using the mean question difficulty and belief for the bin, assuming a uniform distribution of questions). For each worker, the system maintains a hash from each bin to a set of unanswered questions whose difficulty and current belief fall within the bin boundaries — and which are not currently being answered by another worker. The system makes an assignment by popping a question from the nonempty bin with highest utility. $\text{JUGGLER}_{\text{BIN}}$ approximates the performance of $\text{JUGGLER}_{\text{AD}}$ arbitrarily closely as the parameter $C$ increases, and can scale independently of the total number of questions since it does not need to consider all questions in each round.



(a) Low-skilled worker (b) High-skilled worker

Figure 2: Visualization of the IG-based utility of asking questions of (a) a low-skilled worker ($\gamma = 0.71$) and (b) a high-skilled worker ($\gamma = 5.0$), as a function of question difficulty and current belief. Darker color indicates higher expected utility. (Bin-size corresponds to running $\text{JUGGLER}_{\text{BIN}}$ with $C = 21$.)

Note that the order in which $\text{JUGGLER}_{\text{BIN}}$ visits bins is not the same for all workers, and depends on a particular worker's skill. Figure 2 shows, for instance, that it may be more valuable to ask a high-skilled worker a difficult question we know little about (belief close to 0.5) than an easy question whose answer is quite certain (belief close to 0 or 1), but same statement might not hold for a low-skilled worker.

### Variable Worker Response Times

Up to this point, we have assumed that each worker completes his assignment in each round. In practice, however, workers take different amounts of time to respond to questions, and a task router must either force some workers to wait until others finish or make new assignments to subsets of workers as they finish. $\text{JUGGLER}_{\text{RT}}$ described in Algorithm 4, performs the latter service by maintaining a record of outstanding assignments in order to route the most useful new tasks to available workers.

## Experiments

We seek to answer the following questions: (1) How well does our adaptive approach perform in practice? We answer this question by asking oDesk workers to perform a challenging NLP task called *named entity linking* (NEL) (Min and Grishman 2012). (2) How much benefit do we derive

**Algorithm 4** The $\text{JUGGLER}_{\text{RT}}$ algorithm

**Input:** Workers $\mathcal{W}$, prior $P(\mathbf{a})$ over answers, unobserved votes $\mathcal{X}_R$, horizon $T$
$\mathbf{x} \leftarrow \emptyset$
$S_{busy} \leftarrow \emptyset$
**for** $t = 1$ **to** $T$ **do**
  $S_t \leftarrow$ call $\text{JUGGLER}_{\text{OFF}}(T = 1, S = S_{busy})$
  $S_{new} \leftarrow S_t \setminus S_{busy}$ and assign $S_{new}$
  $S_{done} \leftarrow$ retrieve completed assignments
  $S_{busy} \leftarrow (S_{busy} \cup S_{new}) \setminus S_{done}$
  Observe $\mathbf{x}_{S_{done}}$ and set $\mathbf{x} \leftarrow \mathbf{x} \cup \mathbf{x}_{S_{done}}$
  $\mathcal{X}_R \leftarrow \mathcal{X}_R \setminus S_{new}$
  Update prior as $P(\mathbf{a} \mid \mathbf{x})$
**end for**

from an adaptive approach, compared to our offline allocation algorithm? We answer this question by looking at average performance over many experiments with simulated data. (3) How much does our adaptive approach benefit from variations in worker skill or problem difficulty? We answer this question through simulations using different distributions for skill and difficulty. (4) Finally, how well does our approach scale to many workers and tasks? We answer this question by first evaluating our strategy of limiting one worker per task per round, and then evaluating how speed gains from binning impact the quality of results.

