Fair Work: Crowd Work Minimum Wage with One Line of Code

Mark E. Whiting, Grant Hugh, Michael S. Bernstein
Stanford Computer Science
353 Serra Mall
Stanford, California 94305
mwhiting@cs.stanford.edu

Abstract

Accurate task pricing in microtask marketplaces requires substantial effort via trial and error, contributing to a pattern of worker underpayment. In response, we introduce Fair Work, enabling requesters to automatically pay their workers minimum wage by adding a one-line script tag to their task HTML on Amazon Mechanical Turk. Fair Work automatically surveys workers to find out how long the task takes, then aggregates those self-reports and auto-bonuses workers up to a minimum wage if needed. Evaluations demonstrate that the system estimates payments more accurately than requesters and that worker time surveys are close to behaviorally observed time measurements. With this work, we aim to lower the threshold for pro-social work practices in microtask marketplaces.

Microtask crowdsourcing remains a precarious income source for crowd workers (Gray and Suri 2019; Kittur et al. 2013; McInnis et al. 2016; Martin et al. 2014; Hara et al. 2018). On the most popular platforms, no minimum wage rate is enforced. Some requesters, or task authors, explicitly take advantage of this to underpay workers; others are well intentioned but still accidentally underpay because expertise bias leads us to underestimate how challenging our tasks actually are (Hinds 1999). The platforms' own time reports can be notoriously inaccurate in estimating and visualizing the time that workers spend (Rzeszotarski and Kittur 2011). Workers compensate for underpayment by mobilizing tools that help them identify reasonable requesters (Hara et al. 2018; Irani and Silberman 2013; Hanrahan et al. 2015). Requesters, for their part, typically need to put in substantial effort to test their tasks across a variety of possible workers in order to price effectively (Gaikwad et al. 2017; Cheng, Teevan, and Bernstein 2015), and there is no accurate straightforward approach.

In this paper, we offer an approach that only requires a single line of code to help ensure that microtask workers make at least a minimum wage. Existing approaches retain a high threshold of effort (Myers, Hudson, and Pausch 2000); our goal is to drastically lower the effort threshold. If this is feasible, we hope to expand the set of pro-social researchers and requesters who opt in to a norm of fair pay.

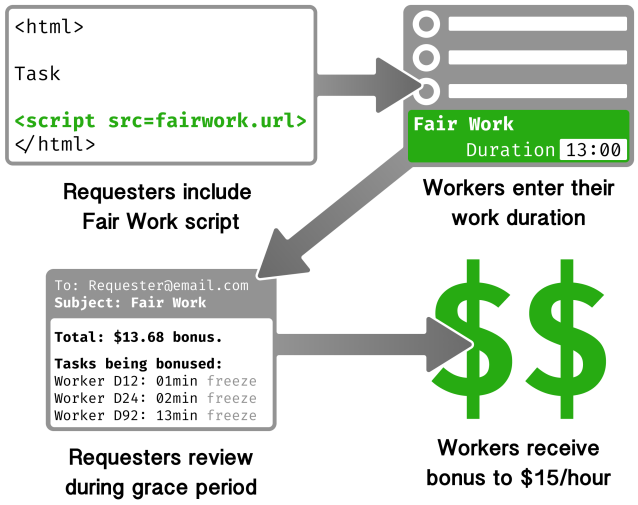

Figure 1: Fair Work uses a custom script to add a duration input to the bottom of Amazon Mechanical Turk tasks. Workers' durations are used to calculate a fair pay and send bonuses when needed.

We introduce Fair Work for Amazon Mechanical Turk (https://fairwork.stanford.edu), a Javascript script that can be added as a single `<script>` tag to any task HTML to help ensure that workers are paid at least minimum wage (Figure 1). Requesters register their API keys with Fair Work, and then add a server-generated Fair Work script tag to any of their tasks. The script adds a self-report time survey to the task asking workers to share how long the task took them to complete, or to click to confirm an automatically detected active work time. Fair Work then aggregates worker time reports in order to estimate the median time it takes workers per task. If the task underpaid minimum wage, Fair Work queues a bonus. Requesters receive a daily summary of the time reports and any queued bonuses; after a twelve-hour grace period to correct any errors or malfeasance, Fair Work uses the requesters' AWS keys to auto-bonus all workers on the task to bring them up to the minimum wage rate. For re-

questers who do not wish to store encrypted API keys on a centralized server, Fair Work is also available open source at https://github.com/StanfordHCI/fairwork and can be deployed privately on Heroku or the requester's own server.

Fair Work departs from several of the norms of similar tools and techniques. Most visibly, it makes no guarantees to protect the requester against malicious workers. Instead, the tool makes a strong assumption that most workers will make their estimates in good faith, relying on medians to filter out any stray false reports. As a last resort, the tool allows requesters to freeze payments for the small minority of workers who persistently cause issues (e.g., reporting two hours on tasks that took most workers only two minutes). The assumption here is that the vast majority of workers are honest and will report honestly or with only slight inflation (Ariely 2008; Hancock, Toma, and Ellison 2007), and that even slight inflation could be considered compensation for the considerable unpaid time that workers spend searching for tasks (Horton and Chilton 2010). In other words, Fair Work is not strategyproof: instead, it asserts an expectation of honesty and goodwill on behalf of all parties who use it, and allows requesters to handle rare deviant behavior rather than assuming many actors are deviants.

We report both a formative and a summative evaluation of the tool. In our formative work, we surveyed American Mechanical Turk Workers about their greatest frustrations with requester behavior on the platform, identifying underpayment as the single largest issue, followed by unfair rejection. A second survey identified a "Fight for Fifteen" $15 per hour wage rate (Rolf 2016) as the desired wage rate. In our summative evaluation, we recruited experienced requesters to provide tasks and attempt to price them at an hourly wage of $\$15$ per hour. We collected 19 unique tasks, added the Fair Work script, and posted the tasks on Amazon Mechanical Turk. We found that requesters mispriced their tasks by $\$7.80$ per hour on average, effectively pricing between 50% and 150% of the intended wage rate, and that workers mildly exaggerate their task completion times.

