# Decision-Theoretic Control of Crowd-Sourced Workflows

**Peng Dai** **Mausam** **Daniel S. Weld**
Dept of Computer Science and Engineering
University of Washington
Seattle, WA-98195
{daipeng,mausam,weld}@cs.washington.edu

## Abstract

*Crowd-sourcing* is a recent framework in which human intelligence tasks are outsourced to a crowd of unknown people ("workers") as an open call (e.g., on Amazon's Mechanical Turk). Crowd-sourcing has become immensely popular with hoards of employers ("requesters"), who use it to solve a wide variety of jobs, such as dictation transcription, content screening, etc. In order to achieve quality results, requesters often subdivide a large task into a chain of bite-sized subtasks that are combined into a complex, iterative workflow in which workers check and improve each other's results. This paper raises an exciting question for AI — could an autonomous agent control these workflows without human intervention, yielding better results than today's state of the art, a fixed control program?

We describe a planner, TURKONTROL, that formulates workflow control as a decision-theoretic optimization problem, trading off the implicit quality of a solution artifact against the cost for workers to achieve it. We lay the mathematical framework to govern the various decisions at each point in a popular class of workflows. Based on our analysis we implement the workflow control algorithm and present experiments demonstrating that TURKONTROL obtains much higher utilities than popular fixed policies.

## Introduction

In today's rapidly accelerating economy an efficient workflow for achieving one's complex business task is often the key to business competitiveness. *Crowd-sourcing*, "the act of taking tasks traditionally performed by an employee or contractor, and outsourcing them to a group (crowd) of people or community in the form of an open call" [18], has the potential to revolutionize information-processing services by quickly coupling human workers with software automation in productive workflows [6].

While the phrase 'crowd-sourcing' was only termed in 2006, the area has grown rapidly in economic significance with the growth of general-purpose platforms such a Amazon's *Mechanical Turk* [12] and task-specific sites for call centers [10], programming jobs [16] and more. Recent research has shown surprising success in solving difficult tasks using the strategy of incremental improvement in an iterative workflow [9]; similar workflows are used commercially to automate dictation transcription and screening of posted content. See Figure 1 for a successful example of a complex task solved using Mechanical Turk — this challenging handwriting was deciphered step by step, with output of one worker feeding as the input to the next. Additional voting HITs were used to assess whether a worker actually improved the transcription compared to the prior effort.

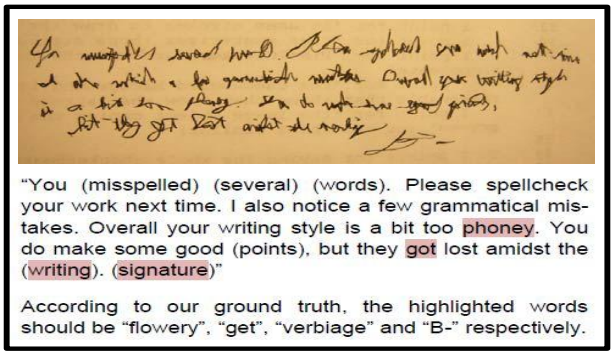


Figure 1: A handwriting recognition task (almost) successfully solved at Mechanical Turk using an iterative workflow. Workers were shown the text written by a human and in a few iterations they deduced the message (with errors highlighted). Figure adapted from [9].

From an AI perspective, crowdsourced workflows offer a new, exciting and impactful application area for intelligent control. Although there is a vast literature on decision-theoretic planning and execution (*e.g.*, [14; 1; 7]), it appears that these techniques have yet to be applied to control a crowd-sourcing platform. While the handwriting example shows the power of collaborative workflows, we still do not know answers to many questions: (1) what is the optimal number of iterations for such a task? (2) how many ballots should be used for voting? (3) how do these answers change if the workers are skilled (or very error prone)?

This paper offers initial answers to these questions by presenting a decision-theoretic planner, which dynamically optimizes iterative workflows to achieve the best quality/cost tradeoff. We make the following contributions:

- We introduce the AI problem of optimization and control of iterative workflows over a crowd-sourcing platform.
- We develop the mathematical theory for optimizing the quality/cost tradeoff for a popular class of workflows.
- We implement an agent, TURKONTROL, for taking deci-

sions at each step of the workflow based on the expected utilities of each action.
- We simulate TURKONTROL in a variety of complex scenarios and find that it behaves robustly. We also show that TURKONTROL decisions result in a significantly higher final utility compared to fixed policies and other baselines.

