Improving Human Sequential Decision-Making with Reinforcement Learning

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

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

Wichinpong Park Sinchaisri
University of California, Berkeley, Haas School of Business, parksinchaisri@haas.berkeley.edu

Workers spend a significant amount of time learning how to make good decisions. Evaluating the efficacy of a given decision, however, can be complicated—e.g., 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 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 experiments where participants manage a virtual kitchen. Our experiments show that the tips generated by our algorithm can significantly improve human performance relative to intuitive baselines. In addition, we discuss a number of empirical insights that can help inform the design of algorithms intended for human-AI interfaces. 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 reinforcement learning, sequential decision-making, human-AI interface

1. Introduction

Workers spend a significant amount of time on the job learning how to make good decisions that improve their performance (Chui et al. 2012). Yet, the impact of a current decision can be long-range—affecting future decisions/rewards in complex ways—making it difficult for them to evaluate the quality of a decision. This is exacerbated by the fact that multiple decisions are often made sequentially, making it hard to determine which decisions are responsible for good outcomes even in hindsight. Many jobs require such sequential decision-making, e.g., doctors ordering tests to optimize patient outcomes (Kleinberg et al. 2015), or workers choosing jobs on gig economy platforms to optimize their daily profits (Marshall 2020, Allon et al. 2023). As a concrete example,

physicians seek to learn good strategies for ordering lab tests, since obtaining test results in a timely fashion is necessary to minimize delays in patient visits; for instance, Song et al. (2017) finds that experienced physicians have learned to order these tests early 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 can be difficult because the connection between when tests are ordered and the overall quality of care is influenced by numerous other decisions made by the physician, as well as unrelated changes in the underlying environment (e.g., hospital congestion).

Learning on the job can significantly impact service quality, since workers likely make sub-optimal decisions during this time. For instance, when surgeons first use new devices, surgery duration increases by roughly a third, which can be costly to both patients and providers (Ramdas et al. 2017). Thus, when 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 instance, some workers are comparatively isolated—e.g., physicians working in rural hospitals 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.

Thus, a natural question arises: can we *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 by gig economy platforms; 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. However, trace data is often extremely noisy, granular, and of tremendous volume, rendering it unreadable to humans. At the same time, recent advances in reinforcement learning have enabled machines to achieve human-level or super-human performance at many challenging sequential decision-making tasks (Mnih et al. 2015, Silver et al. 2016). Thus, we might hope to leverage these techniques to mine high-volume trace data to automatically identify key bottlenecks in current human decision-making, as well as promising tips/advice to improve their performance.

In this paper, we perform a large-scale behavioral experiment to study whether reinforcement learning can be used to infer tips that improve human performance in sequential decision-making tasks. There is now a large body of evidence that machine learning predictions can improve human performance in *one-shot* decision-making—where the current decision does not affect future

outcomes—e.g., bail decisions (Green and Chen 2019), visual question answering (Chandrasekaran et al. 2017, 2018), satellite image analysis (Kneusel and Mozer 2017), and detecting deceptive reviews (Lai and Tan 2019). In these settings, it often suffices to provide the model's prediction to the user, potentially in an interpretable way to improve trust and compliance. However, sequential decision-making settings pose qualitatively different challenges, since current decisions can have long-term consequences and affect future observed states. In particular, we must figure out in *which* states we should intervene, which can be informed by examining bottlenecks in the current human policy. To this end, we devise a novel algorithmic framework for inferring simple tips that, if adopted, can improve the performance of the worker. Our algorithm aims to capture the *discrepancy* between the existing human policy (as captured by historical trace data) and the optimal policy, which helps us identify the most performance-improving tips for key bottlenecks in current human decision-making.

An additional challenge in sequential decision-making is that for these tips to improve performance, the human needs to understand how to operationalize them into their broader workflow. Otherwise, even if they comply with the tip, there is no guarantee that they correctly understand what decisions to make on other time steps to achieve optimal performance. In principle, even if a tip suggests optimal actions for the worker to take, and the worker complies with the tip perfectly, the overall performance could degrade since the worker subsequently makes poor decisions. Thus, our search space of candidate tips must focus on interpretable and actionable information that workers can easily operationalize. Whether humans can actually do so is an empirical question; thus, we conduct a large-scale behavioral experiment that studies how humans perceive and improve their own decision-making over time (given tips from either our algorithm, or via peer feedback or simple descriptive statistics), how they adjust other portions of their workflow to accommodate these changes, and how humans may incorrectly perceive bottlenecks in their own decision-making.

To summarize, two criteria are needed to *actually* improve human decision-making. First, our algorithm must identify sufficiently useful tips to improve performance (assuming humans comply with and effectively operationalize them). Second, humans must be able to understand and comply with our tip and, furthermore, effectively operationalize it by modifying their broader workflow.

Algorithm. Our algorithm builds on the idea of model distillation (Buciluǎ et al. 2006, Hinton et al. 2015) for interpretable reinforcement learning (Verma et al. 2018, Bastani et al. 2018), which involves first training a blackbox decision-making policy using reinforcement learning (Sutton and Barto 2018), and then training an interpretable policy to approximate the blackbox policy. However, unlike prior work, our goal is to infer an interpretable tip that best minimizes the discrepancy between the existing human policy and the blackbox policy, rather than to train the best-performing interpretable policy that is agnostic to the current human policy. Thus, the chosen tip is tailored

to current bottlenecks in the human decision-making policy, and accounts for which actions are consequential for achieving higher performance—i.e., following the tip is expected to improve the long-term performance of the human rather than to simply mimic the optimal policy. In order to easily convey our insights to humans, we design the search space over tips to consist of if-then-else rules. Despite their simplicity, we find that these tips can capture useful insights that are challenging for humans to learn by themselves in complex sequential decision-making problems.

Game. To study these issues, we designed and built a sequential decision-making game where human players manage a virtual kitchen, inspired by the popular game *Overcooked*. Our game is based on the discrete-time job shop scheduling problem, where tasks need to be scheduled to virtual workers; each task consists of subtasks with dependencies (e.g., ingredients must be chopped before cooking) and workers have heterogeneous processing times (e.g., a chef is better at cooking, a server is better at plating). Players must assign subtasks to virtual workers in a way that minimizes the time it takes to complete a set of food orders. Our game is deterministic, making it easy for inexperienced players to learn the optimal strategy from a few interactions. Instead, the difficulty in achieving good performance comes from the game's combinatorial state space, encoding worker availability and subtask completion so far. For instance, they must make forward-looking trade-offs, e.g., deciding whether to greedily assign a worker to a subtask that they are slow to complete, or to leave them idle in anticipation of a more suitable subtask.

Our game captures challenges in a variety of operations problems encountered in the real world. For instance, when assigning tasks to health workers, there can be substitution when patient traffic is high, such as having a nurse practitioner perform tasks usually done by physicians. Another example is delivery workers on a grocery delivery platform choosing which orders to accept, where the worker must account for dependencies (e.g., orders must be picked up before delivery) as well as heterogeneous service times (e.g., bikers have an advantage over drivers in high-traffic locations). More broadly, our game can be viewed as a stylized model of any manager scheduling employees to perform tasks on a daily basis, a gig economy employee scheduling their daily workload, or a project manager assigning subtasks to workers to accomplish a longer-term goal. While these examples typically involve more complex challenges such as stochastic demands, we believe our experimental findings on worker learning and compliance can generalize well to these settings.

Experiment. Our primary contribution is a large-scale randomized controlled experiment in the context of this game; Figure 1 illustrates the high-level setup and flow of the game and Section 3 provides a more detailed description. In particular, we perform a large-scale behavioral study on Amazon Mechanical Turk based on two different configurations of our virtual kitchen environment. In the *normal* configuration, the participant plays three identical instantiations of the environment. In the *disrupted* configuration, the first two instantiations of the environment are identical to the

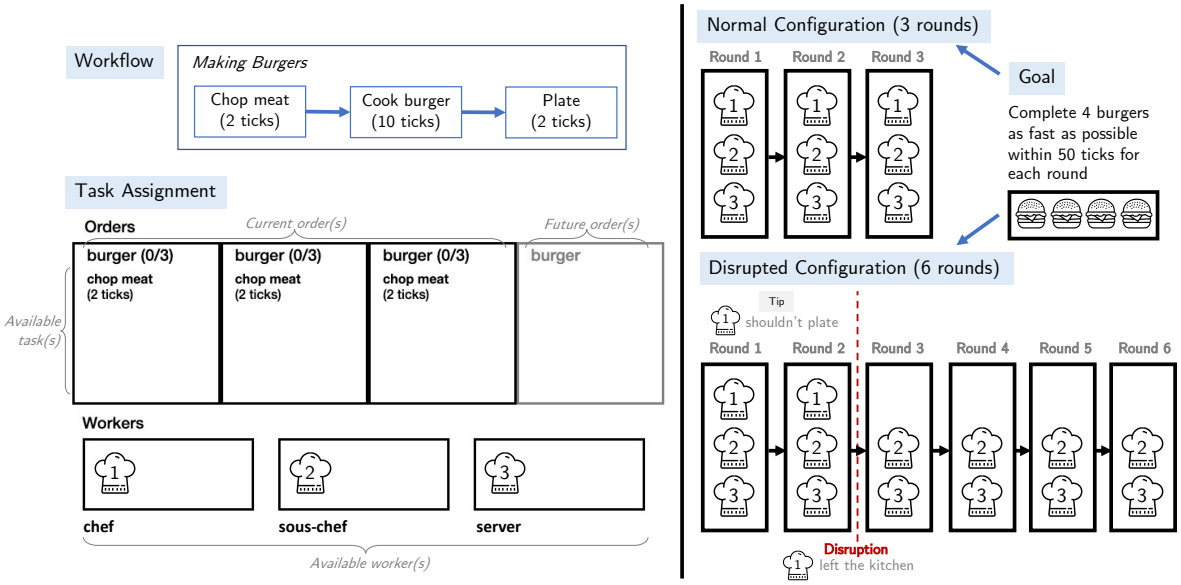


Figure 1 Overview of kitchen management game. The left panel depicts what participants see: (i) the workflow required to complete a burger order, and (ii) the game screen that allows available tasks to be dragged and dropped to one of 3 virtual workers. The right panel depicts the study design: in the normal configuration, participants play the same game for 3 rounds; in the disrupted configuration, participants play the same game for 2 rounds, face a disruption in the kitchen (i.e., the chef leaves), and play the disrupted game for 4 rounds.

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 the right panel of Figure 1. 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 results demonstrate that our algorithm can generate valuable insights that enable human participants to substantially improve their performance compared to counterparts that are not shown the tip or that are shown alternative tips derived from natural baselines. Importantly, we observe that participants do not naively adjust their policy 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. Overall, our findings suggest that reinforcement learning can effectively leverage trace data to infer interpretable and useful insights, and furthermore, can successfully convey these insights to humans to improve their decision-making.

1.1. Related Literature

Identifying performance improvements for human workers. Process improvement has long been a focal point in operations management; scholars have especially identified various difficulties associated with sequential decision-making and learning. Thus, we study process improvement from the perspective of individual workers through sequential decision-making. When workers first experience a new work environment, they may have difficulty adjusting, resulting in various degrees of undesirable performance (Ramdas et al. 2017), e.g., unexpected critical medical incidents slow down ambulance activation among paramedics (Bavafa and Jónasson 2021). The situation is exacerbated when inexperienced workers lack guidelines on how to manage their workflow, resulting in sub-optimal task prioritization and poor productivity (Ibanez et al. 2018). Complexity of workflows also plays a role. Workers tend to focus on immediate challenges and ignore opportunities for learning (Tucker et al. 2002); furthermore, switching between tasks can significantly hurt productivity (Gurvich et al. 2020). Depending on the features of the sequential decision-making problem, workers may generally follow non-optimal policies (Kagan et al. 2021).

A common approach to increase reliability and reduce process variation is to standardize processes and offer best practices (Nonaka and Takeuchi 1995, Pfeffer et al. 2000, Spear 2005). However, creating standards can be challenging (Szulanski 1996, Argote 2012) and time-consuming (Nonaka and Takeuchi 1995). Workers can learn by trial and error (Dorn and Guzdial 2010), but past experience sometimes makes it challenging to identify best practices (Huckman and Pisano 2006, Kc and Staats 2012). Workers can also learn through soliciting peer feedback (Herkenhoff et al. 2018, Jarosch et al. 2019, Song et al. 2017, Brattland et al. 2018) or working alongside experienced peers (Chan et al. 2014, Tan and Netessine 2019); these mechanisms are especially salient when there is familiarity and collaborative experience between workers (Akşin et al. 2021, Kim et al. 2020). However, these ingredients are often not available. 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 logged trace data of historical decisions and outcomes. While recent work has leveraged trace data and machine learning to predict when humans make mistakes in decision-making (Fudenberg and Liang 2019, McIlroy-Young et al. 2020, Fudenberg et al. 2022), they do not offer tips to improve human performance.

Using machine learning to improve one-shot decision-making. As noted earlier, several recent papers have studied whether machine learning can improve human decision-making in the one-shot setting. Key challenges that arise are that humans often erroneously assess their own abilities (Fügener et al. 2022) as well as the predictive model's abilities (Chandrasekaran et al. 2017, 2018, Green and Chen 2019); this in turn can result in unwarranted algorithm aversion (Dietvorst et al. 2015) or algorithm appreciation (Logg et al. 2019). This can be overcome by mechanisms such as

enabling the predictive model to delegate tasks to humans in a user-aware manner (Fügener et al. 2022), training workers on the success/failures of their specific predictive model (Chandrasekaran et al. 2018), capturing the uncertainty of the model's predictions (Kneusel and Mozer 2017), or accounting for systematic human deviations from the model (Sun et al. 2022). Another important lever is improving the interpretability/explainability of the predictive model (Stites et al. 2021, Lu et al. 2019), which allows workers to gain a deeper understanding of the environment and the potential improvement to be obtained (Sull and Eisenhardt 2015, Gleicher 2016). This can be accomplished by using simple model families like decision trees (Breiman et al. 1984, Bertsimas and Dunn 2017) or rule lists (Wang and Rudin 2015, Letham et al. 2015), or by employing post-hoc explanation methods like LIME (Ribeiro et al. 2016).

In contrast to these approaches, we focus on sequential decision-making, which is representative of many real-world workflows and poses qualitatively different challenges. For example, adopting a recommended decision on the current time step affects future states/decisions faced by the worker; as a consequence, compliance with a tip may actually *hurt* performance if the worker is unable to appropriately adjust their future workflow. Algorithmically, it is also more challenging to compute interpretable policies, since the entire sequence of recommended decisions needs to be interpretable. Thus, we propose a novel framework that adapts interpretable reinforcement learning techniques (Puiutta and Veith 2020, Meyer et al. 2014) to compute interpretable tips that bridge the discrepancy between the human's current policy and the optimal policy. We build on a strategy that first trains a high-performance blackbox policy, and then use imitation learning (Ross et al. 2011) to distill this policy into an interpretable one (Verma et al. 2018, Bastani et al. 2018).

1.2. Contributions

Our work contributes to the literature in two ways. First, we propose a novel algorithm for inferring tips for sequential decision-making. Our algorithm leverages techniques from interpretable reinforcement learning to capture the discrepancy between the existing human policy (as captured by trace data) and the optimal policy, thereby identifying the best performance-improving tip targeted towards key bottlenecks in current human decision-making.