## Benchmarks

To evaluate the performance of our adaptive strategy, we compare its implementation in the information gain-based $\text{JUGGLER}_{\text{AD}}$ to a series of increasingly powerful alternatives, some of which are also variants of JUGGLER:

- **Random (Ra).** The random strategy simply assigns each worker a random question in each round.
- **Round robin (RR).** The round robin strategy orders the questions by the number of votes they got so far, and in each round iteratively assigns each worker a random question with the fewest votes.
- **Uncertainty-based (Unc).** The uncertainty-based policy orders workers from the least to most skilled [1] and in each round iteratively matches the least skilled yet-unassigned worker to the unassigned question with the highest label uncertainty, measured by the entropy of the posterior distribution over the labels already received for that question. Note that this strategy differs from our implementation of the information gain-based $\text{JUGGLER}_{\text{AD}}$ in just two ways: (a) it never assigns two workers to the same question in the same round and (b) it ignores question difficulty when performing the assignment, thereby computing a difficulty-oblivious version of information gain. Thus, the uncertainty-based strategy can be viewed as a simpler version of $\text{JUGGLER}_{\text{AD}}$ and is expected to perform similarly to $\text{JUGGLER}_{\text{AD}}$ if all questions have roughly the same difficulty.

[1] For consistency with other approaches (we did not observe a significant ordering effect).

- **Accuracy gain-based (AG).** This policy is identical to the information gain-based $\text{JUGGLER}_{\text{AD}}$, but in every round greedily optimizes expected accuracy gain instead, i.e., seeks to find a single-round question-to-worker assignment $S$ that maximizes

$$\mathbb{E}\left[\sum_{q \in \mathcal{Q}} \max_{a_q}(P(a_q \mid \mathcal{X}_S), 1 - P(a_q \mid \mathcal{X}_S))\right].$$

  Unlike information gain, accuracy gain is not submodular, so the worker allocation procedure that maximizes it greedily does not have a constant error bound even within a single allocation round. Nonetheless, intuitively it is a powerful heuristic that provides a reasonable alternative to information gain in the JUGGLER framework.

In the next subsections, we denote the information gain-based $\text{JUGGLER}_{\text{AD}}$ as IG for ease of comparison with the other benchmarks. While the random and round robin strategies a-priori look weaker than IG, the relative qualitative strength of the other two is not immediately clear and provides important insights into IG's practical operation.

In the experiments that follow, we initially force all benchmarks and IG itself to assign questions that have not yet been assigned, until each question has been asked once, before proceeding normally. This ensures that all predictions are founded on at least one observation and provides empirical benefits.

## Comparison of Adaptive Strategies

In order to measure performance with real workers and tasks, we selected a binary named entity linking task (Lin, Mausam, and Weld 2012; Ling and Weld 2012). Since we wished to compare different policies on the same data, we controlled for variations in worker performance by coming up with a set of 198 questions and hiring 12 oDesk workers, each of whom completed the entire set. This ensured reproducibility. Our different policies could now request that any worker perform any problem as they would on a real citizen science platform. To avoid confounding factors, we randomized the order in which workers answered questions, as well as the order in which possible answers were shown on the screen.

In order to estimate worker skill and problem difficulty, we computed a maximum likelihood estimate of these quantities by running gradient descent using gold answers. Note that in general, these parameters can be estimated with little data by modeling worker communities and task features, as discussed previously.

Recall that in each round, each of our workers is assigned one of the 198 questions. After each round, for each policy we calculate the most likely answers to each question by running the same EM procedure on the worker responses. From the predictions, we can calculate the accuracy achieved by each policy after each round.

Figure 3 compares the performance of our adaptive policies, using the observed votes from the live experiment. Since simulating policy runs using all 12 workers would

result in a deterministic procedure for IG and AG, we average the performance of many simulations that use 8 randomly chosen workers to produce a statistically significant result. One way to compare the performance of our policies is to compute the relative labor savings compared to the round robin policy. First, we compute a desired accuracy as a fraction of the maximum accuracy obtained by asking each worker to answer each question, and then compute the fraction of votes saved by running an adaptive policy compared to the round robin policy. Using this metric, all adaptive approaches achieve 95% of the total possible accuracy using fewer than 50% of the votes required by round robin to achieve that same accuracy.

In order to tease apart some of the relative differences between the adaptive policies, we also generated synthetic data for the 198 questions and 12 workers by randomly sampling worker quality and question difficulty using the parameters we estimated from the live experiment. Figure 4 shows the results of this experiment. IG significantly outperforms AG, which in turn significantly outperforms UNC, in terms of fraction of votes saved compared to round robin in order to achieve 95% or 97% of the total possible accuracy ($p < 0.0001$ using two-tailed paired t-tests). These results demonstrate three important points:

- The largest savings are provided by JUGGLER$_{\text{AD}}$ (IG), followed by AG and UNC.
- All three adaptive algorithms significantly outperform the non-adaptive baselines.
- Although the information gain-based JUGGLER$_{\text{AD}}$ "wins" overall, the other modifications of JUGGLER$_{\text{AD}}$ (AG and UNC) perform very well too, despite providing no theoretical guarantees.

