Learning Best Practices: Can Machine Learning Improve Human Decision-Making?

Hamsa Bastani
Wharton School, Operations Information and Decisions, hamsab@wharton.upenn.edu

Osbert Bastani
University of Pennsylvania, Computer and Information Science, obastani@seas.upenn.edu

Wichinpong Park Sinchaisri
Wharton School, Operations Information and Decisions, swich@wharton.upenn.edu

Preliminary Draft: December 7, 2020
(Latest Version at: `http://wichinpong.com/files/tips.pdf`)

Workers spend a significant amount of time learning how to make good decisions. Evaluating the efficacy of a given decision, however, is quite complicated. For one, decision outcomes are often long-term and relate to the original decision in complex ways. The goal of our paper is to study whether machine learning can be used to infer tips that can help workers learn to make better decisions. Such an algorithm must identify strategies that not only improve worker performance, but that are also interpretable to the human workers so that they can easily understand and follow the tips. We propose a novel machine learning algorithm for inferring interpretable tips that can help users improve their performance in sequential decision-making tasks. We perform a behavioral study to validate our approach. To this end, we designed a virtual kitchen-management game that requires the participant to make a series of decisions to minimize overall service time. Then, we compare the performance of participants shown a tip inferred using our algorithm compared to a control group that is not shown the tip, as well as groups shown either a tip proposed by experienced human workers or a tip inferred by a baseline algorithm. Our experiments show that (i) the tips generated by our algorithm are effective at improving performance, (ii) they significantly outperform the two baseline tips, and (iii) they successfully help participants build on their own experience to discover additional strategies and overcome their resistance to exploring counterintuitive strategies.

Key words: behavioral operations, interpretable machine learning, sequential decision-making, best practices, learning, human-AI interface

1. Introduction

Workers often spend a significant amount of time on the job learning how to make good decisions that improve their performance (Chui et al. 2012). The impact of a current decision can be highly stochastic and affect future decisions/rewards, making it difficult for them to evaluate the quality of a decision. This issue is further exacerbated by the fact that multiple decisions are often made sequentially, making it hard to determine which decisions are responsible for good outcomes. Many jobs require sequential decision-making; for example, doctors making decisions to optimize the long-term outcomes of their patients (Kleinberg et al. 2015) or drivers on ride-hailing platforms optimizing their long-term profits (Marshall 2020). As a concrete example, physicians seek to learn

good strategies for ordering lab tests, since obtaining the appropriate testing results in a timely fashion is necessary to minimize delays in patient visits. Song et al. (2017) finds that experienced physicians have learned to order these tests early on to avoid delays. Despite the simple description of the strategy—"order lab and radiology tests as early in the care delivery process as possible"—learning it on the job is difficult because the connection between when the tests are ordered and the overall quality of care is highly stochastic, and is influenced by other decisions made by the physician as well as unrelated environmental factors such as hospital congestion.

The need to spend time learning on the job has consequences for service quality, since workers likely make suboptimal decisions during this time. For instance, when surgeons first use new devices, surgery duration increases by 32.4% (Ramdas et al. 2017). Thus, whenever possible, workers seek alternative ways to acquire best practices on decision-making. Continuing our example on physician decisions for lab testing, Song et al. (2017) finds that physicians can learn strategies for reducing service time from their better-performing colleagues. This approach is effective precisely because the strategy is simple and easy to communicate, yet time-consuming to discover independently. However, learning from their peers is not always an option for workers; for instance, some workers are comparatively isolated—e.g., physicians working in rural hospitals or operating their own practices or independent workers in the gig economy. In these cases, workers must wastefully spend time independently rediscovering best practices that are already known to their colleagues.

A natural question is whether we can *automatically* discover best practices and convey them to workers to help them improve their performance. In particular, over the past two decades, many domains have accumulated large amounts of *trace data* on human decisions. For example, nearly every physician action is logged in electronic medical record data; every movement of a driver is recorded on a ride-hailing platform; even retail manager decisions on pricing and inventory management are recorded on a daily basis. This data implicitly encodes the collective knowledge acquired by numerous workers about how to effectively perform their jobs. Thus, we might hope to leverage tools from machine learning to mine this data and automatically discover insights that can be used to help workers improve their performance.

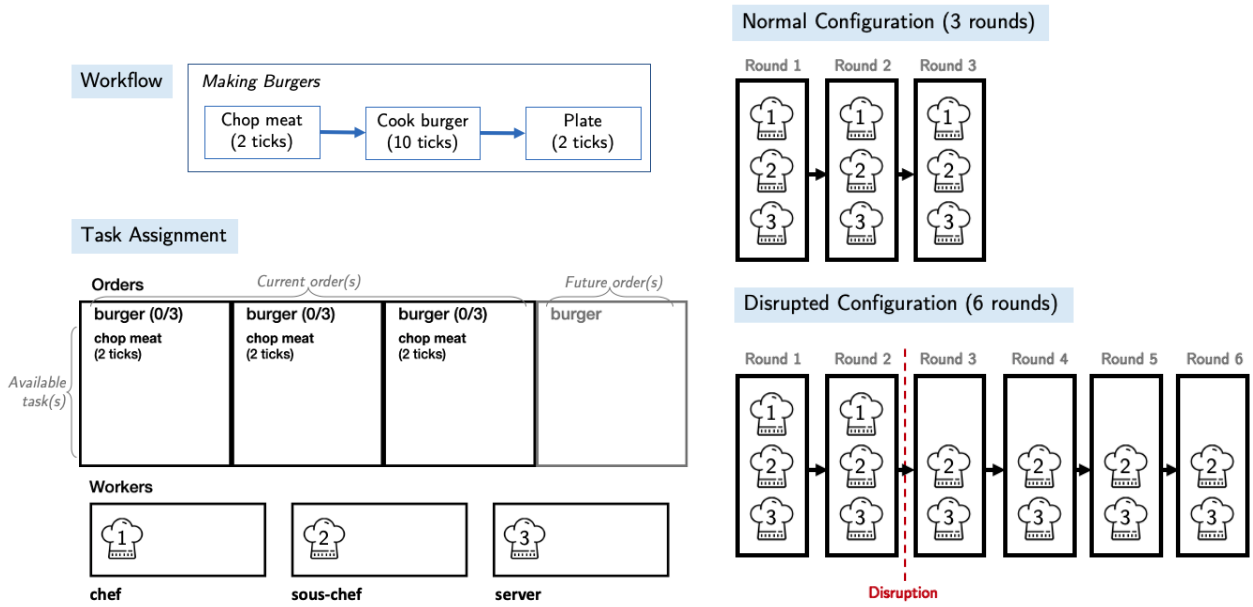
In this paper, we study whether machine learning can be used to infer rules that help improve workers' performance at sequential decision-making tasks. In particular, we propose a novel algorithm for mining useful rules or best practices. Our algorithm automatically learns a decision-making rule that, if correctly followed by the human worker, most improves their performance. It does so in two steps. First, our algorithm uses imitation learning (Abbeel and Ng 2004) to learn a model of the current strategy employed by the human workers. These algorithms are designed to reverse-engineer the strategy employed by humans based on data encoding the actions they take in various states. In particular, we use *Q-learning* (Watkins and Dayan 1992) to learn a neural

network, called the *Q-network*, that approximates the long-term value of the actions taken by the human workers. In addition to encoding the strategy of the human worker, the Q-network also encodes how changes to the human strategy affect their performance. Then, our algorithm leverages the Q-network to learn a decision-making rule that modifies the human worker strategy in a way that most improves their performance. We must carefully design the search space of decision-making rules so that human workers can correctly follow the rule. That is, the rule must be an *interpretable model* whose computation process can be understood by humans (Letham et al. 2015). In particular, we design the search space to consist of if-then-else rules, and use an approach based on *interpretable reinforcement learning* (Bastani et al. 2018) to learn the best rule. Importantly, despite their simplicity, these if-then-else rules can capture useful insights that are challenging for humans to learn by themselves due to the sequential nature of the decision-making problem.

As a case study, we have designed a game where human participants act as managers for a virtual kitchen. An illustration of this task is shown in Figure 1a. In this environment, the human is shown a set of food orders (e.g., burgers, tacos, etc.), each of which is decomposed into a set of subtasks (e.g., chopping, cooking, serving, etc.). To complete an order, the human must assign each subtask to one of the available virtual workers (e.g., chef, server, etc.). The goal is to do so in a way that completes all the orders as quickly as possible. There are two aspects of this environment that make it challenging: (i) each virtual worker has different skills (e.g., the chef cooks quickly but serves slowly), and (ii) the subtasks have dependencies (e.g., the food must be cooked before it is served). As a consequence, the human must balance leveraging the strengths of each virtual worker (i.e., avoid assigning suboptimal subtasks that the virtual worker is slow to complete) and ensuring that none of the workers are idle (i.e., assign suboptimal subtasks to avoid idling the virtual worker). This environment can be thought of as a networked queuing model with heterogeneous servers—i.e., the subtask dependencies are encoded by the network structure and the virtual workers are the heterogeneous servers.

We conduct a behavioral study using Amazon Mechanical Turk (MTurk) workers to test whether our machine-learning algorithm can learn rules that help human workers improve their performance. Our study is based on two different configurations of our virtual kitchen environment. In the "normal" configuration, the MTurk worker plays three identical instantiations of the environment. In the "disrupted" configuration, the first two instantiations of the environment are identical to the ones in the normal configuration, but the remaining four instantiations are modified so that a key worker (namely, the chef) is no longer available. These two configurations are visualized in Figure 1b. The disrupted configuration is particularly challenging for the MTurk workers, since they must "un-learn" preconceived notions about the optimal strategy acquired during the first two instatiations. For each of these configurations, we leverage our algorithm to learn interpretable

Figure 1 Overview of behavioral study: virtual kitchen management.



(a) Workflow and main experimental task

(b) Study configurations

decision-making rules, and then demonstrate how providing this decision-making rule improves the performance of the MTurk workers. Our results show that (i) the tips inferred from our algorithm are effective at significantly improving performance and speeding up learning, (ii) they outperform the tips generated either by previous participants or the baseline algorithm by a significant margin, and (iii) they induce the participants to discover additional optimal strategies beyond what is stated in the tips.