Second, to the best of our knowledge, we conduct the first large-scale behavioral experiment on Amazon Mechanical Turk to understand how reinforcement learning based tips can improve human performance in sequential decision-making problem. Unlike one-shot decision-making, in order to be effective, humans must understand not only the meaning of a tip, but also how to operationalize it into a broader workflow. Our experimental results demonstrate that workers are capable of inferring complex strategies from the limited recommendations provided by our algorithm's tips, but this is not always the case with tips inferred through peer feedback or simple descriptive statistics. We also provide a number of additional insights about how workers comply with tips, as well as how they perceive bottlenecks in their own workflows.

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.

We begin by formalizing the tip inference problem. We model our setting as the human acting to maximize reward in a standard, undiscounted Markov Decision Process (MDP) $\mathcal{M} = (S, A, R, P)$ 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). 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 $r_t = R(s_t, a_t)$, 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 $\zeta = ((s_1, a_1, r_1), ..., (s_T, a_T, r_T))$. We measure the cumulative expected reward of a given policy π as

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

where $D^{(\pi)}$ is the distribution of rollouts induced by using policy π. We denote the human policy π_H, which is not directly observed but can be estimated from historical trace data. It will also be useful to define the optimal policy, $\pi^* = \arg\max_\pi J(\pi)$, which maximizes cumulative reward.

Tips: Now, given the MDP $\mathcal{M}$ and the human policy π_H, our goal is to learn a tip ρ that, conditioned on adoption by the human, most improves the cumulative expected reward. Formally, a tip indicates that in certain states s, the human should use action $\rho(s) \in A$ instead of following their own policy π_H. 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., $\psi(s)$ might be an indicator of whether a sufficient quantity of a certain resource is currently available). In other words, a tip $\rho = (\psi, a)$ says that if the logical predicate ψ is true, then the human should use the action a prescribed by the tip; otherwise, they should use their own policy π_H.

If the human follows this tip exactly, then the resulting policy they use is $\pi_H \oplus \rho$, where we define the operation

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

Here, $\mathbb{1}$ is the indicator function; that is, the human takes action a with probability one if $\psi(s)$ holds, and follows their existing policy otherwise.

REMARK 1. In practice, we find that human adoption of tips varies. However, it is difficult to predict the rate of adoption of a tip prior to offering it. Instead, we focus on identifying the best performance-improving tip *conditioned* on adoption. We find that this strategy works sufficiently well to improve performance in our experiments as long as the human can understand both the tip and its rationale. We give a detailed discussion of compliance with tips in Section 5.2.

Our goal is to compute the tip ρ^* that most improves the human's performance—i.e.,

$$\rho^* = \arg\max_{\rho} J(\pi_H \oplus \rho). \tag{2}$$

This formulation ensures that the chosen tip is *consequential* to improving performance J in Eq. (1). There are many other ways to choose tips, e.g., one can naïvely identify state-action pairs that frequently differ between the human and optimal policies. We illustrate the drawbacks to such an approach in our experiments (see Section 5).

Algorithm: Next, we describe our algorithm for solving Eq. (2). Note that we can simply loop through each candidate tip ρ, but we may lack the data to evaluate $J(\pi_H \oplus \rho)$ without additional assumptions. This is because showing the tip changes the human's behavior, changing the distribution of states $D^{(\pi)}$ they visit to $D^{(\pi \oplus \rho)}$. However, we do not have samples from $D^{(\pi \oplus \rho)}$, which are necessary to estimate Eq. (1). One strategy would be to run an experiment with each tip to obtain these samples, but this is prohibitively expensive. Alternatively, one can consider approximating the unobserved distribution $D^{(\pi \oplus \rho)}$ with the observed distribution $D^{(\pi)}$ when evaluating $J(\pi_H \oplus \rho)$, but this has the unfortunate consequence of removing the dependence on the tip ρ entirely from our optimization problem in Eq. (2), rendering us unable to identify good tips.

Instead, we describe an approximation that is implementable given observed data, and effectively distinguishes between candidate tips; we find that this strategy works well in our experiments. To this end, we leverage the well-studied value- and Q-functions (Watkins and Dayan 1992) (denoted V^* and Q^*, respectively), which can be defined recursively by the Bellman equation:

$$V^*(s) = \max_{a \in A} Q^*(s, a),$$
$$Q^*(s, a) = R(s, a) + \mathbb{E}_{s' \sim p(\cdot|s,a)}[V^*(s')].$$

Intuitively, $V^*(s)$ is the cumulative expected reward accrued from state s when using the optimal policy, and $Q^*(s,a)$ is the cumulative expected reward accrued from s by first taking action a and then using the optimal policy. We can compute both V^* and Q^* using Q-learning (Watkins and Dayan 1992). Now, we can rewrite the objective $J(\pi_H \oplus \rho)$ in Eq. (2) as follows:

LEMMA 1 **(Lemma 2.2, Bastani et al. 2018)**. *For any policy π, we have*

$$J(\pi^*) - J(\pi) = \mathbb{E}_{\zeta \sim D^{(\pi)}}\left[\sum_{t=1}^{T} V_t^*(s_t) - Q_t^*(s_t, \pi(s_t))\right].$$

Applying this lemma to both π_H and $\pi_H \oplus \rho$, and taking the difference, we obtain

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

Letting $\bar{D}_t^{(\pi)}$ be the marginal distribution of s_t in the distribution $D^{(\pi)}$ over rollouts, then

$$J(\pi_H \oplus \rho) - J(\pi_H) = \sum_{t=1}^{T} \mathbb{E}_{s_t \sim \bar{D}_t^{(\pi_H)}}\left[V_t^*(s_t) - Q_t^*(s_t, \pi_H(s_t))\right] - \mathbb{E}_{s_t \sim \bar{D}_t^{(\pi_H \oplus \rho)}}\left[V_t^*(s_t) - Q_t^*(s_t, \pi_H \oplus \rho(s_t))\right].$$

Now, assuming that $\bar{D}_t^{(\pi_H)} \approx \bar{D}_t^{(\pi_H \oplus \rho)}$, we have

$$\begin{aligned} J(\pi_H \oplus \rho) - J(\pi_H) &\approx \sum_{t=1}^{T} \mathbb{E}_{s_t \sim \bar{D}_t^{(\pi_H)}}\left[V_t^*(s_t) - Q_t^*(s_t, \pi_H(s_t))\right] - \mathbb{E}_{\zeta \sim \bar{D}_t^{(\pi_H)}}\left[V_t^*(s_t) - Q_t^*(s_t, \pi_H \oplus \rho(s_t))\right] \\ &= \mathbb{E}_{\zeta \sim D^{(\pi_H)}}\left[\sum_{t=1}^{T} Q_t^*(s_t, \pi_H \oplus \rho(s_t)) - Q_t^*(s_t, \pi_H(s_t))\right]. \end{aligned} \tag{3}$$

Intuitively, this assumption says that the *indirect* effect on performance due to the shift in the state distribution induced by the tip (i.e., from $\bar{D}_t^{(\pi_H)}$ to $\bar{D}_t^{(\pi_H \oplus \rho)}$) is small; instead, the main effect is due to the *direct* effect on performance due to the change in the current human action induced by the tip, which is captured by Eq. (3). In practice, we do not observe that the state distributions shift substantially, suggesting that this is a good approximation.

Next, we approximate the expectation in our objective using observed rollouts (i.e., historical trace data) $\zeta_1, ..., \zeta_k \sim D^{(\pi_H)}$ from the human policy π_H. Thus, our algorithm computes the tip

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

Here, we have dropped the terms $J(\pi_H)$ and $\mathbb{E}_{\zeta \sim D^{(\pi_H)}}\left[\sum_{t=1}^{T} Q_t^*(s_t, \pi_H(s_t))\right]$ since they are constant in ρ; for a given tip $\rho = (\psi, a)$ and action a', we have also defined the operation

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

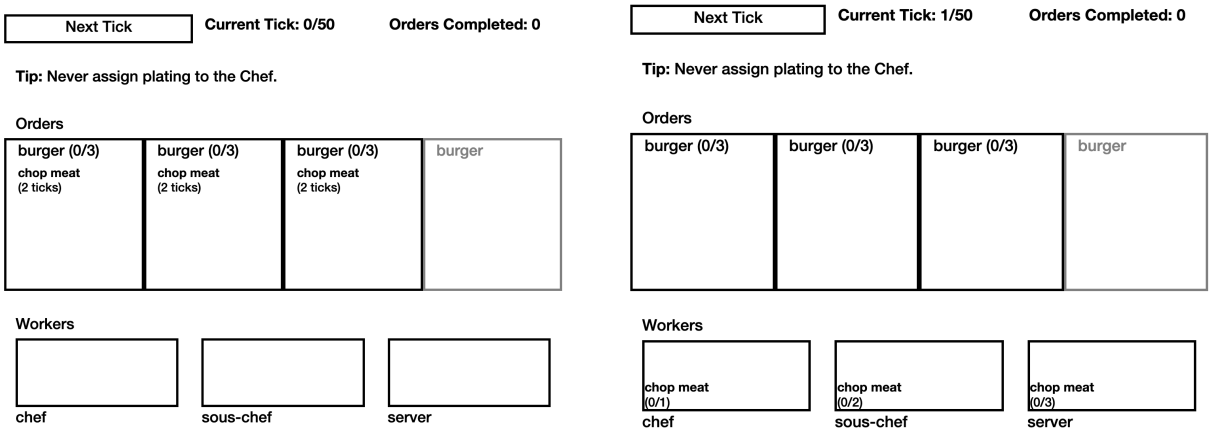
We optimize Eq. (4) by enumerating through candidate tips ρ, evaluating the objective, and selecting the tip $\hat{\rho}$ with the highest objective value.

3. Virtual Kitchen Management Game

Our main empirical question is whether human workers can incorporate tips inferred using our algorithm into their broader decision-making policy. Specifically, our tips only provide partial information about the discrepancy between their policy and the optimal policy; thus, workers must not only comply with our tip (which is the usual challenge in improving human performance at one-shot decision-making problems), but they must implicitly infer additional information about the optimal policy in order to effectively operationalize our tip into their broader workflow. To achieve this goal, our environment was designed with two criteria in mind: (i) it should be possible for humans to compute the optimal policy given sufficient thought, but (ii) the optimal policy should not be obvious. We focused on deterministic environments, where inexperienced workers could reason about the optimal strategy from very few interactions with the environment. While we believe our insights extend to stochastic environments, they intuitively require more experience/interactions for humans to deduce optimal strategies. Finally, we deliberately designed a problem where we can compute the optimal policy (see Appendix A.2 for a description of this policy), which enables us to evaluate human sub-optimality.

In particular, we build on the job shop scheduling problem, where the goal is to schedule jobs to machines in an optimal way, and where there are dependencies between different jobs. To ensure the problem is sufficiently challenging, we introduce additional complexity in the form of heterogeneous machines, where the processing time for different types of jobs varies depending on the machine. To make our problem intuitive to human users, inspired by the popular game *Overcooked*, we represented our decision-making problem as a virtual kitchen management game that can be played by individual human players (see Figure 1). In this game, the player takes the role of a manager of several virtual workers (the "machines")—namely, chef, sous-chef, and server—serving burgers in a virtual kitchen. Each burger consists of a fixed set of subtasks (the "jobs") that must be completed in order—namely, chopping meat, cooking the burger, and plating the burger. The game consists of discrete time steps; on each time step, the player must decide which (if any) subtask to assign to each idle worker. The worker then completes the subtask across a fixed number of subsequent time steps, and then becomes idle again. A burger is completed once all its subtasks are completed, and the player completes the game once four burger orders are completed. The player's goal is to complete the game in as few time steps as possible.

There are two key aspects of the game that make it challenging. First, the subtasks have dependencies—i.e., a subtask can only be assigned once previous subtasks of the same order have already been completed. For example, the "plate burger" task can only be assigned once the "cook burger" task is completed. Second, the virtual workers have heterogeneous skills—i.e., different workers take different numbers of steps to complete different subtasks. For example, the chef is



(a) The initial state where players 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.

Figure 2 Example screenshots from the game.

skilled at chopping/cooking but performs poorly at plating, while the server is the opposite, and the sous-chef has average skill on all subtasks; see Table B.1 in Appendix B for details. Ideally, one would match workers to tasks that they are skilled at to reduce completion time. Thus, the player faces the following dilemma. When a worker becomes available but is not skilled at any of the currently available subtasks, then the player must decide between (i) assigning a suboptimal subtask to that worker, potentially creating a bottleneck, or (ii) leaving the worker idle until a more suitable subtask becomes available. For instance, if the server is idle but all available subtasks are "cook burger", then the player must either (i) assign cooking to the unskilled server, thereby slowing down completion of that burger and eliminating the possibility of assigning plating to the server for the near future, or (ii) leave the server idle until a "plate burger" subtask becomes available. Furthermore, players are not shown the number of steps a worker takes to complete a subtask until they assign the subtask to that worker (see Figure 2 and Appendix D for example game screenshots); instead, they must experiment to learn this information.

We consider two scenarios of the game, differing only in terms of worker availability. In the first scenario, the kitchen is *fully-staffed*, where the human player has access to all three virtual workers (chef, sous-chef, and server). In the second scenario, the human player faces a disruption and the kitchen becomes *understaffed*, with only two virtual workers (sous-chef and server). In both scenarios, the goal is to complete four burgers in as few time steps as possible. We describe how this decision-making problem can be formulated as an MDP and the resulting optimal policies in Appendix A. Note that the optimal policy completes four burgers in 20 and 34 time steps for the fully-staffed and understaffed scenarios, respectively.

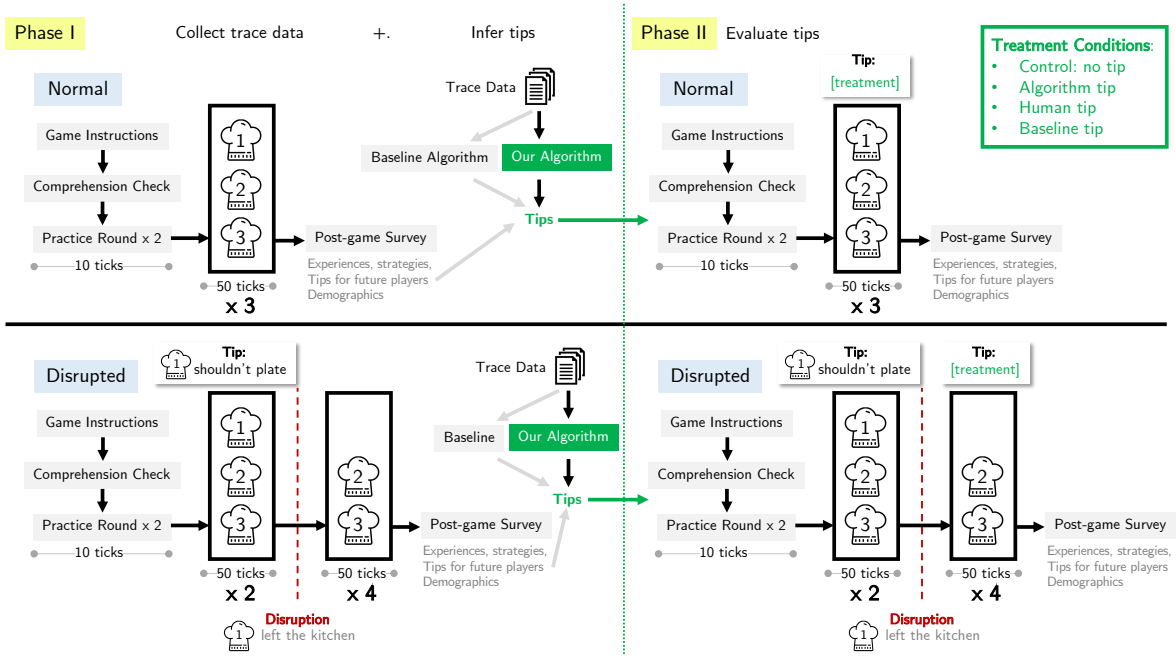


Figure 3 Overview of experimental flow. The top two panels depict Phase I (left) and II (right) for the normal configuration, where each participant plays three fully-staffed scenarios. The bottom two panels depict Phase I (left) and II (right) for the disrupted configuration, where each participant plays two fully-staffed and four understaffed scenarios. Phase II participants are randomly assigned to one of four conditions (control, algorithm, human, and baseline). The set of participants across all four configuration-phase pairs is mutually exclusive.

