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

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

July 2, 2021

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. Surprisingly, even though learning good decision-making strategies is difficult, they can often be expressed in simple and concise forms. Focusing on sequential decision-making, we design a novel machine learning algorithm that is capable of extracting "best practices" from trace data and conveying its insights to humans in the form of interpretable tips. Our algorithm selects the tip that best bridges the gap between the actions taken by the human workers and those taken by the optimal policy in a way that accounts for which actions are consequential for achieving higher performance. We evaluate our approach through a series of randomized controlled user studies where participants manage a virtual kitchen. Our experiments show that the tips generated by our algorithm can significantly improve human performance. In addition, we discuss a number of empirical insights that can help inform the design of algorithms intended for human-AI collaboration. For instance, we find evidence that participants do not simply blindly follow our tips; instead, they combine them with their own experience to discover additional strategies for improving performance.

Key words: behavioral operations, interpretable machine learning, reinforcement learning, sequential decision-making, best practices, 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 in complex ways, 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 high-volume data and automatically discover insights that can be used to help workers improve their performance.

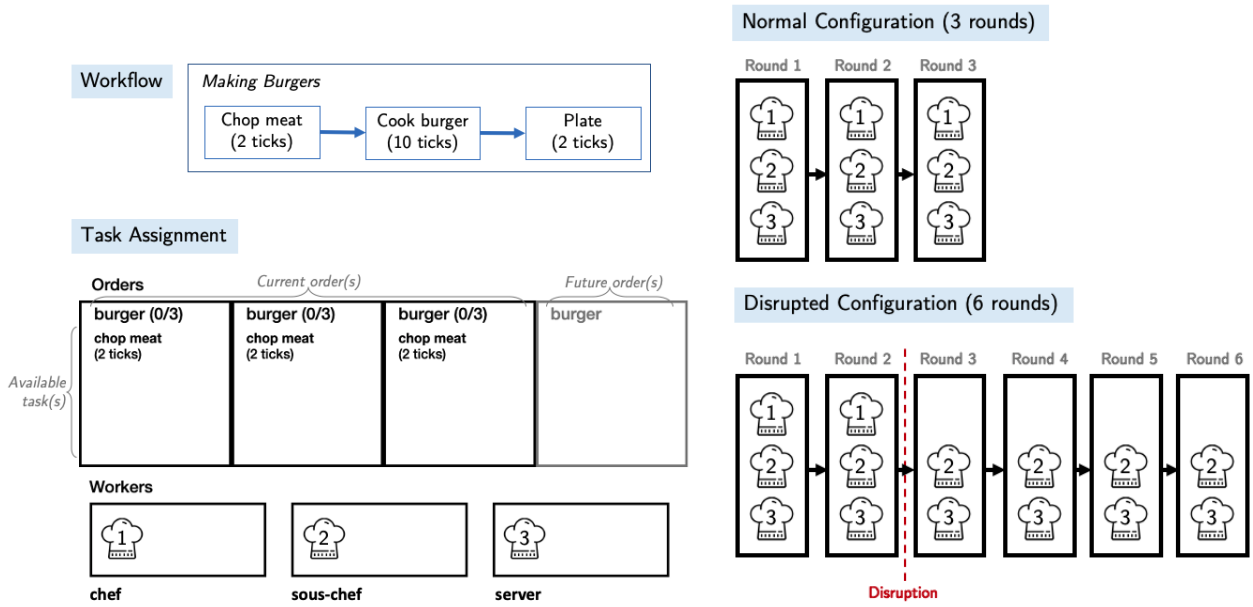
In this paper, we study whether machine learning can be used to infer rules that help improve workers' performance at sequential decision-making tasks. Whereas interpretable machine learning has focused on whether a human can understand the computation performed by a model (Caruana et al. 2015, Doshi-Velez and Kim 2017), our goal is to examine whether these models can convey useful insights to humans. To this end, we devise a novel algorithm for automatically learning decision-making rules or *tips* that, if correctly followed by the human worker, most improves their performance. Our algorithm builds on the idea of using model distillation (Bucilua et al. 2006, Hinton et al. 2015) for interpretable reinforcement learning (Verma et al. 2018, Bastani et al.

2018), where the strategy is to first train a high-performance neural network decision-making policy using reinforcement learning (Sutton and Barto 2018), and then train an interpretable policy to approximate this neural network. Our algorithm first 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 human strategy 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 workers, the Q-network also encodes how changes to the human strategy affect their performance. Then, we derive a novel objective that captures the discrepancy between the actions taken by the human policy compared to the optimal policy. Our algorithm selects an interpretable tip that minimizes this discrepancy if followed by the human; intuitively, this tip best bridges the gap between the humans and the optimal policy. Importantly, this tip does so in a way that accounts for which actions are consequential for achieving higher performance—i.e., it suggests actions are expected to improve their long-term performance of the human rather than simply mimic the neural network. We must also carefully design the search space of tips so that human workers can correctly follow the rule. That is, the tip must be an *interpretable model* whose computation process can be understood by humans (Friedman et al. 2008, Letham et al. 2015). In particular, we design the search space to consist of if-then-else rules. Despite their simplicity, these tips 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 design a game where human players act as managers for a virtual kitchen. An illustration of this task is shown in Figure 1a. In this game, the player must assign subtasks (e.g., chopping, cooking, plating) to virtual kitchen workers (e.g., chef, sous-chef, server) with varying capabilities in a way that optimizes the time it takes to complete a set of food orders. There are two aspects of this game that make it challenging: (i) each virtual kitchen 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 player 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 perform an extensive behavioral study on Amazon Mechanical Turk based on two different configurations of our virtual kitchen environment. In the *normal* configuration, the participant

Figure 1 Overview of behavioral study: virtual kitchen management.



(a) Workflow and main experimental task

(b) Study configurations

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 human participants, 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 tips, and then demonstrate how providing this decision-making rule improves the performance of the participants.

Our experimental results demonstrate that our algorithm can generate novel insights that enable human participants to substantially improve their performance compared to counterparts that are not shown the tip or are shown alternative tips derived from natural baselines. Our tip can speed up learning by up to three rounds of in-game experience, demonstrating that it can significantly reduce the cost of learning. Furthermore, in the presence of in-game disruption, our results suggest that our tip enables the participants to discover additional strategies. Interestingly, participants do not merely adjust their actions by blindly following the tip. Instead, as they gain experience with the game, they increasingly understand the significance of the tip and improve their performance in ways beyond the surface-level meaning of the tip. Our findings suggest that our algorithm infers interpretable and useful insights about the underlying task, and successfully conveys these insights to human participants. Ultimately, we demonstrate how machine learning can be used

to better understand human behavior, and how to leverage this knowledge to improve human decision-making.

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 (2020) 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 (2021) 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 (Caruana et al. 2015, Letham et al. 2015). 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 adopting 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), largely focusing on whether a human can understand why the model is making a certain prediction (Caruana et al. 2015, Letham et al. 2015, Ribeiro et al. 2016, Doshi-Velez and Kim 2017). There has also been work on understanding how humans interpret predictions of machine learning models (Narayanan et al. 2018), and on measuring and improving interpretability (Lage et al. 2018). In contrast, our goal is to understand whether machine learning algorithms can communicate insights to improve human performance at complex sequential decision-making tasks. Finally, incorporating human domain knowledge into algorithms has recently received increased attention (Arvan et al. 2019, Ibrahim

et al. 2020). Our work contributes to these streams of literature in two ways. First, we show that a simple intervention–providing a simple tip–can help speed up and complement workers' learning. Second, we develop a novel algorithm that leverages the readily available trace data and automatically identifies tips designed to help improve human performance at a challenging sequential decision-making task.

2. Inferring Tips via Interpretable Reinforcement Learning

Consider a human making a sequence of decisions to achieve some desired outcome. We study settings where current decisions affect future outcomes—for instance, if the human decides to consume some resources at the current time step, they can no longer use these resources in the future. These settings are particularly challenging for decision-making due to the need to reason about how current actions affect future decisions, making them ideal targets for leveraging tips to improve human performance. In particular, our goal is to provide insights to the human that enable them to improve their performance.

2.1. Background on MDPs

We begin by formalizing the tip inference problem. We model our setting as the human acting to maximize reward in a Markov Decision Process (MDP) $\mathcal{M} = (S, A, R, P, \gamma)$ over a finite time horizon T. Here, S is the state space, A is the action space, R is the reward function, and P is the transition function. Intuitively, a state $s \in S$ captures the current configuration of the system (e.g., available resources), and an action $a \in A$ is a decision that the human can make (e.g., consume some resources to produce an item). More precisely, we represent the human as a decision-making policy π_H mapping states to (possibly random) actions. At each time step $t \in \{1, ..., T\}$, the human observes the current state s_t and selects an action a_t to take according to the probability distribution $p(a_t \mid s_t) = \pi_H(s_t, a_t)$. Then, they receive reward $r_t = R(s, a)$, and the system transitions to the next state s_{t+1}, which is a random variable with probability distribution $p(s_{t+1} \mid s_t, a_t) = P(s_t, a_t, s_{t+1})$, after which the process is repeated until $t = T$. A sequence of state-action-reward triples sampled according to this process is called a *rollout*, denoted $\zeta = ((s_1, a_1, r_1), ..., (s_T, a_T, r_T))$. The human's goal is to act according to a policy π_H that maximizes the cumulative expected reward $J(\pi_H)$, where

$$J(\pi) = \mathbb{E}_{\zeta \sim D^{(\pi)}} \left[\sum_{t=1}^{T} \gamma^t r_t \right],$$

and where $D^{(\pi)}$ is the distribution of rollouts induced by using policy π.