1.1. Related Literature and Contributions

Process improvement has always been one of the major emphases both in the operations management literature and in practice. Our work focuses on process improvement from the perspective of individual workers. Scholars have identified various difficulties associated with learning to improve performance. When first experiencing a new work environment, workers tend to have difficulty adjusting, resulting in various degrees of undesirable performance. For instance, as mentioned earlier, Ramdas et al. (2017) finds that when surgeons first use a new surgical device, surgery duration increases by 32.4%, hurting both their service quality and productivity. Bavafa and Jónasson (2020a) shows that unexpected critical medical incidents slow down the ambulance activation among paramedics. The situation exacerbates when inexperienced workers lack a guideline on how to manage their workflow as their prioritization could often be suboptimal and detrimental to productivity (Ibanez et al. 2017). The complex nature of workflow also plays a role. Workers tend to focus on immediate challenges and ignore opportunities for learning (Tucker et al. 2002) and

switching between tasks could hurt as much as 20% of their productivity (Gurvich et al. 2019). In many collaborative work settings, productivity depends on one's co-workers. Collaboration is particularly challenging in distributed work, where there is considerable uncertainty about others' behaviors (Weisband 2002, Mao et al. 2016). This is especially true in healthcare, where delivery processes involve numerous interfaces and patient handoffs among multiple healthcare practitioners with varying levels of training and prior experiences working together (Hughes et al. 2008, Akşin et al. 2020).

To increase reliability and reduce process variation, process standardization is commonly implemented to form best practices (Nonaka and Takeuchi 1995, Pfeffer et al. 2000, Spear 2005). Process standardization is generally a two-step process: creating the standards and then communicating them. Creating standards and developing knowledge of best practices are known to be hard as they take time (Nonaka and Takeuchi 1995) and knowledge transfer often fails across organizational borders (Szulanski 1996, Argote 2012). A rich literature in operations management and organizational behavior has shown how various aspects of experiences can improve individuals' productivity and performance. For example, professional web developers frequently learn new concepts and strategies by trial and error (Dorn and Guzdial 2010). Past experiences on the same or related tasks, even subtasks, have a significant effect on performance (Huckman and Pisano 2006, Kc and Staats 2012), a variety of experiences could hinder workers' ability to identify best practices (Kc and Staats 2012). Furthermore, Bavafa and Jónasson (2020b) shows that greater prior experience reduces variance of performance. Social interaction is another common way to learn. Therapy workers learn from clients' feedback to adjust their treatment process (Brattland et al. 2018). Workers also learn significantly from their colleagues, particularly those with a high level of knowledge or valuable skills (Herkenhoff et al. 2018, Jarosch et al. 2019). Song et al. (2017) shows that by publicly disclosing relative performance feedback, physicians can better identify their top-performing co-workers, enabling the identification and validation of best practices. Working alongside experienced peers is shown to improve workers' performance (Chan et al. 2014, Tan and Netessine 2019). Team experience and familiarity with one another and with the tasks are associated with both team and individual performance (Akşin et al. 2020, Kim et al. 2020). However, these learning strategies can be inefficient as they rely on the availability of experts and knowledge of best practices. Given well-documented difficulties in learning on the job and identifying best practices, our work proposes an effective approach to automatically extract best practices from logs of historical decisions and outcomes.

Besides identifying best practices, effectively sharing and encouraging workers to adopt them are known to be challenging (Tucker et al. 2007). One way to improve such knowledge transfer is to structure it as a simple rule. The clarity of simple rules allows workers to gain deeper understanding

of the environment and potential improvement (Sull and Eisenhardt 2015, Gleicher 2016). A simple training intervention is also shown to improve decision-making by persistently reducing cognitive biases (Morewedge et al. 2015, Sellier et al. 2019). Thanks to the fast-growing advancement of artificial intelligence, machine-learning models have demonstrated great success in learning complex systems and making predictions that help guide high-stakes decision-making in various domains, from healthcare to criminal justice. However, most commonly used black-box models do not provide users with transparency, accountability, or explanations. The lack of human understanding of how algorithms work poses serious problems to society (Rudin 2019) and leads to aversion to adopt these tools (Dawes et al. 1989, Dietvorst et al. 2015). In recent years, significant efforts have been dedicated towards the development of models that are inherently interpretable (e.g., see Murdoch et al. (2019) for an in-depth review of methods and applications of interpretable machine learning). Incorporating human domain knowledge into algorithms has also received increased attention recently (Arvan et al. 2019, Ibrahim et al. 2020). Our work contributes to this stream of literature in two ways. First, we show that a simple intervention–providing a simple tip–can help improve worker performance over time and speed up their learning process. Second, we develop a novel algorithm that leverages the largely untapped potential of worker trace data to complement existing training programs and learning among workers.

2. Learning Interpretable Rules for Improving Human Performance

In this section, we describe our algorithm for computing rules designed to improve the performance of human workers. We begin by formulating this problem as an MDP, and then describe our algorithm for solving this problem.

2.1. Background on MDPs

Because we are focused on sequential decision-making, we assume the decision-making problem is modeled by a Markov Decision Process (MDP). In particular, consider an MDP $M = (S, A, P, R, \gamma)$, where S is a finite set of states, A is a finite set of actions, $P : S \times A \times S \to \mathbb{R}$ encodes the transition probabilities (i.e., $P(s' \mid s, a)$ is the probability of transitioning from state s to state s' upon taking action a), $R : S \to \mathbb{R}$ denotes rewards (i.e., $R(s)$ is the reward obtained in state s), and $\gamma \in (0, 1)$ is a discount factor. In addition, we assume there is a deterministic initial state $s_0 \in S$. Finally, we also assume there is a finite time horizon $T \in \mathbb{N}$ after which the rewards are zero.

Now, given a stochastic policy $\pi : S \times A \to \mathbb{R}$ (i.e., $\pi(a \mid s)$ is the probability of taking action a in state s), a *rollout* in M uses π to generate a random sequence of state-action-reward tuples $\zeta = ((s_0, a_0, r_0), ..., (s_T, a_T, r_T))$, where $a_t \sim \pi(\cdot \mid s_t)$, $r_t = R(s_t)$, and $s_{t+1} \sim P(\cdot \mid s_t, a_t)$. We use $D^{(\pi)}$ to denote the distribution over rollouts using π. Note that if the MDP transitions P and the policy π are both deterministic, then ζ is also deterministic.

The *cumulative expected reward* of π is

$$J(\pi) = \mathbb{E}_{\zeta \sim D^{(\pi)}}\left[\sum_{t=0}^{T} \gamma^t r_t\right].$$

Finally, the value function $V^{(\pi)} : S \to \mathbb{R}$ and Q function $Q^{(\pi)} : S \times A \to \mathbb{R}$ of π are the unique solutions to the recursive system of equations

$$V^{(\pi)}(s) = \mathbb{E}_{\pi(a|s)}[Q^{(\pi)}(s,a)]$$
$$Q^{\pi}(s,a) = R(s) + \gamma \cdot \mathbb{E}_{P(s'|s,a)}[V^{(\pi)}(s')],$$

respectively. Intuitively, $V^{(\pi)}(s)$ is the cumulative expected reward of using π if the initial state is s, and $Q^{(\pi)}(s,a)$ is the cumulative expected reward of using π from state s, but where the first action taken is fixed to be a.

2.2. Problem Formulation

Given an MDP and a human acting in that MDP, our goal is to learn a decision-making rule that most improves the performance of the human. In particular, we model the human as executing a *human policy* π_H; we measure their performance as the cumulative expected reward $J(\pi_H)$ that they achieve. To ensure that the decision-making rule can be understood by the human worker, we restrict to learning rules of the form

if [state constraint] then [action].

Note that this rule specifies the action to take in a portion of the state space; in the remainder of the state space, the human should continue to make decisions using their own policy π_H. More precisely, assuming we have a mapping $\phi : S \to \{0,1\}^d$ of states to a set of binary properties $\phi(s)_i$, then a state constraint is a predicate $\psi : S \to \{0,1\}$ of the form

$$\psi(s) = (\phi(s)_{i_1} = b_1) \wedge ... \wedge (\phi(s)_{i_k} = b_k).$$

Then, the rule is a pair $\rho = (\psi, a)$ of a predicate ψ and an action a. Intuitively, this rule says to take action a in state s if s satisfies ψ (i.e., $\psi(s) = 1$); otherwise, the human should take an action using their own policy π_H. More precisely, given a stochastic policy π and a rule $\rho = (\psi, a)$, we define the *rule-following policy* $\pi \oplus \rho : S \times A \to \mathbb{R}$

$$(\pi \oplus \rho)(s,a) = \begin{cases} \mathbb{1}(a = a') & \text{if } \psi(s) = 1 \\ \pi(s,a) & \text{otherwise.} \end{cases}$$

In particular, $\pi_H \oplus \rho$ represents the setting where the human worker exactly follows rule ρ—i.e., they use the action recommended by ρ when applicable and use their own policy π_H otherwise.[1] Finally,

[1] Although we rank rules assuming humans follow our tips exactly, this is not the case in practice. Nevertheless, our behavioral experiments demonstrate that our tips significantly improve performance relative to other types of tips.

given a class of rules $\rho \in \mathcal{R}$, our goal is to choose the one that most improves the performance of the human worker—i.e.,

$$\rho^* = \arg\max_{\rho \in \mathcal{R}} J(\pi_H \oplus \rho). \tag{1}$$

To guide our algorithm for learning ρ^*, we assume we are given an *expert policy* π_* that achieves high performance $J(\pi_*)$. In principle, we can compute the exact optimizer $\pi_* = \arg\max_\pi J(\pi)$ using dynamic programming. However, this approach is computationally intractable for large state spaces. Instead, we can use techniques such as model-free reinforcement learning (Watkins and Dayan 1992, Sutton et al. 2000) to compute π_* that approximately optimizes $J(\pi)$. These approaches rely on our assumption that the MDP structure is known; when it is unknown, our algorithm can instead leverage sampled rollouts from a human expert.