We also compared these policies in the context of JUGGLER$_{\text{RT}}$, as shown in Figure 5. In order to simulate worker response times, we fit log-normal distributions to the observed response times and sampled from those distributions. Although the relative benefits are smaller, the ranking of policies remains the same as in the pure round-based setting, demonstrating the promise of our approach to generalize to the more realistic setting of variable worker response times.

Figure 6 summarizes the relative savings of our policies in each of these three scenarios outlined above—fixed votes, simulated votes, and simulated votes with variable response times. Since there is a clear advantage to IG across scenarios, in the following sections we conduct additional simulations to further characterize its performance and answer our remaining empirical research questions.

## Benefit of Adaptivity

In order to determine the benefit of adaptivity, we generated synthetic data for 50 questions and 12 workers by randomly sampling worker quality and question difficulty.

Figure 7 shows the relative labor savings of JUGGLER$_{\text{AD}}$ and JUGGLER$_{\text{OFF}}$ compared to the round robin policy. The desired accuracy is again computed as a fraction of the maximum accuracy obtained by asking

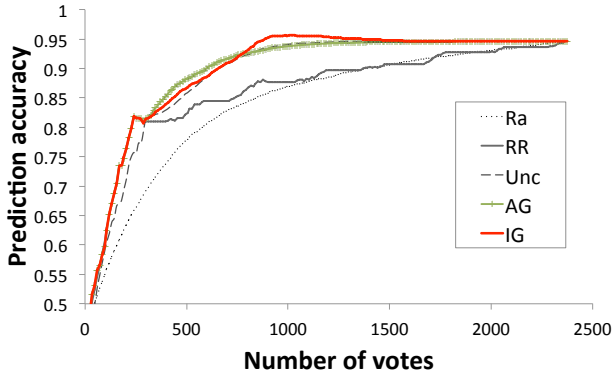



Figure 3: In live experiments, JUGGLER$_{\text{AD}}$ (IG) reaches 95% of the maximum attainable accuracy with only 48% of the labor employed by the commonly used round robin policy (RR).

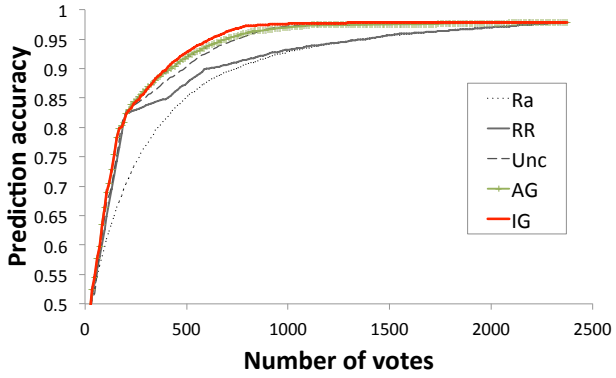



Figure 4: In experiments with simulated data using worker skills and question difficulties estimated from the live experiment, JUGGLER$_{\text{AD}}$ (IG) outperforms the accuracy gain (AG) and uncertainty-based (UNC) policies.

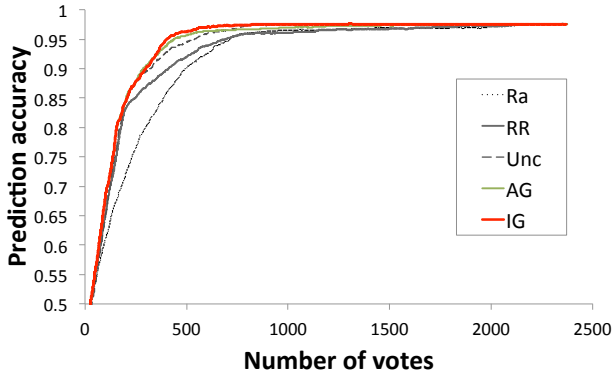



Figure 5: In live experiments with variable worker response times, JUGGLER$_{\text{RT}}$ (IG) also outperforms the other policies, despite the need to make assignments in real time as workers complete tasks.