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

Now, given the MDP $\mathcal{M}$ along with the human policy π_H, our goal is to learn a decision-making rule or tip ρ that most improves the performance of the human as measured by the cumulative expected reward $J(\pi_H)$. Formally, a tip indicates that in certain states s, the human should use action $\rho(s) \in A$ instead of their own policy π_H. For simplicity, we assume that the human always follows the tip; while this assumption does not always hold in practice, we find that it works well as long as the human can understand the tip along with its rationale. Thus, we consider tips in the form of a single, interpretable rule:

$$\rho(s) = \text{if } \psi(s) \text{ then take action } a,$$

where $a \in A$ is an action and $\psi(s) \in \{\text{true}, \text{false}\}$ is a logical predicate over states $s \in S$—e.g., it might say that a sufficient quantity of a certain resource is currently available. Intuitively, a tip $\rho = (\psi, a)$ says that if the condition ψ is satisfied, then the human should use action a; otherwise, they should use their default action $\pi_H(s)$.

Note that this tip 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 tip is a pair $\rho = (\psi, a)$ of a predicate ψ and an action a.

Assuming the human follows this tip exactly, then the resulting policy they use is $\pi_H \oplus \rho$, where

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

In particular, $\pi_H \oplus \rho$ represents the setting where the human 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, given a class of rules $\rho \in \mathcal{R}$, our goal is to choose the one that most improves the human's performance—i.e.,

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

[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.

The tip inference problem is to compute ρ^*. 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. Tip Inference 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 $\rho \in \mathcal{R}$.

We first compute the optimal Q-function Q^* by (approximately) solving the Bellman equations

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

using Q-learning (Watkins and Dayan 1992), where we parameterize Q^* using a neural network with a single hidden layer. Then, we leverage the following result from Bastani et al. (2018):

Lemma 1. *For any policy π, we have*

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

We can use this result to rewrite the objective $J(\pi_H \oplus \rho)$ in (1) as follows:

$$J(\pi_H \oplus \rho) = \mathbb{E}_{\zeta \sim D^{(\pi_H \oplus \rho)}}\left[\sum_{t=1}^{T} Q^*(s_t, a_t)\right].$$

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 tip ρ does not drastically change the human's decisions or the distribution over rollouts of the human–i.e., $D^{(\pi_H \oplus \rho)} \approx D^{(\pi_H)}$. Then, we have

$$\begin{aligned} J(\pi_H \oplus \rho) &= \mathbb{E}_{\zeta \sim D^{(\pi_H \oplus \rho)}}\left[\sum_{t=1}^{T} Q^*(s_t, a_t)\right] \\ &\approx \mathbb{E}_{\zeta \sim D^{(\pi_H)}}\left[\sum_{t=1}^{T} Q^*(s_t, (a_t \oplus \rho)(s_t)\right], \end{aligned}$$

where for a given tip $\rho = (\psi, a)$ and action a', we have

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

Furthermore, we approximate the expectation in our objective using sampled rollouts $\zeta_1, ..., \zeta_k \sim D^{(\pi_H)}$ from the human policy π_H, where $\zeta_i = ((s_{i,1}, a_{i,1}, r_{i,1}), ..., (s_{i,T}, a_{i,T}, r_{i,T}))$. Thus, our algorithm computes the tip

$$\hat{\rho} = \arg\max_{\rho} \frac{1}{k} \sum_{i=1}^{k} \sum_{t=1}^{T} Q^*(s_{i,t}, (a_{i,t} \oplus \rho)(s_{i,t})), \tag{2}$$

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

$$\hat{Q} = \arg\min_{Q \in \mathcal{Q}} \sum_{i=1}^{h} \sum_{t=1}^{T} (Q(s_{i,t}, a_{i,t}) - Q_{i,t})^2 \qquad \text{where} \qquad Q_{i,t} = \sum_{\tau=t+1}^{T} r_{i,\tau}.$$

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

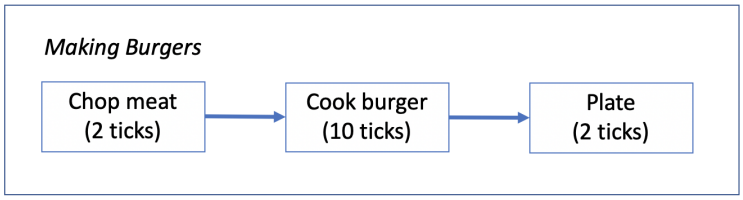
$$\hat{\rho} = \arg\max_{\rho \in \mathcal{R}} \frac{1}{k} \sum_{i=1}^{k} \sum_{t=1}^{T} \hat{Q}(s_{i,t}, (a_{i,t} \oplus \rho)(s_{i,t})). \tag{3}$$

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-management game that can be played by individual human users.

In this game, the user takes the role of a manager of several virtual workers (namely, chef, sous-chef, and server) producing food orders (all burgers) in a virtual kitchen. Each order itself consists of a fixed set of subtasks (namely, chop meat, cook burger, or plate burger). The game consists of a fixed number of steps or *ticks*; on each step, the user must decide which (if any) subtask to assign to an idle worker. The worker becomes busy for a number of subsequent time steps, after which the subtask is completed and the worker becomes idle again. An order is completed once all its subtasks are completed, and the game is complete once all orders are completed. The user's goal is to complete the game in as few time steps as possible.

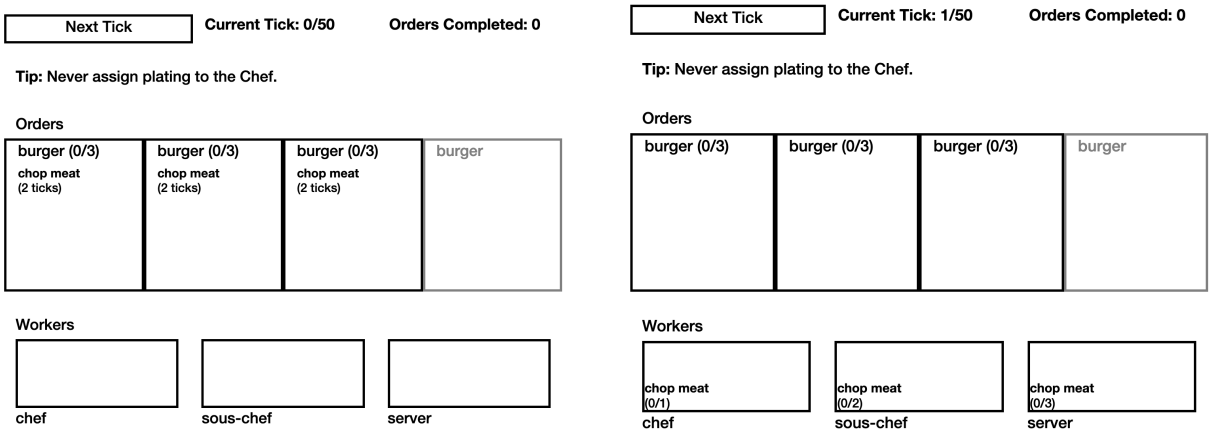
Figure 2 **Subtasks required to make a burger with a median processing time for each subtask.**



There are two key aspects of the game that make it challenging for the human to perform well. First, as illustrated in Figure 2, the subtasks have dependencies—i.e., some subtasks of an

order can only be assigned to a worker once other subtasks of the same order have already been completed. In particular, "cook burger" can only be assigned once "chop meat" is completed, and "plate burger" can only be assigned once "cook burger" is completed. Second, the virtual workers are heterogeneous—i.e., different workers take different numbers of time steps to complete different subtasks. In particular, 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. As a consequence, the user faces the following dilemma. Suppose a worker becomes idle, but there is no available subtask that that worker completes quickly. Then, the user must either assign a suboptimal subtask to that worker, or leave the worker idle until such a subtask to become available. For instance, if the server is idle but all available subtasks are "cook burger", then the user must either assign "cook burger" to the server, or leave the server idle until a "serve burger" task becomes available. Furthermore, users are not shown the exact number of steps each worker takes to complete each subtask, only the median number of steps taken. The true number of steps for a given worker is only revealed if they assign the subtask to that worker (see Figure 3 for example game screenshots). Thus, the user must experiment to learn this information over time.

Figure 3 Example screenshots from the game.



(a) The initial state where users observe available subtasks, median times to completion, and three idle virtual workers. The interface also shows the current tick, time limit, current progress, and potential 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.

3.1. Formulation as an MDP

We describe the MDP encoding our kitchen game. At a high level, the states encode progress towards completing all the food orders, the actions encode the currently available assignments of

subtasks to workers, and the rewards encode the number of steps taken to complete all orders. More specifically, the states encode (i) which subtasks have been completed so far across all orders, and (ii) which subtask has been assigned to each virtual worker (if any), as well as how many steps remain to complete this subtask. Next, the actions consist of all possible assignments of available subtasks (i.e., have not yet been assigned) to available virtual workers (i.e., not currently working on any subtask). Finally, the reward is -1 at each step, until all orders are completed; thus, the total number of steps taken to complete all orders is the negative reward.