2.3. Rule Learning Algorithm

Now, we describe our algorithm for maximizing the objective in (1). Ideally, our algorithm would simply enumerate $\rho \in \mathcal{R}$, compute $J(\pi_H \oplus \rho)$, and return the rule ρ that achieves the highest score. The key challenge is how to compute the value of the objective $J(\pi_H \oplus \rho)$ in (1) for a candidate tip $\rho \in \mathcal{R}$. First, we leverage the following result from Bastani et al. (2018):

LEMMA 1. *For any policy π, we have*

$$J(\pi) = \mathbb{E}_{\zeta \sim D^{(\pi)}}\left[\sum_{t=0}^{T} Q^{(\pi_*)}(s_t, a_t)\right],$$

We can use this result to rewrite the objective $J(\pi_H \oplus \rho)$ in (1). However, we do not have access to samples $\zeta \sim D^{(\pi_H \oplus \rho)}$. To address this issue, we use an approximation where we assume that the distribution over rollouts of the human user is not significantly affected by the rule—i.e., $D^{(\pi_H \oplus \rho)} \approx D^{(\pi_H)}$. Then, we have

$$\begin{aligned} J(\pi_H \oplus \rho) &= \mathbb{E}_{\zeta \sim D^{(\pi_H \oplus \rho)}}\left[\sum_{t=0}^{T} Q^{(\pi_*)}(s_t, a_t)\right] \\ &\approx \mathbb{E}_{\zeta \sim D^{(\pi_H)}}\left[\sum_{t=0}^{T} Q^{(\pi_*)}(s_t, (a_t \oplus \rho)(s_t)\right], \end{aligned}$$

where given rule $\rho = (\psi, a)$ and action a', we define

$$(a' \oplus \rho)(s) = \begin{cases} a & \text{if } \psi(s) = 1 \\ a' & \text{otherwise.} \end{cases}$$

Next, we can approximate this objective using sampled rollouts based on the human policy—i.e., given samples $\zeta^1, ..., \zeta^k \sim D^{(\pi_H)}$, we have

$$\hat{\rho} = \arg\max_{\rho \in \mathcal{R}} \frac{1}{k} \sum_{i=1}^{k} \sum_{t=1}^{T} Q^{(\pi_*)}(s_t^i, (a_t^i \oplus \rho)(s_t^i)).$$

The remaining challenge is that we do not have access to the Q-function $Q^{(\pi_*)}$ of the expert policy π_*. We can learn an estimate $\hat{Q}$ of $Q^{(\pi_*)}$ using supervised learning based on sampled rollouts $\zeta \sim D^{(\pi_*)}$. In particular, given samples $\zeta^1, ..., \zeta^h \sim D^{(\pi_*)}$, we solve the optimization problem

$$\hat{Q} = \arg\min_{Q \in \mathcal{Q}} \sum_{i=1}^{h} \sum_{t=0}^{T} \left(Q(s_t^i, a_t^i) - Q_t^i\right)^2 \qquad \text{where} \qquad Q_t^i = \sum_{\tau=t+1}^{T} r_\tau^i.$$

Here, Q_t^i is an unbiased estimate of $Q^{(\pi^*)}(s_t^i, a_t^i)$. For instance, we could choose $\mathcal{Q}$ to be a random forest or a neural network. Then, our objective becomes

$$\hat{\rho} = \arg\max_{\rho \in \mathcal{R}} \frac{1}{k} \sum_{i=1}^{k} \sum_{t=1}^{T} \hat{Q}(s_t^i, (a_t^i \oplus \rho)(s_t^i)).$$

3. Case Study: Virtual Kitchen Management Game

We seek to evaluate whether our algorithm can reliably improve worker performance in a controlled environment. To this end, we have developed a sequential decision-making task in the form of a virtual kitchen game that can be played by individual human users.

In this game, the human user takes a role of a manager of a virtual kitchen. The overall goal is to complete a fixed set of n food orders (e.g., burgers, tacos, etc.), where order $j \in \{1, ..., n\}$ consists of k_j subtasks (e.g., chopping, cooking, serving, etc.). To complete an order, the human user must assign each of these subtasks to one of the available virtual workers (e.g., chef or server), ideally accounting for the heterogeneous skillset of each worker. The game operates in discrete time steps called *ticks*. On each tick, the human user can assign any of the available subtasks to any of the available virtual workers, where (i) a subtask is available if all its prerequisites have been completed but it has not yet been assigned, and (ii) a virtual worker is available if they are not currently working on another subtask. Importantly, the human user can choose not to assign any subtask to a virtual worker even if they are available—e.g., to strategically wait and assign the worker a more appropriate subtask that will become available at a later tick. On each tick, the human user makes their desired assignments, and then clicks a "next tick" button on the screen; upon clicking this button, the assignments are made and the game is incremented to the next tick. This process repeats until all orders are complete. The goal of the human user is to assign all subtasks to the virtual workers in a way that minimizes the total number of ticks it takes to complete all the orders.

There are two aspects of the game which make it challenging to play optimally. First, the subtasks have dependencies—e.g., the burger must be cooked before it can be served. Second, the virtual workers have different skills that affect how long they take to complete a subtask—e.g., a chef can cook quickly but is slow at serving, whereas a server can serve quickly but is slow at cooking. Thus, the human user must make trade-offs such as deciding whether to greedily assign a worker to a task that they are slow to complete, or leave them idle in anticipation of an upcoming task they can complete quickly.

3.1. Formulation as an MDP

At a high level, the states encode the progress towards completing all the food orders, the actions encode the available assignments of subtasks to workers, and the rewards encode the number of ticks the user takes to complete all the orders. More specifically, the states encode the following information: (i) in all the orders, which subtasks have been completed so far, and (ii) which subtask has been assigned to each virtual worker (if any), and how many ticks remain for the virtual worker to complete that subtask. Next, the actions consist of all possible assignments of available subtasks to available virtual workers. Finally, the reward is -1 at each tick, until all orders are completed—thus, the total number of ticks taken to complete all orders is the negative reward.

3.2. Search Space of Tips

Next, we describe the search space of tips (i.e., rules) that are considered by our algorithm. Each tip is actually composed of a set of rules inferred by our algorithm. Recall that our algorithm considers rules in the form of an if-then-else statement that says to take a certain action in a certain state. One challenge is the combinatorial nature of our action space—there can be as many as $\frac{k!}{(k-m)!}$ actions, where m is the number of workers and $k = \sum_{j=1}^{n} k_j$ is the total number of subtasks. The large number of actions can make the rules very specific—e.g., simultaneously assigning three distinct subtasks to three of the virtual workers. Instead, we decompose the action space and consider assigning a single subtask to a single virtual worker. To be precise, we include three features in the predicate ϕ: (i) the subtask being considered, (ii) the order to which the subtask belongs, and (iii) the virtual worker in consideration. Then, our algorithm considers rules of the form

$$\text{if } (\text{order} = o \ \wedge \ \text{subtask} = s \ \wedge \ \text{virtual worker} = w) \text{ then } (\text{assign } (o, s) \text{ to } w),$$

where o is an order, s is a subtask, and w is a virtual worker.

Even with this action decomposition, we found that these rules are still too challenging for human users to internalize since the rules are very specific. Instead, we post-process the rules inferred by our algorithm by aggregating over tuples (o, s, w) that have the same s and w—e.g., instead of considering two separate rules[2]

$$\text{if } (\text{order} = \text{burger}_1 \ \wedge \ \text{subtask} = \text{cooking} \ \wedge \ \text{virtual worker} = \text{chef}) \text{ then } (\text{assign } (\text{burger}_1, \text{cooking}) \text{ to chef}),$$

$$\text{if } (\text{order} = \text{burger}_2 \ \wedge \ \text{subtask} = \text{cooking} \ \wedge \ \text{virtual worker} = \text{chef}) \text{ then } (\text{assign } (\text{burger}_2, \text{cooking}) \text{ to chef}),$$

we merge them into a tip

$$\text{assign cooking to chef 2 times.}$$

In other words, a tip is a combination of rules $\rho = (\rho_1, ..., \rho_k)$. The score that we assign to such a tip is $J(\rho) = \sum_{i=1}^{k} J(\rho_i)$. Then, we choose the tip that achieves the highest score.

[2] We experimented with *combinations* of rules in exploratory pilots, and found that MTurk workers were unable to operationalize and comply with such complex tips even though they might be part of an optimal strategy.

3.3. Rule Inference Algorithm Implementation

First, we describe how we compute the expert Q-function Q^*. In principle, we could use dynamic programming to solve for the optimal value function V^*, and then compute the optimal Q-function based on V^*. However, while our state space is finite, it is still too large for dynamic programming to be tractable. Instead, we use the policy gradient algorithm (which is widely used for model-free reinforcement learning) as a heuristic to learn an expert policy π_* for our MDP (Sutton et al. 2000).

At a high level, the policy gradient algorithm searches over a family of policies π_θ parametrized by $\theta \in \Theta \subseteq \mathbb{R}^{d_\Theta}$; typically, π_θ is a deep neural network, and θ is the corresponding vector of neural network parameters. This approach requires featurizing the states in the MDP—i.e., constructing a feature mapping $\phi : S \to \{0,1\}^d$. Then, the neural network policy π_θ takes as input the featurized state $\phi(s)$, and outputs an action $\pi_*(\phi(s)) \in A$ to take in state s.[3] Then, the policy gradient algorithm performs stochastic gradient descent on the objective $J(\pi_\theta)$, and outputs the best policy $\pi_* = \pi_{\theta^*}$. In general, $J(\pi_\theta)$ is nonconvex so this algorithm is susceptible to local minima, but it is well known that it performs exceptionally well in practice.

In our implementation, the state features include the availability of each sub-task (for each order), the current status of each worker, and the time index. We take π_θ to be a neural network with 50 hidden units; to optimize $J(\pi_\theta)$, we take 10,000 stochastic gradient steps with a learning rate of 0.001. In addition, since our MDP has finite horizon, we use a discount factor of $\gamma = 1$.

Once we have computed π_*, we use the supervised learning algorithm described in Section 2 to learn an estimate $\hat{Q}$ of the optimal policy's $Q^{(\pi_*)}$; specifically, we choose $\hat{Q}$ to be a random forest (Breiman 2001). The random forest operates over the same featurized states as the neural network policy—i.e., it has the form $\hat{Q}(\phi(s), a) \approx Q^{(\pi_*)}(s, a)$.

Finally, we apply our algorithm to inferring rules on state-action pairs collected from observing human users playing our game. Because our goal is to help human users improve, we restrict our data to the bottom 25% of human users in terms of performance. In addition, we apply two additional postprocessing steps to the set of candidate rules. First, we eliminate rules that apply in less than 10% of the states occurring in the human trace data—i.e., the predicate $\psi(s) = 1$. This step eliminates high-variance rules that have large benefits, but are useful only a small fraction of the time. Second, we eliminate rules that, when they apply, disagree with the expert policy more than 50% of the time—i.e., for a rule (ψ, a), $\psi(s) = 1$ and $a \neq \pi^*(s)$. This step eliminates rules that have large benefits on average, but sometimes offer incorrect advice that can confuse the human user (or cause them to distrust our tips).

[3] To be precise, $\pi_*(\phi(s))$ outputs a probability $\pi_*(a \mid \phi(s))$ for each action $a \in A$ of taking a in state s. We take the action a with the highest probability.