## Background

While the ideas in our paper are applicable to different workflows, for our case study we choose the iterative workflow introduced by Little *et al.* [9] depicted in Figure 2. This particular workflow is representative of a number of flows in commercial use today; at the same time, it is moderately complex making it ideal for first investigation.

Little's chosen task is iterative text improvement. There is an initial job, which presents the worker with an image and requests an English description of the picture's contents. A subsequent iterative process consists of an *improvement job* and *voting jobs*. In the improvement job, a (different) worker is shown this same image as well as the current description and is requested to generate an improved English description. Next $n \geq 1$ ballot jobs are posted ("Which text best describes the picture?"). Based on a majority opinion the best description is selected and the loop continues. Little *et al.* have shown that this iterative process generates better descriptions for a fixed amount than allocating the total reward to a single author.

Little *et al.*, support an open-source toolkit, TurKit, that provides a high-level mechanism for defining moderately complex, iterative workflows with voting-controlled conditionals. However, TurKit doesn't have built-in methods for monitoring the accuracy of workers; nor does TurKit automatically determine the ideal number of voters or estimate the appropriate number of iterations before returns diminish. Our mathematical framework in the next section answers these and other questions.

## Decision Theoretic Optimization

The agent's control problem for a workflow like iterative text improvement is defined as follows. As input the agent is given an initial artifact (or a job description for requesting one), and the agent is asked to return an artifact which maximizes some payoff based on the quality of the submission. Intuitively, something is high quality if it is better than most things of the same type. For engineered artifacts (including English descriptions) one may say that something is high quality if it is difficult to improve. This suggests measuring quality of an artifact in terms of units we call a *quality improvement probability* (QIP), which we denote by $q \in [0,1]$. An artifact with QIP $q$ means an average dedicated worker has probability $1-q$ of improving the artifact. In our initial model, we assume that requesters will express their utility as a function $U$ from QIP to dollars.

The QIP of an artifact is never exactly known – it is at best estimated based on domain dynamics and observations (like vote results). Thus, it is a POMDP problem – the decisions need to be taken based on our belief of the QIP. Moreover, since QIP is a real number, it is a POMDP in continuous state space [2]. These kind of POMDPs are especially hard to solve for realistic problems. We overcome the computational bottleneck by performing limited lookahead search to make planning more tractable.

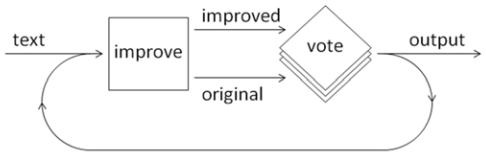

Figure 2: Flowchart for the iterative text improvement task, reprinted from [9].

Figure 3 summarizes a high level flow for our planner's decisions. At each step we track our belief in QIPs ($q$ and $q'$) of the previous ($\alpha$) and the current artifact ($\alpha'$) respectively. Each decision or observation gives us new information, which is reflected in the QIP posteriors. These distributions also depend on the accuracy of workers, which we also incrementally estimate based on their previous work.

Based on these distributions we estimate expected utilities for each action. This lets us answer questions like (1) when to terminate the voting phase (thus switching attention to artifact improvement), (2) which of the two artifacts is the best basis for subsequent improvements, and (3) when to stop the whole iterative process and submit the result to the requester.

Below, we present our mathematical analysis in detail. It is divided into three key stages: QIP posteriors after improvement or a new ballot, utility computations for the available actions and finally the decision making algorithm and implementation details.