In sum, this paper presents Fair Work as a low-threshold approach that requesters can use to help encourage fair wage rates. We further contribute a formative evaluation demonstrating that underpayment remains a pervasive issue for workers, and a summative evaluation supporting our claims that requesters struggle to price effectively and that workers will generally report honestly.

Related Work

Studies of worker motivation on Mechanical Turk have focused increasingly on wages. Early surveys reported multiple overlapping motivators (Ipeirotis 2010; Kaufmann, Schulze, and Veit 2011) including intrinsic enjoyment, and detailed variations in motivation across different geographic regions (Antin and Shaw 2012). However, more recent work has called out wages more visibly as a central matter of importance for workers (Kaplan et al. 2018; Martin et al. 2014; Gray and Suri 2019). New participant populations in the crowd work ecosystem immediately gravitate to extrinsic financial motivation as the primary reason to participate (Brewer, Morris, and Piper 2016).

Worker wages have become a focal point of study and attention. Estimated wages on Amazon Mechanical Turk are low (Horton and Chilton 2010); data-driven analyses suggest that the mean and median hourly wage are about $2 to $3 (Hara et al. 2018), less than half of the U.S. federal minimum wage despite 75% of active workers residing in the United States (Ipeirotis 2010). Rejections and time spent searching for tasks contribute to this low wage (Chilton et al. 2010; McInnis et al. 2016). There are now calls pushing for researchers to pay workers at least minimum wage (Silberman et al. 2018). The platform Prolific Academic (https://www.prolific.co/) asks requesters to pay a minimum wage, though many others such as Amazon Mechanical Turk do not.

Over time, understandings of the crowd worker experience have grown to more fully encompass the broad spectrum of social and workplace issues that workers face. These issues are modern instantiations of the historical practice of piecework (Alkhatib, Bernstein, and Levi 2017). Much of the modern understanding has been built up through studies of worker online communities such as Turker Nation, Reddits /r/mturk, and MTurk Crowd. In these communities, workers share information on how to make money (Martin et al. 2014) and support each other (Gray et al. 2016). Beyond this, workers face structural challenges ranging from precarity from work rejections (McInnis et al. 2016) to second-language issues (Gupta et al. 2014).

Worker communities have taken many steps to improve their environment. Collective action is one extremely visible option (Bederson and Quinn 2011). Workers have sued crowdsourcing platforms (Otey 2015) and engaged in strikes (Conger, Xiuzhong Xu, and Wichter 2019) in part to seek higher wages, and have engaged in collective action online to call attention to their situation (Salehi et al. 2015). They also adopt tools to help avoid the worst parts of the marketplace. For example, tools such as Turkopticon (Irani and Silberman 2013) allow workers to share information about low-quality requesters; TurkBench (Hanrahan et al. 2015), Crowd-workers (Callison-Burch 2014), and TurkScanner (Saito et al. 2019) all surface information about tasks and completion times so that workers can plan their work. Such tools are best co-designed with workers, rather than "cast[ing] Turkers as dopes in the system" (Irani and Silberman 2016). Fair Work focuses on tools for requesters, with the goal of helping fix the wage problem upstream.

Current requester tools are focused on helping requesters overcome low-quality work or malicious workers. One set of tools supports recruitment, for example modulating whether another task is needed to obtain high confidence labels (Sheng, Provost, and Ipeirotis 2008), inserting known answers into the question stream to test workers, recruiting workers quickly (Bernstein et al. 2011; Lasecki et al. 2011), or classifying likely inattentive workers (Rzeszotarski and Kittur 2011). A smaller set of techniques have been developed for helping requesters design tasks (Gaikwad et al. 2017; Bragg, Weld, and others 2018) and price tasks (Cheng, Teevan, and Bernstein 2015). While these tasks do attempt to lower the threshold for requesters to author effective tasks, they all require the requester to launch the task, gather feed-

back and data, and iterate. We hypothesize that many requesters want to pay fairly, but are put off by the requirement to iterate and prefer a solution that they do not need to babysit. So, unlike prior work, we designed Fair Work to not require any intervention by paying a post-hoc bonus to workers. Workers can see the Fair Work interface on their tasks, and understand that the effective pay rate may be higher than advertised.

This prior work together paints a picture where wages are a primary motivator for workers on platforms such as Amazon Mechanical Turk, but where wages are disappointingly meager. Workers have tools to navigate the situation, but often underpayment is due to requester error; many requesters wish to pay fairly, but payment is not their priority when rushing to push out their work. Fair Work contributes beyond this prior by introducing the first tool for no-intervention payment for requesters, and by communicating a clear promise of minimum wage payment based on worker reports.

Formative Surveys

While wage issues have been detailed carefully by prior work, several questions remained unanswered:

1. How do wage issues compare, in workers' eyes, to other requester behaviors?
2. What wage rate do workers feel is a fair compensation?
3. How do workers feel about the accuracy of self-reporting their time spent on a task?
4. How do workers feel that differences of opinion should be resolved if requesters feel that workers are not reporting time honestly?

We performed a series of two surveys on Amazon Mechanical Turk to answer these questions. Our first survey was launched to 112 workers in the United States — the vast majority of workers are in the U.S. (Ipeirotis 2010).[1] The survey paid $0.90 per response, which based on our estimates would target the minimum wage in our location. After filtering out incomplete or unusable data, we were left with N=101 responses.

By aggregating freetext responses from pilot surveys and by drawing on prior work (Irani and Silberman 2013), we identified a list of requester behaviors that workers found frustrating: (i) Approving work too slowly, (ii) Blocking workers, (iii) Broken HITs, (iv) Not being responsive to questions and emails, (v) Rejecting work unfairly, (vi) Underpayment, (vii) Violating Mechanical Turk's Terms of Service, and (viii) Violating worker privacy. In our survey, we asked workers to first select up to three behaviors from the list above that they felt were the most *common* errors, and then to select up to three behaviors that they felt were the most *frustrating* errors. Reports of frustrating and common requestor behaviors were highly correlated ($R >= 0.9$), so we only report on the results for most frustrating in this paper, as those capture workers' affective experiences. The

[1]We also replicated these results with the additional restriction that workers had completed at least 10,000 tasks on the platform.