3.4. Baseline Algorithm

A simpler approach is to directly imitate the optimal policy rather than indirectly using the Q function. In particular, given rollouts $\hat{D}$ from a policy π, we compute the frequency of state-action pairs in $\hat{D}$—i.e.,

$$\hat{C}(\psi, a) = \log\left(1 + \#\{(\psi, a) \in \hat{D}\}\right),$$

where $\#\{(s,a) \in \hat{D}\}$ is the number of times the action a was taken when the state constraint ψ was active in one of time steps in a rollout in D^π. Then, the baseline algorithm selects the rule "if $\psi(s)$ then a" with the highest $\hat{C}(\psi, a)$. Then, we post-process these tips in the same way as we process the tips inferred using our algorithm. This comparison evaluates the importance of accounting for the reward structure when selecting good tips.

4. Experimental Design

We perform a behavioral study to evaluate whether the tips inferred using our algorithm can help human participants improve their performance at complex sequential decision-making tasks—specifically, in our virtual kitchen game. Our goal is to address the following research questions:

- Can our algorithm infer tips that help participants improve their performance on a complex sequential decision-making task?
- How do the tips inferred by our algorithm compare to other approaches, including tips suggested by experienced human participants as well as a baseline algorithm-inferred tip?
- Does the inferred tip help improve human performance solely because the participant follows the tip, or does it induce them to improve in additional ways?

Recall that our tip inference algorithm requires data on the human performance so it can focus on conveying information not already known by the humans. For example, if it is fairly common for human participants to learn not to assign a lengthy cooking task to a virtual server, our algorithm will search for other rules that are less obvious but vital to performance improvement. Thus, our behavioral experiment proceeds in two phases. First, we collect data on the actions taken by the humans participants when they are not provided with any tips, and use our algorithm in conjunction with this data to infer tips. Second, we evaluate whether providing these tips can significantly improve the performance of subsequent human participants, compared both to a control group of participants who are not provided with any tips as well as two other groups of participants who are each provided with tips of different sources (i.e., previous participants or a baseline algorithm).

Each human user plays the game several times in sequence, allowing them to learn good strategies over time. For each time the user plays the game, we need to specify the orders that must be completed as well as the available virtual workers; we refer to this specification as a *scenario*. Furthermore, we refer to the overall sequence of scenarios played by the user as a *configuration*.

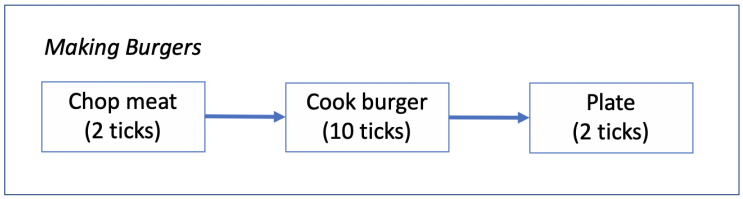
We evaluate our algorithm based on two different configurations of our queueing game that are designed to evaluate different conditions under which tips might be useful. First, the *normal configuration* consists of a single scenario we refer to as the *fully-staffed scenario*. Thus, our goal is to infer tips that help the human users fine-tune their performance at this scenario. Second, the *disrupted configuration* starts with the fully-staffed scenario, but then switches to a modified scenario called the *understaffed scenario*. Intuitively, we expect the human workers to acclimate to the fully-staffed scenario; thus, they may have difficulty adapting to the understaffed scenario where the high-level strategy is very different. Thus, our goal is to infer tips that convey shifts in strategy that are needed to perform well in the new scenario.

For a given configuration, we additionally vary the tip that is shown to the user. Potential tips include: no tip (i.e., the control), our algorithm-inferred tip, a baseline tip, and a human-suggested tip. Our goal is to understand how the choice of tip affects the performance of the human users in the context of that configuration.

4.1. Virtual Kitchen Scenarios & Configurations

Our experimental design is based on two scenarios of the virtual kitchen, differing in terms of which workers are available. In the *fully-staffed scenario*, the human user has access to three virtual workers, whereas in the *understaffed scenario*, the human user only has access to two workers. The scenarios are identical in terms of the orders that must be completed. The orders are all the same dish—specifically, they must complete four burger orders. To complete a single burger order requires three subtasks: (i) chopping meat, (ii) cooking burger, and (iii) plating. Each subtask can only be started once the previous one has been completed. The subtasks in the burger order are illustrated in Figure 2.

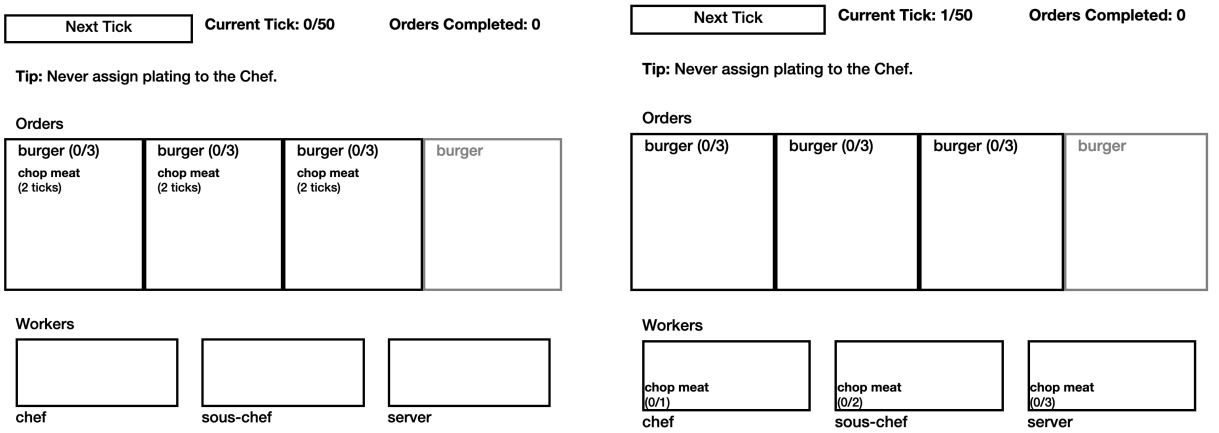
Figure 2 **Subtasks required to make a burger with a median processing time for each subtask.**



There are three possible virtual workers in the kitchen: chef, sous-chef, and server. The chef is fastest at chopping and cooking, but is slowest at plating. The sous-chef is a "jack-of-all-trades", who can perform all tasks at an intermediate speed. Finally, the server is fastest at plating, but slowest at chopping and cooking. Participants are not given the exact number of ticks each worker takes to complete each subtask. Instead, when a subtask becomes available, participants are shown the median number of ticks required to complete that subtask. The true number of ticks for a

given worker is only revealed if they assign the subtask to that worker. The human user must experiment to learn this information by playing the game. Figure 3 shows two screenshots of the game. While the optimal policy in these types of games are typically quite complex and must be solved approximately, we find that a significant number of human users are able to learn very efficient policies by playing the game 3-6 times in a row. For instance, they identify bottleneck subtasks, and learn to mitigate them by assigning these subtasks to their most capable virtual worker.

Figure 3 Example screenshots from the game.



(a) The initial state where participants observe available subtasks from current orders, median times to completion, and three idle virtual workers. The interface also shows the current tick, time limit, current progress, and treatment-specific tip.

(b) The next state after all three previously available subtasks were assigned to the virtual workers and the true completion times were realized, revealing different levels of virtual workers' skills.

As discussed above, we consider two different scenarios for our virtual kitchen. First, in the *fully-staffed scenario*, the human user has access to all three virtual workers—i.e., the chef, sous-chef, and server. In contrast, in the understaffed scenario, the user has access to only two virtual workers—namely, the sous-chef and server.

Finally, we describe the two configurations of our game used in our study. First, in the normal configuration, the human user plays the fully-staffed scenario for three rounds. Second, in the disrupted configuration, they first play two rounds of the fully-staffed scenario, followed by four rounds of the understaffed scenario.

4.2. Experimental Procedure

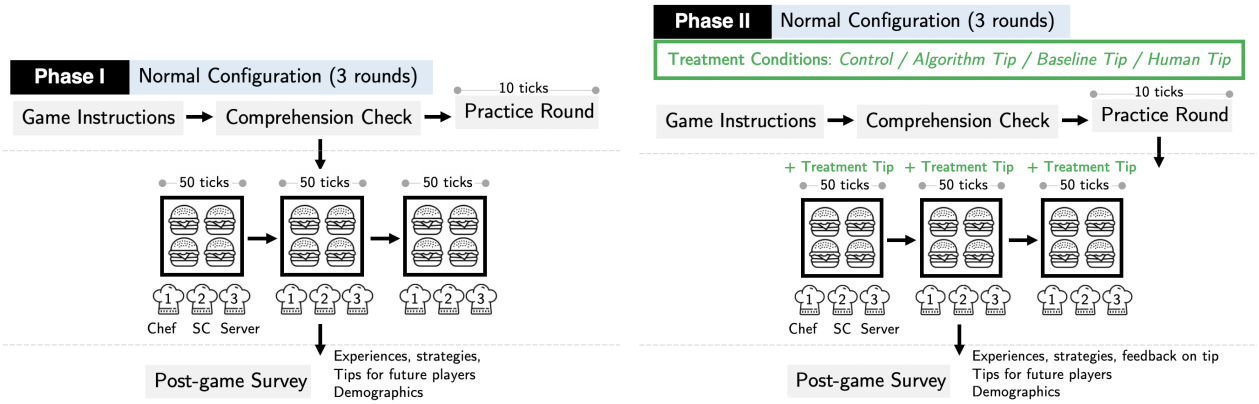
We perform separate experiments for each configuration of our game. The high-level structure of our experimental design for each configuration is the same; they differ in terms of when we

show tips to the user and which tips we show. For each configuration, our experiment proceeds in two phases. Before starting our game, each human user is shown a set of game instructions and comprehension checks; then, they play a practice scenario twice (with an option to skip the second one).[4]

In the first phase, we recruit 200 participants via Amazon Mechanical Turk (MTurk) to play the game. At the end of all rounds of play, we give users a post-game survey where we ask several questions regarding their experience with the game. Additionally, we ask the human user to suggest a tip for future players. In particular, we show them a pre-determined list of tips and ask for the one that they believe would improve the performance of future players. This list of tips is constructed by merging three types of tips: (i) all possible tips of the format described in Section 3.2 (e.g., "Chef shouldn't plate."), (ii) generic tips that arise frequently in our exploratory user studies (e.g., "Keep everyone busy at all time."), (iii) a small number of manually constructed tips obtained by studying the optimal policy (e.g., "Chef should chop as long as there is no cooking task"). Importantly, this list always contains the top tip inferred using our algorithm.