3.2. Search Space of Tips

Next, we describe the search space of tips 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 tips 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 $k!/(k-m)!$ 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 tips 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. More precisely, 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 tips 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 tips are still too complicated for human users to internalize. Thus, we post-process the tips inferred by our algorithm by aggregating over tuples (o, s, w) that have the same s and w.[2] For example, instead of considering two separate tips

$$\begin{aligned}
&\text{if } (\text{order} = \text{burger}_1 \ \wedge \ \text{subtask} = \text{cooking} \ \wedge \ \text{virtual worker} = \text{chef}) \\
&\qquad \text{then } (\text{assign } (\text{burger}_1, \text{cooking}) \text{ to chef}) \\
&\text{if } (\text{order} = \text{burger}_2 \ \wedge \ \text{subtask} = \text{cooking} \ \wedge \ \text{virtual worker} = \text{chef}) \\
&\qquad \text{then } (\text{assign } (\text{burger}_2, \text{cooking}) \text{ to chef}),
\end{aligned}$$

we merge them into a tip

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

In other words, a tip is a combination of tips $\rho = (\rho_1, ..., \rho_k)$. Finally, the score our algorithm assigns to such a tip is $J(\rho) = \sum_{i=1}^{k} J(\rho_i)$; then, it chooses the tip with the highest score.

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

3.3. Tip Inference Algorithm Implementation

Next, we describe how our algorithm computes the optimal Q-function Q^* for the kitchen game MDP. 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 parameterized by $\theta \in \Theta \subseteq \mathbb{R}^{d_\Theta}$; typically, π_θ is a 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 has been shown to perform well in practice.

For the kitchen game MDP, we use state features including whether each subtask of each order is available, the current status of each virtual worker, and the current time step. 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 a finite horizon, we use a discount factor of $\gamma = 1$.

Once we have computed π_*, we use our tip inference algorithm described in Section 2.3 to learn an estimate $\hat{Q}$ of the expert Q-function $Q^{(\pi_*)}$ for π_*. We choose $\hat{Q}$ to be a random forest (Breiman 2001). It 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 tips on state-action pairs collected from observing human users playing our game. Since our goal is to help human users improve their performance, we restrict the training dataset to the bottom 25% performing human users. In addition, we apply two post-processing steps to the set of candidate tips. First, we eliminate tips that apply in less than 10% of the (featurized) states that occur in the human dataset. This step eliminates high-variance tips that may have large benefit, but are useful only a small fraction of the time; we omit such tips since our estimates of their quality tend to have very high variance. Second, we eliminate tips that disagree with the expert policy more than 50% of the time—i.e., for a tip (ψ, a), we have $\psi(s) = 1$ and $a \neq \pi^*(s)$ for more than 50% of state-action pairs in the human dataset. This step

[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. Once the neural network has been trained, we always take the action a with the highest probability.

eliminates tips that have large benefits on average, but frequently offer incorrect advice that can confuse the human user or cause them to distrust our tips.

4. Experimental Design

We perform an extensive pre-registered[4] behavioral study via Amazon Mechanical Turk (MTurk) to address the following research questions:

- Can our algorithm infer tips that can help participants significantly improve their performance on complex sequential decision-making tasks—specifically, in our virtual kitchen game?
- How do the tips inferred by our algorithm compare to the control group, as well as compare to two natural baselines: (i) a baseline algorithm that naïvely tries to match the optimal policy, and (ii) tips suggested by experienced human participants?
- Does the inferred tip help improve performance solely because the participant follows the tip, or does it induce them to improve in additional ways?

Baselines: Our first baseline is a naïve algorithm that, given rollouts $\hat{D}$ from the optimal policy π^*, computes the frequency of state-action pairs in $\hat{D}$—i.e.,

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

where $\#\{(\psi, a) \in \hat{D}\}$ 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 $\hat{D}$. Then, it selects

$$\hat{\rho}_{\text{bl}} = \arg\max_{\rho} \frac{1}{k} \sum_{i=1}^{k} \sum_{t=1}^{T} C^*(s_{i,t}, (a_{i,t} \oplus \rho)(s_{i,t})). \tag{4}$$

Intuitively, this objective ignores the structure of the MDP implicitly encoded in the Q-function, and instead directly tries to imitates the optimal policy. For our second baseline, we show each participant a comprehensive list of candidate tips and ask them to select the one they believe would most improve the performance of future players.

Experimental phases: 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 the 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. In *Phase I*, we have participants play the game without tips, and collect data on the actions they take. Then, we use the tip inference algorithm in conjunction with this data to generate a tip; our algorithm and the baseline algorithm use this data to focus on conveying information not already known by the humans, whereas the human tip depends on the suggestions by participants in Phase I. Then, in *Phase II*, we evaluate whether providing our tips can significantly improve the performance of a new set of participants, compared to the control group, as well as compared to the two baselines.

[4] The full pre-registration document for our study is available at `https://aspredicted.org/blind.php?x=8ye5cb`.

Scenarios and configurations: Each human participant plays the game several times in sequence, allowing them to learn good strategies over time. For each time the participant 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 participant as a *configuration*.

We evaluate our algorithm based on two different configurations of our game that are designed to evaluate different conditions under which tips might be useful. First, the *normal configuration* consists of three rounds of a single scenario we refer to as the *fully-staffed scenario*, in which the participant has access to all three virtual workers—i.e., chef, sous-chef, and server. Thus, our goal is to infer tips that help the participants fine-tune their performance at this scenario. Second, the *disrupted configuration* starts with the fully-staffed scenario for two rounds, but then switches to a modified scenario called the *understaffed scenario*, in which the participant has access to only two virtual workers—i.e., sous-chef and server, for four more rounds. Intuitively, we expect the participants 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.

4.1. Experimental Procedure

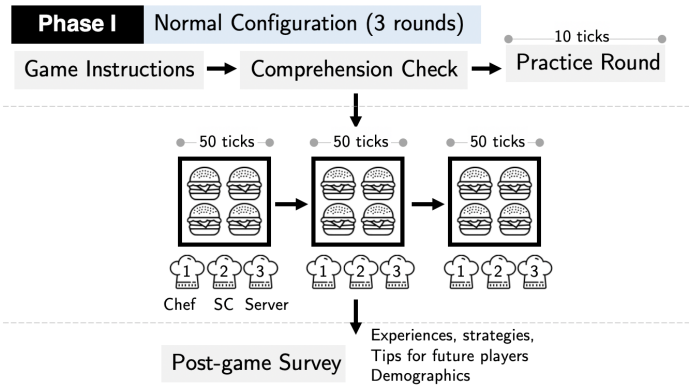
We perform separate experiments for each of the two configurations of our game. The high-level structure of our experimental design for each configuration is the same; they differ in terms of when we show tips to the participant and which tips we show. Before starting our game, each participant is shown a set of game instructions and comprehension checks; then, they play a practice scenario twice (with an option to skip the second one). 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. Then, they proceed to play the scenarios for the current configuration. Finally, after completing all scenarios, we give each participant a post-game survey that includes several questions regarding their experience with the game.

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 steps taken to complete each round. The bonus ranges from \$0.15 to \$0.75 per round.

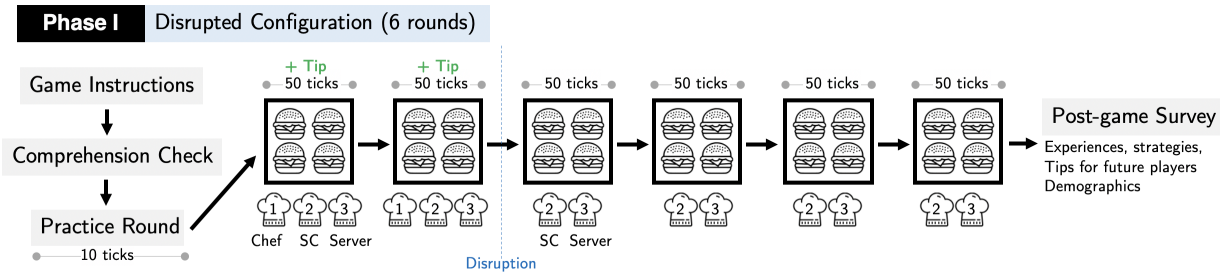
Phase I: For each configuration, we recruit 200 participants via MTurk to play the game. Figure 4 illustrates the study flow for Phase I. As part of the post-game survey, we ask the participants to suggest a tip for future players. In particular, we show each participant a comprehensive list of candidate tips and ask them to select the one they believe would most improve the performance

of future players. This list of tips is constructed by merging three types of tips: (i) all possible tips in the search space considered by our algorithm (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.

Figure 4 Study flow for Phase I.



(a) Normal configuration



(b) Disrupted configuration

Tip inference: Next, we use the data from the final round played by the participants to infer tips in three ways: (i) use our tip inference algorithm in conjunction with the data from Phase I, (ii) do the same with the naïve algorithm, and (iii) rank the candidate tips in the post-game survey based on the number of votes by the participants.

For the normal configuration, 183 participants[5] successfully completed the game. The top three tips inferred from each of the sources are reported in Table 1. For the algorithm tip, "Chef should never plate" is selected as it is expected to be the most effective at shortening completion time (2.43 steps). For the baseline tip, our naïve algorithm selects "Chef should chop once" as it is the

[5] They are 34.6 years old on average, 57.38% are female, and 73.22% have at least a two-year degree.

most frequently observed state-action pair in the data. Finally, for the human tip, "Strategically leave some workers idle" received the most votes among the participants (28.42%). It is worth noting that all of the tips most voted by past players are in line with the optimal strategy. 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. The second and third tips reflect the information participants could learn from assigning different tasks to different workers during the game: the server spends the most time cooking while the chef spends the most time plating.