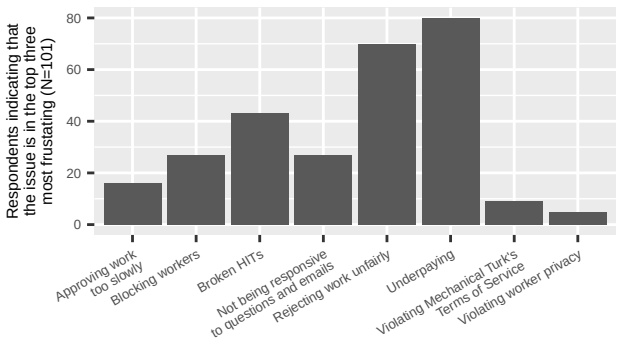

Figure 2: Workers report that underpayment is the single most frustrating requester behavior they experience on Amazon Mechanical Turk.

survey also asked workers to choose what they felt would be the fairest minimum wage for tasks out of the following options: (i) No Minimum Wage, (ii) Fight for Fifteen (Rolf 2016): $15/hr, (iii) Washington DC Minimum Wage: $13.25/hr, (iv) Highest State Minimum Wage: $11.50/hr, (v) Federal Minimum Wage: $7.25/hr, (vi) Requester Location Minimum Wage, (vii) Worker Location Minimum Wage, and (viii) Other. We also asked workers to explain why they felt that the minimum wage they picked was fair in their view.

The second block of survey questions asked workers about honest reporting. We first asked whether workers feel that they themselves might exaggerate how long a task takes in order to increase their wage. Then, drawing inspiration from the Bayesian Truth Serum (Prelec 2004), we asked them what percentage of other workers they felt might exaggerate their time on task to increase their wages.

Across all surveyed requester behaviors, workers reported underpayment as the single most frustrating issue (Figure 2), with 79% of workers including it in their top three most frustrating requester behaviors. Unfair rejection trailed it with 69% of workers reporting it in their top three, and broken HITs with 42%. All other issues were at or below one quarter of workers reporting them. The dominance of the underpayment issue is a striking evolution of workers' points of view over the past half decade since Irani and Silberman found that fewer than a third of Mechanical Turk workers even mentioned underpayment as an issue (2013).

Prior work a half decade ago suggested that workers oriented around a fair pay rate of the federal minimum wage, $7.25 per hour (Martin et al. 2014). Our data suggest that this norm has also shifted drastically, with the "Fight for Fifteen" (Rolf 2016) $15 per hour (36% of responses) as the most popular response as to what constitutes a fair wage. Federal minimum wage (15%) and no minimum wage (6%) were far less popular options.

In qualitative responses, workers focused on both ethical imperatives and practical requirements in justifying their choice:

"The push for fifteen dollars an hour minimum wage

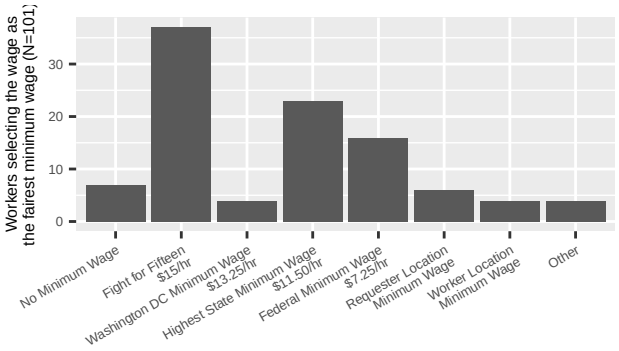

Figure 3: The "fight for fifteen" (Rolf 2016) wage of $15 per hour was perceived by workers as the fairest wage for their labor.

> is a push to make work and wages livable; if people are working on Mechanical Turk and taking it seriously like a professional skill (which I do) they should be compensated a fair living wage."
>
> "It's ludicrous to think that anything under $15 is a remotely livable wage. You can't live off of that. Skipping meals, forgoing doctor visits, living in awful conditions, working insanely long hours are all part of the current wage scheme. Working online is still working! It's right in the name. Why should they be exempt from labor laws?"
>
> "If you factor in self-employment taxes, 13.25/hr goes from good pay to fair."

The minority of workers who disagreed felt that minimum wages should not be enforced, or that it should not be universalized:

> "I don't think it's fair to force someone to pay someone else. Workers and Requesters are free to work together or not. If Requesters price the work too low, no one will do it. Then they can raise the rate they're paying. But if the work is getting done, it's clearly worth someone's time to do it. Low pay is frustrating but that's life."
>
> "Because minimum wage is not the same across the country and should not be treated as so."

30% of workers reported that they might stretch their own time reports occasionally. They felt on average that 50% of workers would do so ($\sigma = 25\%$). Prior work suggests that, when given a chance to cheat, many people do cheat, but only by slightly fudging the truth rather than engaging in big lies (Ariely 2008). In the Evaluation section, we will directly measure the extent of this time exaggeration.

From this first survey, we took a clear design imperative that addressing low wages was the most important first goal for a requester tool, and that $15 per hour was the appropriate fair wage standard around which to anchor our system. However, there still remained questions of how to handle situations where requesters and workers disagreed about the veracity of time reports. So, we launched a second survey (N=31) to the same population as the first survey. This second survey asked workers to rank three options for how to manage situations when a requester believes that a worker has not reported honestly: (i) Requesters freeze bonus payments, worker is notified that they have been flagged for suspected falsification of time via email, and then workers can reply in certain period of time to discuss the situation; (ii) When a requester flags worker for suspected falsification of time, all other workers who have previously used Fairwork are notified via email and can weigh in on what action should be taken; and (iii) If a worker is flagged three times for suspected falsification of time, bonus payments are automatically stopped for the worker. We also surveyed workers using Likert scales to rate the importance of possible features of an arbitration system.