Next, we use the data from the final round played by the human users to infer tips. We do so using each our main algorithm and the baseline algorithm to infer tips, which we refer to as "our tip" and the "baseline tip", respectively. In addition, we also choose a human-suggested tip that is the one most frequently recommended in the post-game survey.

Figure 4 Study flow for the normal configuration.



(a) Phase I

(b) Phase II

In the second phase, we evaluate the effectiveness of the each of the inferred tips based on the data from the first phase. In this phase, human users are randomly assigned to one of four arms,

[4] The practice scenario is meant to familiarize participants with the game mechanics and the user interface. In this scenario, they manage three identical chefs to make a single, simple food order. This food order is significantly different than the burger order used in the main game.

which differ in terms of the tip that is shown to the user. These arms include the *control arm* (i.e., no tip), the *algorithm arm* (i.e., the tip inferred by our algorithm), the *baseline arm* (i.e., the tip inferred by the baseline algorithm), and the *human arm* (i.e., the tip most frequently recommended by human users). We recruited 350 MTurk users to play each arm in each configuration, totaling 2,800 users.

The specific tips we show on each round depends not just on the arm, but also varies from round to round depending on the configuration. For the normal configuration (i.e., the human user plays the fully-staffed scenario for three rounds), we show the tip for the current arm on all three rounds. Figures 4a and 4b illustrate the study flow for the normal configuration. However, for the disrupted configuration configuration (i.e., they first play two rounds of the fully-staffed scenario, followed by four rounds of the understaffed scenario), the tip for the current arm is specific to the understaffed scenario. Thus, we only show the tip for the current arm on rounds 3-6. In all arms, for rounds 1 and 2, we show the tip inferred using our algorithm for the fully-staffed scenario from the normal configuration. By doing so, we help the human users learn more quickly to play the fully-staffed scenario before switching to the understaffed scenario. Figures 5 and 6 illustrate the study flow for the disrupted configuration.

Figure 5 Study flow for Phase I of the disrupted configuration.

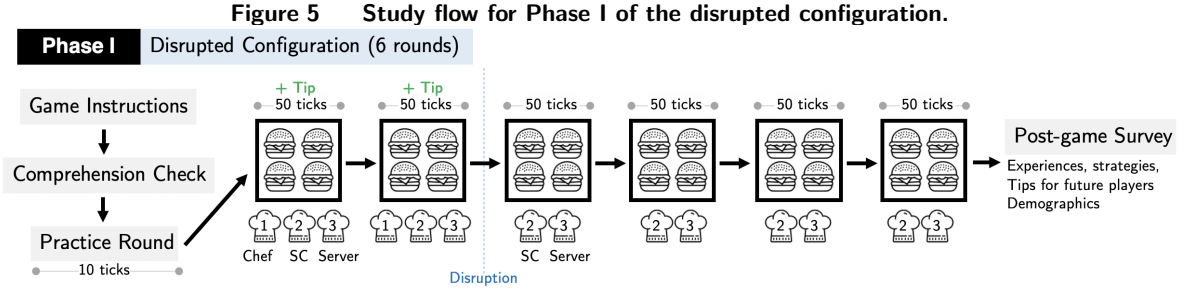
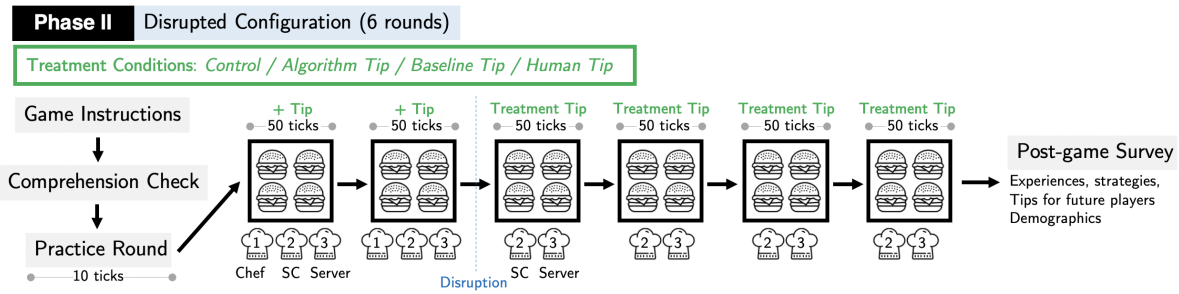


Figure 6 Study flow for Phase II of the disrupted configuration.



Each participant receives a participation fee of \$0.10 for each round they completed. We also provide a bonus based on their performance, measured by the number of ticks taken to complete each round. The bonus ranges from \$0.15 to \$0.75 per round. The full pre-registration document for our study is available at `https://aspredicted.org/blind.php?x=8ye5cb`.

4.3. Hypotheses

We are interested in addressing three sets of hypotheses with our experiment.

1. Do humans perform better with the tip from our algorithm compared to receiving no tips?
   - H1a: In the normal configuration, participants who receive the tip generated by our algorithm will perform better than those not receiving any tips.
   - H1b: In the disrupted configuration, participants who receive the tip generated by our algorithm will perform better than those not receiving any tips.
2. Do humans perform better with a tip from our algorithm compared to the tip most frequently suggested by other humans who have completed the game?
   - H2a: In the normal configuration, participants who receive the tip generated by our algorithm will perform better than those receiving the tip most frequently suggested by past participants who also played the normal configuration.
   - H2b: In the disrupted configuration, participants who receive the tip generated by our algorithm will perform better than those receiving the tip most frequently suggested by past participants who also played the disrupted configuration.
3. Do humans perform better with the tip from our algorithm compared to the tip from a baseline tip mining algorithm?
   - H3a: In the normal configuration, participants who receive the tip generated by our algorithm will perform better than those receiving the tip generated by the baseline.
   - H3b: In the disrupted configuration, participants who receive the tip generated by our algorithm will perform better than those receiving the tip generated by the baseline.

To do so, we perform six two-sample one-sided t-tests to compare the distributions of number of ticks to completion of the final round of each configuration. We also consider additional secondary outcome measures including the learning rate (e.g., performance across rounds), the fraction of participants who completed each round of the game by taking the optimal number of steps, and how well the participants complied with the provided tip and learned additional optimal strategies.

5. Experimental Results

We report the performance of the tips generated by our algorithm compared to that of the other treatment groups. Our results demonstrate that despite their simplicity and conciseness, the tips inferred using our algorithm capture strategies that are hard for the participants to learn, and can significantly improve their performance. In addition, we investigate how participants' learning process over the rounds is impacted by the different tips they are shown. Our findings suggest that the participants are not blindly following our tip, but combine our tip with their own experience to improve performance, and furthermore build on our tip to discover additional strategies beyond what is stated in the tip.

5.1. Phase I: Inferring Tips

Normal configuration. 183 participants successfully completed the game in the first phase of the normal configuration. The average age of the participants is 34.6 years old, 57.38% are female, and 67.73% have at least a two-year degree. The top three tips generated by our algorithm (with the expected performance improvement) are "Chef should never plate" (2.43 fewer ticks on average), "Server plates three times" (0.53 ticks), and "Server should skip chopping once" (0.18 ticks). The baseline algorithm yielded different tips; the top three tips from the baseline algorithm are: "Chef should chop once", "Server should plate three times", and "Sous-chef should plate twice". The three most common tips chosen by participants in Phase I of the normal configuration are "Strategically leave some workers idle" (28.42%), "Server shouldn't cook" (21.31%), and "Chef shouldn't plate" (13.11%). All of these human-suggested tips are in line with the optimal policy. The first tip captures the key strategy that some virtual workers should be left idle rather than assigned to a time-consuming task. However, it is less specific than the other tips or tips from our framework. The second and third tips reflect the fact that the server takes the most time cooking and the chef takes the most time plating, which participants could learn from assigning these tasks to them during the game.

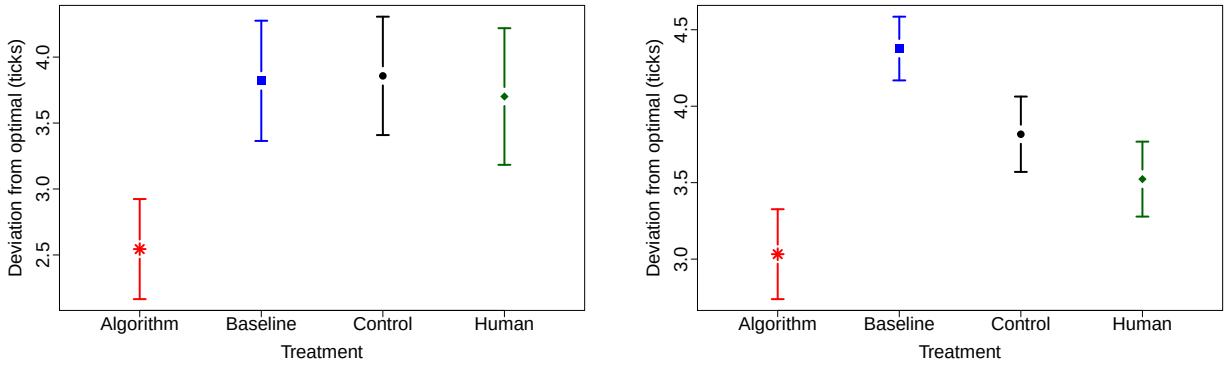
Disrupted configuration. 172 participants successfully completed the first phase of the disrupted configuration. They are 36.4 years old on average, 61.63% are female, and 77.91% received at least a two-year degree. The top three tips generated by our algorithm are"Server should cook twice" (2.32 fewer ticks on average), "Sous-chef should plate once" (0.55 ticks), and "Server should chop once" (0.18 ticks). The first and third tips are in line with the optimal policy. The second tip is slightly off as in the optimal policy Sous-chef and Server should each plate twice. Lastly, the baseline algorithm yielded slightly different tips; the top three tips from the baseline algorithm are: "Sous-chef should plate twice", "Sous-chef should chop three times", and "Server should cook twice". All of these tips are in line with the optimal policy. The three most common tips suggested by participants are "Server should cook once/Sous-chef should cook three times" (28.48%), "Server shouldn't cook/Sous-chef should do all cooking" (23.84%), and "Keep everyone busy at all time/Do not leave anyone idle" (16.86%). Interestingly, the first two tips are not optimal. In the optimal policy, Sous-chef and Server should each cook twice. The third tip is in line with the optimal policy as no worker should be left idle in the disrupted scenario, but it is much less specific than the first two tips or those from our framework.