Table 1 Top three tips inferred from different sources for the normal configuration.

Normal	**Tip #1**	Tip #2	Tip #3
Algorithm	**Chef should never plate**	Server plates three times	Server should skip chopping once
Baseline	**Chef should chop once**	Server should plate three times	Sous-chef should plate twice
Human (% voted)	**Strategically leave some workers idle** (28.42%)	Server should never cook (21.31%)	Chef should never plate (13.11%)

For the disrupted configuration, 172 participants[6] successfully completed the game. Table 2 reports the top three tips inferred from each of the sources. The best algorithm tip is "Server should cook twice" with the expected completion time reduction of 2.32 steps. Interestingly, only 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. For the baseline arm, all three tips are in line with the optimal policy, and the naïve algorithm chooses "Sous-chef should plate twice". Finally, for the human tip, "Server should cook once" or "Sous-chef should cook three times" got the most votes. Unlike in the normal configuration, the top two human tips here are not part of the optimal policy. In the optimal policy, the sous-chef and the server should each cook twice. The third human tip does align with the optimal policy; however, it is much less specific than the other tips. This highlights the increased difficulty for humans to identify the optimal strategy in the disrupted configuration compared to the normal configuration.

Phase II. We next evaluate the effectiveness of each of the inferred tips. In this phase, participants are randomly assigned to one of four arms, which differ in terms of the tip that is shown to them. 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 naïve algorithm), and the *human arm* (i.e., the tip with most votes from past players). We recruited 350 MTurk workers to play each arm in each configuration, totaling to 2,800 participants.

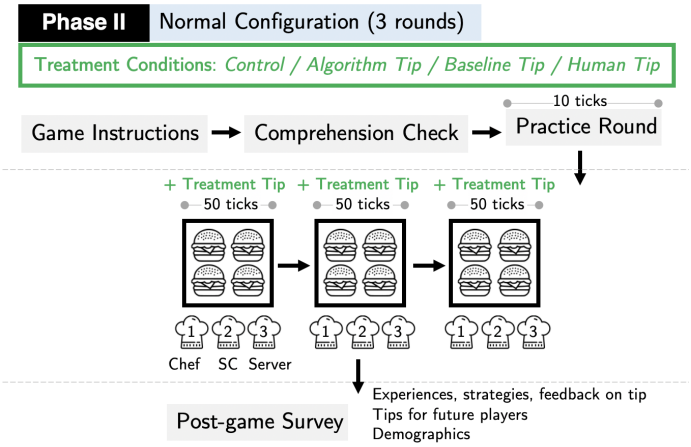
[6] They are 34 years old on average, 61.63% are female, and 77.91% have at least a two-year degree.

Table 2 Top three tips inferred from different sources for the disrupted configuration.

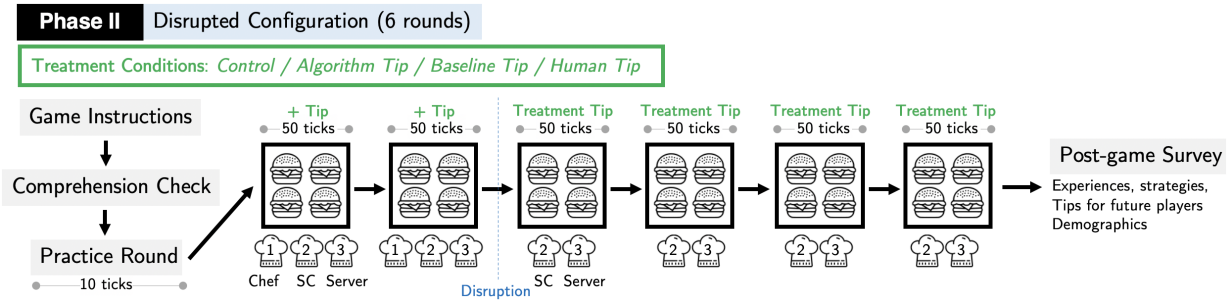
Disrupted	**Tip #1**	Tip #2	Tip #3
Algorithm	**Server should cook twice**	Sous-chef should plate once	Server should chop once
Baseline	**Sous-chef should plate twice**	Sous-chef should chop three times	Server should cook twice
Human (% voted)	**Server should cook once** (28.48%)	Server should never cook (23.84%)	Keep everyone busy (16.86%)

The specific tips we show in each round depends not just on the arm, but also varies from round to round depending on the configuration. For the normal configuration, we show the tip for the current arm in all three rounds. However, for the disrupted configuration, the tip for the current arm is specific to the understaffed scenario. Thus, we only show the tip for the current arm in 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 participant learn more quickly to play the fully-staffed scenario before switching to the understaffed scenario. Figure 5 illustrates the study flow for the second phase.

Figure 5 Study flow for Phase II.



(a) Normal configuration



(b) Disrupted configuration

4.2. Hypotheses and Overview of Analysis

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 previous participants 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 previous 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 previous 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 algorithm.
   - 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 algorithm.

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 complete each round of the game by taking the optimal number of steps, and how well the participants comply with the provided tip and learn additional optimal strategies.

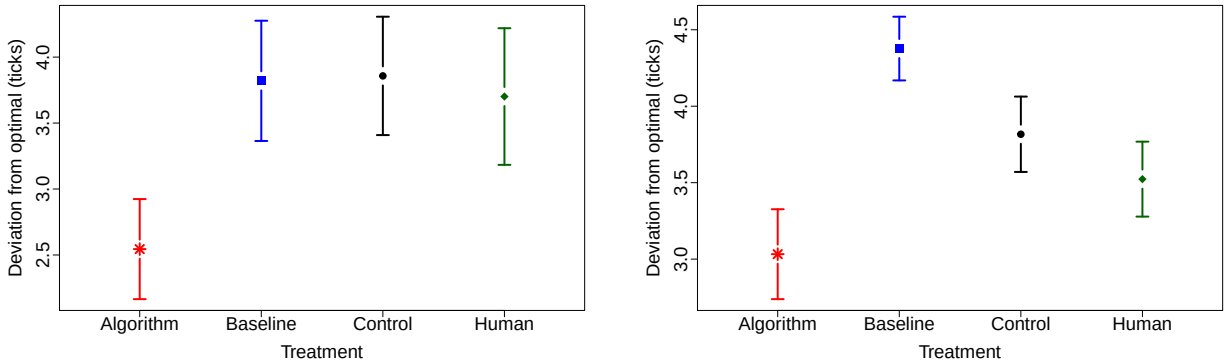
5. Experimental Results

Next, we describe our experimental results. Overall, our results demonstrate that despite the simplicity and conciseness of our tips, they capture strategies that are hard for participants to learn and can significantly improve their performance. In addition, we find evidence that the participants were not blindly following our tips, but combine them with their own experience to improve performance. Finally, we also find evidence that participants build on our tips by discovering additional strategies beyond the ones stated in the tips.

5.1. Our Tips Substantially Improve Performance

We assess performance by comparing the completion time in the final round. Overall, our algorithm was the most successful at improving performance. For the normal configuration, 1,317 participants[7] successfully completed the game. Participants shown our tip completed the final round in 22.54 steps on average (optimal is 20 steps), significantly outperforming those in other arms: 23.86 (control group, $t(329) = -4.397$, $p < 0.0001$), 23.73 (human tip, $t(312) = 3.628$, $p = 0.0002$), and 23.82 (naïve algorithm, $t(334) = -4.232$, $p < 0.0001$); see Figure 6a. In general, a substantial fraction of participants learned to play the game optimally after three rounds. 34.59% of those in our algorithm arm achieved optimal performance in the final round, while 24.44% to 28.90% of the other groups could do the same.

Figure 6 Performance of participants across conditions in the last round of Phase II.



(a) Normal configuration (optimal: 20 ticks)

(b) Disrupted configuration (optimal: 34 ticks)

In the disrupted configuration, 1,011 participants[8] successfully completed the game. Participants shown our tip completed the final rounds in 37.05 steps, again significantly outperforming those in other arms: 37.92 (control group, $t(243) = -4.361$, $p < 0.0001$), 37.53 (human tip, $t(246) = -2.52$, $p = 0.0061$), and 38.40 (naïve algorithm, $t(246) = -7.348$, $p < 0.0001$); see Figure 6b. We found similar results for the optimal performance rate. Only participants given our tip achieved optimal performance: 18.79% of participants in our algorithm arm played optimally, compared to 1.14% of the human tip arm, 0.99% of the naïve algorithm arm, and 0.51% of the control arm. As there were no significant differences in performance across treatments when playing the initial fully-staffed rounds, remarkably poor performance outside of our algorithm arm reflects the difficulty of managing the understaffed scenario.

Our results also help us understand the reasons behind the success of our tip. Intuitively, the naïve algorithm performs poorly since it blindly tries to mimic the optimal policy rather than focus

[7] They are 33.3 years old on average, 51.03% are female, and 67.73% have at least a two-year degree.

[8] They are 34.9 years old on average, 60.14% are female, and 70.43% have at least a two-year degree.

on accounting for consequential decisions. For instance, in the disrupted configuration, it infers the tip "Sous-chef should plate twice", which actually reduces performance. The actions suggested by this tip occur at the end of the game, which is too late to significantly benefit overall performance. In contrast, our algorithm focuses on mimicking decisions made by the optimal policy that have long-term benefits.