64% of respondents selected the first option — a freeze followed by an email discussion — as their top choice. The other options were far less popular: 26% selected the peer jury, and 10% selected the three-strikes approach. In explaining their decisions, the workers made clear that conversation is the right way for workers and requesters to engage:

> "I prefer the option that allows the worker to discuss the situation because there are many reasons that could lead to a faulty completion time including an adblocker, broken script, or a mistake."

As seen in prior work (Whiting et al. 2017), a minority of "lone wolf" workers wanted the system to avoid any reliance on other workers' reports or judgments:

> "involving other workers is always a terrible idea. Have you met some of us??!"

Of the set of characteristics for an arbitration system, the highly-rated ones involved having an opportunity for workers to explain themselves ($\mu = 6.10/7, \sigma = 1.74$) and a well-outlined procedure ($\mu = 5.93, \sigma = 1.61$); the least popular characteristic was consulting other workers ($\mu = 1.52, \sigma = 2.01$).

Based on the results of this survey, we designed Fair Work to allow requesters to temporarily freeze payment and then engage in communication with requesters before unfreezing payments.

Fair Work

Fair Work is a Javascript tool that requesters can insert into their Amazon Mechanical Turk tasks in order to ensure that their workers are paid at least a minimum wage. Its main goal is to provide a low-threshold (low-effort) solution for requesters who wish to be pro-social in their payment strategies but find traditional payment approaches challenging and error-prone. In naming this platform, we drew on the language of advocacy around fair wages. We acknowledge that what constitutes a "fair wage" is and will always be contested, and we hope that the wage that Fair Work offers will likewise update over time.

At a high-level, Fair Work provides a HTML `<script>` element that requesters add to the HTML of their HIT. Once the page loads, Fair Work inserts an iframe into the task, which workers can use to self-report the time it takes them

to complete the task. Daily, the Fair Work server aggregates the median time report per task across workers, compares that median rate to the minimum wage rate, and bonuses all workers who completed the task if necessary to ensure that their earnings are at least minimum wage.

Requesters who want to use Fair Work must first register, which allows Fair Work to issue bonuses on their behalf. To do so, they must agree to the Fair Work IRB and upload their AWS keys. The AWS keys, which Fair Work uses to track HIT completion and send bonuses, are encrypted and stored on the server. After registration, the requester receives a unique `<script>` tag that they can copy and paste into the HTML of their tasks either in the Mechanical Turk web interface or in any self-hosted tasks on their own server. Upon inserting the Fair Work `<script>` tag, the requester will see a Fair Work iframe containing a description of Fair Work and the worker IRB agreement appended to the bottom of their task.

Once the task is published, workers previewing the task will also be able to see the same Fair Work iframe. Workers who accept the task will see a checkbox to agree to the Fair Work IRB. The workers who agree to the IRB will be presented with an input box where they can record the amount of time it took them to complete the task. We opt for a self-report time rather than automatically tracked duration because many tasks occur in other windows (e.g., Qualtrics surveys) or otherwise interfere with efforts to automatically track time. Fair Work prompts the time report by asking:

> Get paid fairly: This requester is using the Fair Work script to bring pay rates up to the minimum wage of $15/hr. Fair Work does this by measuring completion times and then auto-bonusing workers to meet the desired hourly wage if needed.
>
> Please report how much active work time this task took you, rounded to the nearest half minute. If you are far off from the median time report across all workers for this task, indicating a lack of good-faith estimation, you may be removed from Fair Work bonus eligibility. Bonuses are sent out daily. By participating, you acknowledge and consent to the Fair Work IRB.

Workers can then enter the number of minutes it took them to complete the task into the Fair Work textbox. Fair Work also provides two accelerator options: (i) estimated time, a time updated to reflect the length of active time the user has been active on the tab (Zissman 2019); (ii) last time, the time that the worker reported on the last HIT they completed of this task type.

Once per day, for each task type, Fair Work will aggregate all the reported times per worker, take the median per worker to calculate an estimated time per worker for that task, then take the median of those estimated times across workers to estimate the effective time required to complete the task. Given the effective time and the actual payment for the task, Fair Work calculates the bonus necessary to bring payment up to $15/hr. For example, if the requester originally set the payment for the task at $1 but the median reported time across workers was 20 minutes, then the correct payment should have been $5, and Fair Work would bonus each worker who did the task $4 per task.

Once per day, Fair Work sends an email to the requester summarizing its calculations and the total bonus amount that it will be sending to workers. To help the requester audit how the bonuses were calculated, the email breaks down the median time that each worker reported on the task. If the requester does nothing, Fair Work sends the bonus to each worker — including those who completed the task but never reported times on the task — twelve hours after the summary email is sent.

Fair Work pays the bonuses to the workers through the Mechanical Turk API using the requester's AWS keys. The message attached to the bonus contains similar information as the requester received, summarizing the estimated time per task and why they are receiving the funds. If the requester does not have enough funds in their account to pay the bonus, both the worker and requester receive an email that the requester is short funds, and the system tries again later to re-send the bonus.

If the requester notices a worker who has reported egregiously long times, they may freeze payment to that worker while they resolve the situation. Freezing has the effect of removing that worker's reports from the pending estimates, and blocking the worker from future Fair Work reports for that requester, until the requester removes the freeze. The Fair Work interface stresses that freezing should only be done if it is necessary. During the twelve hour grace period after receiving the summary email, requesters may place a freeze by clicking a freeze link next to that worker's reports in the summary email. Requesters write an explanation of the freeze, which is sent to the worker in an email notifying them that they have been frozen. The email contains the requester's email address, allowing the worker to contact the requester to manage any disputes. When a requester issues a freeze, Fair Work recalculates the estimated time for all pending tasks excluding the frozen worker's time reports and sends a new email with revised pending bonus payments.