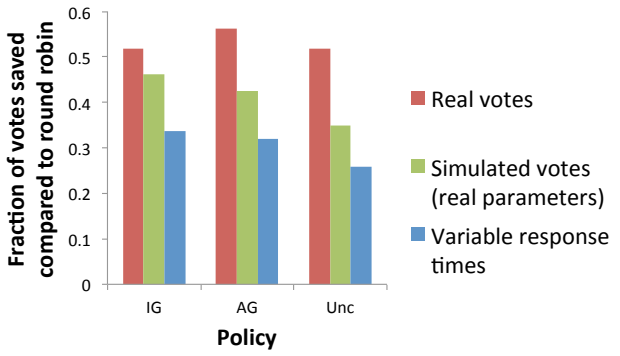



Figure 6: Summary of the fraction of savings of IG, AG, and UNC compared to the round robin policy for reaching an accuracy of 95% of the total achievable accuracy (from asking each worker each question).

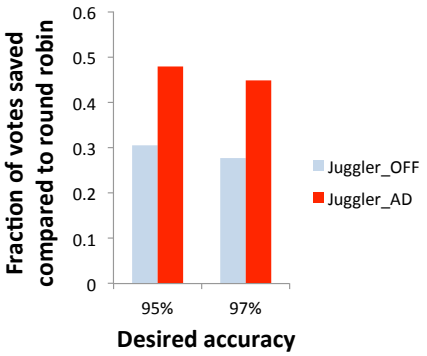



Figure 7: The "adaptivity gap." $\text{JUGGLER}_{\text{AD}}$ saves significantly more human labor than $\text{JUGGLER}_{\text{OFF}}$ by observing and responding to responses adaptively.

each worker to answer each question. Our simulation draws question difficulties uniformly and inverse worker skills $1/\gamma \sim \mathcal{N}(0.79, 0.29)$, a realistic distribution that fits estimates from the live experiment. $\text{JUGGLER}_{\text{AD}}$ saves significantly more labor than $\text{JUGGLER}_{\text{OFF}}$ underscoring the value of adaptive routing compared to offline.

## Varying Worker Skills and Task Difficulties

We now investigate how various parameter distributions affect adaptive performance. Again, we generated synthetic data for 50 questions and 12 workers. Figure 8 shows that the relative savings of $\text{JUGGLER}_{\text{AD}}$ over round robin increases as the pool of workers becomes more diverse. We control the experiment by again sampling difficulties uniformly and using our earlier worker skill distribution, but varying the standard deviation from $\sigma = 0$ to $\sigma = 0.2$. Like before, accuracies are computed as a fraction of the total achievable accuracy given a set of workers. Our algorithms perform better for larger values of $\sigma$ because they are able to exploit differences between workers to make the best assignments.

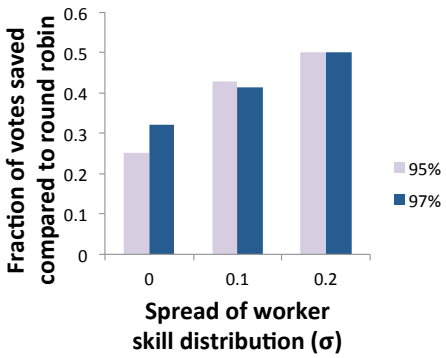



Figure 8: The benefit provided by $\text{JUGGLER}_{\text{AD}}$ increases as the pool of workers becomes more diverse.

Increasing the diversity of question difficulties while keeping the skill distribution fixed produces similar savings. For example, sampling difficulties from a Beta distribution with a single peak produces relative savings over round robin that are significantly less than sampling uniformly or from a bimodal distribution.