4. Experimental Design

We investigate how humans interpret and follow the tips inferred by our algorithm in the context of our virtual kitchen management game, using pre-registered behavioral experiments involving Amazon Mechanical Turk (AMT) workers.[1] We describe our experimental design in this section.

4.1. Overview

Figure 3 summarizes our experiment, which proceeds in two phases. In Phase I, we recruit AMT workers to play our game without showing them any tips, and collect trace and survey data on their behavior. This phase enables us to collect historical data that would normally already be available for an existing decision-making task, which we use to infer tips.

Next, Phase II is our actual randomized controlled experiment; in this phase, we again recruit AMT workers to play our game, but this time, we randomize each participant into one of four *advice conditions*, and show them a tip that depends on their advice condition (namely, the tip inferred using our algorithm, two alternative tips, and a control group where they are not shown any tip). We measure the performance of the participants, with the goal of determining whether our approach improves over the three alternatives. We describe the four advice conditions below.

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

In both phases, each participant plays a sequence of three or six *rounds* of our virtual kitchen management game; each round is one instance of our game that is completely independent of the other rounds. The number of rounds is determined by the *game configuration* they are assigned to (normal vs. disrupted), which we described below. By having the participant play multiple rounds instead of a single one, we can study both how performance varies with the tip they are shown, as well as how it evolves across games as participants gain experience.

In summary, Phase I is purely to gather data for computing tips; in this phase, participants are randomly assigned to one of two conditions (game configuration). Then, Phase II is our main experiment, which uses a 2 (game configuration) $\times$ 4 (advice condition) between-subjects design; in this phase, participants are assigned randomly to the eight total conditions (two game configuration conditions times four advice conditions). See additional details on the experimental design (e.g., details on inferred tips, performance-based pay) in Appendix B, participant demographics in Appendix C, and screenshots of our game in Appendix D.

Game configurations. In both phases of our experiment, participants are randomized into one of two game configurations, each of which determines a sequence of rounds of our game:

- Normal configuration: Each participant plays three rounds of the fully-staffed scenario
- Disrupted configuration: Each participant plays two rounds of the fully-staffed scenario, followed by four rounds of the understaffed scenario (i.e., the chef is no longer available), for a total of six rounds.

Intuitively, the normal configuration studies whether tips can help human participants fine-tune their performance. In contrast, the disrupted configuration is designed to show how tips can help participants adapt to novel situations where the optimal strategy substantially changes. The disrupted scenario is the more interesting one, since disruptions often cause workers to struggle to adapt (Ramdas et al. 2017, Bavafa and Jónasson 2021), making tips especially useful.

Advice conditions. In Phase II, participants are randomly assigned not only to a game configuration, but also one of four advice conditions:

- "Control group" condition: Participants are not shown any tips.
- "Our algorithm" condition: Participants are shown the tip inferred by our algorithm.
- "Human" condition: Similar to peer feedback, participants are shown the tip most frequently suggested by Phase I participants after they have completed all rounds of our game.
- "Baseline algorithm" condition: Participants are shown a tip derived by a baseline algorithm that leverages simple descriptive statistics to identify the state-action pair where human participants and the optimal policy most frequently differ.

These advice conditions, described in more detail in Section 4.2, are chosen to illustrate how our algorithmic approach compares to and complements worker learning in practice.

Phase I details: In Phase I, we have $N = 183$ participants for the normal configuration, and $N = 172$ participants for the disrupted configuration.

Phase II details: In Phase II, we have $N = 1,317$ participants for the normal configuration, and $N = 1,011$ participants for the disrupted configuration. In the normal configuration, Phase II participants are shown the tip for their advice condition for the fully-staffed scenario on all rounds. In the disrupted configuration, they are shown the tip designated by their condition for the understaffed scenario (the last 4 rounds). In the first two rounds of the disrupted configuration, our goal is to quickly acclimate participants to the fully-staffed scenario in a way that is consistent across conditions. Thus, we show our algorithm tip for the fully-staffed scenario—"Chef should never plate"—across all conditions (including control) for the first two rounds; we choose this tip because, as we show in Section 5, it most quickly improves human performance in the fully-staffed scenario. After the disruption, we inform participants that the optimal strategy has now changed due to the chef's departure.

Participant recruitment and pay. We recruited participants on the Amazon Mechanical Turk (AMT) platform. Each participant can only participate once across both phases and all conditions—i.e., no participant has prior experience with any version of the game. Participants are compensated a flat rate for completing the study, plus a relatively large performance-based bonus determined by how quickly they complete each round of the game (see Appendix B.4 for details).

Hypotheses. Our main outcomes of interest are the average performance in the final round of the game (i.e., the average number of time steps taken by participants to complete all orders in the final round they play), as well as the fraction of participants who ultimately learn the optimal policy. The final round is the the fourth round of the normal configuration and the sixth round of the disrupted configuration. Then, our main hypothesis is that for each of the two game configurations, participants in the "our algorithm" advice condition (i.e., shown the tip inferred using our algorithm) outperform participants in the other three advice conditions. In addition to our main hypothesis, we also examine participant behaviors in response to different tips, particularly their compliance, and how they learn to improve their decision-making beyond the provided tips.

4.2. Advice Conditions

Control group. The "control group" condition represents settings where best practices are not readily available, so workers must learn over time based on their own experience; indeed, we observe that performance improves over time without any tips in this condition.

Our algorithm. The "our algorithm" condition represents our approach. In particular, we use the tip $\hat{\rho}$ inferred using our algorithm (Eq. (4)) based on the trace data obtained in Phase I. Additional details are provided in Appendix A.

Human. The "human" condition represents settings where one can obtain advice on best practices from more experienced peers (e.g., as in Song et al. 2017). We use Phase I to do so. In particular, each participant in Phase I is shown a comprehensive list of candidate tips at after completing all rounds of our game, and is asked to select the tip they believe would most improve the performance of future players. This list is constructed by merging three types of tips:

1. all possible tips of the format described in Appendix A.3 (e.g., "Chef should not plate"),
2. a small number of generic player tips that arose frequently in our exploratory pilot studies (e.g., "Keep everyone busy at all times"), and
3. 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").

Our algorithm's tip is always contained in this list, as part of the first category above. This list contained 13–14 tips (depending on the configuration), which we found to be a reasonable length that did not overwhelm participants in our pilot studies. We take the most frequently chosen tip as the "human tip", capturing the wisdom of the (experienced) crowd. We also considered several variations, such as taking the tip recommended by the best performing human participants, but these variations all resulted in the same tip; see Appendix C.4 for details.

The human tip is designed to demonstrate how our algorithmic approach can exceed the capabilities of humans to offer useful advice, capturing the limitations of relying on peers for advice.

Baseline algorithm. The "baseline algorithm" condition illustrates a naïve use of descriptive statistics on historical trace data to provide tips—simply looking for frequent differences between the human and optimal policies, rather than leveraging interpretable reinforcement learning to identify the most consequential actions for improving performance. In particular, given rollouts $\zeta_1^*, ..., \zeta_h^* \sim D^{(\pi^*)}$ sampled using the optimal policy, we let $C^*(s,a)$ denote the number of times state-action pair (s,a) occurs across these rollouts. Then, given the observed rollouts (i.e., historical trace data from human decision-making) $\zeta_1, ..., \zeta_k \sim D^{(\pi_H)}$, the baseline algorithm selects the tip

$$\hat{\rho}_{\text{bl}} = \arg\max_{\rho} \frac{1}{k} \sum_{i=1}^{k} \sum_{t=1}^{T} C^*(s_{i,t}, a_{i,t}). \tag{5}$$

In other words, our baseline optimizes the same objective but with Q^* replaced with C^*. Intuitively, this baseline strategy tries to directly imitate the optimal policy, whereas our strategy prioritizes state-action pairs that are more relevant to achieving high rewards. In this condition, we show participants the tip $\hat{\rho}_{\text{bl}}$ inferred by the baseline algorithm (Eq. (5)) based on the Phase I data.

This baseline algorithm ignores the sequential nature of our decision-making problem. It is designed to highlight the complexity of sequential structure compared to the one-shot decision-making setting studied in prior work, and in particular, the importance of accounting for this sequential structure when inferring tips.

5. Experimental Results

Despite their simplicity and conciseness, we find that our tips can significantly improve participant performance since they capture strategies that are hard for participants to learn; in contrast, alternative tips have varying empirical shortcomings that reduce their effectiveness. We also describe how participant compliance—a key ingredient to ultimately improving decisions—varies across tips. Finally, we find evidence that participants do not blindly follow our tips, but combine them with their own experience to discover additional strategies beyond our tips. Figure 4a shows the tips inferred in each condition for each configuration using trace and survey data from Phase I.

5.1. Performance: *Our Tips Substantially Improve Performance*

Figure 4 shows performance results across all four conditions and both configurations. Figure 4b & 4c show participant performance in the final round of our game, Figure 4d & 4e show how performance improves across rounds, and Figure 4f & 4g show the fraction of participants achieving optimal performance across rounds. For each configuration, we report performance as the excess ticks (time) taken over the optimal policy, normalized by the optimal policy's ticks, i.e.,

$$\frac{\#\text{ ticks taken} - \text{optimal }\#\text{ ticks}}{\text{optimal }\#\text{ ticks}}.$$

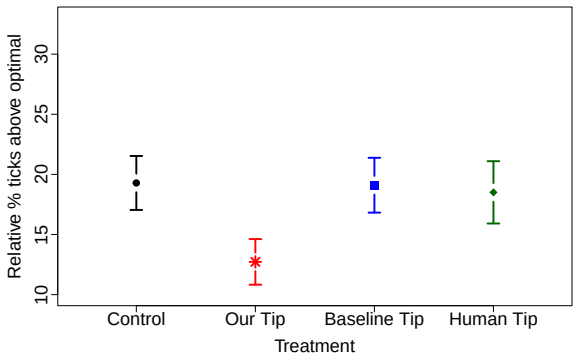
Results in terms of the raw number of ticks are shown in Figure C.1 in Appendix C.2.

The normal configuration is relatively easy for participants—a substantial fraction (24%) discover the optimal policy by the final round without the aid of tips (control group). As shown in Figure 4b, participants shown our tip completed the final round in 22.5 steps on average, significantly outperforming participants in the control group ($t(329) = -4.397,\ p < 10^{-4}$), those shown the human-suggested tip ($t(312) = -3.628,\ p = 2 \times 10^{-4}$), and those shown the tip from the baseline algorithm ($t(334) = -4.232,\ p < 10^{-4}$).[2] Our tip speeds up learning by at least one round compared to the other conditions—i.e., the performance of participants given our tip on round k was similar to or better than the performance of participants in other conditions on round $k+1$ (Figure 4d). Our tip also helped more participants (35%) achieve optimal performance (20 steps) in the final round, compared to 24-29% in other conditions.

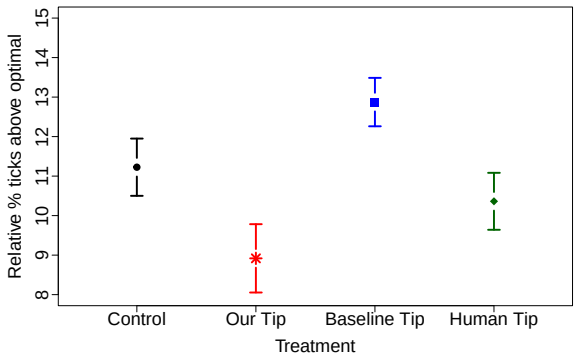
The disrupted configuration is substantially harder, since participants must adapt to the more counter-intuitive understaffed scenario. Perhaps as a consequence, participants benefit much more from tips: those in the control group took four rounds to achieve the same level of performance as those shown our tip on the first round. Participants shown our tip completed the final round in 37.1 steps, again significantly outperforming participants in the control group ($t(243) = -4.361,\ p < 10^{-4}$), those shown the human-suggested tip ($t(246) = -2.52,\ p = 6 \times 10^{-3}$), and those shown the

[2] Results remain highly statistically significant under a Bonferroni correction for multiple hypothesis testing.

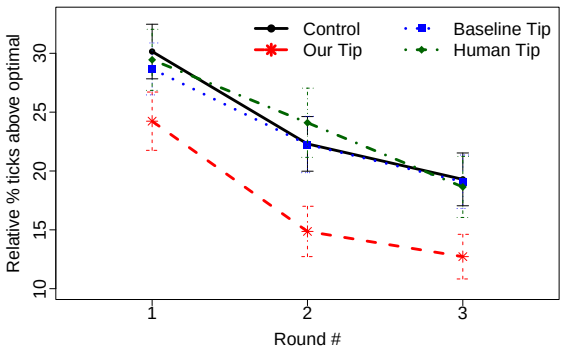
(a) Tips for each condition and configuration



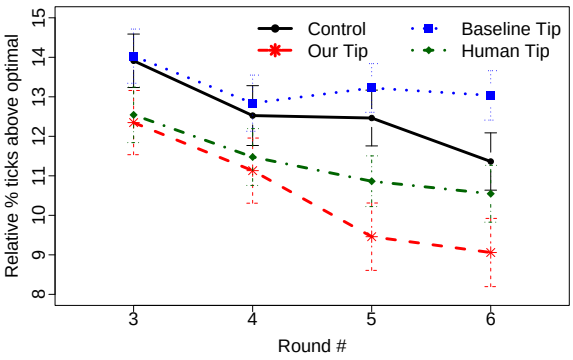
(b) Final Round Performance (Normal)



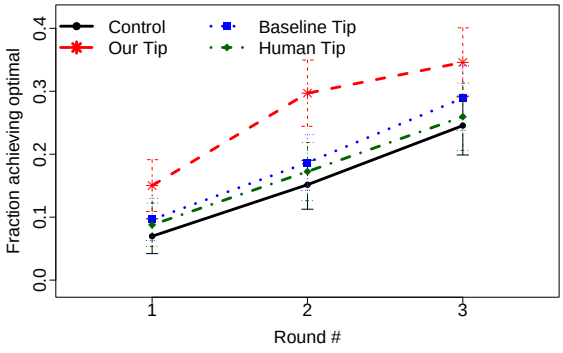
(c) Final Round Performance (Disrupted)



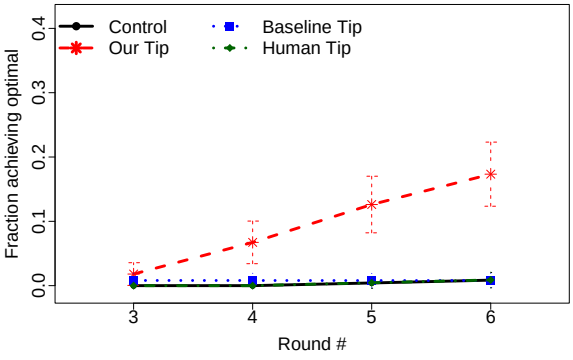
(d) Performance over Time (Normal)



(e) Performance over Time (Disrupted)



(f) Fraction Achieving Optimal (Normal)



(g) Fraction Achieving Optimal (Disrupted)

Figure 4 Phase II Participant Performance. The top row shows the tips derived for each condition and configuration based on Phase I data. Remaining rows depict various views of participant performance across conditions in the normal (left) and disrupted (right) configurations. The top row shows performance in the last round of the configuration, the second row shows how participant performance improves over time, and the third row shows the fraction of participants who execute an optimal policy over time.

tip from the baseline algorithm ($t(246) = -7.348,\ p < 10^{-4}$). In the disrupted configuration, the baseline tip actually *reduces* participant performance, likely because participants struggle to operationalize it. More starkly, 19% of participants shown our tip achieved optimal performance (34 steps) in the final round, compared to less than 1% in all other conditions—i.e., our tip uniquely helps participants learn to play optimally. Note that there were no significant differences in performance across conditions when playing the two fully-staffed rounds in the disrupted configuration. Therefore, the relatively worse performance under other conditions reflect the informativeness of alternative tips.