### QIP Tracking

Suppose we have an artifact $\alpha$, with an unknown QIP $q$ and a prior[1] density function $f_Q(q)$. Suppose a worker $x$ takes an improvement job and submits another artifact $\alpha'$, whose QIP is denoted by $q'$. Since $\alpha'$ is a suggested improvement of $\alpha$, $q'$ depends on the initial quality $q$. Moreover, a higher accuracy worker $x$ may improve it much more so it depends on $x$. We define $f_{Q'|q,x}$ as the conditional quality distribution of $q'$ when worker $x$ improved an artifact of quality $q$. This function describes the dynamics of the domain. With a known $f_{Q'|q,x}$ we can easily compute the prior on $q'$ from the law of total probability:

$$f_{Q'}(q') = \int_0^1 f_{Q'|q,x}(q') f_Q(q) dq. \quad (1)$$

While we do have priors on the QIPs of both the new and the old artifacts, we do not know for sure whether the new artifact is an improvement over the old or not. The worker may have done a good job or a bad job. Even if it is an improvement we need to assess how good of an improvement it is. Our workflow at this point tries to gather evidence to answer these questions by generating ballots and asking new workers a question: "Is $\alpha'$ a better answer than $\alpha$ for the

[1]We will estimate a QIP distribution for the very first artifact by a limited training data. Later, posteriors of the previous iteration will become priors of the next.

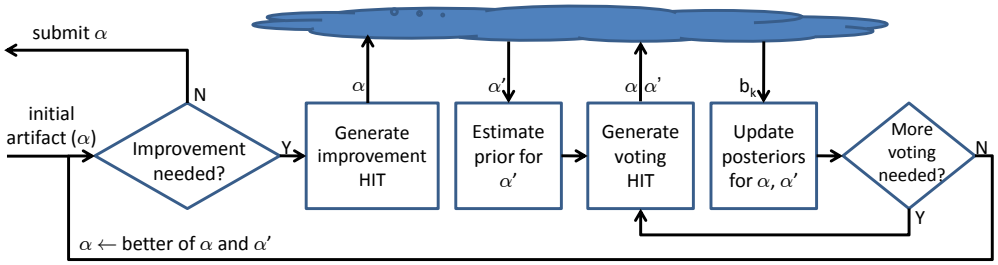


Figure 3: Computations needed by TURKONTROL for control of an iterative-improvement workflow.

original question?". Say we ask $n$ workers and their votes are $\overrightarrow{\mathbf{b}^n} = b_1, \ldots, b_n$, where $b_i \in \{1, 0\}$.

Based on these votes we compute the posteriors in QIP, $f_{Q|\overrightarrow{\mathbf{b}^n}}$ and $f_{Q'|\overrightarrow{\mathbf{b}^n}}$. These posteriors have three roles to play. First, more accurate beliefs lead to a higher probability of keeping the better artifact for subsequent phases. Second, within the voting phase confident beliefs help us decide when to stop voting. Third, a high QIP belief also helps us decide when to quit the iterative process and submit.

In order to accomplish this we make some assumptions. First, we assume each worker $x$ is diligent, so she answers all ballots to the best of her ability. Still she may make mistakes, and we have full knowledge of her accuracy. Second, we assume that several workers will not collaborate adversarially to defeat the system.

These assumptions might lead one to believe that the probability distributions for worker responses ($P(b_i)$) are independent of each other. Unfortunately, this independence is violated due to a subtlety. The reason is that even though the different workers are not collaborating a mistake by one worker changes the error probability of others. This happens because a mistake gives evidence that the question may be intrinsically hard and hence, difficult for others to get it right also. To get around this we introduce *intrinsic difficulty* ($d$) of our question ($d \in [0, 1]$). It depends on whether the two QIPs are very close or not. Closer the two artifacts the more difficult it is to judge whether one is better or not. We define the relationship between the difficulty and QIPs as

$$d(q, q') = 1 - |q - q'|^{\mathcal{M}} \tag{2}$$

We can safely assume that given $d$ the probability distributions will be independent of each other.

Moreover, each worker's accuracy will vary with the problem's difficulty. We define $a_x(d)$ as the accuracy of the worker $x$ on a question of difficulty $d$. We will expect everyone's accuracy to be monotonically decreasing in $d$. It will approach random behavior as questions get really hard, *i.e.*, $a_x(d) \to 0.5$ as $d \to 1$. Similarly, as $d \to 0$, $a_x(d) \to 1$. We use a group of polynomial functions $\frac{1}{2}[1 + (1 - d)^{\gamma_x}]$ for $\gamma_x > 0$ to model $a_x(d)$ under these constraints. It is easy to check that this polynomial function satisfies all the conditions when $d \in [0, 1]$. Note that smaller the $\gamma_x$ the more concave the accuracy curve, and thus greater the expected accuracy for a fixed $d$.