## Scaling to Many Tasks and Workers

In order to empirically validate the speed and allocation effectiveness of our proposed scaling strategies, we experimented with data simulating a larger number of workers and questions (100 and 2000, respectively) using the same question difficulty and worker skill settings as in the earlier adaptivity experiment. Without imposing a limit on the number of simultaneously-operating workers per task, it is infeasible to run $\text{JUGGLER}_{\text{AD}}$ on problems of this size. We first checked to see if there was a drop in question-answering performance caused by limiting the number of workers assigned to a given task in each round. We found no statistically significant difference between limiting the number of workers per task to 1, 2, or 3 (at the 0.05-significance level using one-tailed paired t-tests to measure the fraction of votes saved to reach various thresholds). This result matches our experience running smaller experiments.

Restricting the number of workers per task to a small number reduces $\text{JUGGLER}_{\text{AD}}$'s scheduling time to 10–15 seconds[2] per round (scheduling 100 workers), but this is still significantly longer than the fraction-of-a-second scheduling times for our earlier experiments with 12 workers and 198 questions. $\text{JUGGLER}_{\text{BIN}}$, by contrast, is able to schedule workers in fewer than 3 seconds for any number of bins we tried ($20^2$, $40^2$, $80^2$, and $160^2$). Indeed, $\text{JUGGLER}_{\text{BIN}}$ makes assignments using $O(|\mathcal{W}|^2)$ computations, which is drastically better than $O(|\mathcal{W}||\mathcal{Q}|)$ for the unbinned version when there are many questions. Although our runtime analysis for $\text{JUGGLER}_{\text{BIN}}$ includes an additional term on the order of $|\mathcal{W}|C^2$, in practice this cost is much smaller since we do not need to traverse all bins in order to find a valuable

[2] All reported runtimes use an unoptimized implementation on a single machine with two 2.66 GHz processors and 32 GB of RAM.

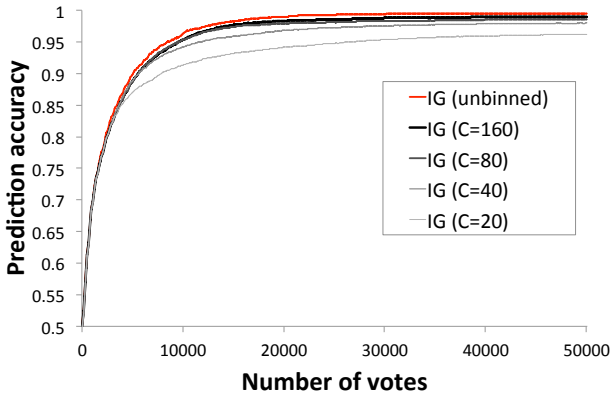



Figure 9: Increasing the parameter $C$ governing the number of bins enables JUGGLER$_{\text{BIN}}$ to closely approximate JUGGLER$_{\text{OFF}}$.

question to assign to a worker.

Finally, the performance of JUGGLER$_{\text{BIN}}$ approaches that of the unbinned algorithms as we increase the value of the parameter $C$, since smaller bins more closely approximate the true utility values. Figure 9 shows the results of this experiment. JUGGLER$_{\text{BIN}}$ asymptotes at a slightly lower accuracy than the unbinned approach, suggesting that more sophisticated approaches like adaptive binning may produce further gains.

## Related Work

Previous work has considered task routing in a number of restricted settings. Some approaches (Karger, Oh, and Shah 2011a; 2011b; 2013) assume worker error is independent of problem difficulty and hence gain no benefit from adaptive allocation of tasks.

Most adaptive approaches, on the other hand, assign problems to one worker at a time, not recognizing the cost of leaving a worker idle (Donmez, Carbonell, and Schneider 2009; Yan et al. 2011; Wauthier and Jordan 2011; Chen, Lin, and Zhou 2013). Other adaptive approaches assume that a requester can immediately evaluate the quality of a worker's response (Ho and Vaughan 2012; Tran-Thanh et al. 2012), or wait for a worker to complete all her tasks before moving on to the next worker (Ho, Jabbari, and Vaughan 2013). Some, unlike ours, also assume that any given question can be assigned to just one worker (Tran-Thanh et al. 2012). Moreover, the adaptive approach that best models real workers optimizes annotation accuracy by simply seeking to discover the best workers and use them exclusively (Chen, Lin, and Zhou 2013).