Implementation

Fair Work is implemented as a Django application that runs on Heroku. Requesters' AWS keys are stored using Fernet symmetric encryption on the server, and unencrypted when needed using a private key. When signing up for Fair Work, the system uses the AWS keys to generate a script embed that uses that requester's AWS account ID (e.g., `<script src="https://fairwork.stanford.edu/fairwork.js?aws_account=12345"></script>`). If an AWS account ID is used on tasks that are not created by that account, the script errors. The script, on page load, appends an `iframe` to the end of the DOM. This iframe, which loads on the Fair Work server, contains the worker IRB agreement and time report interface. Time reports are sent asynchronously as workers type them. Iframes are necessary in this environment to manage modern browsers' restrictions on cross-site scripting.

The code is available under the MIT open-source license at https://github.com/StanfordHCI/fairwork, and can be deployed by any requester to a private server (Heroku or otherwise) if they do not want the Fair Work server to hold their

encrypted AWS keys. The system alerts workers if the task they are looking at does not use the Fair Work server. We ask requesters who use the open-source code to not disable this warning or lower the minimum wage.

Design space and tradeoffs

In creating Fair Work, we faced a number of value-laden decisions and tradeoffs. In this section, we briefly review these decisions to support future researchers and practitioners in the space.

Minimum Wage Our survey indicated that workers felt it was fairest to pay $15 per hour today; however, requesters might find this rate high. How do we adjudicate this?

We experimented with local minimum wage based on requester location and worker location. Ultimately, we felt that a requester-based minimum wage would be confusing for workers, since they would not be able to get a simple guarantee from the presence of the Fair Work script: requesters living in the city of Chicago, for example, would be paying $12.00/hr, but those just a mile outside of Chicago would only be paying the Illinois state minimum wage of $8.25/hr. The same goes for workers: paying each worker based on their local minimum wage, e.g. with the Min-Wage Retrieval API (Mankar, Shah, and Lease 2017), could be confusing to requesters, and might lead to geographic bias in recruitment.

So, we sought a single approach that could be applied consistently. We experimented with an expected value national wage by weighing local minimum wages in the United States by the proportion of the population that lived in each locality, but this resulted in an $8.78 wage, far below workers' fair wage feedback in the survey. Ultimately, we decided (1) that a single minimum wage provided the clearest mental model to workers and requesters, and that (2) requesters using the tool were doing so with the goal of being pro-social, and so many would either already be in $15/hr localities or be aligned with the $15/hr efforts in their locality. So, we decided that a $15/hr minimum wage was the clearest and fairest option for Fair Work.

Scheduling How quickly should Fair Work bonus workers? How much time should requesters have to audit the results before the bonuses are sent? The answers to these two questions are in tension: workers need prompt payment (Irani and Silberman 2013) and requesters need time to ensure that Fair Work's estimates are reasonable. In addition, because requesters may change tasks over time without updating the HITGroup (the ID that the platform uses to group tasks), Fair Work needs some short period of hysteresis over which to aggregate time self-reports: this allows the estimated wage to update as the task changes.

In this case, we started with a constraint on how long workers feasibly wait for approval and payment. Based on interactions with workers, we set this limit at about one day after work is approved. Then, we worked backward to identify a feasible window for requesters to vet the results. Typically, there is nothing for requesters to do, so we chose to give requesters a twelve-hour window. In other words: every twenty-four hours, Fair Work calculates bonuses for any tasks that have been completed in the previous twenty-four hours. After that, requesters have twelve hours to intervene if needed before the bonuses are sent out. Lengthening this schedule might stress workers but would give requesters more flexibility; shortening it would stress requesters but give workers more certainty.

Freezing Fair Work defaults to trusting that workers will submit honest time reports and thus the earliest version of the system had no capacity for freezing bonuses. However, testing this early prototype reminded us that some workers would misunderstand or abuse the system. Therefore, we added the freeze system as a way to give requesters a grace period to ensure nothing looks wrong, and to control any bad actors.

We examined several alternatives. The first was worker adjudication: if a requester feels that a worker is lying, they flag the report, and both the worker and requester's explanations go to a panel of impartial workers to adjudicate and decide. The second was a three-strikes rule: if several requesters independently flagged the worker, that worker would be excluded from all Fair Work bonuses going forward. Surveyed workers far preferred the approach that encouraged individual conversation between the worker and requester, which is why we developed the freeze framework to start an email conversation. For now, our team adjudicates any arguments that cannot be worked out interpersonally; however, we have not yet been asked by any worker or requester to intervene in this way.

Effective wage calculation Fair Work calculates the effective wage rate for a task by taking the median time report per worker, then taking the median across workers. Why is this approach fair, if a median means that half of workers will be earning below this rate?

An alternative architecture might bonus workers individually based on their own self-reported time. This would be a reasonable approach and a viable future work direction for the project, should requesters prefer it. The computed median could then be used for any workers who do not report their time. We chose a unified estimate per task for Fair Work because it would be a familiar mental model for workers and requesters: each task has an estimated wage rate and is paid based on that.

A second alternative might be to use a statistic rather than the median, such as 25th percentile or the mean. The median has the important advantage of being more resilient to outliers. When there are many reports, individual strategic behavior has no practical effect and requesters do not need to worry about freezing. When there are a small number of reports, the median could be skewed, but in this case it is easy for requesters to identify and freeze the few problematic outliers. This prevents the requester from needing to be burdened with freezing a large number of reports. More complex methods could have flagged outliers automatically, but we felt that it would make the methodology less interpretable to the requester; if requesters do not understand how bonuses are calculated, it will be difficult for them to trust our system and gauge whether or not the bonuses are equitable. Furthermore, freezing would be much harder for

requesters because they would not know how freezing one extraneous time report will affect the bonus recalculation.

Evaluation

To evaluate Fair Work, we sought to answer two questions. First, does Fair Work support requesters in avoiding task underpayment? Second, how accurate are workers in their time estimates they provide to Fair Work?