Shortcomings of baseline tips. As noted earlier, this tip tries to mimic the optimal policy rather than focusing on consequential actions; thus, we expect these tips to be less valuable to participants (for improving performance) than our algorithm's tips. Participants complied with both the baseline algorithm's tips and our algorithm's tips at similar rates (see Section 5.2).

However, the baseline algorithm's tip is still derived from the optimal policy, so it is surprising that it performs *worse* than the control condition in the disrupted configuration. In fact, in Section 5.3, we show that participants who received our algorithm's tips also learned the strategy encoded in the baseline algorithm's tip; however, participants who received the baseline algorithm's tip did not learn the strategy encoded in our algorithm's tip in both configurations. Thus, the problem is not with the *content* of the baseline algorithm's tip, but rather that participants struggle to *operationalize* the baseline tip into their workflow (without knowing our algorithm's tip).

In particular, when participants apply a tip, they shift to new unseen portions of the state space, and must also learn to act well in those states. By focusing on "high-value" states and critical performance bottlenecks, our algorithm more easily enables participants' off-distribution learning. For example, in the disrupted configuration, the baseline algorithm's tip "Sous-chef should plate twice" suggests actions that occur late in the game (hindering participants' ability to explore and alter their strategy) and does not focus on the critical performance bottleneck (cooking). In contrast, our algorithm's tip "Server should cook twice" frees the sous-chef to plate later in the game (a strategy—not explicitly conveyed in our tip—that participants automatically learn when given our tip). However, targeting early decisions alone is not sufficient to help participants learn. In the normal configuration, although the baseline algorithm's tip targets an *earlier* action ("Chef should chop once") compared to our algorithm's tip ("Chef shouldn't plate"), it fails to help participants learn the entire optimal strategy (see Section 5.3) because it does not address the important bottleneck (keeping the chef from the lengthy task of plating).

Shortcoming of human tips. While the human-suggested tips consistently improve performance compared to the control group, they can be overly general or incorrect. In the normal configuration, Phase I participants did not translate their strategy into a specific tip—i.e., their suggested tip

"Strategically leave some workers idle" captures a strategy needed to perform better but fails to convey any necessary details to identify the optimal strategy. Alternatively, in the disrupted configuration, Phase I participants provided an *incorrect* tip, suggesting "Server should cook once", whereas the optimal policy actually assigns the server to cook twice (as suggested by our tip)—i.e., participants identified the correct direction of change in response to the under-staffing disruption, but at an insufficient magnitude. The tips chosen by participants are remarkably consistent across different participant subgroups—e.g., top performers from Phase I vs. all participants (see Appendix C.5)—and generally fail to capture counter-intuitive properties of the optimal policy. Perhaps due to their more intuitive nature, participants are substantially *more* likely to comply with the human tip than with our algorithm's tip (see Section 5.2). Thus, our results suggest that the worse performance of the human tip is due to the sub-optimal quality of the chosen tip.[3]

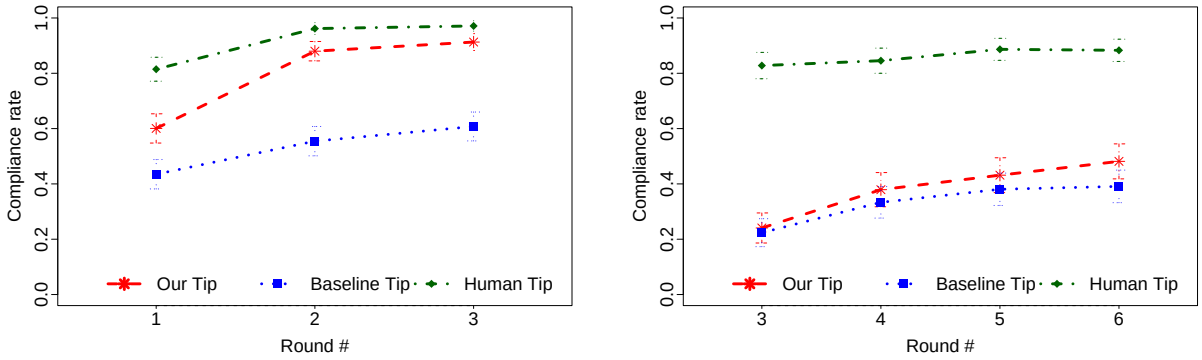
5.2. Compliance: *Humans Comply with Tips More over Time*

As discussed earlier, the effectiveness of a tip critically depends on whether humans are able to understand it and implement it effectively. This involves both complying with the tip's suggested actions as well as modifying other portions of their strategy to make full use of the tip. First, we examine compliance with the tips. Note that participants were not informed of the source of the tip (i.e., algorithm or human), so any variation in compliance is due to the content of the tip, rather than behavioral reactions to its source (e.g., algorithmic aversion, see Dietvorst et al. 2015).

Figure 5 shows the fraction of participants that complied with the tip they were offered in each condition. Specifically, we measure the fraction of participants that act in a way that is consistent with the tip they are shown.[4] We see that participants increasingly comply with the tips shown over time—as they gain experience, and better understand the significance of the tip—in all conditions. Compliance with the baseline algorithm's tip was relatively low in both configurations, suggesting that participants did not find it as useful. Alternatively, compliance with the human-suggested tip was higher than compliance with our algorithm's tip, particularly in the disrupted configuration. Based on participants' post-game feedback, we found that this is likely because the human-suggested tip better matches human intuition (since it is devised by humans). The disrupted configuration is illustrative. Although our algorithm's tip is correct (unlike the human-suggested tip), it is highly counter-intuitive, hurting adoption. For example, in the disrupted scenario, our tip

[3] Note that human participants have a slightly different tip search space than our algorithm. However, this discrepancy cannot be the source of the performance difference, since in the disrupted configuration, both our algorithm's tip and the human tip are present in both search spaces; participants then chose an incorrect tip.

[4] For the human tip in the normal configuration ("Strategically leave some worker(s) idle"), we measured compliance by identifying if the participant ever skipped a potential task assignment when at least one virtual kitchen worker was idle and there was at least one available subtask. Note that we cannot perfectly certain if such "skipping" was strategic, but given that participants were financially incentivized to complete each round as fast as possible, we expect that participants would not skip an assignment unless they were being strategic.



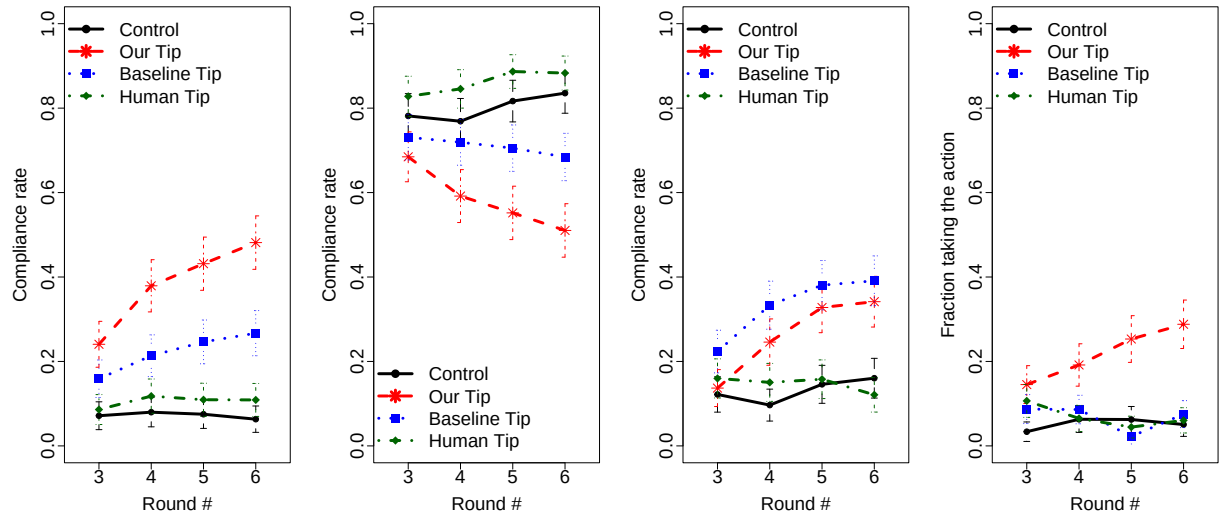
(a) Normal Configuration (b) Disrupted Configuration

Figure 5 Compliance with Tips. Participant compliance in Phase II with the respective tip they were shown in each condition for the normal (left) and disrupted (right) configurations over time.

"Server should cook twice" may appear unreasonable since the server is very slow at cooking; in fact, participants *just* learned to never assign the server cooking in the fully-staffed scenario prior to the disruption. Yet, having the server cook twice is the only way to achieve optimal performance in the understaffed scenario; in contrast, the human-suggested tip is to only have the server cook once, which is a less sharp departure from the previously employed policy. As participants gain experience with the new understaffed scenario, they grow to appreciate the value of our algorithm's tip (i.e., compliance with our algorithm's tip more than doubles over the four rounds).

Our results suggest that participants do not blindly follow tips; instead, they only follow them if they believe that the suggested strategy is effective. Qualitative evidence from post-game survey responses suggested that participants ignored tips they did not agree with (see Section 5.5). Thus, compliance is a function not just of the interpretability of the tip (which is unchanged across conditions), but also the strategy it encodes. When the optimal strategy is counter-intuitive, we observe an intrinsic trade-off between the optimality of the tip and compliance with the tip. Even in the disrupted configuration, our algorithm's tip succeeds despite much lower compliance (relative to the human-suggested tip) since it suggests a highly effective strategy; as seen in Figure 4g, participants that understand this strategy can achieve optimal performance (whereas essentially none of the participants in the other conditions were able to do so). Interestingly, as we show in the next subsection, participants in the *control group* also comply with the human-suggested tip at a high rate—i.e., the human-suggested tip largely captures behaviors that are likely to be adopted even in the absence of tips; in contrast, our algorithm's tip allows participants to learn new strategies that they may not learn otherwise.

One may be able to improve compliance with algorithmic tips through additional levers. For instance, financial incentives (Giuffrida and Torgerson 1997) and providing social nudges by setting norms (Benjamin et al. 2010) can improve human compliance with recommendations or guidelines;



(a) Our Tip: *"Server Cooks Twice"* (b) Human Tip: *"Server Cooks Once"* (c) Baseline Tip: *"Sous-chef Plates Twice"* (d) Unshown Tip: *"Server Chops Once"*

Figure 6 Learning beyond Tips. Panels (a)-(c) show the rate at which participants in each condition cross-comply with each offered tip over time in the disrupted configuration. Panel (d) shows analogous results for a rule that is part of the optimal policy but was not shown as a tip in any condition.

however, it is important to design these approaches in a context-specific way to avoid null or even negative effects (see, e.g., Beshears et al. 2015, Chang et al. 2021). We perform a follow-up experiment testing these levers to improve compliance with our algorithm's tip (see Section 5.4).

5.3. Learning Beyond Tips: *Our Tips Help Humans Learn to Perform Optimally*

One of the critical challenges in sequential decision-making is that the human must learn strategies *beyond* the provided tip to achieve good performance throughout their workflow, since the tip only captures a portion of the optimal policy. To study whether humans learn the optimal policy, we examine what kinds of strategies they learn beyond the specific tips they were shown. More precisely, we study *cross-compliance*, which is the compliance of the participant to tips other than the one they were shown. Naïvely, there is no reason to expect participants to cross-comply with a tip that we did not show them beyond the cross-compliance exhibited by the control group. Thus, any cross-compliance beyond that of the control group measures how a tip enables participants to learn strategies outside of the stated tip. Assuming these strategies are consistent with the optimal policy, cross-compliance serves as a way to measure participants' progress towards operationalizing the tip effectively throughout their broader workflow.

We focus on the disrupted configuration since it is more challenging for participants, leading to more interesting cross-compliance patterns.[5] Figure 6 shows the cross-compliance of participants

[5] In the easier normal configuration, participants in all conditions cross-comply with all other tips (which are all part of the optimal policy), but they achieve higher cross-compliance when shown our algorithm's tip; see Appendix C.3.

in each condition with the different tips (algorithm, baseline, human), as well as a new tip ("Server chops once") not shown to any participants. This new tip is part of the optimal policy for the understaffed scenario used in the disrupted configuration. Participants in the human and control groups only comply with the human tip. Indeed, the human-suggested tip actually contradicts the optimal policy; thus, despite effectively operationalizing the tip, participants are prevented from learning the other tips which are part of the optimal policy.[6] Participants shown the baseline tip only have high compliance with the baseline tip, indicating that the baseline tip could not help participants uncover the rest of the optimal policy; although the baseline tip is part of the optimal policy, it fails to help participants discover strategies beyond the tip itself, since it does not focus on high-value states and critical bottlenecks (see our earlier discussion in Section 5.1). In contrast, participants who received our algorithm's tip have high cross-compliance with *all* parts of the optimal policy (i.e., the baseline tip and the unshown tip); furthermore, our algorithm is the only condition where cross-compliance with the sub-optimal human tip decreases over time. That is, our tip uniquely enables participants to combine the tip with their own experience to discover useful strategies (that are consistent with the optimal policy) beyond what is stated in the tip.