Note that given knowledge of $d$ we can compute the likelihood of a worker answering "Yes". We consider the $i^{th}$ worker $x_i$ who has accuracy $a_{x_i}(d)$. We calculate $P(b_i = 1 \mid q, q')$ as:

$$\begin{aligned} \text{If } q' > q \; P(b_i = 1 | q, q') &= a_{x_i}(d(q, q')), \\ \text{If } q' \leq q \; P(b_i = 1 | q, q') &= 1 - a_{x_i}(d(q, q')). \end{aligned} \tag{3}$$

We first derive the posterior distribution given one more ballot $b_{n+1}$, $f_{Q|\overrightarrow{\mathbf{b}^{n+1}}}(q)$ based on existing distributions $f_{Q|\overrightarrow{\mathbf{b}^n}}(q)$ and $f_{Q'|\overrightarrow{\mathbf{b}^n}}(q)$. We abuse notation slightly, using $\overrightarrow{\mathbf{b}^{n+1}}$ to symbolically denote that $n$ ballots are known and we will receive another ballot (value currently unknown) in the future. By applying the Bayes rule we get

$$f_{Q|\overrightarrow{\mathbf{b}^{n+1}}}(q) \propto P(b_{n+1} \mid q, \overrightarrow{\mathbf{b}^n}) f_{Q|\overrightarrow{\mathbf{b}^n}}(q) \tag{4}$$

$$= P(b_{n+1} \mid q) f_{Q|\overrightarrow{\mathbf{b}^n}}(q) \tag{5}$$

Equation 5 is based on the independence of workers. Now we apply the law of total probability on $P(b_{n+1} \mid q)$ :

$$P(b_{n+1} \mid q) = \int_0^1 P(b_{n+1} \mid q, q') f_{Q'|\overrightarrow{\mathbf{b}^n}}(q') dq' \tag{6}$$

The same sequence of steps can be used to compute the posterior of $\alpha'$.

$$f_{Q'|\overrightarrow{\mathbf{b}^{n+1}}}(q') \propto P(b_{n+1} \mid q', \overrightarrow{\mathbf{b}^n}) f_{Q'|\overrightarrow{\mathbf{b}^n}}(q') \tag{7}$$

$$= P(b_{n+1} \mid q') f_{Q'|\overrightarrow{\mathbf{b}^n}}(q') \tag{8}$$

$$= \left[ \int_0^1 P(b_{n+1} | q, q') f_{Q|\overrightarrow{\mathbf{b}^n}}(q) dq \right] f_{Q'}(q')$$

*Discussion:* Why should our belief in the quality of the previous artifact change (posterior of $\alpha$) based on ballots comparing it with the new artifact? This is a subtle, but important point. If the improvement worker (who has a good accuracy) was unable to create a much better $\alpha'$ in the improvement phase that must be because $\alpha$ already has a high QIP and is no longer easily improvable. Under such evidence we should increase QIP of $\alpha$, which is reflected by the posterior of $\alpha$, $f_{Q|\overrightarrow{\mathbf{b}}}$. Similarly, if all voting workers unanimously thought that $\alpha'$ is much better than $\alpha$, it means the ballot was very easy, *i.e.*, $\alpha'$ incorporates significant improvements over $\alpha$ and the QIPs should reflect that.

This computation helps us determine the prior QIP for the artifact in the the next iteration. It will be either $f_{Q|\overrightarrow{\mathbf{b}}}$ or $f_{Q'|\overrightarrow{\mathbf{b}}}$ (Equations 5 and 8), depending on whether we decide to keep $\alpha$ or $\alpha'$.

## Utility Estimations

We now discuss the computation for the utility of an additional ballot. At this point, say, we have already received $n$ ballots ($\overrightarrow{\mathbf{b}^n}$) and we have posteriors of the two artifacts $f_{Q|\overrightarrow{\mathbf{b}^n}}$ and $f_{Q'|\overrightarrow{\mathbf{b}^n}}$ available to us. We use $U_{\overrightarrow{\mathbf{b}^n}}$ to denote the expected utility of stopping now, *i.e.*, without another ballot and $U_{\overrightarrow{\mathbf{b}^{n+1}}}$ to denote the utility after another ballot. $U_{\overrightarrow{\mathbf{b}^n}}$ can be easily computed as the maximum expected utility we get from the two artifacts $\alpha$ and $\alpha'$:

$$U_{\overrightarrow{\mathbf{b}^n}} = max\{E[U(Q|\overrightarrow{\mathbf{b}^n})], E[U(Q'|\overrightarrow{\mathbf{b}^n})]\}, \text{ where} \quad (9)$$

$$E[U(Q|\overrightarrow{\mathbf{b}^n})] = \int_0^1 U(q) f_{Q|\overrightarrow{\mathbf{b}^n}}(q) dq \quad (10)$$

$$E[U(Q'|\overrightarrow{\mathbf{b}^n})] = \int_0^1 U(q') f_{Q'|\overrightarrow{\mathbf{b}^n}}(q') dq' \quad (11)$$

Using $U_{\overrightarrow{\mathbf{b}^n}}$ we need to compute the utility of taking an additional ballot, $U_{\overrightarrow{\mathbf{b}^{n+1}}}$. The $n+1^{th}$ ballot, $b_{n+1}$, could be either "Yes" or "No". The probability distribution $P(b_{n+1} \mid q, q')$ governs this, which also depends on the accuracy of the worker (see Equation 3). However, since we do not know which worker will take our ballot job, we assume anonymity and expect an average worker $\overline{x}$ with the accuracy function $a_{\overline{x}}(d)$. Recall from Equation 2 that difficulty, $d$, is a function of the similarity in QIPs. Because $q$ and $q'$ are not exactly known, probability of getting the next ballot is computed by applying law of total probability on the joint probability $f_{Q,Q'}(q, q')$

$$P(b_{n+1}) = \int_0^1 \left[ \int_0^1 P(b_{n+1}|q, q') f_{Q'|\overrightarrow{\mathbf{b}^n}}(q') dq' \right] f_{Q|\overrightarrow{\mathbf{b}^n}}(q) dq.$$

These allow us to compute $U_{\overrightarrow{\mathbf{b}^{n+1}}}$ as follows ($c_b$ is the cost of a ballot)

$$U_{\overrightarrow{\mathbf{b}^{n+1}}} = max\{E[U(Q|\overrightarrow{\mathbf{b}^{n+1}})], E[U(Q' \mid \overrightarrow{\mathbf{b}^{n+1}})]\} - c_b$$

$$E[U(Q \mid \overrightarrow{\mathbf{b}^{n+1}})] = \int_0^1 \left( \sum_{b_{n+1}} U(q) f_{Q|\overrightarrow{\mathbf{b}^{n+1}}}(q) P(b_{n+1}) \right) dq$$

We can write a similar equation for $E[U(Q' \mid \overrightarrow{\mathbf{b}^{n+1}})]$.

Similarly, we can compute the utility of an improvement step. We already have access to current beliefs on the QIP of $\alpha$ and $\alpha'$. Based on those and Equation 9 we can choose $\alpha$ or $\alpha'$ as the better artifact. The belief of the chosen artifact acts as $f_Q$ for Equation 1 and we can estimate a new prior $f_{Q'}$ after an improvement step. Expected utility of improvement will be $\max\left(\int_0^1 U(q) f_Q(q) d(q), \int_0^1 U(q') f_{Q'}(q') d(q')\right) - c_{imp}$. Here $c_{imp}$ is the cost an improvement HIT.

**Decision Making:** At any step we can either choose to do an additional vote, choose the better artifact and attempt another improvement or submit the artifact. We already described computations for utilities for each option. For a greedy 1-step lookahead policy we can simply pick the best of the three options.

Of course, a greedy policy may be much worse than the optimal. We can compute a better policy by an $l$-step lookahead algorithm where we evaluate all sequences of $l$ decisions, find the best sequence based on our utilities and then execute the first action of the sequence and repeat.

**Updating Difficulty and Worker Accuracy:** The agent updates its estimated $d$ before each decision point based on its estimates of QIPs as follows:

$$d^* = \int_0^1 \int_0^1 d(q, q') f_Q(q) f_{Q'}(q') dq dq' = \int_0^1 \int_0^1 (1 - |q, q'|^{\mathcal{M}}) f_Q(q) f_{Q'}(q') dq dq' \quad (12)$$

After completing each iteration we have access to estimates for $d^*$ and the believed answer. We can use this information to update our record on the quality of each worker. In particular, if someone answered a question correctly then she is a good worker (and her $\gamma_x$ should decrease) and if someone made an error in a question her $\gamma_x$ should increase. Moreover the increase/decrease amounts should depend on the difficulty of the question. The following simple update strategy may work:

1. If a worker answered a question of difficulty $d$ correctly then $\gamma_x \leftarrow \gamma_x - d\delta$.
2. If a worker made an error when answering a question then $\gamma_x \leftarrow \gamma_x + (1 - d)\delta$.