Finally, while the human arm performs better than the control group, it could not reach the performance of our tip. The human suggested tips suffer from humans' inability to translate their strategy into a concise and specific tip and to adapt their strategy to disruption. In the normal configuration, the human suggested tip "Strategically leave some workers idle", while capturing the key strategy, is more shallow and less specific than other tips, leaving participants to figure out the actual strategy. In the disrupted configuration, their inability to *un-learn* and adapt their strategy to the disrupted environment becomes apparent. 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. Thus, the success of our tip is due in part to how it offers specific advice that helps humans adapt to disruption and exploring potentially counterintuitive strategies[9].

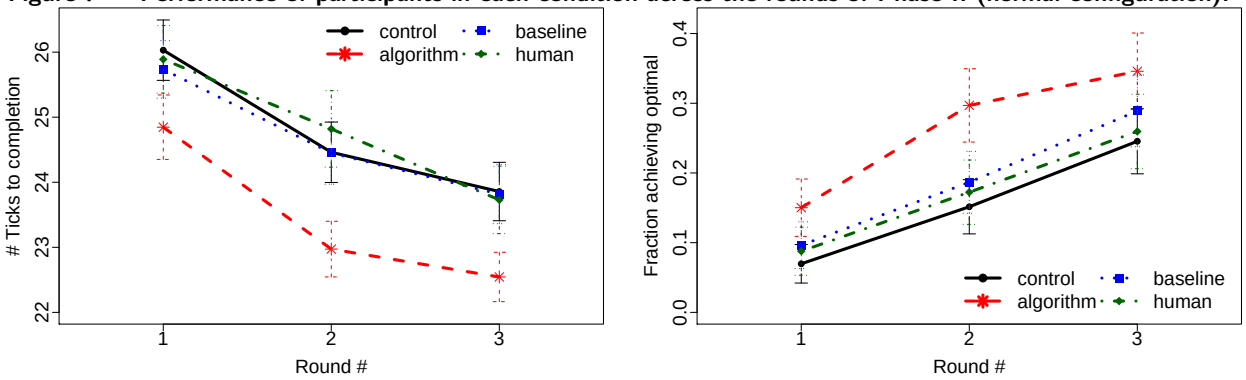
5.2. Learning over Time: Our Tips Speed Up Learning

Next, we study how performance improves across rounds as participants learn better strategies. In particular, tips can be thought of as a substitute for learning, reducing the number of rounds needed for participants to achieve a certain performance level. Recall that participants in the normal configuration had three game rounds over which they could learn and improve, while those

[9] 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.

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.

Figure 7 Performance of participants in each condition across the rounds of Phase II (normal configuration).



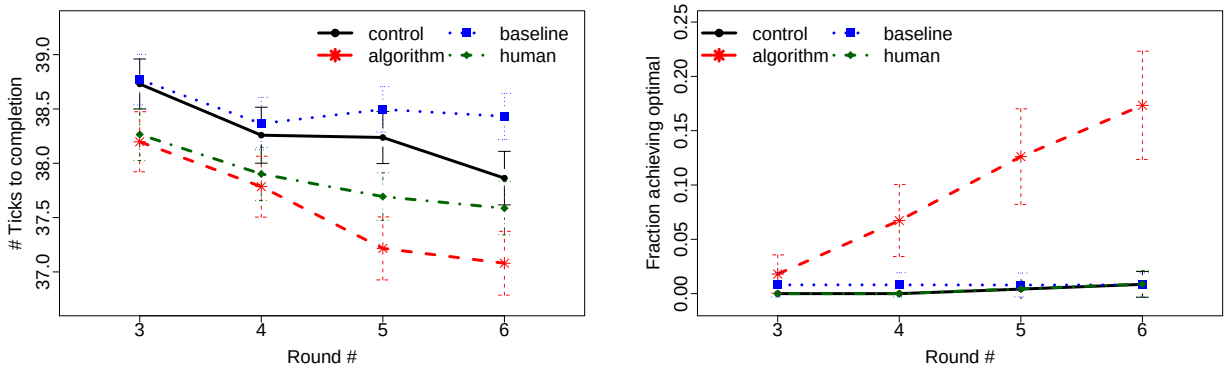
(a) Ticks to completion

(b) Fraction achieving optimal

In the normal configuration, our tip speeds up learning by at least one round compared to any of the other arms. From Figure 7a, we observe that participants in all arms improved over the three rounds. Although participants in each of the three treatment arms received their corresponding tip 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 speeds up learning by at least one round compared to any of the other arms—i.e., the performance of participants given our tip on the kth round was similar to the performance of participants in alternative arms on the $(k+1)$th round. Figure 7b displays the fraction achieving this optimal performance for each arm over the rounds. As before, we observe that overall participants improved over the three rounds, and that a significantly larger fraction of participants in the algorithm arm could reach the optimal solution.

The improved speed of learning was even more apparent under the disrupted configuration as illustrated in Figure 8a: while all participants improved over the rounds, those in the control group took four rounds to achieve the same level of performance as those provided with our tip on the first round. Note that the human arm initially improves performance comparably to our arm; however, it levels off towards the end whereas our arm continues to improve. In addition, Figure 8b shows the fraction of participants that achieved optimal performance. Interestingly, only participants given our tip achieved optimal performance in any round. These results suggest that our tip, while simple and concise, 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.

Figure 8 Performance of participants in each condition in the last four rounds of Phase II (disrupted configuration).



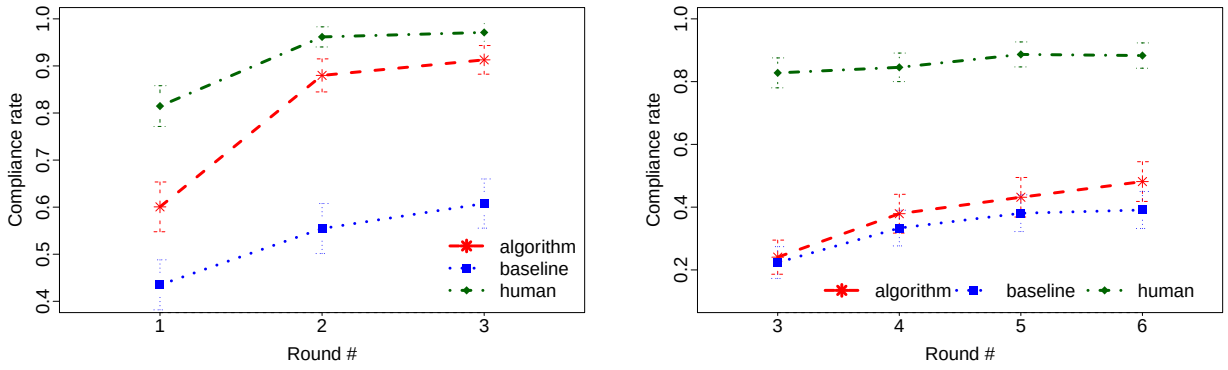
(a) Ticks to completion

(b) Fraction achieving optimal

5.3. Complying to Tips: Human Users are More Compliant over Time

The effectiveness of a tip critically depends on whether the participant follows it; to better understand this relationship, we study how well participants complied with tips across arms. Importantly, participants were not informed of the source of the tips, so variation in compliance is entirely due to the contents of the tips.

Figure 9 Compliance rate across the rounds of Phase II.



(a) Normal configuration

(b) Disrupted configuration

In the normal configuration, we find that participants increasingly comply with tips across rounds in all arms, as can be seen in Figure 9a. However, in the final round, a significantly higher fraction participants complied with our tips and the human suggested tips compared to the naïve algorithm tips, suggesting that participants determined that the naïve algorithm did not suggest a useful tip. In the disrupted configuration, the compliance of the human suggested tip ("Server cooks once") is significantly higher than the others, likely because it is the most intuitive. In contrast, our tip ("Server cooks twice") is counter-intuitive since the server is slow at cooking; nevertheless, it is

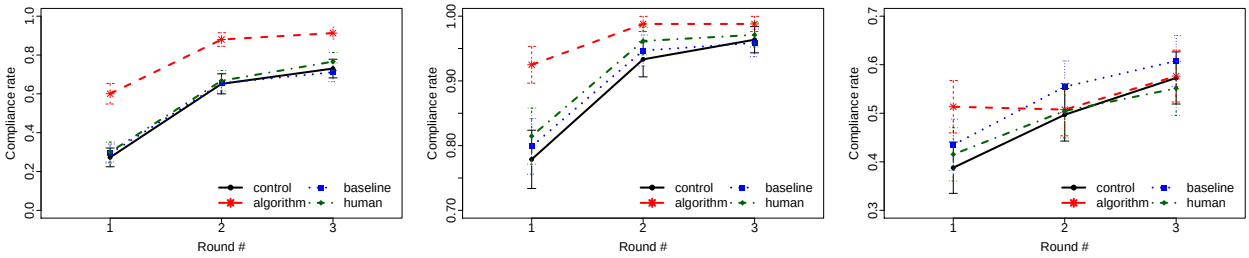
more effective at improving performance when followed. These results suggest that participants are not blindly following tips; instead, they only follow the tip if it suggests a strategy that makes sense to them. Furthermore, they show that our strategy is effective even though our tips are inferred under the assumption that the participant exactly follows the tip. Intuitively, we believe our approach remains effective since our objective of identifying a tip that maximizes long-term payoff is consistent with the idea that participants only follow the tip if it encodes an effective strategy; that is, they follow the strategy as long as they can understand it and it is effective.