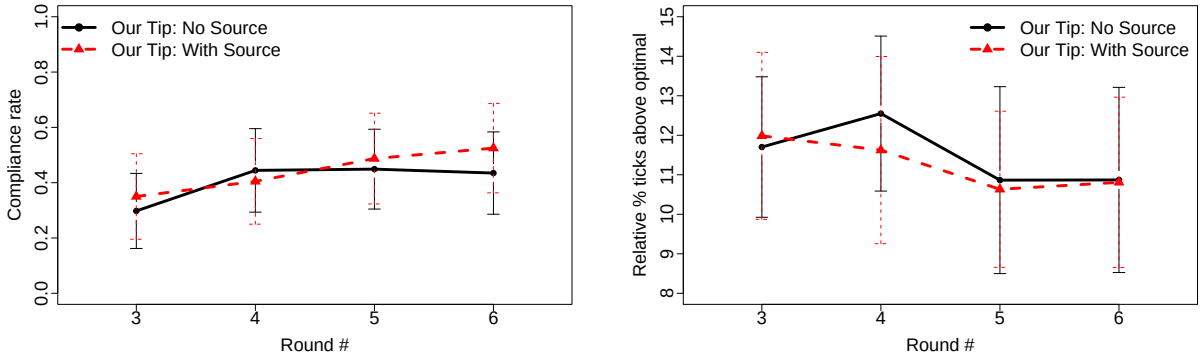
5.4. Compliance: *Algorithm Aversion and Interventions to Improve Compliance*

Compliance in sequential decision-making depends on many factors. Thus far, we have found evidence that intuitive tips can improve compliance, while difficulty operationalizing tips into one's broader workflow can hurt compliance. In practice, additional levers such as trust and incentives may also play a significant role in compliance. We conduct two follow-up studies to examine how these levers may affect our results in real-world scenarios.

Trust in Algorithms. While our experiments thus far did not reveal the *source* of the tip (i.e., whether it is generated by an algorithm), workers may be able to infer this information in real-world contexts, potentially resulting in algorithm aversion (Dietvorst et al. 2018)—i.e., where humans are mistrustful of algorithmic advice. To this end, we perform a pilot study in the disrupted configuration to evaluate the impact of algorithm aversion on compliance. We randomly assign participants into one of two conditions: "Our Algorithm: No Source" and "Our Algorithm: With Source". The "Our Algorithm: No Source" condition is identical to the "Our Algorithm" condition in our main study, i.e., the participant is shown "Tip: Server should cook twice" during the understaffed rounds. In contrast, in the "Algorithm: With Source" condition, the participant is instead shown: "Tip from AI Algorithm: Server should cook twice. The AI algorithm analyzes past players' strategies and chooses the best tip that would help improve your performance."

[6] Note that participants in the control and human conditions comply with the human tip at similar rates, i.e., the human tip suggests behaviors that are highly likely to be adopted even in the absence of tips.

We recruited 200 participants via AMT, of which 90 successfully completed the study and passed all comprehension and attention checks. We find that there are no statistically or economically significant differences in the results between the two conditions according to compliance rate (Figure 7a) or final round performance (Figure 7b). Providing the source of the tip in fact has a directionally *positive* impact on compliance, suggesting that it is unlikely that we would observe algorithm aversion. One potential explanation is that, given the complex nature of the task, knowing that the tip came from an algorithm could increase humans' likelihood in adopting the tip similar to the phenomenon of algorithm appreciation documented by Logg et al. (2019). Note that the final round performance is essentially identical whether or not the source was provided.



(a) Compliance Rate with Our Algorithm's Tip

(b) Performance over Time

Figure 7 Source of Tip. Participant compliance with our algorithm's tip ("Server should cook twice") (left) and participant performance (right) whether the information about the source of the tip was provided.

Financial/Social Incentives. Next, we conducted a follow-up full-scale experiment to determine whether we can improve compliance with our algorithm's tip through financial or social interventions.[7] Focusing again on the disrupted configuration, we investigated the following four interventions in the under-staffed rounds of the disrupted configuration:

1. "Pay" condition: For rounds 3 and 4 (first two under-staffed rounds), we pay the participant the maximum pay for each round *if* they successfully complied with the tip (i.e., server cooked twice). The pay scheme returns to the original performance-based one in rounds 5 and 6.
2. "Social" condition: We add the following to the tip for all four under-staffed rounds—"While this tip may appear counter-intuitive, the majority of best players adopted this rule, enabling them to achieve the optimal performance of 34 ticks." Past research has shown that information about social norms can change human behavior, e.g., residents consumed less energy after learning that their neighbors had better energy consumption ratings (Allcott 2011).

[7] This follow-up study is pre-registered at `https://aspredicted.org/blind.php?x=85D_1RB`

3. "Pay-Social" condition: Participants receive both the Pay and Social interventions.
4. "Curriculum" condition: In round 3 (e.g., the first disrupted round), we present the Human tip from the original study ("Server should cook once") instead of our algorithm's tip. Then, in rounds 4 through 6, we present our algorithm's tip ("Server should cook twice"). The rationale of this intervention is to slowly move the participant's strategy from not letting the server cook any burgers to something in between (letting the server cook once) before telling them the more counter-intuitive algorithm tip (letting the server cook twice).

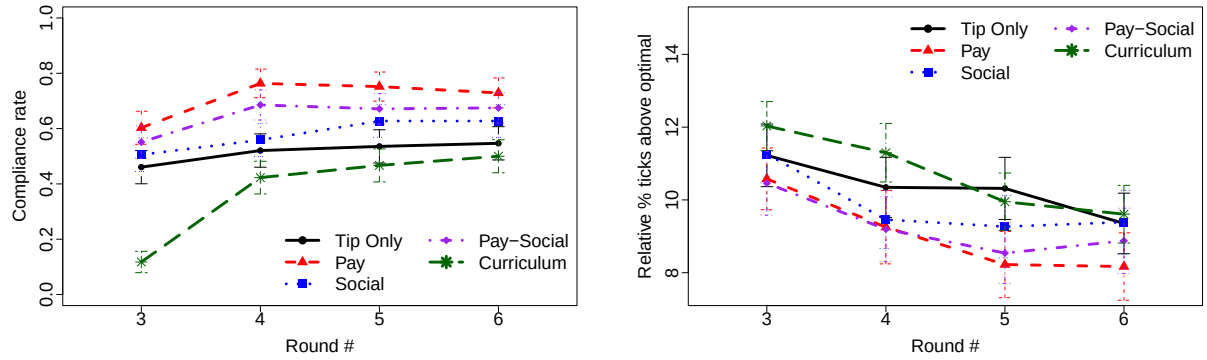
We recruited 1,967 participants via AMT, of which 1,496 successfully completed the study and passed all the comprehension and attention checks. Participants were randomly assigned into one of five conditions: "Tip Only" (identical to the Algorithm arm in the original study), "Pay", "Social", "Pay-Social", and "Curriculum" interventions. The main outcome of interest here is the compliance rate in the final round of the game. Figure 8a shows compliance rates by condition across all four rounds. We find that any combination of "Pay" and "Social" interventions improves compliance with our algorithm's tip. Paying people to follow the tip for two rounds appears to be effective at getting them to try out the seemingly counter-intuitive tip and such compliance sticks around when we no longer compensate merely for their compliance. Both "Pay" and "Pay-Social" interventions significantly improve compliance in round 6 compared to the "Tip Only" condition. Providing social information alone does improve compliance, but the effect is not statistically significant. Finally, slowly easing people toward our algorithm's tip in the "Curriculum" intervention does not seem to improve compliance by the end of the game; providing an intermediate step between human intuition and the optimal action might even backfire as it may slow down the ability of humans to adapt to the new environment. We note that the performances in the final round of the game across all five conditions are not statistically significantly different (see Figure 8b); however, the "Pay" condition appears to have the best performance.

5.5. Participant Comments on The Provided Tips

Lastly, we manually and independently code participants' sentiment towards the tip they received—positive, neutral, or negative—based on their post-game survey responses to the question *"What did you think about the tip for these last [three/four] rounds and how did you incorporate it in your strategy?"*. Figure 9 exhibits the breakdown of responses in each condition and configuration.[8] For robustness, we also performed these sentiment analyses using two natural language processing approaches, VADER and BERTweet, and find qualitatively similar insights (see Appendix C.5).

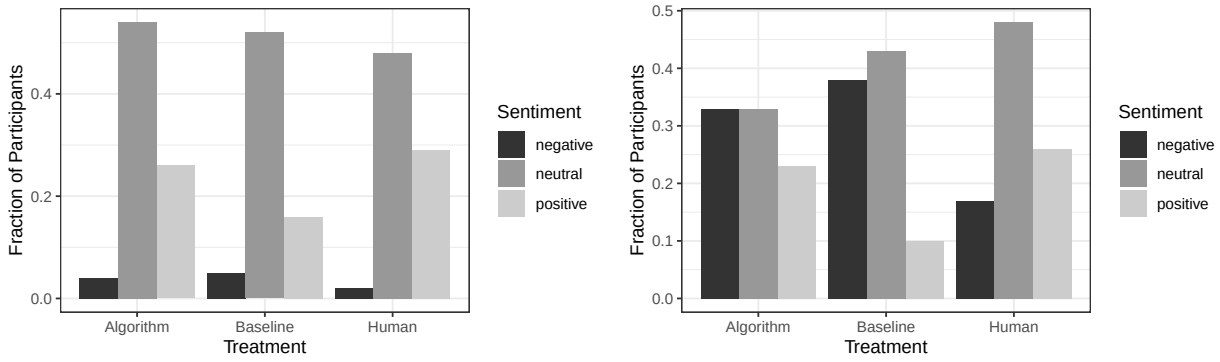
We first observe that, for both configurations, more (fewer) participants in the human condition responded positively (negatively) to the tips compared to the other conditions. In other words,

[8] We excluded unrelated/uninformative participant responses, so fractions per condition may not add to 1.



(a) Compliance Rate with Our Algorithm's Tip (b) Performance over Time

Figure 8 Interventions for Compliance. Participant compliance with our algorithm's tip ("Server should cook twice") (left) and participant performance (right) in each intervention across the four disrupted rounds.



(a) Normal Configuration (b) Disrupted Configuration

Figure 9 Participant sentiment on the provided tips in post-game survey.

human participants selected tips that would likely be accepted by other humans; these tips offer natural strategies that match human intuition, e.g., we observe that they are often adopted even in the control group. On the other hand, the algorithmic and baseline tips are necessarily part of the optimal (rather than human) policy, and can therefore be counter-intuitive; this is especially apparent in the disrupted configuration, where these tips received substantially more negative feedback (and therefore lower compliance rates, as observed in Figure 5b). However, performance and compliance improved over time, implying that it took participants time and effort to correctly incorporate these tips into their workflow and execute an optimal strategy. This is supported by selected excerpts from participant comments presented in Table 1.

Many participants felt that the human tip was more accurate since it better matched their intuition, and disagreed with tips that they found counter-intuitive. Some participants even found the human tip to be counter-intuitive since it did not match their intuition from the fully-staffed scenario in prior rounds (i.e., it asks the server to cook once instead of not at all); this result is matched by a post-game survey question for the normal configuration where participants were asked

Disrupted Configuration	Our Algorithm's Tip *"Server should cook twice"*	Human Tip *"Server should cook once"*
Positive	• "It was very helpful. It made me focus on making sure the server cooked more even if that was not his obvious strength." • "I ignored the tip at first, but later I used the tip and it helped me complete the tasks quickly." • "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." • "I did not listen to it at first because I didn't believe that it would actually help but it did." • "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."	• "It seemed pretty much essential to have server cook once." • "I thought it was smart and I used it exclusively." • "It was accurate, and I implemented the tip." • "I felt that tip was valid, as the server primarily is useful plating/ chopping. I only had him cook once." • "It helped because she could cook one burger but any more than that and your ticks would be too high."
Negative	• "I think it was a bad tip. I couldn't figure out how to incorporate it successfully." • "Seemed counterintuitive." • "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." • "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." • "It was not needed since the server took so much longer to cook."	• "It was not helpful, because it does not specify when the server should cook." • "I used the tip but I don't think it was helpful. The server took long to cook." • "I don't agree with this tip." • "It was not terribly helpful. I tried to incorporate but it did not seem to help" • "It stunk honestly. The server takes forever to cook."

Table 1 Selected excerpts from participant comments on the provided tips (disrupted configuration).

to imagine how their strategy would change in the under-staffed scenario (see Appendix B.5). As a consequence, compliance and performance suffered. Importantly, we observe that even participants who successfully understood our algorithm's tip (and viewed it favorably at the end) claimed that they did not comply with the tip in earlier rounds of the understaffed scenario. Rather, they needed time to experiment with and without the tip in order to learn its value.

Our results suggest that, to achieve high compliance, it is not sufficient for the participant to just understand the action suggested by the tip (a major focus of the literature on interpretable machine learning); they also have to believe that the suggested action will help improve performance, and be given sufficient time to learn how to correctly incorporate the tip into their workflow.

6. Concluding Remarks

We have proposed a novel reinforcement learning algorithm for automatically identifying interpretable tips designed to help improve human sequential decision-making. Our large-scale behavioral study demonstrates that the tips inferred by our algorithm can successfully improve human

performance at challenging sequential decision-making tasks, speeding up learning by up to three rounds of in-game experience. Furthermore, we find evidence that participants combine our tips with their own experience to discover additional strategies beyond those stated in the tip. In other words, our algorithm is capable of identifying concise insights and communicating them to humans in a way that expands and improves their knowledge. To the best of our knowledge, our work is the first to empirically demonstrate that reinforcement learning based tips can be used to improve human sequential decision-making.

An important ingredient in our framework is incorporating trace data to identify succinct 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 to provide personalized offerings to their customers, but the data on their own employees is underused. Trace data is often noisy and too granular to be readable by and immediately useful to 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 experienced or high-performing workers can be used with reinforcement learning to identify good strategies.

Furthermore, we provide a number of insights that can aid the design of human-AI interfaces. First, a significant factor in the performance of a tip is whether humans comply with that tip. Prior work has studied compliance from the perspective of *algorithm aversion* (i.e., whether humans trust other humans more than algorithms) (Eastwood et al. 2012, Dietvorst et al. 2015, 2018), as well as interpretability (i.e., whether the human understands the tip) (Doshi-Velez and Kim 2017, Lage et al. 2018, Rudin 2019). Our results suggest that human compliance additionally depends on whether humans believe (based on their intuition and past experience) that the tip improves performance. Second, it takes time for humans to correctly operationalize and adopt the tip—humans need experience to understand why the tip is correct and to discover complementary strategies that further improve their performance. Thus, there is an opportunity for human-AI interfaces to help humans *gradually* adapt their behavior to improve performance. Third, as evidenced by the baseline tips, even tips that are part of the optimal policy can hurt participant performance if they focus on actions that are not consequential; avoiding such tips is important since it can cause participants to lose trust in machine learning models. We anticipate that human-AI interfaces will become increasingly prevalent as machine learning algorithms are deployed in real-world settings to help humans make consequential decisions, and a better understanding of how to design trustworthy interfaces will be critical to ensuring that these interfaces ultimately improve human sequential decision-making.

References

Akşin Z, Deo S, Jónasson JO, Ramdas K (2021) Learning from many: Partner exposure and team familiarity in fluid teams. *Management Science* 67(2):854–874.

Allcott H (2011) Social norms and energy conservation. *Journal of public Economics* 95(9-10):1082–1095.

Allon G, Cohen MC, Moon K, Sinchaisri WP (2023) Managing multihoming workers in the gig economy. *Available at SSRN 4502968* .

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

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 (2021) Recovering from critical incidents: Evidence from paramedic performance. *Manufacturing & Service Operations Management* 23(4):914–932.

Benjamin DJ, Choi JJ, Strickland AJ (2010) Social identity and preferences. *American Economic Review* 100(4):1913–28.