We recruited Amazon Mechanical Turk requesters who had posted at least 2 different tasks in the past, and collected 19 unique tasks types that they had previously created. We relaunched them on Amazon Mechanical Turk while using the Fair Work script.

We launched at least 10 assignments (tasks for different workers) for each of up to 10 different HITs (specific tasks) per task type. We made the tasks available to Amazon Mechanical Turk workers with any relevant qualifications used by the requesters on the tasks they provided. When no qualifications were specified by the requester, we used workers with a minimum of 1000 accepted tasks on the platform. We restricted tasks to workers based in the USA. This method resulted in a total of 736 completed tasks for the experiment. We additionally asked each requester to estimate a piecework rate for their task if they were intending the task's hourly wage to be $15 per hour, in alignment with the pay rate goal in Fair Work. A total of 259 unique workers contributed to this experiment.

Extending the initial design of Fair Work, the system we used for our experimentation also included a timer script that measured the number of seconds the task was a focused window and the worker was interacting with the window (Zissman 2019) to provide baseline comparison between reported and actual work times. We instrumented the system to measure the actual time on the task, independent of whether the workers self reported their work times in Fair Work, allowing us to also consider how selection effects may impact workers' actual median work times. For our analyses, we filtered out two task types that required completion off-tab or in other ways that the automatic time tracking would be inaccurate.

How accurately do requesters price?

We compared the target pay rate of $15 per hour to the median observed completion times on the tasks. Requesters mispriced their tasks by $7.80 ($\sigma = \$3.50$) on average across all the unique tasks we evaluated. In other words, the average task had an error on the order of paying roughly 50% of the target wage ($7.20) or 150% of the target wage ($22.80). Requester inaccuracies in setting the correct price for their tasks both over and undershot the Fair Work rates (Figure 4). More tasks underpaid than overpaid, with 68% underpaid.

This result reinforces that even when trying to target a specific wage, requesters often underestimate how long the task takes and thus underpay. Some requesters were able to price effectively, and some overpaid, but the plurality would feature Fair Work price adjustments.

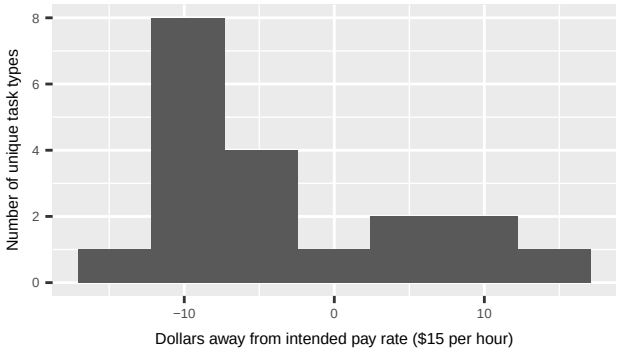

Figure 4: When targeting a $15 per hour wage, requesters tend to assume they are paying more than they actually are.

How accurately do workers report durations?

259 of the workers on the task opted in to data collection via our IRB agreement. Of the 736 tasks these workers completed, 50% had at least one self-reported work duration for the worker on that task. Workers reported 1 duration per task type on average ($\sigma = 1.5$).

Comparing workers' median self reported work times ($\mu = 180, \sigma = 360$) to their median observed timer times ($\mu = 100, \sigma = 110$) per task via a paired t-test was significant, $t(186) = -6.02$, $p < .001$. The median difference for an individual worker between their observed and self-reported times is 38% of the observed time (Figure 5). This would mean, for example, a 60 second reported duration and an 43 second recorded duration. This result is consistent with prior work (Ariely 2008), a "generous rounding up" of timing rather than outright exploitation. The effect of this inflation is to shift the percentage of workers that achieve the Fair Work minimum wage. If workers' self-reports were exactly the same as their observed times, then using the median would suggest that 50% of workers would make more than the minimum wage and 50% would make less. By comparing workers' individual recorded times to the aggregate self-report estimates, we find that this inflation means that, in practice, 78% of workers are making at least the minimum wage.

Comparing average observed times between workers who did ($\mu = 132.74, \sigma = 132.33$) and did not ($\mu = 99.15, \sigma = 107.28$) self-report Fair Work times, we do not observe a significant difference with an unpaired t-test ($t(22.647) = 0.69963, p = 0.4913$). So, the workers who self-report times do not appear to be different in task completion times than those who do not report, indicating that using those who do report times to generalize appears to be a reasonable assumption.

Responses to Fair Work

We aggregate three sources of qualitative insight about Fair Work: responses from community outreach, responses from requesters when eliciting tasks for experiments, and responses from workers as they utilized the platform.

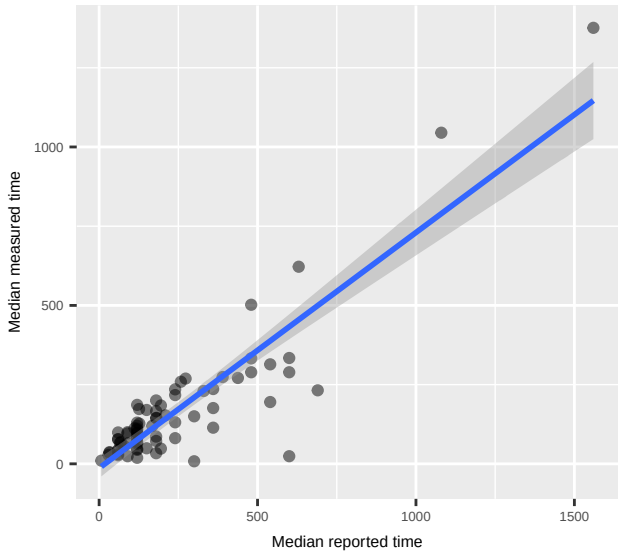

Figure 5: Workers report slightly longer task completion times, on average, than observed.