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

A key question is understanding how humans internalized and actualized the strategies encoded in the tips we inferred. We study this question in two ways. First, we examine *cross-compliance*, which is the compliance of the participant to alternative tips other than the one we showed them. Naïvely, there is no reason to expect participants to cross-comply with an alternative tip (assuming it does not overlap with the tip shown), beyond the cross-compliance of the control group to that tip. Thus, cross-compliance measures how showing one tip can enable participants to discover strategies beyond what is stated in that tip. Finally, we investigate whether the tips could help participants uncover the structure of the optimal policy beyond the simple rules they stated.

Cross-compliance. We find that our tips had high cross-compliance than other tips in both configurations. For the normal configuration, we find that participants across all arms learn not to assign plating to the chef (Figure 10a), strategically leave some virtual workers idle (Figure 10b), and let the chef chop only once (Figure 10c). These tips are all consistent with the optimal policy, suggesting that participants generally learn over time to improve their performance regardless of the treatment. Interestingly, participants in the algorithm arm have similar or higher cross-compliance compared to the other arms. This result suggests that our tip is the most effective as the information it encompasses the information conveyed by the other tips.

Figure 10 Cross-compliance rate across the rounds of Phase II (normal configuration).

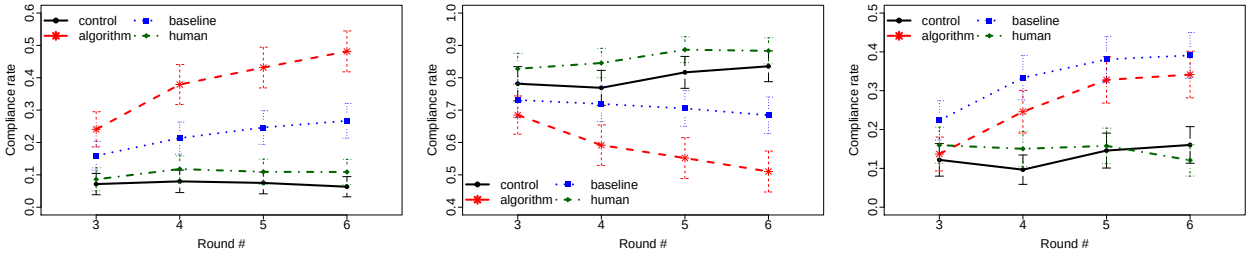


(a) Algorithm: "Chef shouldn't plate" (b) Human: "Leave some idle" (c) Baseline: "Chef chops once"

For the disrupted configuration, the cross-compliance of the human and control arms with our tip remained flat (Figure 11a). We observe a similar trend for the naïve algorithm tip (Figure

11c). These results suggest that participants do not naturally learn this strategy over time, most likely since it is counterintuitive. In particular, simple tips can greatly improve human performance by capturing counterintuitive strategies that take a great deal of experimentation to discover. Finally, Figure 11b shows the cross-compliance 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.

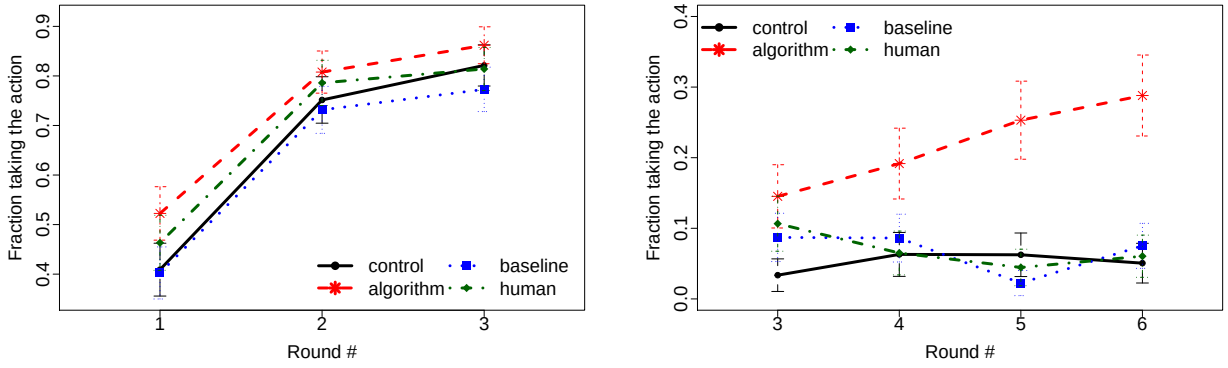
Figure 11 Cross-compliance rate across the rounds of Phase II (disrupted configuration).



(a) Algorithm: "Server cooks twice" (b) Human: "Server cooks once" (c) Baseline: "Sous-chef plates twice"

Uncovering the optimal policy. 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. For instance, the fraction of participants in each arm that never assigned cooking to the server in each round, as if they were following the tip "Server shouldn't cook", increases over time and within each round the fractions are not statistically different among the arms (see Figure 12a). This result suggests that people learned about this rule by themselves across all arms.

Figure 12 Fraction of participants taking optimal action beyond the tips they were shown across the rounds.



(a) Fully-staffed: "Server shouldn't cook" (b) Under-staffed: "Server chops once"

Furthermore, for the under-staffed scenario, the optimal policy requires balanced assignment of cooking and plating tasks to both sous-chef and server while assigning most of the chopping tasks to sous-chef. A key rule beyond our tip is thus "Server chops once". We find that only participants in our arm cross-complied with this auxiliary tip (see Figure 12b), again demonstrating that our tip enables participants to discover strategies beyond what is stated in the tip.

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 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 presence of in-game disruption, our results suggest that our tip enables the participants to discover additional strategies beyond the given 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. Such data is often noisy and too granular to be readable by and immediately useful for humans. Our machine learning 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, personalizing tips to individual workers could greatly improve the performance of our tip inference algorithm—ideally, we would infer tips personalized for different skill levels and individual worker characteristics. Next, 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, which would enable us to better understand 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 collaborative or competitive settings. Work teams are a common source of disruption. When one member leaves, similar to the chef leaving the kitchen in our game, there will be spillovers on other members' productivity and performance (Moon et al. 2019). 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. Similarly, potential congestions could be incorporated into the algorithm to infer tips for settings in which workers may compete for shared resources. 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. Our empirical results suggest that following the tip depends not just on whether the human trusts the algorithm, but on whether the human understands why the proposed tip is reasonable. This insight brings a novel dimension to human trust in artificially intelligent systems.

Acknowledgments

This research was supported in part by the Wharton Behavioral Lab, the Mack Institute for Innovation Management, and the Wharton Risk Management and Decision Processes Center.

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 (2020) Recovering from critical incidents: Evidence from paramedic performance. *Manufacturing & Service Operations Management* .

Bavafa H, Jónasson JO (2021) The variance learning curve. *Management Science* 67(5):3104–3116.

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.

Bucilua C, Caruana R, Niculescu-Mizil A (2006) Model compression. *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 535–541.

Caruana R, Lou Y, Gehrke J, Koch P, Sturm M, Elhadad N (2015) Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 1721–1730.

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.

Doshi-Velez F, Kim B (2017) Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608* .

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.

Friedman JH, Popescu BE, et al. (2008) Predictive learning via rule ensembles. *Annals of Applied Statistics* 2(3):916–954.

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.

Hinton G, Vinyals O, Dean J (2015) Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* .

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.

Lage I, Ross AS, Kim B, Gershman SJ, Doshi-Velez F (2018) Human-in-the-loop interpretability prior. *arXiv preprint arXiv:1805.11571* .

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/`.

Moon K, Bergemann P, Brown D, Chen A, Chu J, Eisen E, Fischer G, Loyalka PK, Rho S, Cohen J (2019) Manufacturing productivity with worker turnover. *Available at SSRN 3248075* .

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.

Narayanan M, Chen E, He J, Kim B, Gershman S, Doshi-Velez F (2018) How do humans understand explanations from machine learning systems? an evaluation of the human-interpretability of explanation. *arXiv preprint arXiv:1802.00682* .

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.

Ribeiro MT, Singh S, Guestrin C (2016) " why should i trust you?" explaining the predictions of any classifier. *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 1135–1144.

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, Barto AG (2018) *Reinforcement learning: An introduction* (MIT press).

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.

Verma A, Murali V, Singh R, Kohli P, Chaudhuri S (2018) Programmatically interpretable reinforcement learning. *International Conference on Machine Learning*, 5045–5054 (PMLR).

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.

Appendix A: Additional Details of the Experiment

Table A.1 Participants' demographic and gameplay information.

	Phase I: *Normal*	Phase II: *Normal*	Phase I: *Disrupted*	Phase II: *Disrupted*
Total	183	1,317	172	1,011
Mean age [range]	34.6 [18, 76]	33.3 [18, 74]	34 [19, 76]	34.9 [16, 84]
Female	57.38%	51.03%	61.63%	60.14%
$\geq$ 2-year degree	73.22%	67.73%	77.91%	70.43%
Median duration	18.82 minutes	20.50 min	27.80 min	26.80 min
Found the game difficult	60.66%	50.04%	70.93%	64.99%
Never played similar games	45.36%	43.82%	46.51%	43.52%

A.1. Participants' feedback on the provided tips.