Bertsimas D, Dunn J (2017) Optimal classification trees. *Machine Learning* 106(7):1039–1082.

Beshears J, Choi JJ, Laibson D, Madrian BC, Milkman KL (2015) The effect of providing peer information on retirement savings decisions. *The Journal of finance* 70(3):1161–1201.

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.

Breiman L, Friedman J, Stone CJ, Olshen RA (1984) *Classification and regression trees* (CRC press).

Buciluǎ 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.

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

Chandrasekaran A, Prabhu V, Yadav D, Chattopadhyay P, Parikh D (2018) Do explanations make vqa models more predictable to a human? *arXiv preprint arXiv:1810.12366* .

Chandrasekaran A, Yadav D, Chattopadhyay P, Prabhu V, Parikh D (2017) It takes two to tango: Towards theory of ai's mind. *arXiv preprint arXiv:1704.00717* .

Chang T, Jacobson M, Shah M, Pramanik R, Shah SB (2021) Financial incentives and other nudges do not increase covid-19 vaccinations among the vaccine hesitant. Technical report, National Bureau of Economic Research.

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

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.

Fudenberg D, Kleinberg J, Liang A, Mullainathan S (2022) Measuring the completeness of economic models. *Journal of Political Economy* 130(4):956–990.

Fudenberg D, Liang A (2019) Predicting and understanding initial play. *American Economic Review* 109(12):4112–41.

Fügener A, Grahl J, Gupta A, Ketter W (2022) Cognitive challenges in human–artificial intelligence collaboration: Investigating the path toward productive delegation. *Information Systems Research* 33(2):678–696.

Giuffrida A, Torgerson DJ (1997) Should we pay the patient? review of financial incentives to enhance patient compliance. *Bmj* 315(7110):703–707.

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

Green B, Chen Y (2019) The principles and limits of algorithm-in-the-loop decision making. *Proceedings of the ACM on Human-Computer Interaction* 3(CSCW):1–24.

Gurvich I, O'Leary KJ, Wang L, Van Mieghem JA (2020) Collaboration, interruptions, and changeover times: Workflow model and empirical study of hospitalist charting. *Manufacturing & Service Operations Management* 22(4):754–774.

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.

Hutto C, Gilbert E (2014) Vader: A parsimonious rule-based model for sentiment analysis of social media text. *Proceedings of the international AAAI conference on web and social media*, volume 8, 216–225.

Ibanez MR, Clark JR, Huckman RS, Staats BR (2018) Discretionary task ordering: Queue management in radiological services. *Management Science* 64(9):4389–4407.

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

Kagan E, Leider S, Sahin O (2021) Dynamic decision-making in operations management. *Johns Hopkins Carey Business School Research Paper* (21-13).

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, Tong J, Peden C (2020) Admission control biases in hospital unit capacity management: How occupancy information hurdles and decision noise impact utilization. *Management Science* 66(11):5151–5170.

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

Kneusel RT, Mozer MC (2017) Improving human-machine cooperative visual search with soft highlighting. *ACM Transactions on Applied Perception (TAP)* 15(1):1–21.

Lage I, Ross AS, Kim B, Gershman SJ, Doshi-Velez F (2018) Human-in-the-loop interpretability prior. *Advances in neural information processing systems* 31.

Lai V, Tan C (2019) On human predictions with explanations and predictions of machine learning models: A case study on deception detection. *Proceedings of the conference on fairness, accountability, and transparency*, 29–38.

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.

Lu J, Lee D, Kim TW, Danks D (2019) Good explanation for algorithmic transparency. *Available at SSRN 3503603* .

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

McIlroy-Young R, Sen S, Kleinberg J, Anderson A (2020) Aligning superhuman ai with human behavior: Chess as a model system. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1677–1687.

Meyer G, Adomavicius G, Johnson PE, Elidrisi M, Rush WA, Sperl-Hillen JM, O'Connor PJ (2014) A machine learning approach to improving dynamic decision making. *Information Systems Research* 25(2):239–263.

Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G, et al. (2015) Human-level control through deep reinforcement learning. *Nature* 518(7540):529–533.

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

Pérez JM, Giudici JC, Luque F (2021) pysentimiento: A python toolkit for sentiment analysis and socialnlp tasks. *arXiv preprint arXiv:2106.09462* .

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

Puiutta E, Veith EM (2020) Explainable reinforcement learning: A survey. *International cross-domain conference for machine learning and knowledge extraction*, 77–95 (Springer).

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.

Ross S, Gordon G, Bagnell D (2011) A reduction of imitation learning and structured prediction to no-regret online learning. *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 627–635 (JMLR Workshop and Conference Proceedings).

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.

Silver D, Huang A, Maddison CJ, Guez A, Sifre L, Van Den Driessche G, Schrittwieser J, Antonoglou I, Panneershelvam V, Lanctot M, et al. (2016) Mastering the game of go with deep neural networks and tree search. *Nature* 529(7587):484–489.

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.

Stites MC, Nyre-Yu M, Moss B, Smutz C, Smith MR (2021) Sage advice? the impacts of explanations for machine learning models on human decision-making in spam detection. *International Conference on Human-Computer Interaction*, 269–284 (Springer).

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

Sun J, Zhang DJ, Hu H, Van Mieghem JA (2022) Predicting human discretion to adjust algorithmic prescription: A large-scale field experiment in warehouse operations. *Management Science* 68(2):846–865.

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* 15(2):122–137.

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

Wang F, Rudin C (2015) Falling rule lists. *Artificial intelligence and statistics*, 1013–1022 (PMLR).

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

Appendix A: Tip Inference Algorithm

We first discuss how we formulate the Markov Decision Process (MDP) for our virtual kitchen-management game and the overall structure of the optimal policies for both the fully-staffed and understaffed scenarios. Then, we provide detailed information on the design and implementation of our tip inference algorithm.

A.1. MDP Formulation

In our virtual kitchen MDP, 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. 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). 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.

A.2. Optimal Policies

We summarize the optimal policy for each scenario. Note that the optimal policy for the understaffed scenario is more counter-intuitive than that for the fully-staffed scenario.

Fully-staffed scenario. Here, the participant has access to all three virtual workers. The optimal number of ticks to complete this scenario is 20. The key insights to achieving optimality 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; they should instead wait to be assigned to plating the first cooked burger, (iv) the chef should never plate, (v) the sous-chef must plate exactly one of the burgers, and (vi) none of the three workers should be left idle except in the previous cases.

Understaffed scenario. Here, the participant has access to only two virtual workers (e.g., the sous-chef and the server). The optimal number of ticks to complete this scenario is 34. The keys insights to achieving optimality 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, even though the server is slow at cooking, (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, in that order, (v) the sous-chef must chop three of the four burgers and the server must chop one, (vi) both workers must plate two burgers each, even though the sous-chef is slower at plating, (vii) the second cooked burger must not be served until the third and fourth burgers are cooked, and (viii) both workers must be kept busy at all times.

A.3. Search Space of Tips

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

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

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.[9] In particular, consider a tip $\rho = (\psi, a)$ with state predicate ψ and action a, where $a = (o, s, w)$ is a tuple consisting of a subtask s of an order o that is to be assigned to worker w. Our algorithm first aggregates all tips of the form $\rho = (\psi, (o, s, w))$ with the same subtask-worker pair (s, w), to obtain a list $R_{s,w} = \{\rho_1, ..., \rho_k\}$ for each (s, w) pair. This (s, w) pair is converted into a tip by counting the number of distinct orders o that occur across $\rho \in R_{s,w}$; if j different orders o occur, then the tip becomes

$$\text{assign } s \text{ to } w, \ j \text{ times.}$$

For example, instead of considering two separate tips

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

we merge them into a tip

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

Next, the score our algorithm assigns to the aggregated tip $R_{s,w}$ is $J(R_{s,w}) = \sum_{\rho \in R_{s,w}} J(\rho)$. Finally, our algorithm chooses the tip $R_{s,w}$ with the highest score.

A.4. Tip Inference Procedure

Next, we describe how our algorithm computes optimal tips for our MDP. While our state space is finite, it is still too large for dynamic programming to be tractable. Instead, we use the policy gradient algorithm to (heuristically) learn an expert policy π_* (Sutton et al. 2000), which uses gradient descent to optimize a policy π_θ with parameters $\theta \in \Theta \subseteq \mathbb{R}^{d_\Theta}$; we choose π_θ to be a neural network. This approach requires that we construct a feature map $\phi : S \to \{0, 1\}^d$. Then, π_θ takes as input the featurized state $\phi(s)$, and outputs a categorical distribution $\pi_*(a \mid \phi(s))$ over actions $a \in A$. Then, the policy gradient algorithm performs stochastic gradient descent on the objective $J(\pi_\theta)$, and outputs the best policy $\pi_* = \pi_{\theta^*}$. For the kitchen

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

game MDP, we use state features including whether each subtask of each order is available, the current status of each 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.

Once we have computed π_*, we use our tip inference algorithm to learn an estimate $\hat{Q}$ of the 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—indeed, our expected improvement is much higher for the bottom 25% (3.6 tips faster for normal, 4.4 ticks faster for disrupted) than for everyone (2.1 ticks faster for normal, 1.8 ticks faster for disrupted), demonstrating that our tip is expected to be most effective for the bottom quartile of participants. In Appendix C.4, we show that our algorithm is robust to this choice, i.e., it produces the same tips if we instead consider the bottom 50% of participants or all participants.

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 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. In Appendix C.4, we show that this second elimination step is robust to the choice of threshold.

A.5. Adapting Our Tips to Other Domains

Broadly speaking, a challenge in interpretable machine learning is that the space of interpretable models must be tailored to each new domain, to ensure that the model captures insights relevant to that domain in a human-interpretable way. For our virtual kitchen management game, we have tailored our tips to convey useful information by first inferring if-then rules, and then aggregating these rules into useful tips. The design decisions include both the post-processing steps used to prune and aggregate tips as well as the feature map over states used to infer tips. We arrived at this tradeoff since we wanted tips that could be easily read and understood by human participants while conveying useful information for improving decision-making. The specific choices we made and the post-processing steps we used were informed by our pilot studies.

When applying our algorithm to a new domain, our approach must be adapted so it infers tips that are useful for that domain. In general, the goal should be to produce tips that are as informative as possible under the condition that a human worker can understand what the tip is trying to convey in a reasonable amount of time. For tasks where individual decisions must be made quickly, the tip must be very succinct and easy to understand; in these settings, post-processing strategies such as ours may be necessary to ensure the human understands the tip. Otherwise, more detailed tips such as the original if-then rules can be used.

Finally, we briefly comment on when we expect our algorithm's tip to outperform both the human tip and the baseline algorithm's tip. As our results demonstrate, the human tip tends to have higher compliance since it is usually more intuitive, yet it might be sub-optimal in settings where the optimal policy is

complex/unintuitive. As a consequence, we expect the human tip to be more effective when the optimal strategy is intuitive; alternatively, one can imagine scenarios where the optimal policy is simply too complex for the human to determine (even with our algorithm's tip), making it better to go with a more intuitive but less effective strategy. For the baseline tip, we expect it to only be effective when the sequential structure is relatively unimportant for achieving good performance (e.g., in well-mixed MDPs), and the human can focus on achieving good immediate reward. In this case, a strategy that directly tries to maximize immediate rewards may also be effective.

Appendix B: Additional Details on Experimental Design

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 food order (different than the burger order used in the main game). Then, they proceed to play the scenarios for the current configuration. Table B.1 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.

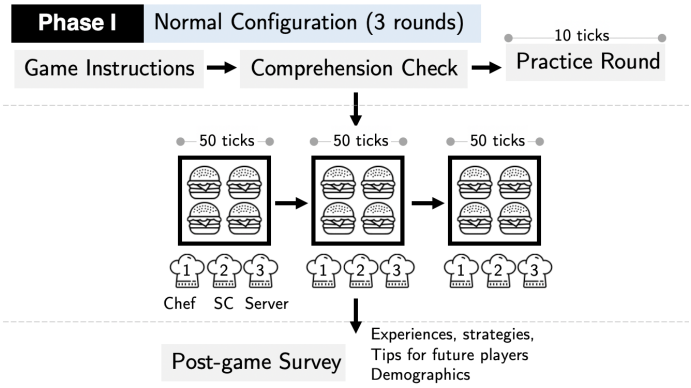
	Chopping meat	Cooking burger	Plating burger
Chef	1	4	6
Souf-chef	2	8	2
Server	3	12	1

Table B.1 The (deterministic) number of time steps each virtual worker requires to complete a given subtask.

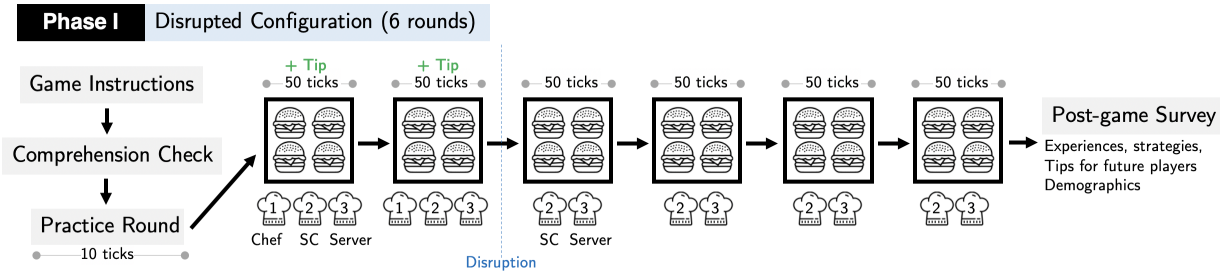
After completing all scenarios, we give each participant a post-game survey regarding their experience with the game. Each participant receives a participation fee of \$0.10 for each round they complete; they also receive a performance-based bonus based on the number of time steps taken to complete each round. The bonus ranges from \$0.15 to \$0.75 per round. Participants provided informed consent, and all study procedures were approved by our institution's Institutional Review Board.

B.1. Phase I

For each configuration, we recruited 200 participants via Amazon Mechanical Turk. 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 pilot 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.



(a) Normal configuration



(b) Disrupted configuration

Figure B.1 Study flow for Phase I.

B.2. Inferred Tips

Next, we use participant data from the final round 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 baseline algorithm, and (iii) rank the candidate tips in the post-game survey based on the number of votes by the participants. As shown in Appendix C.4, the human tips are robust to the participant subgroup used to contruct them—i.e., we get the same tips if we restrict only to top performers.

For the normal configuration, 183 participants[10] successfully completed the game. The top three tips inferred from each of the sources are reported in Table B.2. 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 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 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.

[10] They are 35 years old on average, 57% are female, and 68% have at least a two-year degree.

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%)	Server should never cook (21%)	Chef should never plate (13%)

Table B.2 Top three tips inferred from different sources for the normal 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%)	Server should never cook (24%)	Keep everyone busy (17%)

Table B.3 Top three tips inferred from different sources for the disrupted configuration.

For the disrupted configuration, 172 participants[11] successfully completed the game. Table B.3 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. The baseline algorithm chooses "Sous-chef should plate twice" and the human tip "Server should cook once" (equivalently "Sous-chef should cook three times") got the most votes. Unlike the normal configuration, the top two human tips are not part of the optimal policy. In the optimal policy, sous-chef and 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.