5.2. Phase II: Our Tips Significantly Improve Performance

Normal configuration. 1,317 participants successfully completed the game. The mean age is 33.3 years old, 51.03% are female, and 67.73% completed a degree beyond high school. In the final

round of the game, participants from the algorithm condition completed the game in 22.54 ticks on average, outperforming those in other conditions (see Figure 7a). The average numbers of ticks to completion for other conditions are: 23.86 ticks (control), 23.73 ticks (human), and 23.82 ticks (baseline). The performance of the algorithm arm is statistically significantly better than that of the control arm ($t = -4.397, p = 0.00000647$), the human arm ($t = 3.628, p = 0.000158$), and the baseline arm ($t = -4.232, p = 0.0000135$).

Figure 7 Performance of participants across conditions in the last round of Phase II.



(a) Normal configuration (optimal: 20 ticks)

(b) Disrupted configuration (optimal: 34 ticks)

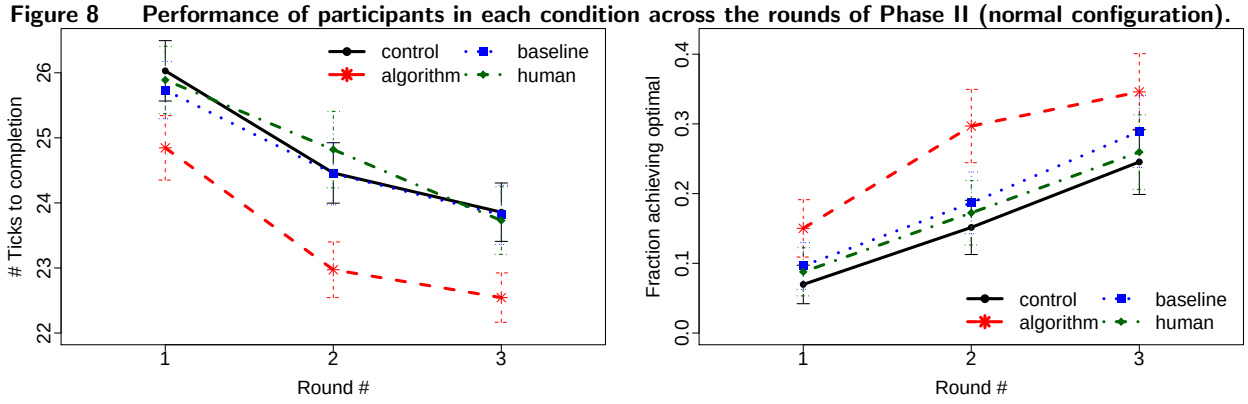
Disrupted configuration. 1,011 participants successfully completed the game. On average, participants are 34.9 years old, 60.14% are female, and 70.43% have at least a two-year degree. 244 were in the control arm, 247 in the algorithm arm, 249 in the human arm, and 271 in the baseline arm.The average number of ticks to complete the final round are: 37.92 ticks (control), 37.05 ticks (algorithm), 37.53 ticks (human), and 38.40 ticks (baseline) (see Figure 7b). The performance of the algorithm arm is statistically significantly better than that of the control arm ($t = -4.361, p = 0.00000806$), the human arm ($t = -2.52, p = 0.00605$), and the baseline arm ($t = -7.348, p < 1E-12$).

5.3. Learning Over Time: Our Tips Speed Up Learning

Next, we study how participants learn strategies over time. Recall that participants in the normal configuration had three game rounds over which they could learn and improve, while those in the disrupted configuration had four rounds (not counting the initial two rounds with the fully-staffed scenario). In addition to the number of ticks to complete all orders, we also examine the fraction of participants that reach the optimal reward—i.e., completing the game in 20 ticks for the fully-staffed scenario or in 34 ticks for the understaffed scenario).

Normal configuration. Figure 8a illustrates the performance measured by the number of ticks taken to complete each round for participants in each condition. First, we observe that participants in all arms improved over the three rounds. Although participants in each of the three arms (but not the control) received tips starting from the first round, only those receiving our tip performed significantly better the control in any of the three rounds. In fact, our tip appears to speed up learning by at least one round compared to any of the other rounds—i.e., the performance of participants given our tip on the kth round is similar to the performance of participants in alternative arms on the $(k+1)$th round. Figure 8b displays the fraction achieving this optimal performance for each arm over the rounds. As before, we observe that all participants improved over the three rounds, and that our tip was the only one to significantly improve performance in any round.

Figure 8 Performance of participants in each condition across the rounds of Phase II (normal configuration).



(a) Ticks to completion

(b) Fraction achieving optimal

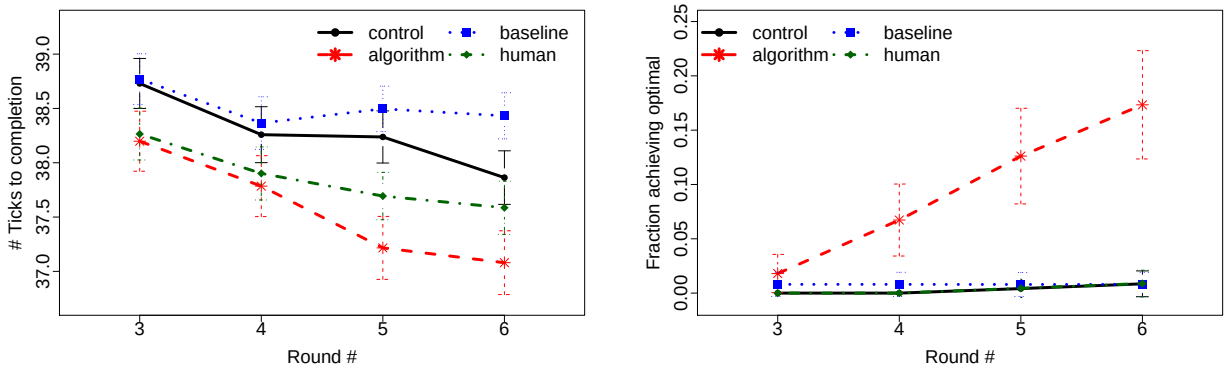
Disrupted configuration. Figure 9a shows the performance measured by the number of ticks taken to complete each round for participants in each arm. As before, we find that participants improve over the rounds. The participants in the control arm took four rounds to achieve the same level of performance as the participants shown our tip on the first round. Thus, our optimal tip speeds up learning by four rounds. In addition, Figure 9b shows the fraction of participants that achieved optimal performance. As can be seen, only participants given our tip achieved optimal performance with any significance. By the last round of the game, 18.79% of participants in the algorithm arm played optimally, compared to 1.14% of the human arm, 0.99% of the baseline arm, and 0.51% of the control arm.

In this case, the baseline tip ("Sous-chef should plate twice") actually reduces performance. The actions suggested by this tip only concerns the last stage of the sequence, which is inline with our intuition that focusing on mimicking the optimal policy is not sufficient to improve performance. In

contrast, we need to focus on mimicking consequential decisions made by the optimal policy—e.g., our tip focuses on early actions that can improve long-term performance.

Next, note that the human arm actually starts off improving performance by about the same amount as our arm, but levels off whereas participants shown our tip continue to improve performance. In other words, it takes several rounds for participants to make good use of our tips. These results suggest that our tip—while simple and concise in nature—encodes a complex underlying strategy that the participants come to understand when they combine it with their own experience playing the game. In contrast, the human-suggested tip encodes a more shallow strategy that quickly improves performance but does not lead to deeper insight over time. We study this phenomenon in more detail in the subsequent sections.

Figure 9 Performance of participants in each condition in the last four rounds of Phase II (disrupted configuration).



(a) Ticks to completion

(b) Fraction achieving optimal

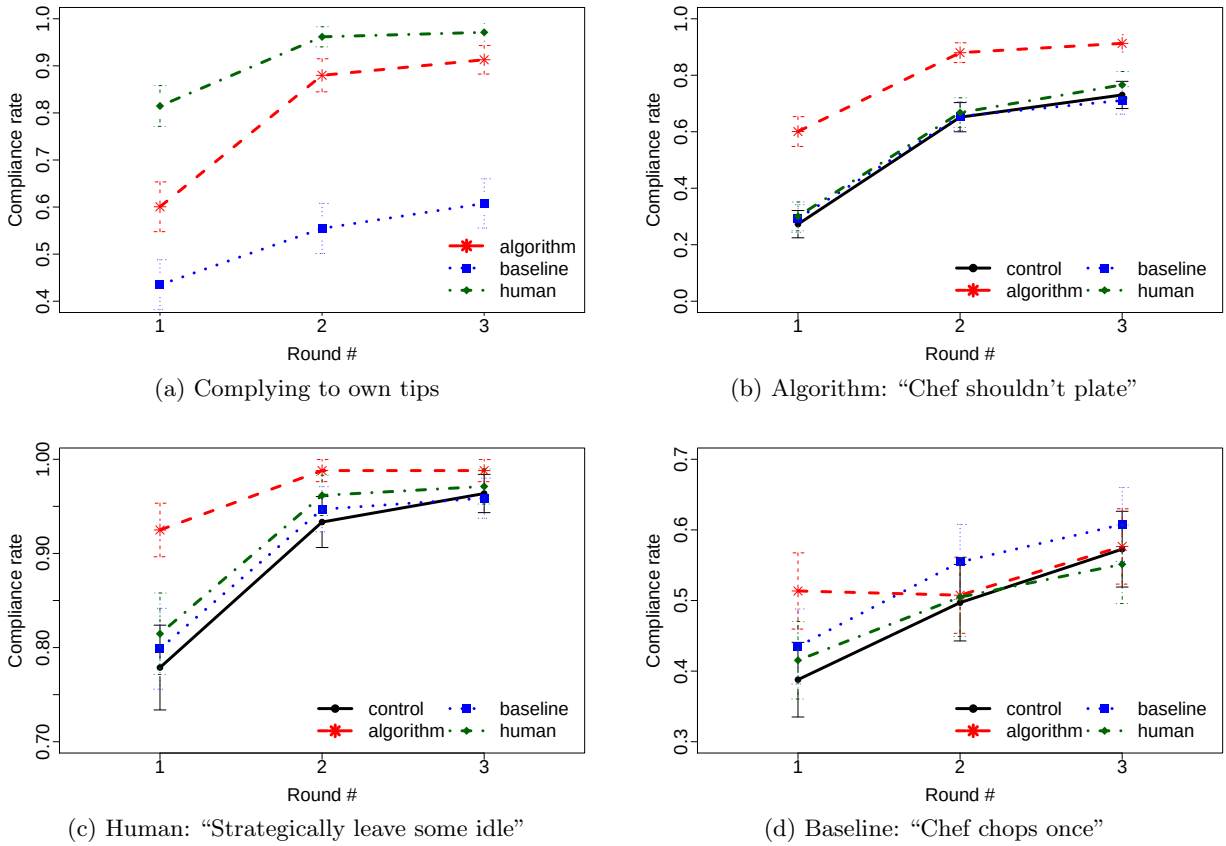
Lastly, we believe that one potential obstacle to uncovering the optimal strategy during the understaffed scenario is the participants' inability to "un-learn" and adapt their strategy to the disrupted environment. In the first two rounds with the fully-staffed scenario, participants have potentially learned each worker's skill level and developed a strategy to assign tasks based on such knowledge. A long task (e.g., cooking burger) is often assigned to the highly skilled worker (e.g., chef), while the least skilled worker (e.g., server) is reserved to carry on a short task (e.g., plating). Once the disruption took place, the majority of participants kept their original strategy—i.e., not assigning chopping or cooking tasks to the server. After four rounds of the understaffed scenario, a fraction of participants learned that leaving the server idle was suboptimal. The human-proposed tip ("Server should cook once") suggests that participants were able to adjust their strategy towards the optimal one after four rounds. However, this tip is not aggressive enough to achieve the optimal performance—as indicated by our tip ("Server should cook twice"). In optimal play, the server actually needs to perform a significantly larger share of the subtasks than the