We use $\delta$ to represent the learning rate, which we could slowly reduce over time so that the accuracy of a worker approaches an asymptotic distribution.

**Implementation:** In a general model such as ours maintaining a closed form representation for all these continuous functions may not be possible. Uniform discretization is the simplest way to approximate these general functions. However, for efficient storage and computation TURKONTROL could employ the piecewise constant/piecewise linear value function representations or use particle filters. Even though approximate both these techniques are very popular in the literature for efficiently maintaining continuous distributions [11; 4] and can provide arbitrarily close approximations. Because some of our equations require double integrals and can be time consuming (*e.g.*, Equation 12) these compact representations help in overall efficiency of the implementation.

## Experiments

This section aims to empirically answer the following questions: 1) How deep should be an agent's lookahead to best tradeoff between computation time and utility? 2) Does TURKONTROL make better decisions compared to TurKit? 3) Can our planner outperform an agent following a well-informed, fixed policy?

**Experimental Setup** We set the maximum utility to be 1000 and use a convex utility function $U(q) = 1000\frac{e^q - 1}{e - 1}$ with $U(0) = 0$ and $U(1) = 1000$. We assume the quality of the initial artifact follows a Beta distribution $\text{Beta}(1, 9)$, which implies that the mean QIP of the first artifact is 0.1.

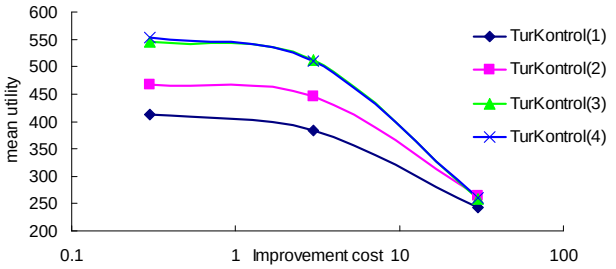


Figure 4: Average net utility of TURKONTROL with various lookahead depths calculated using 10,000 simulation trials on three sets of (improvement, ballot) costs: (30,10), (3,1), and (0.3,0.1). Longer lookahead produces better results, but 2-step lookahead is good enough when costs are relatively high: (30,10).

Suppose the quality of the current artifact is $q$, we assume the conditional distribution $f_{Q'|q,x}$ is Beta distributed, with mean $\mu_{Q'|q,x}$ where:

$$\mu_{Q'|q,x} = q + 0.5[\ (1-q) \times (a_x(q) - 0.5) \\ +q \times (a_x(q) - 1)\ ]. \qquad (13)$$

and the conditional distribution is Beta$(10\mu_{Q'|q,x}, 10(1 - \mu_{Q'|q,x}))$. We know a higher QIP means it's less likely the artifact can be improved. We model results of an improvement task, in a manner akin to ballot tasks; the resulting distribution of qualities is influenced by the worker's accuracy and the improvement difficulty, $d = q$.

We fix the ratio of the costs of improvements and ballots, $c_{imp}/c_b = 3$, because ballots take less time. We set the difficulty constant $\mathcal{M} = 0.5$. In each of the simulation runs, we build a pool of 1000 workers, whose error coefficients, $\gamma_x$, follow a bell shaped distribution with a fixed mean $\gamma$. We also distinguish the accuracies of performing an improvement and answering a ballot by using one half of $\gamma_x$ when worker $x$ is answering a ballot, since answering a ballot is an easier task, and therefore a worker should have higher accuracy.