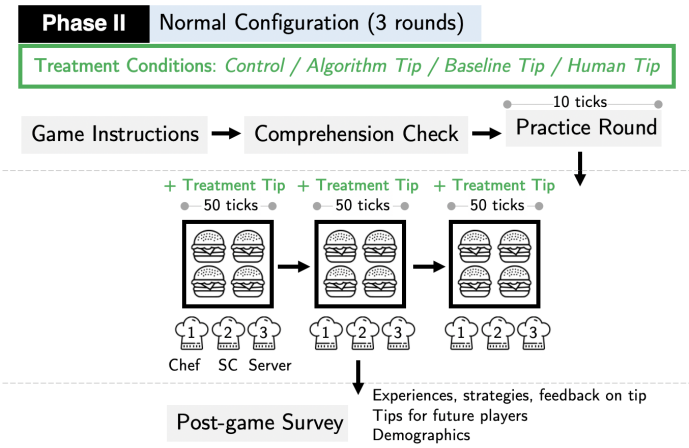
B.3. Phase II

Next, we evaluate the effectiveness of these tips. In this phase, participants are randomly assigned to one of 4 conditions (control, baseline, algorithm, human). We recruited 350 AMT users to play each condition in each configuration, totaling to 2,800 users. The specific tips we show in each round depends not just on the condition, but also varies from round to round depending on the configuration. For the normal configuration, we show the tip for the current condition in all three rounds. However, for the disrupted configuration, the tip for the current condition is specific to the understaffed scenario. Thus, we only show the tip for the current condition in rounds 3–6; in all conditions, for rounds 1 and 2, we show the tip inferred by our algorithm for the fully-staffed scenario from the normal configuration. By doing so, we ensure that the tip shown during the fully-staffed scenario does not bias our evaluation of the tip for the understaffed scenario.

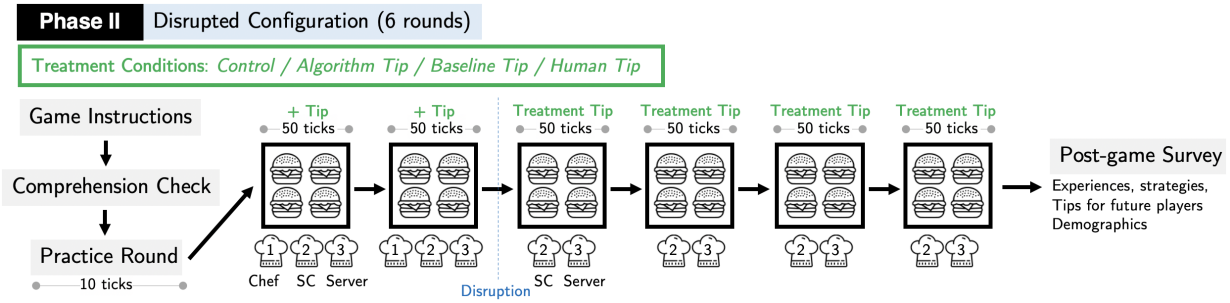
B.4. Pay Schemes

Normal configuration. In Phase I, participants received \$0.30 as a base pay for their participation. In addition, they could earn a performance-based bonus for each of the three rounds of the game. The optimal (e.g.,

[11] They are 36 years old on average, 62% are female, and 78% have at least a two-year degree.



(a) Normal configuration



(b) Disrupted configuration

Figure B.2 Study flow for Phase II.

shortest possible) completion time is 20 time steps and the maximum time allowed is 50 time steps. The bonus is as follows: \$0.75 if completing the round in exactly 20 time steps, \$0.35 if completing the round in 21 to 22 time steps, \$0.15 if completing the round in 23 to 26 time steps, or no bonus otherwise. The total pay ranges from \$0.30 to \$2.55, with a mean of \$1.00, a median of \$0.95, and a standard deviation of \$0.56. The sum of the total pay is \$182.15 (183 participants). In Phase II, which was conducted well into the COVID-19 pandemic, we kept the same base pay but slightly increased the tiered bonus: \$1.25 if completing the round in exactly 20 time steps, \$0.60 if completing the round in 21 to 22 time steps, \$0.25 if completing the round in 23 to 26 time steps, or no bonus otherwise. The total pay ranges from \$0.30 to \$4.05, with a mean of \$1.63, a median of \$1.40, and a standard deviation of \$1.03. The sum of the total pay is \$2,149.70 (1,317 participants).

Disrupted configuration. In both phases, participants received \$0.60 as a base pay for their participation. In addition, they could earn a performance-based bonus for each of the six rounds of the game. For the first two rounds, in which they managed a fully-staffed kitchen, the bonus scheme is the same as that of Phase I of the normal configuration. For the last four rounds, in which they managed an understaffed kitchen (optimal completion time is 34 time steps), the bonus is as follows: \$0.75 if completing the round in exactly 34 time steps, \$0.35 if completing the round in 35 to 36 time steps, \$0.15 if completing the round in 37 to 38 time steps, or no bonus otherwise. In Phase I, the total pay ranges from \$0.60 to \$3.30, with a mean of \$1.63, a

median of $1.55, and a standard deviation of $0.60. The sum of the total pay is $279.55 (172 participants). In Phase II, the total pay ranges from $0.60 to $4.50, with a mean of $1.81, a median of $1.75, and a standard deviation of $0.68. The sum of the total pay is $1,829.25 (1,011 participants).

B.5. Hypothetical Disruption

In the post-game survey of both phases of the normal configuration, participants were asked to imagine a hypothetical understaffed scenario where the chef was no longer available in the kitchen and select the best tip that they believed would most help improve performance in such disruption. Note that these participants did not experience a disruption during their gameplay. The list of tips presented to them is the same as the one offered to the participants in the disruption configuration. Consistently in both phases, the tip that received the most votes is "Server shouldn't cook". Again, this is likely due to the fact that, after three rounds of managing the virtual kitchen under the fully-staffed scenario, the participants potentially learned the optimal policy that the server should not be assigned to cook any burger. Without the actual experience of managing the disruption, they appeared to be biased towards their strategy learned in the fully-staffed scenario, which felt more intuitive to them. This observation highlights one of the key insights of our study that humans' intuition could be far away from the optimal policy, making them less likely to comply to the counter-intuitive tip inferred from our algorithm.

Appendix C: Additional Details on Experimental Results

C.1. Pilot Studies

We ran small-scale pilot studies in late 2019 and early 2020 both online via AMT and in person at our institutions' behavioral lab. Following best practices, the main objectives of these pilots were to finalize the design of the game and ensure feasibility, but not to estimate treatment effects. For example, we investigated how long it would take the participant to finish each round of the game, calibrated the dishes and their cooking tasks/sequences, figured out how to portray different virtual kitchen workers, and improved the user interface of the game. We also experimented with different framing and presentation of tips to get a preliminary understanding of how participants would notice and respond to the tips. Once we identified our final design of the game, we pre-registered our main study and started collecting data in Summer 2020.

C.2. Participant Demographics and Performances

Table C.1 exhibits participants' demographic and gameplay information across our studies. The four groups of participants are not significantly different from one another, except that those playing the disrupted configuration spent slightly longer time to complete the game and found the game to be slightly more difficult, compared to the normal configuration. Tables C.2 and C.3 show the average performance in each round across phases and treatment conditions for normal and disrupted configurations, respectively.

Figure C.1 illustrates participant performance in Phase II of our primary experiment in both the normal and disrupted configurations (same information as Figure 4 but in terms of raw ticks) but in the number of ticks instead of the relative percentage of ticks above optimal. Analogously, Figure C.2 illustrates participant performance in terms of raw ticks for our two follow-up experiments in Section 5.4.

	Phase I: Normal	Phase II: Normal	Phase I: Disrupted	Phase II: Disrupted
Total	183	1,317	172	1,011
Mean age [range]	35 [18, 76]	33 [18, 74]	34 [19, 76]	35 [16, 84]
Female	57%	51%	62%	60%
$\geq$ 2-year degree	73%	68%	78%	70%
Median duration	19 minutes	21 min	28 min	27 min
Found the game difficult	61%	50%	71%	65%
Never played similar games	45%	44%	47%	44%

Table C.1 Participants' demographic and gameplay information.

	Phase I	Phase II: Control	Phase II: Algorithm	Phase II: Baseline	Phase II: Human
Round 1	25.73	26.03	25.04	26.01	26.16
Round 2	25.02	24.46	23.29	24.71	25.06
Round 3	23.74	23.86	22.99	24.04	24.06

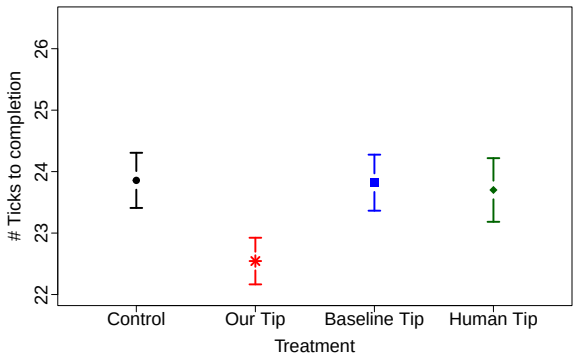
Table C.2 Average performance by treatment condition and round (normal configuration).

	Phase I	Phase II: Control	Phase II: Algorithm	Phase II: Baseline	Phase II: Human
Round 1	24.35	24.26	24.18	24.77	24.64
Round 2	22.87	22.38	22.95	23.08	22.69
Round 3	38.75	38.73	38.19	38.74	38.26
Round 4	38.39	38.21	37.77	38.38	37.85
Round 5	38.25	38.17	37.25	38.42	37.62
Round 6	37.96	37.82	37.14	38.37	37.62

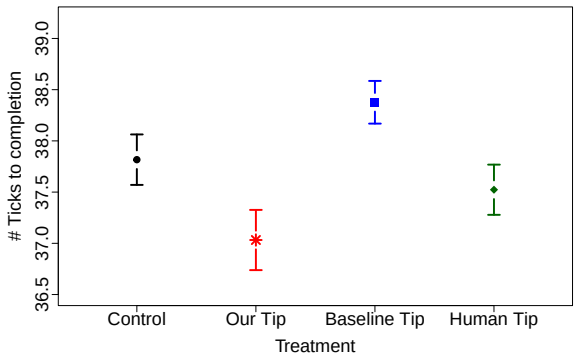
Table C.3 Average performance by treatment condition and round (disrupted configuration).

C.3. Learning Beyond Tips under The Normal Configuration

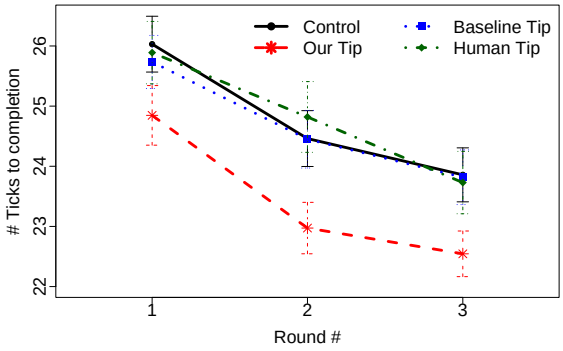
Compared to the disrupted configuration, the normal configuration is much easier. We find that participants across all conditions cross-comply with all other tips: not to assign plating to the chef (Figure C.3a), strategically leave some virtual workers idle (Figure C.3b), and let the chef chop only once (Figure C.3c). These tips are all consistent with the optimal policy, suggesting that participants generally learn over time to improve their performance regardless of the condition. Interestingly, participants in the algorithm condition have similar or higher cross-compliance compared to the other conditions. This result suggests that our tip is the most effective as the information it encompasses the information conveyed by the other tips. 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 recover these optimal strategies as they gain more experience with the game. For instance, the fraction of participants in each arm that never assign 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



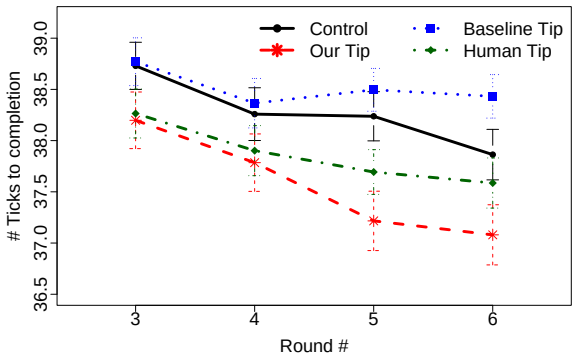
(a) Final Round Performance (Normal)



(b) Final Round Performance (Disrupted)

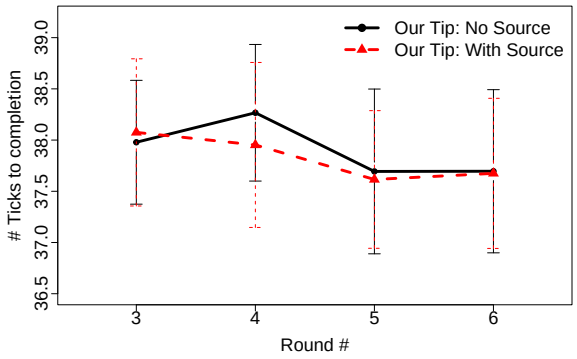


(c) Performance over Time (Normal)

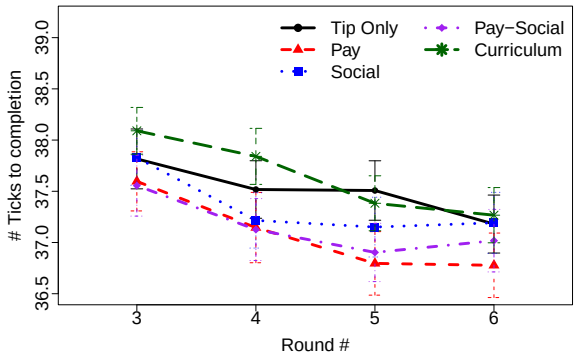


(d) Performance over Time (Disrupted)

Figure C.1 Phase II Participant Performance. The subfigures depict various views of participant performance across conditions in the normal (left) and disrupted (right) configurations. The top row shows performance in the last round of the configuration, the bottom row shows how participant performance improves over time.



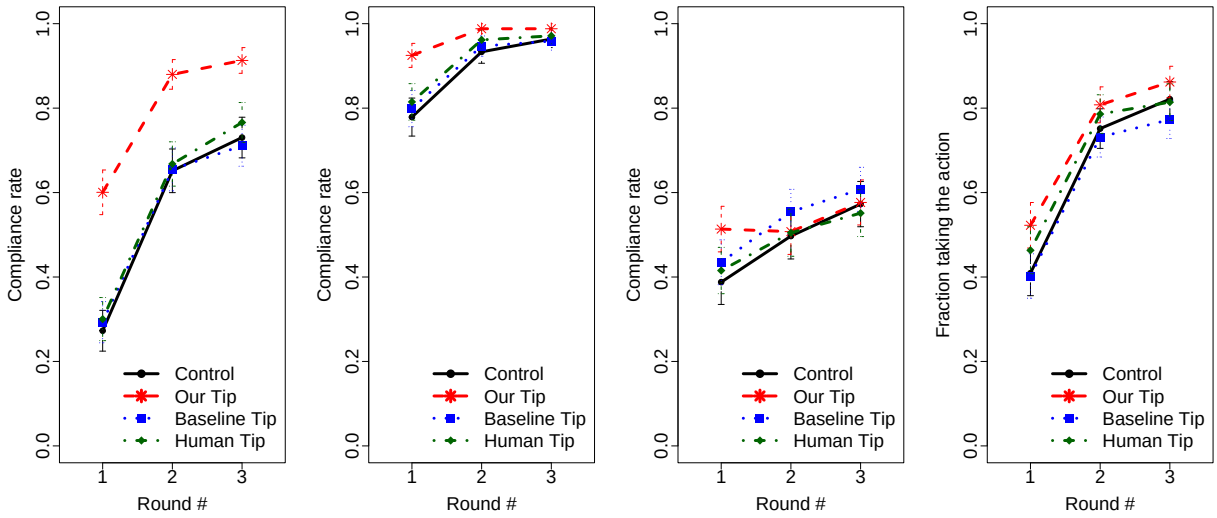
(a) Source of Tip



(b) Interventions for Compliance

Figure C.2 Phase II Participant Performance for Follow-Up Studies. The subfigures depict performance over time across conditions in the Source of Tip (left) and the Interventions for Compliance (right) studies.

statistically different among the arms (see Figure C.3d). This result suggests that participants can uncover this unshown rule by themselves across all conditions.