human tip suggests. Another evidence for this explanation is observed in the post-game survey of both phases of the normal configuration. Although participants in these studies did not experience a disruption, they were asked to imagine a hypothetical understaffed scenario and select the best tip that they expected to help improve performance in such disruption. The tip that received the most votes is "Server shouldn't cook". Without the actual experience of managing the disruption, participants appeared to be biased towards their strategy learned in the normal scenario. Thus, we believe the success of our tip is due in part to how it helps human decision-makers overcome their resistance to exploring counterintuitive strategies.

5.4. Complying to Tips: Human Users are More Compliant Over Time

Next, we study the compliance rate of each tip—i.e., the fraction of participants who follow a given tip. We consider both *direct compliance*, which is the compliance rate of participants of the tip they were shown, and *indirect compliance*, which is the compliance rate of participants to a tip different than the one they were shown (if any).

Figure 10 Compliance rate across the rounds of Phase II (normal configuration).



(a) Complying to own tips

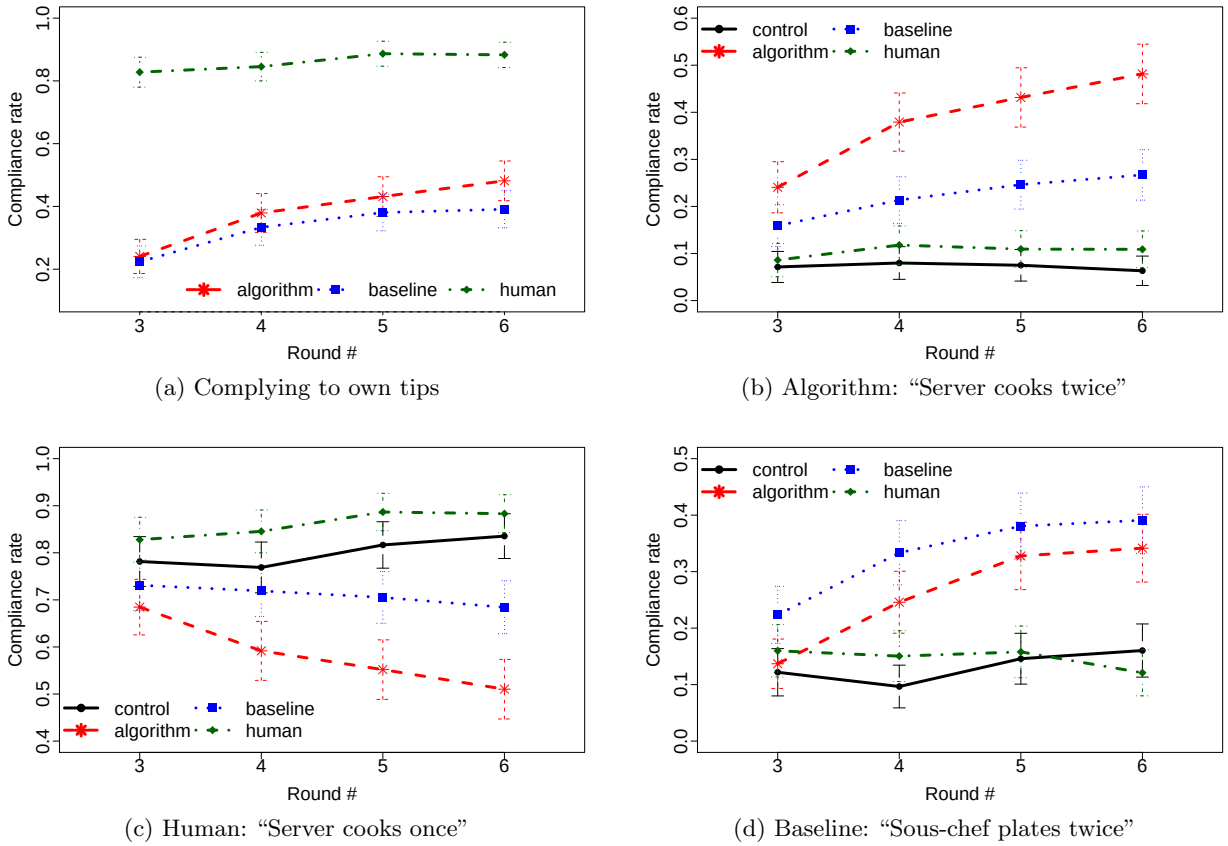
(b) Algorithm: "Chef shouldn't plate"

(c) Human: "Strategically leave some idle"

(d) Baseline: "Chef chops once"

Normal configuration. Figure 10 shows the compliance rate of each tip across the arms. First, as can be seen in Figure 10a, participants increasingly comply to their respective tip over time. By the final round, a significantly higher fraction of human and algorithm participants complied than those in the baseline condition. This result suggests that the participants determined that the baseline tip did not suggest an especially effective strategy. Thus, participants are not blindly following tips, but rather trying to understand the strategy implications behind the tips, and only following the tip if the strategy makes sense and is effective. Next, Figure 10b shows that participants across all arms learn not to assign plating to the chef. Similarly, Figures 10c & 10d show that participants learn to leave some virtual workers idle or let the chef chop once, respectively. These rules are all consistent with the optimal policy, showing that participants learn over time to improve their performance regardless of the treatment. Interestingly, participants shown our tip have similar or higher compliance than the other arms even on tips they were not shown. This result suggests why our tip might be more effective, since the information it conveys encompasses the information conveyed by the other tips.

Figure 11 Compliance rate across the rounds of Phase II (disrupted configuration).



(a) Complying to own tips

(b) Algorithm: "Server cooks twice"

(c) Human: "Server cooks once"

(d) Baseline: "Sous-chef plates twice"

Disrupted configuration. Figure 11 shows the compliance rate of each tip across the arms. As before, Figure 11a shows that participants increasingly comply to their respective tips across rounds. In this case, the compliance of participants in the human arm to their tip is significantly higher than the other two arms. This is likely because among the three, the human tip "Server cooks once" is the most intuitive—e.g., our tip "Server cooks twice" is counterintuitive since the server is slow at cooking. Next, Figure 11b shows the compliance rate of all arms to our tip. As expected, participants shown this tip have by far the highest rate of compliance. The human and control arms do not increase compliance with this tip at all, suggesting that participants do not naturally learn this strategy over time, most likely due to the fact that it is counterintuitive. This result demonstrates how even simple tips can greatly improve human performance because they capture conceptual strategies that take a great deal of experimentation to discover. Figure 11d exhibits similar trends, where the algorithm and baseline arms increase adherence to the baseline tip whereas the human and control arms do not. Finally, Figure 11c shows the compliance rate of each arm to the human tip. Interestingly, compliance of the algorithm arm to this strategy actually decreases over time; indeed, the tip suggested by humans is a suboptimal strategy, so complying with this tip leads to worse performance.

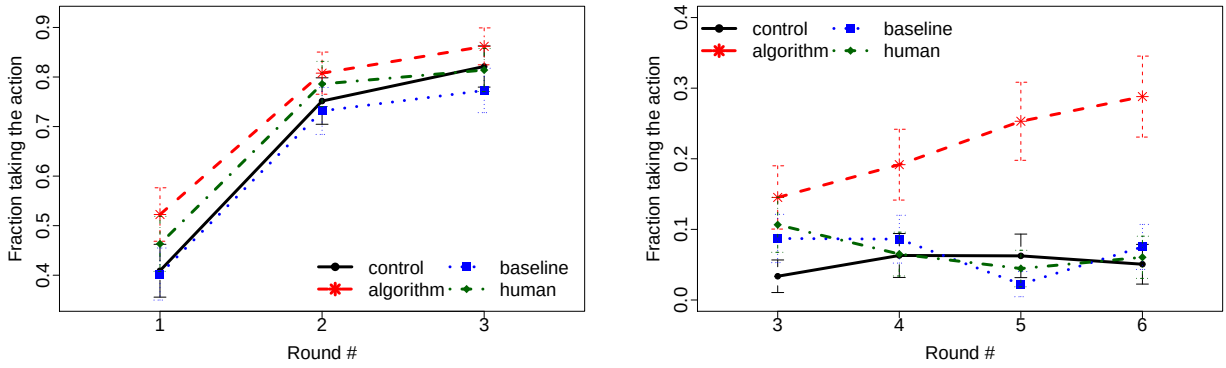
5.5. Learning Beyond Tips: Our Tips Help Users Learn to Play Optimally

Finally, we further investigate how the participants learn by examining their compliance with rules beyond the tips inferred in our study.

Normal configuration. At a high level, the optimal policy for the fully-staffed scenario has the chef cook most of the dishes, has the server plate most of the dishes, and never assigns the chef to plate or the server to cook. We observe that participants generally recovered these optimal strategies as they played more rounds. Figure 12a shows the fraction of participants in each condition that never assigned cooking to the server in each round, as if they were following the tip "Server shouldn't cook". Overall, the compliance fractions increase over time and within each round the fractions are not statistically different among the conditions. This result suggests that people learned about this rule by themselves across all arms.