**Picking the Best Lookahead Depth** We first run 10,000 simulation trials with average error coefficient $\gamma$=1 on three pairs of improvement and ballot costs — (30,10), (3,1), and (0.3,0.1) — trying to find the best lookahead depth $l$ for TURKONTROL. Figure 4 shows the average *net utility*, the utility of the submitted artifact minus the payment to the workers, of TURKONTROL with different lookahead depths, denoted by TurKontrol($l$). Note that there is always a performance gap between TurKontrol(1) and TurKontrol(2), but the curves of TurKontrol(3) and TurKontrol(4) generally overlap. We also observe that when the costs are high, such that the process usually finishes in a few iterations, the performance difference between TurKontrol(2) and deeper step lookaheads is negligible. Since each additional step of lookahead increases the computational overhead by an order of magnitude, we limit TURKONTROL' lookahead to depth 2 in subsequent experiments.

**The Effect of Poor Workers** We now consider the effect of worker accuracy on the effectiveness of agent control policies. Using fixed costs of (30,10), we compare the average net utility of three control policies. The first is TurKontrol(2). The second, TurKit, is a fixed policy from the literature [9]; it performs as many iterations as possible until its fixed allowance (400 in our experiment) is depleted and on each iteration it does at least two ballots, invoking a third only if the first two disagree. Our third policy, TurKontrol(fixed), combines elements from decision theory with a fixed policy. After simulating the behavior of TurKontrol(2), we compute the integer mean number of iterations, $\mu_{imp}$ and mean number of ballots, $\mu_b$, and use these values to drive a fixed control policy ($\mu_{imp}$ iterations each with $\mu_b$ ballots), whose parameters are tuned to worker fees and accuracies.


Figure 5: Net utility of three control policies averaged over 10,000 simulation trials, varying mean error coefficient, $\gamma$. TurKontrol(2) produces the best policy in every cases.

Figure 5 shows that both decision-theoretic methods work better than the TurKit policy, partly because TurKit runs more iterations than needed. A Student's t-test show all differences are statistically significant with $p$ value 0.01. We also note that the performance of TurKontrol(fixed) is very similar to that of TurKontrol(2), when workers are very inaccurate, $\gamma$=4. Indeed, in this case TurKontrol(2) executes a nearly fixed policy itself. In all other cases, however, TurKontrol(fixed) consistently underperforms TurKontrol(2). A Student's t-test results confirm the differences are all statistically significant for $\gamma < 4$. We attribute this difference to the fact that the dynamic policy makes better use of ballots, *e.g.*, it requests more ballots in late iterations, when the (harder) improvement tasks are more error-prone. The biggest performance gap between the two policies manifests when $\gamma$=2, where TurKontrol(2) generates 19.7% more utility than TurKontrol(fixed).

**Robustness in the Face of Bad Voters** As a final study, we considered the sensitivity of the previous three policies to increasingly noisy voters. Specifically, we repeated the previous experiment using the same error coefficient, $\gamma_x$, for each worker's improvement *and ballot* behavior. (Recall, that we previously set the error coefficient for ballots to one half $\gamma_x$ to model the fact that voting is easier.) The resulting graph (not shown) has the same shape as that of Figure 5 but with lower overall utility. Once again, TurKontrol(2) continues to achieve the highest average net utility across all settings. Interestingly, the utility gap between the two TURKONTROL variants and TurKit is consistently bigger for all $\gamma$ than in the previous experiment. In addition, when $\gamma$=1, TurKontrol(2) generates 25% more utility than TurKontrol(fixed) — a bigger gap seen in the previous experiment. A Student's t-test shows all that the dif-

ferences between TurKontrol(2) and TurKontrol(fixed) are significant when $\gamma < 2$ and the differences between both TURKONTROL variants and TurKit are significant at all settings.

## Related Work

Automatically controlling a crowd-sourcing system may be viewed as an agent-control problem where the crowd-sourcing platform embodies the agent's environment. In this sense, previous work on the control of software agents, such as the Internet Softbot [5] and CALO [13] is relevant. However, in contrast to previous systems, our situation is more circumscribed; hence, a narrower form of decision-theoretic control suffices. In particular, there are a small number of parameters for the agent to control when interacting with the environment.

Several researchers are studying crowd-sourcing systems from different perspectives. Ensuring accurate results is one essential challenge in any crowd-sourcing system. Snow *et al.* [15] observe that for five linguistics tasks, the quality of results obtained by voting a small number inexperienced workers can exceed that of a single expert, depending on task, but they provide no general method for determining the number of voters *a priori*.