Workers appear interested by the idea of Fair Work, but some remain skeptical about the degree to which it will be used by requesters if not enforced by the platform. For example, a small number of worker responses on Twitter after our announcement of Fair Work represented a critical perspective that Fair Work would have to be implemented by Amazon Mechanical Turk to hold value. On the other hand, workers who interacted with Fair Work through our experiment responded very pragmatically, indicating interest, but also caution, since many of the experimental tasks were underpaid by their requesters:

> "I read with interest your Fair Work document; however, I am finding that your hits on mturk generally pay much less."

Requesters on the other hand in general reflected positively on the potential for Fair Work to help them price their tasks more accurately. One requester exclaimed reflected this when being providing tasks for our experiment:

> "When I make new tasks, I'm always afraid that I'm either going to annoy workers by paying too little, or be wasting money by paying way too much."

Some requesters reflected that the price rate was substantially higher than their usual hourly wage goal, but also noted that they had little insight as to the effective hourly wage workers on their tasks actually earn. One requester stated "I usually ask a friend to do my task and see how long they take," when asked about pricing methodology. However, they later mentioned that they had no idea if this method would give workers suitable wages.

Overall our qualitative results suggest that workers and requesters stand to benefit from using Fair Work but that buy-in is a two sided problem. The challenges of earning a fair wage on Amazon Mechanical Turk and of pricing tasks fairly is reflected in the responses of both groups. This suggests that the intention of the Fair Work system is desired by the community, but also that finding ways to ensure it is implemented and widely adopted will remain a challenge.

Discussion

One limitation of Fair Work is that, if only a few workers report times for a task, the estimate can easily become inaccurate. In this sense, it is in workers' best interest to report, and in requesters' best interest to request that workers fill in the report.

One way to view our data is that requesters often underprice their tasks, and that workers often over-estimate their time on task. This equilibrium, if it generalizes, could certainly help explain some of the friction that exists on the marketplace. Clearly there are a myriad of different individual experiences. However, finding a process that both parties agree is fair may help ameliorate the situation.

Our results suggest that Fair Work can aid requesters in ensuring that they meet a target wage, and support workers in achieving a minimum wage. However, it is important to consider ecosystem-level impacts. We must consider the possibilities that Fair Work might not achieve its goals, or that Fair Work will be adopted at a rate sufficient to see such effects.

For example, requesters might reduce the price on their tasks to make sure they don't overpay, and rely on Fair Work to compensate. This would complicate the usefulness of pricing signals for workers on the marketplace. In addition, for tools like Fair Work to achieve wide-scale change, it would require a focused group, such as researchers, to commit to using it in their work, essentially agreeing to enforcing a minimum wage for research conducted on Mechanical Turk in their work or in their peer review process. On the worker side, we cannot assess the extent to which Fair Work might encourage collusion within the worker community to systematically over report working times or to encourage slower work in general.

Future work must also investigate and iterate on how workers and requesters navigate disputes. Currently, if workers and requesters cannot reach agreement, the system encourages them to raise it to the tool administrators for an additional opinion. However, models based on peer juries may offer a more neutral and just system.

Conclusion

This paper introduces Fair Work, a tool that enables Amazon Mechanical Turk requesters to pay consistent hourly wages by adding only a single line of code to their tasks. Through a survey of workers, we validated the claim that underpayment is still a widespread issue, motivating the creation of Fair Work. We showed that the system prices more accurately than requesters, and that it leverages a signal, self reported work durations, that serves as a reasonable measure of actual work rates. Though Fair Work remains in its infancy, our hope is that other requesters will broadly adopt Fair Work,

and that the worker community will come to expect fair pay on all tasks in online labor markets.

Acknowledgments

We thank the workers and requesters who participated in this project, as well as those whose experiences influenced its creation. This project was supported by a National Science Foundation award IIS-1351131 and a Sloan Research Fellowship.

References

Alkhatib, A.; Bernstein, M. S.; and Levi, M. 2017. Examining crowd work and gig work through the historical lens of piecework. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, 4599–4616. ACM.

Antin, J., and Shaw, A. 2012. Social desirability bias and self-reports of motivation: a study of amazon mechanical turk in the us and india. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2925–2934. ACM.

Ariely, D. 2008. *Predictably irrational*. Harper Collins.

Bederson, B. B., and Quinn, A. J. 2011. Web workers unite! addressing challenges of online laborers. In *CHI'11 Extended Abstracts on Human Factors in Computing Systems*, 97–106. ACM.

Bernstein, M. S.; Brandt, J.; Miller, R. C.; and Karger, D. R. 2011. Crowds in two seconds: Enabling realtime crowd-powered interfaces. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, 33–42. ACM.

Bragg, J.; Weld, D. S.; et al. 2018. Sprout: Crowd-powered task design for crowdsourcing. In *The 31st Annual ACM Symposium on User Interface Software and Technology*, 165–176. ACM.

Brewer, R.; Morris, M. R.; and Piper, A. M. 2016. Why would anybody do this?: Understanding older adults' motivations and challenges in crowd work. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, 2246–2257. ACM.

Callison-Burch, C. 2014. Crowd-workers: Aggregating information across turkers to help them find higher paying work. In *Second AAAI Conference on Human Computation and Crowdsourcing*.

Cheng, J.; Teevan, J.; and Bernstein, M. S. 2015. Measuring crowdsourcing effort with error-time curves. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, 1365–1374. ACM.

Chilton, L. B.; Horton, J. J.; Miller, R. C.; and Azenkot, S. 2010. Task search in a human computation market. In *Proceedings of the ACM SIGKDD workshop on human computation*, 1–9. ACM.

Conger, K.; Xiuzhong Xu, V.; and Wichter, Z. 2019. Uber drivers' day of strikes circles the globe before the company's i.p.o. *New York Times*.

Gaikwad, S.; Chhibber, N.; Sehgal, V.; Ballav, A.; Mullings, C.; Nasser, A.; Richmond-Fuller, A.; Gilbee, A.; Gamage, D.; Whiting, M.; et al. 2017. Prototype tasks: improving crowdsourcing results through rapid, iterative task design. *arXiv preprint arXiv:1707.05645*.