The Generalized Task Markets (GTM) framework (Shahaf and Horvitz 2010) has much in common with our problem; they seek to find a coalition of workers whose multi-attribute skills meet the requirements of a job. However, the GTM approach assumes a binary utility model (tasks are completed or not) – it does not reason about answer accuracy. Kamar et al. (2012) also focus on citizen science, but don't perform task routing.

## Conclusions and Future Work

In citizen science and other types of volunteer crowdsourcing, it is important to send hard tasks to experts while still utilizing novices as they become proficient. Since workers are impatient, a task router should allocate questions to *all available workers in parallel* in order to keep workers engaged. Unfortunately, choosing the optimal set of tasks for workers is challenging, even if worker skill and problem difficulty are known.

This paper introduces the JOCR framework, characterizing the space of task routing problems based on adaptivity, concurrency and amount of information known. We prove that even offline task routing is NP-hard, but submodularity of the objective function (maximizing expected value of information) enables reasonable approximations. We present JUGGLER$_{\text{OFF}}$ and JUGGLER$_{\text{AD}}$, two systems that perform parallel task allocation for the offline and adaptive settings respectively. Using live workers hired on oDesk, we show that our best routing algorithm uses just 48% of the labor required by the commonly used round-robin policy on a natural language named-entity linking task. Further, we present JUGGLER$_{\text{BIN}}$, a system that can scale to many workers and questions, and JUGGLER$_{\text{RT}}$, a system that handles variable worker response times and yields similar empirical savings.

Much remains to be done. While we have been motivated by volunteer labor from a citizen science scenario, we believe our methods extend naturally to paid crowdsourcing where the utility function includes a linear cost component. In the future we hope to relax the assumptions of perfect information about workers and tasks, using techniques from multi-armed bandits to learn these parameters during the routing process. We also wish to study the case where several different requesters are using the same platform and the routing algorithm needs to balance workers across problem types.

## Acknowledgements

We thank Christopher Lin and Adish Singla for helpful discussions, as well as the anonymous reviewers for insightful comments. This work was supported by the WRF / TJ Cable Professorship, Office of Naval Research grant N00014-12-1-0211, and National Science Foundation grants IIS-1016713, IIS-1016465, and IIS-1420667.

## References


Badanidiyuru, A., and Vondrak, J. 2014. Fast algorithms for maximizing submodular functions. In *Proceedings of SODA '14*.

Chen, X.; Lin, Q.; and Zhou, D. 2013. Optimistic knowledge gradient policy for optimal budget allocation in crowdsourcing. In *Proceedings of the 30th International Conference on Machine Learning (ICML '13)*.

Dai, P.; Mausam; and Weld, D. S. 2010. Decision-theoretic control of crowd-sourced workflows. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI '10)*.

Donmez, P.; Carbonell, J. G.; and Schneider, J. 2009. Efficiently learning the accuracy of labeling sources for selective sampling. In *KDD*.

Filmus, Y., and Ward, J. 2012. A tight combinatorial algorithm for submodular maximization subject to a matroid constraint. In *53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS '12)*.

Golovin, D., and Krause, A. 2011. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *Journal of Artificial Intelligence Research* (2005):1–53.

Ho, C.-J., and Vaughan, J. W. 2012. Online task assignment in crowdsourcing markets. In *AAAI*.

Ho, C.-J.; Jabbari, S.; and Vaughan, J. W. 2013. Adaptive task assignment for crowdsourced classification. In *Proceedings of the 30th International Conference on Machine Learning (ICML '13)*, 534–542.

Kamar, E.; Hacker, S.; and Horvitz, E. 2012. Combining human and machine intelligence in large-scale crowdsourcing. In *Proceedings of the Eleventh International Conference on Autonomous Agents and Multiagent Systems (AAMAS '12)*.

Karger, D. R.; Oh, S.; and Shah, D. 2011a. Budget-optimal crowdsourcing using low-rank matrix approximations. In *Conference on Communication, Control, and Computing*.

Karger, D. R.; Oh, S.; and Shah, D. 2011b. Iterative learning for reliable crowd-sourcing systems. In *NIPS*.