Several tools are being developed to facilitate parts of this crowd-sourcing process. For example, TurKit [9], `Crowdflower.com`'s *CrowdControl*, and `Smartsheet.com`'s *Smartsourcing* provide simple mechanisms for generating iterative workflows, and integrating crowd-sourced results into an overall workflow. All these tools provide operational conveniences rather than any decision support.

Crowdsourcing can be understood as a form of human computation, where the primary incentive is economical. Other incentive schemes include fun, altruism, reciprocity, reputation, *etc.* In projects such as Wikipedia and open-source software development, community-related motivations are extremely important [8]. Von Ahn and others have investigated *games with a purpose* (GWAP), designing fun experiences that produce useful results such as image segmentation, optical character recognition [17]. Crowdflower has integrated Mechanical Turk job streams into games [3] and developed a mechanism whereby workers can donate their earnings to double the effective wage of crowd-workers in impoverished countries — thus illustrating the potential to combine multiple incentive structures.

## Conclusions

We introduce an exciting new application for artificial intelligence — control of crowd-sourced workflows. Complex workflows have become a commonplace in crowd-sourcing and are regularly employed for high quality output. We use decision-theory to model a popular class of iterative workflows and define equations that govern the various steps of the process. Our agent, TURKONTROL, implements our mathematical framework and uses it to optimize and control the workflow. Our simulations show that TURKONTROL is robust in a variety of scenarios and parameter settings, and results in higher utilities than previous, fixed policies.

We believe that AI has the potential to impact the growing thousands of requesters who use crowd-sourcing by making their processes more efficient. To realize our mission we plan to perform three important, next steps. First, we need to develop schemes to quickly and cheaply learn the many parameters required by our decision-theoretic model. Secondly, we need to move beyond simulations, validating our approach on actual MTurk workflows. Finally, we plan to release a user-friendly toolkit that implements our decision-theoretic control regime and which can be used by requesters on MTurk and other crowd-sourcing platforms.

## References


[1] D. Bertsekas. *Dynamic Programming and Optimal Control, Vol 1, 2nd ed.* Athena Scientific, 2000.

[2] E. Brunskill, L.Kaelbling, T.Lozano-Perez, and N. Roy. Continuous-state POMDPs with hybrid dynamics. In *ISAIM'08*, 2008.

[3] Getting the gold farmers to do useful work, October 2009. `http://blog.doloreslabs.com/`.

[4] A. Doucet, N. De Freitas, and N.J. Gordon. *Sequential Monte Carlo Methods in Practice*. Springer, 2001.

[5] O. Etzioni and D. Weld. A softbot-based interface to the Internet. *C. ACM*, 37(7):72–6, 1994.

[6] L. Hoffmann. Crowd control. *C. ACM*, 52(3):16–17, March 2009.

[7] L. Kaebling, M. Littman, and T. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99–134, 1998.

[8] S. Kuznetsov. Motivations of contributors to Wikipedia. *ACM SIGCAS Computers and Society*, 36(2), June 2006.

[9] Greg Little, Lydia B. Chilton, Max Goldman, and Robert C. Miller. TurKit: Tools for Iterative Tasks on Mechanical Turk. In *Human Computation Workshop (HComp2009)*, 2009.

[10] Contact center in the cloud, December 2009. `http://liveops.com`.

[11] Mausam, Emmanuel Benazera, Ronen Brafman, Nicolas Meuleau, and Eric Hansen. Planning with continuous resources in stochastic domains. In *IJCAI'05*, 2005.

[12] Mechanical turk is a marketplace for work, December 2009. `http://www.mturk.com/mturk/welcome`.

[13] B. Peintner, J. Dinger, A. Rodriguez, and K. Myers. Task assistant: Personalized task management for military environments. In AAAI Press, editor, *IAAI-09*, 2009.

[14] S. Russell and E. Wefald. *Do the Right Thing*. MIT Press, Cambridge, MA, 1991.

[15] Rion Snow, Brendan O'Connor, Daniel Jurafsky, and A. Ng. Cheap and fast — but is it good? evaluating non-expert annotations for natural language tasks. In *EMNLP'08*, 2008.

[16] Topcoder, December 2009. `http://topcoder.com`.

[17] Luis von Ahn. Games with a purpose. *Computer*, 39(6):92–94, 2006.

[18] Crowdsourcing, December 2009. `http://en.wikipedia.org/wiki/Crowdsourcing`.