Gray, M. L., and Suri, S. 2019. *Ghost Work: How to Stop Silicon Valley from Building a New Global Underclass*. Eamon Dolan Books.

Gray, M. L.; Suri, S.; Ali, S. S.; and Kulkarni, D. 2016. The crowd is a collaborative network. In *Proceedings of the 19th ACM conference on computer-supported cooperative work & social computing*, 134–147. ACM.

Gupta, N.; Martin, D.; Hanrahan, B. V.; and O'Neill, J. 2014. Turk-life in india. In *Proceedings of the 18th International Conference on Supporting Group Work*, 1–11. ACM.

Hancock, J. T.; Toma, C.; and Ellison, N. 2007. The truth about lying in online dating profiles. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, 449–452. ACM.

Hanrahan, B. V.; Willamowski, J. K.; Swaminathan, S.; and Martin, D. B. 2015. Turkbench: Rendering the market for turkers. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, 1613–1616. ACM.

Hara, K.; Adams, A.; Milland, K.; Savage, S.; Callison-Burch, C.; and Bigham, J. P. 2018. A data-driven analysis of workers' earnings on amazon mechanical turk. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 449. ACM.

Hinds, P. J. 1999. The curse of expertise: The effects of expertise and debiasing methods on prediction of novice performance. *Journal of experimental psychology: applied* 5(2):205.

Horton, J. J., and Chilton, L. B. 2010. The labor economics of paid crowdsourcing. In *Proceedings of the 11th ACM conference on Electronic commerce*, 209–218. ACM.

Ipeirotis, P. G. 2010. Demographics of mechanical turk. *NYU Working Paper*. CEDER-10-01.

Irani, L. C., and Silberman, M. 2013. Turkopticon: Interrupting worker invisibility in amazon mechanical turk. In *Proceedings of the SIGCHI conference on human factors in computing systems*, 611–620. ACM.

Irani, L. C., and Silberman, M. 2016. Stories we tell about labor: Turkopticon and the trouble with design. In *Proceedings of the 2016 CHI conference on human factors in computing systems*, 4573–4586. ACM.

Kaplan, T.; Saito, S.; Hara, K.; and Bigham, J. P. 2018. Striving to earn more: a survey of work strategies and tool use among crowd workers. In *Sixth AAAI Conference on Human Computation and Crowdsourcing*.

Kaufmann, N.; Schulze, T.; and Veit, D. 2011. More than fun and money. worker motivation in crowdsourcing-a study on mechanical turk. In *AMCIS*, volume 11, 1–11. Detroit, Michigan, USA.

Kittur, A.; Nickerson, J. V.; Bernstein, M.; Gerber, E.; Shaw, A.; Zimmerman, J.; Lease, M.; and Horton, J. 2013. The future of crowd work. In *Proceedings of the 2013 confer-*

ence on Computer supported cooperative work, 1301–1318. ACM.

Lasecki, W. S.; Murray, K. I.; White, S.; Miller, R. C.; and Bigham, J. P. 2011. Real-time crowd control of existing interfaces. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, 23–32. ACM.

Mankar, A.; Shah, R. J.; and Lease, M. 2017. Design activism for minimum wage crowd work. *arXiv preprint arXiv:1706.10097*.

Martin, D.; Hanrahan, B. V.; O'Neill, J.; and Gupta, N. 2014. Being a turker. In *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing*, 224–235. ACM.

McInnis, B.; Cosley, D.; Nam, C.; and Leshed, G. 2016. Taking a hit: Designing around rejection, mistrust, risk, and workers' experiences in amazon mechanical turk. In *Proceedings of the 2016 CHI conference on human factors in computing systems*, 2271–2282. ACM.

Myers, B.; Hudson, S. E.; and Pausch, R. 2000. Past, present, and future of user interface software tools. *ACM Transactions on Computer-Human Interaction (TOCHI)* 7(1):3–28.

Otey. 2015. Otey v. crowdflower, inc. et al. https://dockets.justia.com/docket/california/candce/4:2012cv05524/260287/.

Prelec, D. 2004. A bayesian truth serum for subjective data. *science* 306(5695):462–466.

Rolf, D. 2016. *The fight for fifteen: The right wage for a working America*. New Press, The.

Rzeszotarski, J. M., and Kittur, A. 2011. Instrumenting the crowd: using implicit behavioral measures to predict task performance. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, 13–22. ACM.

Saito, S.; Chiang, C.-W.; Savage, S.; Nakano, T.; Kobayashi, T.; and Bigham, J. 2019. Turkscanner: Predicting the hourly wage of microtasks. *arXiv preprint arXiv:1903.07032*.

Salehi, N.; Irani, L. C.; Bernstein, M. S.; Alkhatib, A.; Ogbe, E.; Milland, K.; et al. 2015. We are dynamo: Overcoming stalling and friction in collective action for crowd workers. In *Proceedings of the 33rd annual ACM conference on human factors in computing systems*, 1621–1630. ACM.

Sheng, V. S.; Provost, F.; and Ipeirotis, P. G. 2008. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 614–622. ACM.

Silberman, M. S.; Tomlinson, B.; LaPlante, R.; Ross, J.; Irani, L.; and Zaldivar, A. 2018. Responsible research with crowds: pay crowdworkers at least minimum wage. *Commun. ACM* 61(3):39–41.

Whiting, M. E.; Gamage, D.; Gaikwad, S. N. S.; Gilbee, A.; Goyal, S.; Ballav, A.; Majeti, D.; Chhibber, N.; Richmond-Fuller, A.; Vargus, F.; et al. 2017. Crowd guilds: Worker-led reputation and feedback on crowdsourcing platforms. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*, 1902–1913. ACM.

Zissman, J. 2019. Timeme.js. https://github.com/jasonzissman/TimeMe.js/. Accessed: 2019-05-27.