As discussed in §5.3, the effectiveness of a tip highly depends on whether the participant follows it. We observe that the compliance of the more intuitive tip is significantly higher than the others. In the disrupted configuration, the tips inferred from our algorithm can be considered counter-intuitive since the server is slow at cooking. Here, we further explore how the participants responded to the tips presented to them. In the post-game survey, we asked the participants the following question: "What did you think about the tip for these last [three/four] rounds and how did you incorporate it in your strategy?" We then manually code these responses. Tables A.2 and A.3 exhibit the breakdown of participants in each arm based on the coded responses for normal and disrupted configurations, respectively. Note that participants were not informed of the source of the tips, so variation in compliance is entirely due to the contents of the tips. We also verified whether the participants saw the tip and asked them to write the content of such tip in the survey.

Table A.2 Participants' coded feedback on the provided tips (normal configuration).

Normal	Algorithm	Baseline	Human
	"Chef never plates"	*"Chef chops once"*	*"Leave some idle"*
(N1) Positive	25.87%	16.33%	29.23%
(N2) Negative	4.20%	5.44%	1.92%
(N3) Neutral	53.85%	51.70%	48.08%

Selected excerpts from the algorithm arm (disrupted configuration):

(D1) Positive:

- "It was very helpful. It made me focus on making sure the server cooked more even if that was not his obvious strength." (`1Q5tRaFyI9DwIn4`)
- "I ignored the tip at first, but later I used the tip and it helped me complete the tasks quickly." (ID: `279WDYHBVZVOSzI`)

Table A.3 Participants' coded feedback on the provided tips (disrupted configuration).

Disrupted	Algorithm *"Server cooks twice"*	Baseline *"Sous-chef plates twice"*	Human *"Server cooks once"*
(D1) Positive	23.10%	10.19%	25.87%
(D2) Negative	33.10%	37.58%	16.78%
(D3) Neutral	32.76%	42.99%	47.90%

- "At first I didn't follow it because it seemed counter intuitive since they're slow. But then I had trouble, so I tried it and came out ahead." (`10HkPUkR6o0qDFT`)
- "I did not listen to it at first because I didn't believe that it would actually help but it did." (`3CBo6Xi6oHuTN6H`)
- "The tip was helpful. Without it, I think I would have tried to complete the task without the Server cooking, which would have left someone idle for a long time." (`3oTJ1TnTGYqgsM7`)

(D2) Negative:

- "I think it was a bad tip. I couldn't figure out how to incorporate it successfully." (`2vk1hthUj2uRVYM`)
- "Seemed counterintuitive." (ID: `3G6Zg1V3LIORZCS`)
- "It did not help me. I did not use it for round 1, I used it for round 2 and it made me do worse, so round 3 I tried it again and was still unable to do well, so the last round I ignored the tip." (3GjhwJlOLvwYfOJ)
- "I don't think it helped. I thought having the sous chef cook 3 times would take too long and the point at which I tried it, I decided last minute to have the server cook twice. So I don't think it told me anything useful." (`4Ni66NMtRgJjOUx`)
- "It was not needed since the server took so much longer to cook." (`10uOrh9zAwHCC7P`)

(D3) Neutral:

- "I followed it completely." (`DV8DPipSHvSV8R3`)
- "I made server cook twice later on once I got used to the game." (`DV8DPipSHvSV8R3`)
- "I followed it to the letter." (`2YLp6zcyRzpPZS9`)

Selected excerpts from the human arm (disrupted configuration):

(D1) Positive:

- "It seemed pretty much essential to have server cook once." (`1DBAQnmYnuJkDMp`)
- "I thought it was smart and I used it exclusively." (`beijQ8guDyExa5r`)
- "It was accurate, and I implemented the tip." (`1pA8wDYgWc9hbIt`)
- "I felt that tip was valid, as the server primarily is useful plating/chopping. I only had him cook once." (`1rvkYTwgAjD0z4z`)
- "It helped because she could cook one burger but any more than that and your ticks would be too high." (`d6YSuigdikyaNdT`)

(D2) Negative:

- "It was not helpful, because it does not specify when the server should cook." (`1rDt0QKW6AkWso7`)
- "I used the tip but I don't think it was helpful. The server took long to cook." (`1jkpPZj85v3t8rP`)
- "I don't agree with this tip." (`1OiP5M1rCIsdHoN`)
- "It was not terribly helpful. I tried to incorporate but it did not seem to help" (`2BhT2VK2KivCy3X`)
- "It stunk honestly. The server takes forever to cook." (`21zfL2T3oQ3I1qj`)

(D3) Neutral:

- "I followed it and let the server cook once." (`2bHvWLqGlWZAsJs`)
- "Not sure if it helped or not." (`1OZuqmQLbSY7DZb`)
- "Just tried to adhere to it. Only had the server cook if he wasn't doing anything else." (`24AuDOtoF5XFJbU`)

We make a few notable observations. For both configurations, a substantially larger fraction of participants in the human arm responded positively to the tips compared to other arms. Similarly, a substantially smaller fraction of participants in the human arm viewed the tips negatively. These may suggest that human participants were effective at selecting a tip that would be also accepted by other humans. For the disrupted configuration, which is much more challenging that the normal one, the sentiment towards the content of the tip appears to be highly dependent on how intuitive it is and how easy it is to implement such tip. The algorithm tip and the human tips are similar and it appeared that the participants correctly recognized that the task distribution has to change after the in-game disruption. In the first two rounds with three virtual workers, participants had potentially learned each worker's skill level and developed a strategy based on such knowledge. Thus, they likely learned that the server should not cook. When the chef left the kitchen, they initially kept employing their original policy, but subsequently started to realize that the server does need to cook. The algorithm tip therefore appeared to be too aggressive or counterintuitive for the participants to adopt. The baseline tip was even less favorable in this case; it not only contradicts the preconceived notion that the server should only plate but also makes a recommendation for a subtask that is only available later on in the game. These findings in part explain the varying compliance across the arms and highlight the important tradeoff between intuitiveness and effectiveness at improving performance.

Appendix B: Additional Details of the Virtual Kitchen-Management Game

Table A.4 exhibits the number of time steps needed for each of the virtual workers to complete each of the subtasks required to complete a single burger order.

Table A.4 Virtual workers' skill matrix.

Completion time	Chopping meat	Cooking burger	Plating burger
Chef	1	4	6
Souf-chef	2	8	2
Server	3	12	1

B.1. Summary of the Optimal Policies

Here, we summarize the structure of the optimal policy for each scenario of the game. Note that there exists more than one exact sequence of decisions that belong to the optimal policies. In both scenarios, the participant must serve all four burger orders to complete each round.

Fully-staffed scenario: In this scenario, the participant has access to all three virtual workers. The optimal number of steps needed to complete the round is 20 ticks. The keys to achieving the optimal performance are: (i) all three workers should be assigned to chopping in the first time step, (ii) the chef must cook three of the burgers and the sous-chef must cook one (i.e., the second burger), (iii) the server should never cook and must be kept idle when the third burger becomes available for cooking; she should instead be assigned to plating the first cooked burger, (iv) the chef should never plate, (v) the sous-chef must plate one of the burgers, and (vi) all three workers should not be left idle except in the conditions stated earlier.

Understaffed scenario: In this scenario, the participant has access to only two virtual workers, namely, the sous-chef and the server. The optimal number of steps needed to complete the round is 34 ticks. The keys to achieving the optimal performance are: (i) both workers should be assigned to chopping in the first time step, (ii) the sous-chef and the server must cook two burgers each, (iii) the sous-chef must choose chopping over cooking after finishing her first chopping task, (iv) the server's first three tasks must be chopping, cooking, and cooking, (v) the sous-chef must plate three of the four burgers and the server must plate one, (vi) the second cooked burger must not be served until the third and fourth burgers are cooked, and (vii) both workers must be kept busy at all times. Compared to the fully-staffed scenario, the optimal policy for the understaffed case is much more complex and counterintuitive.

Appendix C: Example Screenshots from the Experiment

Figure A.1 Screenshots of the game introduction.