Karger, D. R.; Oh, S.; and Shah, D. 2013. Efficient crowdsourcing for multi-class labeling. In *Proceedings of the ACM SIGMETRICS/International Conference on Measurement and Modeling of Computer Systems*, 81–92.

Kolobov, A.; Mausam; and Weld, D. S. 2013. Joint crowdsourcing of multiple tasks. In *Proceedings of the First AAAI Conference on Human Computation and Crowdsourcing (HCOMP '13)*.

Krause, A., and Golovin, D. 2014. Submodular function maximization. In *Tractability: Practical Approaches to Hard Problems*. Cambridge University Press. To appear.

Krause, A., and Guestrin, C. 2005. Near-optimal Nonmyopic Value of Information in Graphical Models. In *Conference on Uncertainty in Artificial Intelligence (UAI)*.

Law, E., and von Ahn, L. 2011. *Human Computation*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers.

Lin, C.; Mausam; and Weld, D. S. 2012. Dynamically switching between synergistic workflows for crowdsourcing. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI '12)*.

Ling, X., and Weld, D. S. 2012. Fine-grained entity resolution. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI '12)*.

Lintott, C.; Schawinski, K.; Slosar, A.; Land, K.; Bamford, S.; Thomas, D.; Raddick, M. J.; Nichol, R. C.; Szalay, A.; Andreescu, D.; Murray, P.; and VandenBerg, J. 2008. Galaxy zoo : Morphologies derived from visual inspection of galaxies from the sloan digital sky survey. *MNRAS* 389(3):1179–1189.

Mao, A.; Kamar, E.; Chen, Y.; Horvitz, E.; Schwamb, M. E.; Lintott, C. J.; and Smith, A. M. 2013. Volunteering versus work for pay: Incentives and tradeoffs in crowdsourcing. In *Proceedings of the First AAAI Conference on Human Computation and Crowdsourcing (HCOMP '13)*.

Min, B., and Grishman, R. 2012. Challenges in the knowledge base population slot filling task. In *LREC*, 1137–1142.

Nemhauser, G. L.; Wolsey, L. A.; and Fisher, M. L. 1978. An analysis of approximations for maximizing submodular set functions-II. *Mathematical Programming Study* 8:73–87.

Raghunathan, K.; Lee, H.; Rangarajan, S.; Chambers, N.; Surdeanu, M.; Jurafsky, D.; and Manning, C. 2010. A multi-pass sieve for coreference resolution. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, 492–501. Stroudsburg, PA, USA: Association for Computational Linguistics.

Shahaf, D., and Horvitz, E. 2010. Generalized task markets for human and machine computation. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI '10)*.

Stoyanov, V.; Gilbert, N.; Cardie, C.; and Riloff, E. 2009. Conundrums in noun phrase coreference resolution: Making sense of the state-of-the-art. In Su, K.-Y.; Su, J.; and Wiebe, J., eds., *ACL/IJCNLP*, 656–664. The Association for Computer Linguistics.

Tran-Thanh, L.; Stein, S.; Rogers, A.; and Jennings, N. R. 2012. Efficient crowdsourcing of unknown experts using multi-armed bandits. In *ECAI*, 768–773.

Venanzi, M.; Guiver, J.; Kazai, G.; Kohli, P.; and Shokouhi, M. 2014. Community-based bayesian aggregation models for crowdsourcing. In *Proceedings of the World Wide Web Conference*.

Vondrak, J. 2008. Optimal approximation for the submodular welfare problem in the value oracle model. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC '08)*.

Wauthier, F., and Jordan, M. 2011. Bayesian bias mitigation for crowdsourcing. In *Advances in Neural Information Processing Systems 24 (NIPS '11)*. 1800–1808.

Welinder, P.; Branson, S.; Belongie, S.; and Perona, P. 2010. The multidimensional wisdom of crowds. In *Advances in Neural Information Processing Systems 23 (NIPS '10)*. 2424–2432.

Whitehill, J.; Ruvolo, P.; Wu, T.; Bergsma, J.; and Movellan, J. 2009. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Advances in Neural Information Processing Systems 22 (NIPS '09)*. 2035–2043.

Yan, Y.; Rosales, R.; Fung, G.; and Dy, J. G. 2011. Active learning from crowds. In *ICML*.