Disrupted configuration. At a high level, the optimal policy for the understaffed scenario requires balanced assignment of cooking and plating tasks to both sous-chef and server. A key rule beyond the one inferred by our algorithm is "Server chops once". As can be seen in Figure 12b, participants in the algorithm arm learned this behavior even though they were not shown this tip. In contrast, participants in the remaining arms do not appear to learn this behavior over time. This result demonstrates that the participants are learning to perform strategies beyond the ones immediately suggested by our tip.

Figure 12 **Fraction of participants taking optimal action beyond provided tips across the rounds of Phase II.**



(a) Normal: "Server shouldn't cook"

(b) Disrupted: "Server chops once"

6. Concluding Remarks

We have proposed a novel machine learning algorithm for automatically identifying interpretable tips designed to help improve human decision-making. Our behavioral study demonstrates that the tips inferred using our algorithm can successfully improve human performance at a challenging sequential decision-making task. In particular, our results for the normal configuration suggest that our tip can speed up learning by up to three rounds of in-game experience, demonstrating that our tip can significantly reduce the cost of learning. Furthermore, in the disrupted configuration, our results suggest that our tip enables the human participants to discover additional strategies beyond the tip. In other words, the benefit of tips comes not just from having the human follow the letter of the tip, but from how the human builds on the tip to discover additional insights.

An important ingredient in our framework is the incorporation of trace data to identify pieces of information that are most likely to help improve the performance of an average worker. Modern-day organizations have benefited from using customer data to inform new product strategies and provide personalized offerings to their customers, but the data on their own employees is underused. Our framework provides techniques to leverage the largely untapped potential of readily available trace data in pinpointing areas of performance improvement and identifying new practices. Even when the true optimal strategy is unknown, trace data of workers with high experience or good performance can be used to identify good strategies. In recent years, a growing number of organizations have adopted a gig economy employment model or allowed for remote work in response to worker preferences for flexibility and independence. To compensate for the lack of interactions among workers, firms can employ our algorithm to learn best practices from the highly performing workers and then provide tips to help individuals improve.

There are several important directions for future work. First, incorporating personalization to individual workers could greatly improve the performance of our tip inference algorithm. Our optimal tips were chosen based on the expected improvement among the bottom quartile of performers

in the game. Ideally, we would instead infer tips personalized for different skill levels and individual worker characteristics. Furthermore, in our approach, we only inferred tips at one point in time. In practice, our approach could also be performed every time additional data is collected. Then, an important question is understanding the long-term benefits of our approach and understanding how it affects learning behavior over a longer period of time. Another promising direction is extending our algorithm to a collaborative setting. We have only studied how individual workers learn to improve performance, but a similar approach may help teams improve their collaboration and optimize information sharing. Finally, future work is needed to study how to better convey machine-generated tips to improve compliance. Recent work has documented human aversion to advice made by algorithms (Eastwood et al. 2012, Dietvorst et al. 2015) and shown certain conditions that alleviate such aversion (Dietvorst et al. 2018, Logg et al. 2019). In our study, a fraction of participants chose to forgo our tip and continue using their own strategy. Finding ways to build trust and encourage compliance is an important ingredient for ensuring our tips help people improve.

References

Abbeel P, Ng AY (2004) Apprenticeship learning via inverse reinforcement learning. *Proceedings of the twenty-first international conference on Machine learning*, 1.

Akşin Z, Deo S, Jónasson JO, Ramdas K (2020) Learning from many: Partner exposure and team familiarity in fluid teams. *Management Science* .

Argote L (2012) *Organizational learning: Creating, retaining and transferring knowledge* (Springer Science & Business Media).

Arvan M, Fahimnia B, Reisi M, Siemsen E (2019) Integrating human judgement into quantitative forecasting methods: A review. *Omega* 86:237–252.

Bastani O, Pu Y, Solar-Lezama A (2018) Verifiable reinforcement learning via policy extraction. *Advances in neural information processing systems*, 2494–2504.

Bavafa H, Jónasson JO (2020a) Recovering from critical incidents: Evidence from paramedic performance. *Manufacturing & Service Operations Management* .

Bavafa H, Jónasson JO (2020b) The variance learning curve. *Available at SSRN* .

Brattland H, Høiseth JR, Burkeland O, Inderhaug TS, Binder PE, Iversen VC (2018) Learning from clients: A qualitative investigation of psychotherapists' reactions to negative verbal feedback. *Psychotherapy Research* 28(4):545–559.

Breiman L (2001) Random forests. *Machine learning* 45(1):5–32.

Chan TY, Li J, Pierce L (2014) Learning from peers: Knowledge transfer and sales force productivity growth. *Marketing Science* 33(4):463–484.

Chui M, Manyika J, Bughin J (2012) The social economy: Unlocking value and productivity through social technologies. Technical report, McKinsey Global Institute.

Dawes RM, Faust D, Meehl PE (1989) Clinical versus actuarial judgment. *Science* 243(4899):1668–1674.

Dietvorst BJ, Simmons JP, Massey C (2015) Algorithm aversion: People erroneously avoid algorithms after seeing them err. *Journal of Experimental Psychology: General* 144(1):114.

Dietvorst BJ, Simmons JP, Massey C (2018) Overcoming algorithm aversion: People will use imperfect algorithms if they can (even slightly) modify them. *Management Science* 64(3):1155–1170.

Dorn B, Guzdial M (2010) Learning on the job: characterizing the programming knowledge and learning strategies of web designers. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 703–712.

Eastwood J, Snook B, Luther K (2012) What people want from their professionals: Attitudes toward decision-making strategies. *Journal of Behavioral Decision Making* 25(5):458–468.

Gleicher M (2016) A framework for considering comprehensibility in modeling. *Big data* 4(2):75–88.

Gurvich I, O'Leary KJ, Wang L, Van Mieghem JA (2019) Collaboration, interruptions, and changeover times: Workflow model and empirical study of hospitalist charting. *Manufacturing & Service Operations Management* .

Herkenhoff K, Lise J, Menzio G, Phillips G (2018) Knowledge diffusion in the workplace. Technical report, July 2018. Working Paper, University of Minnesota.

Huckman RS, Pisano GP (2006) The firm specificity of individual performance: Evidence from cardiac surgery. *Management Science* 52(4):473–488.

Hughes RG, et al. (2008) Patient safety and quality. *an evidence-based handbook for nurses* 2008.

Ibanez MR, Clark JR, Huckman RS, Staats BR (2017) Discretionary task ordering: Queue management in radiological services. *Management Science* 64(9):4389–4407.

Ibrahim R, Kim SH, Tong J (2020) Eliciting human judgment for prediction algorithms. *Available at SSRN* .

Jarosch G, Oberfield E, Rossi-Hansberg E (2019) Learning from coworkers. Technical report, National Bureau of Economic Research.

Kc DS, Staats BR (2012) Accumulating a portfolio of experience: The effect of focal and related experience on surgeon performance. *Manufacturing & Service Operations Management* 14(4):618–633.

Kim SH, Song H, Valentine M (2020) Staffing temporary teams: Understanding the effects of team familiarity and partner variety. *Available at SSRN 3176306* .

Kleinberg J, Ludwig J, Mullainathan S, Obermeyer Z (2015) Prediction policy problems. *American Economic Review* 105(5):491–95.

Letham B, Rudin C, McCormick TH, Madigan D, et al. (2015) Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *The Annals of Applied Statistics* 9(3):1350–1371.

Logg JM, Minson JA, Moore DA (2019) Algorithm appreciation: People prefer algorithmic to human judgment. *Organizational Behavior and Human Decision Processes* 151:90–103.

Mao A, Mason W, Suri S, Watts DJ (2016) An experimental study of team size and performance on a complex task. *PloS one* 11(4):e0153048.

Marshall A (2020) Uber changes its rules, and drivers adjust their strategies. URL `https://www.wired.com/story/uber-changes-rules-drivers-adjust-strategies/`.

Morewedge CK, Yoon H, Scopelliti I, Symborski CW, Korris JH, Kassam KS (2015) Debiasing decisions: Improved decision making with a single training intervention. *Policy Insights from the Behavioral and Brain Sciences* 2(1):129–140.

Murdoch WJ, Singh C, Kumbier K, Abbasi-Asl R, Yu B (2019) Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences* 116(44):22071–22080.

Nonaka I, Takeuchi H (1995) *The knowledge-creating company: How Japanese companies create the dynamics of innovation* (Oxford university press).

Pfeffer J, Sutton RI, et al. (2000) *The knowing-doing gap: How smart companies turn knowledge into action* (Harvard business press).

Ramdas K, Saleh K, Stern S, Liu H (2017) Variety and experience: Learning and forgetting in the use of surgical devices. *Management Science* 64(6):2590–2608.

Rudin C (2019) Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence* 1(5):206–215.

Sellier AL, Scopelliti I, Morewedge CK (2019) Debiasing training improves decision making in the field. *Psychological science* 30(9):1371–1379.

Song H, Tucker AL, Murrell KL, Vinson DR (2017) Closing the productivity gap: Improving worker productivity through public relative performance feedback and validation of best practices. *Management Science* 64(6):2628–2649.

Spear SJ (2005) Fixing health care from the inside, today. *Harvard business review* 83(9):78.

Sull DN, Eisenhardt KM (2015) *Simple rules: How to thrive in a complex world* (Houghton Mifflin Harcourt).

Sutton RS, McAllester DA, Singh SP, Mansour Y (2000) Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 1057–1063.

Szulanski G (1996) Exploring internal stickiness: Impediments to the transfer of best practice within the firm. *Strategic management journal* 17(S2):27–43.

Tan TF, Netessine S (2019) When you work with a superman, will you also fly? an empirical study of the impact of coworkers on performance. *Management Science* 65(8):3495–3517.

Tucker AL, Edmondson AC, Spear S (2002) When problem solving prevents organizational learning. *Journal of Organizational Change Management* .

Tucker AL, Nembhard IM, Edmondson AC (2007) Implementing new practices: An empirical study of organizational learning in hospital intensive care units. *Management science* 53(6):894–907.

Watkins CJ, Dayan P (1992) Q-learning. *Machine learning* 8(3-4):279–292.

Weisband S (2002) Maintaining awareness in distributed team collaboration: Implications for leadership and performance. *Distributed work* 311–333.