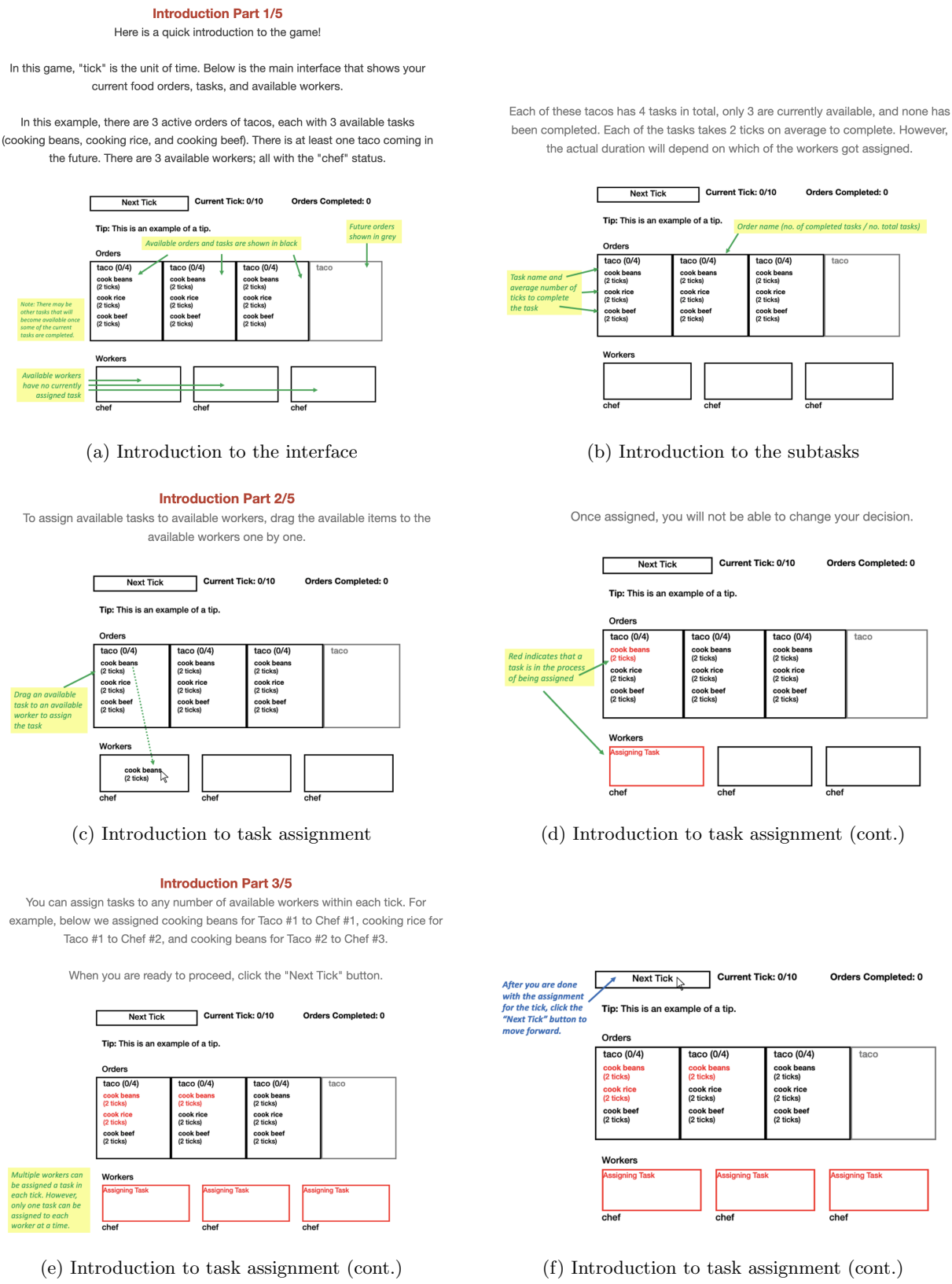
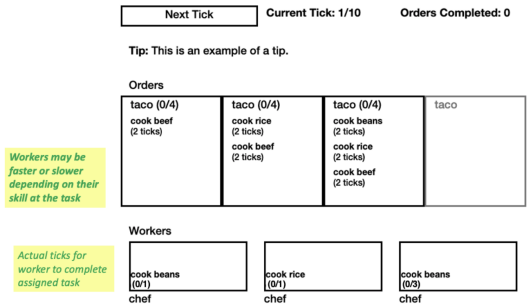


Figure A.2 Screenshots of the game introduction (continued).

Introduction Part 4/5

Each worker has different skills and will perform faster or slower than the other workers depending on the assigned task. You can learn each person's skill by assigning them different tasks and seeing how they perform.

Below, Chef #1 needs 1 tick to cook beans while Chef #3 needs 3 ticks to cook beans. Chef #2 needs 1 tick to cook rice.

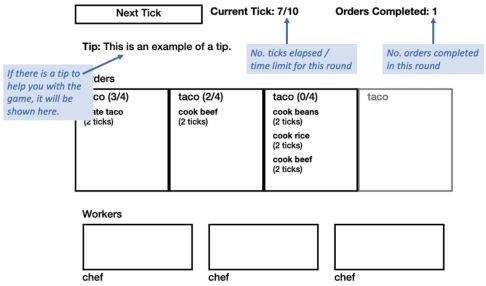


(a) Introduction to workers' skill levels

Introduction Part 5/5

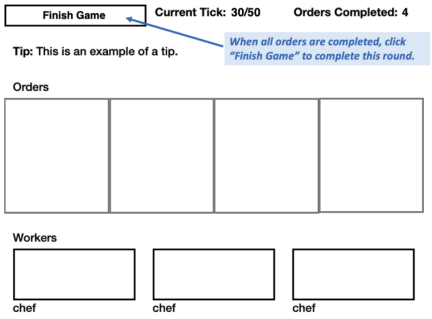
You can keep track of your progress by checking at the information at the top: the current tick, the time limit, and the number of completed orders so far.

In some scenarios, we might have a tip to help with your decisions. If available, the tip will be shown right above the list of food orders.



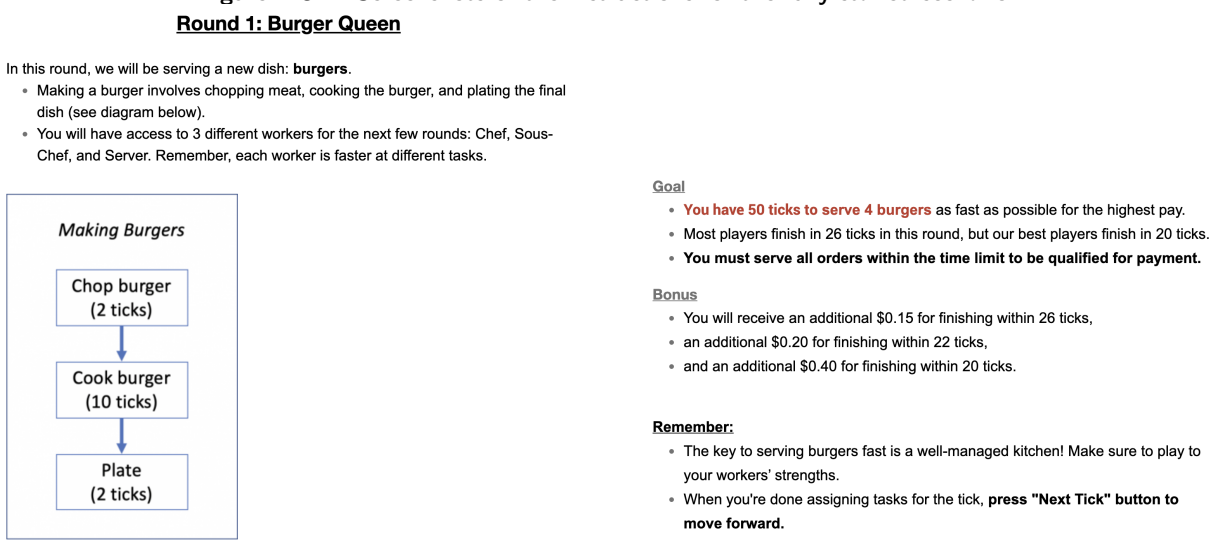
(b) Introduction to the tip

After you completed all the required dishes for the round, you will see a "Finish Game" button that you can click to move on to the next round.



(c) Introduction to round completion

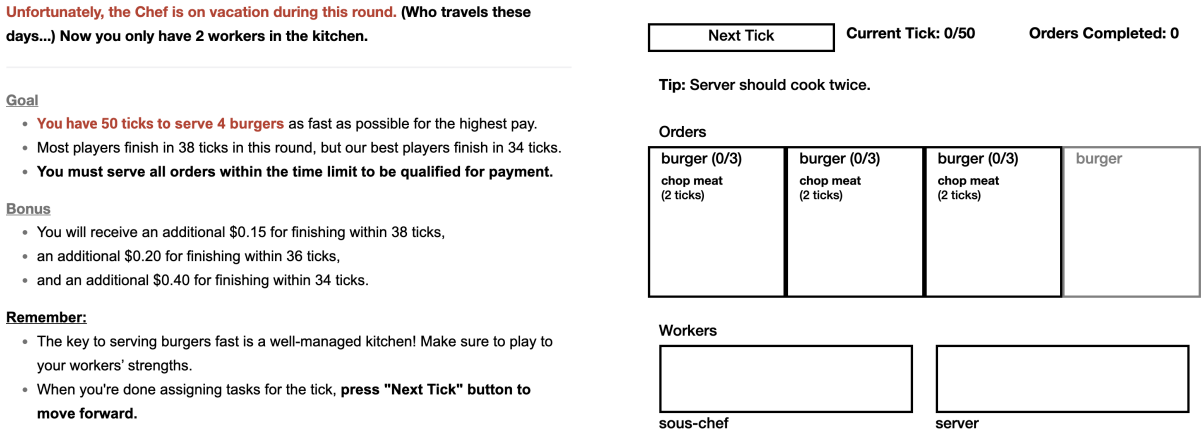
Figure A.3 Screenshots of the instructions for the fully-staffed scenario.



(a) Burger's subtasks and available workers

(b) Goal, incentives, and reminder

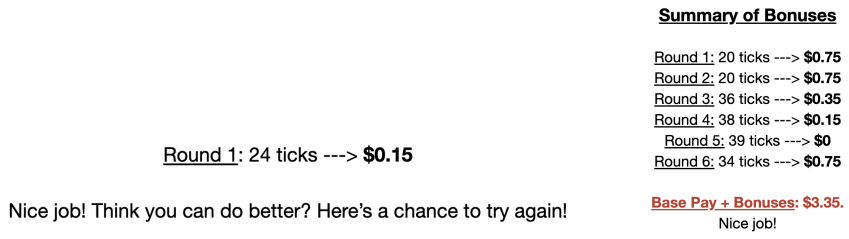
Figure A.4 Screenshots of the instructions for the understaffed scenario.



(a) Updated instructions following the in-game disruption

(b) Game interface (with the algorithm tip)

Figure A.5 Screenshots of the pay information.



(a) Individual round pay information

(b) Summary of total pay