(a) Our Tip: "Chef Shouldn't Plate" (b) Human Tip: "Leave Some Workers Idle" (c) Baseline Tip: "Chef Chops Once" (d) Unshown Tip: "Server Shouldn't Cook"

Figure C.3 Learning beyond Tip (Normal Configuration). Panels (a)-(c) show the rate at which participants in each condition cross-comply with each offered tip over time in the normal configuration. Panel (d) shows analogous results for a rule that is part of the optimal policy but was not shown as a tip in any condition.

C.4. Robustness Checks for Tip Construction

We now examine the robustness of our results to several design choices.

Robustness to subpopulation used to construct human tip. In our experiment, we desginated the human tip as the highest-voted tip among *all* participants from Phase I. However, one may hypothesize that peer feedback is more effective if it is inferred from *top* performers. To this end, we examine what tips we would have derived if we had restricted to the highest-voted suggestion by the (absolute top, top 5%, top 10%, and top 25%) of Phase I performers. We define performance "Top X% performers" refers to the Phase I participants who had the highest performance in Round 6 (final round). The human tip remained the *same* across all of these subgroups in both configurations; in other words, focusing on best performers would not change our results, since we would still identify the same human tips (see Table C.4).

Threshold for elimination	Normal configuration	Disrupted configuration
Everyone (Original)	Leave some idle	Server cooks once
Best performers	Leave some idle	Server cooks once
Top 5% performers	Leave some idle	Server cooks once
Top 10% performers	Leave some idle	Server cooks once
Top 25% performers	Leave some idle	Server cooks once

Table C.4 Top tips for the "Human" condition based on various subpopulations

Robustness of our algorithm's tip to varying quantiles of human trace data. As described in Appendix A.4, for computing our algorithm's tip, we trained on the bottom 25% of participants since the goal of our paper is to help improve the performance of workers who are weakest at the given problem. Indeed, our expected improvement is much higher for the bottom 25% (3.6 tips faster for normal, 4.4 ticks faster for disrupted)

than for everyone (2.1 ticks faster for normal, 1.8 ticks faster for disrupted), demonstrating that our tip is expected to be most effective for the bottom quartile of participants. To analyze the robustness of our approach, we considered two alternatives: using the bottom 50% or using everyone. As shown in Table C.5, our algorithm produces the same tips using these alternative strategies.

Table C.5 Top tips by our algorithm based on varying quantiles of Phase I human trace data

Criteria for tip selection	Normal configuration	Disrupted configuration
Bottom 25% (Original)	Chef should never plate	Server cooks twice
Bottom 50%	Chef should never plate	Server cooks twice
Everyone	Chef should never plate	Server cooks twice

Robustness of our algorithm's tip to elimination threshold. As described in Appendix A.4, for computing our algorithm's tip, we used a post-processing step where we pruned tips that that disagree with the optimal policy more than 50% of the time. To evaluate robustness, we now consider constructing our algorithm's tip based on alternative thresholds of 30% and 70%. Results are shown in Table C.6; as can be seen, the tip is robust to the choice of this threshold.

Threshold for elimination	Normal configuration	Disrupted configuration
30%	Leave some idle	Server cooks once
50% (Original)	Leave some idle	Server cooks once
70%	Leave some idle	Server cooks once

Table C.6 Top tips for the "Human" condition by various elimination criteria

C.5. Sentiment Analysis of Participants' Qualitative Responses Using Machine Learning

In Section 5.5, we conducted a sentiment analysis for the qualitative responses to the post-study survey question "How did you incorporate the provided tip into your gameplay?"; we presented results through manual coding by a human who is not part of the research team. To ensure robustness, we consider alternatively using two natural language processing approaches for the same task.

First, we use a lexicon and rule-based sentiment analysis tool called VADER (Hutto and Gilbert 2014). VADER provides an API *polarity_scores* that returns four values for each set of texts, including *pos* for positive sentiment, *neg* for negative sentiment, *neu* for neutral sentiment, and a compound score. The compound score is normalized to be between -1 (most extremely negative) and $+1$ (most extremely positive). According to the developer of VADER, among these four scores, the compound score is "the one most commonly used for sentiment analysis by most researchers, including the authors."

However, in practice, although VADER generates a reasonable score for most inputs, it does not perform as well for more complex content. For example, it might classify a response as positive because it sees the word "like" repeatedly, when the word is actually being used as a preposition or conjunction. To address this issue, we instead consider a pre-trained BERT model to classify sentiments. Specifically, we used BERTweet Base Sentiment Analysis, a RoBERTa model trained on English tweets (Pérez et al. 2021). We used the pipeline API provided by Hugging Face's Transformers library. Because the model returns three separate

scores: *Positive*, *Neutral*, and *Negative*, to get a final compound score as we had before, we calculated $Positive - Negative$.

Table C.7 exhibits all the scores from our VADER and BERTweet analyses using the responses among participants in the disrupted configuration of Phase II. We find that our results are highly consistent with our initial approach using the human coder—i.e., the human tip consistently has higher positive, lower negative, and higher compound scores than our algorithm's tip across both approaches, suggesting that participants consistently perceive our algorithm's tip to be less favorable.

Disrupted	VADER+	VADER	VADER−	VADER	B+	B	B−	BERTweet
Algorithm	0.0922	0.8515	0.0563	0.0256	0.2113	0.4712	0.3176	−0.1063
Human	0.1042	0.8601	0.0358	0.1305	0.2473	0.6100	0.1427	0.1046
Baseline	0.0979	0.8339	0.0682	0.0600	0.1315	0.5029	0.3657	−0.2342

Table C.7 Positive, neutral, negative, and compound sentiment scores from VADER and BERTweet, using Phase II's disrupted configuration results. $+$, , $-$ refer to positive, neutral, and negative scores, respectively.

Appendix D: Screenshots of Our Virtual Kitchen Management Game

Finally, we provide screenshots to illustrate our experimental design. Figures D.1 and D.2 show the introduction to the task shown to participants explaining various concepts in the game. Figures D.3 and D.4 show instructions for the fully-staffed and understaffed scenarios, respectively, shown to participants. Finally, Figure D.5 shows the payment information shown to the participants.

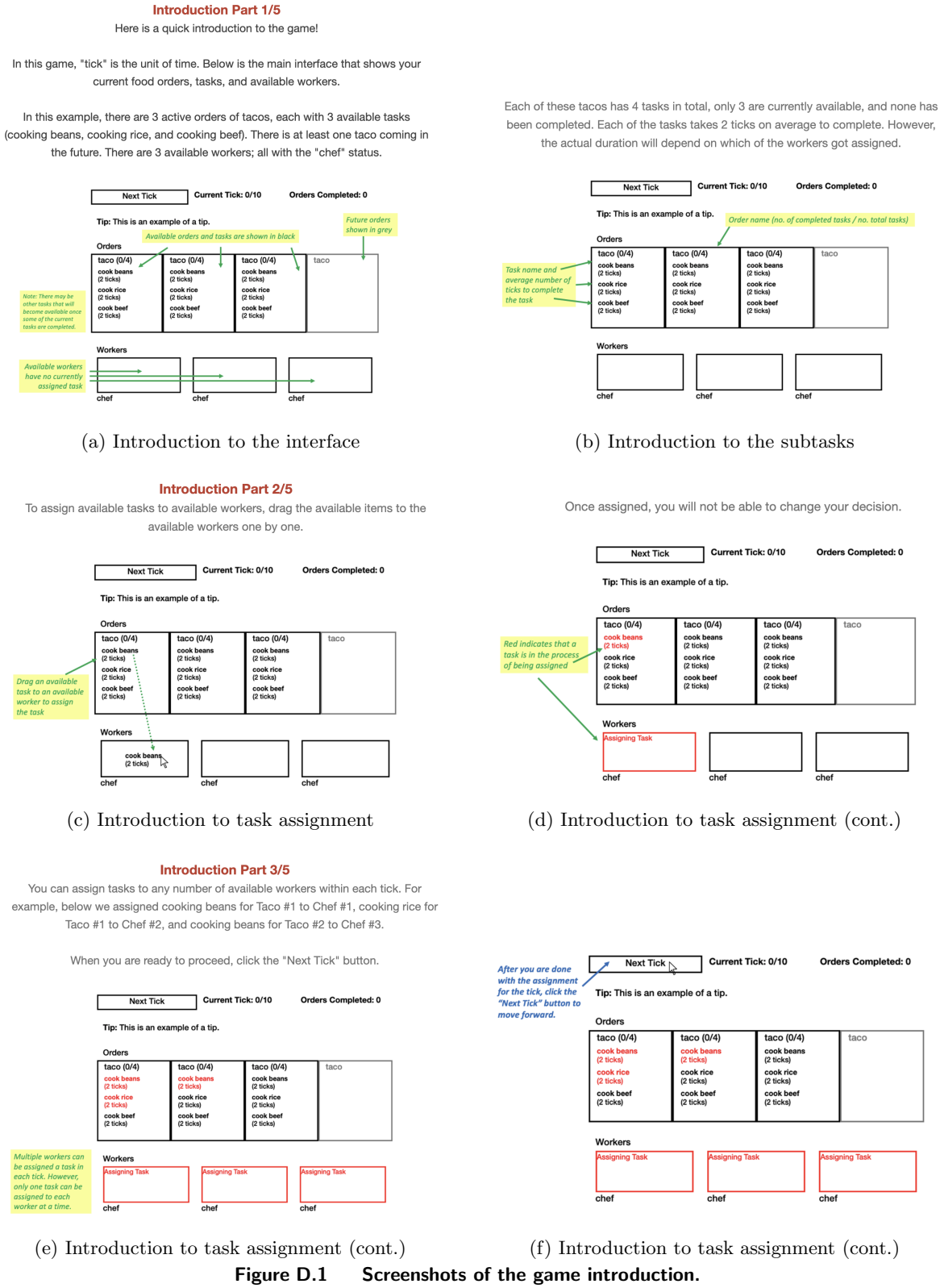
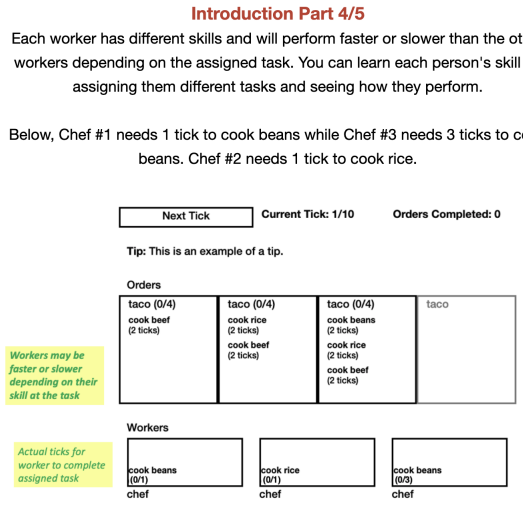
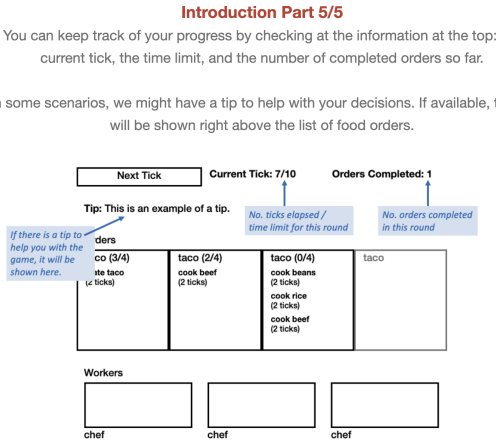


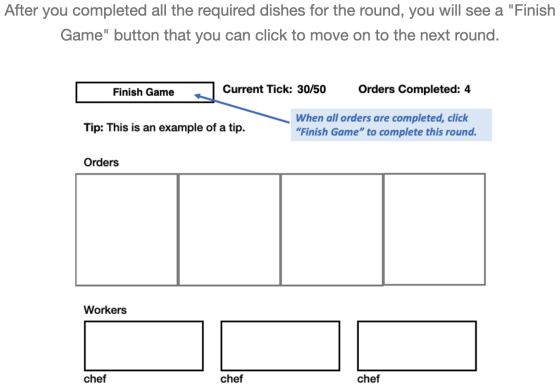
Figure D.1 Screenshots of the game introduction.



(a) Introduction to workers' skill levels

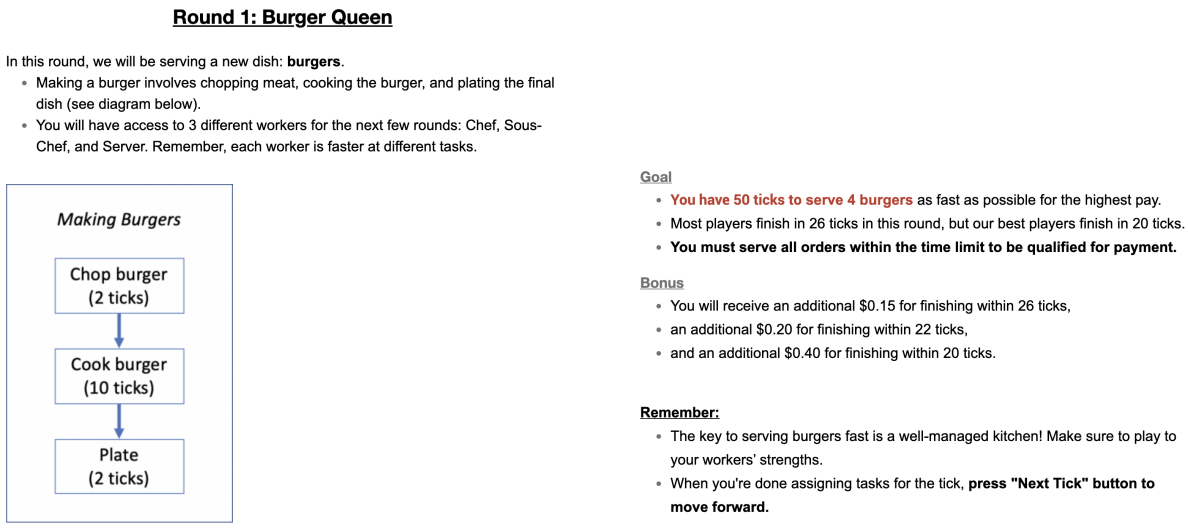


(b) Introduction to the tip



(c) Introduction to round completion

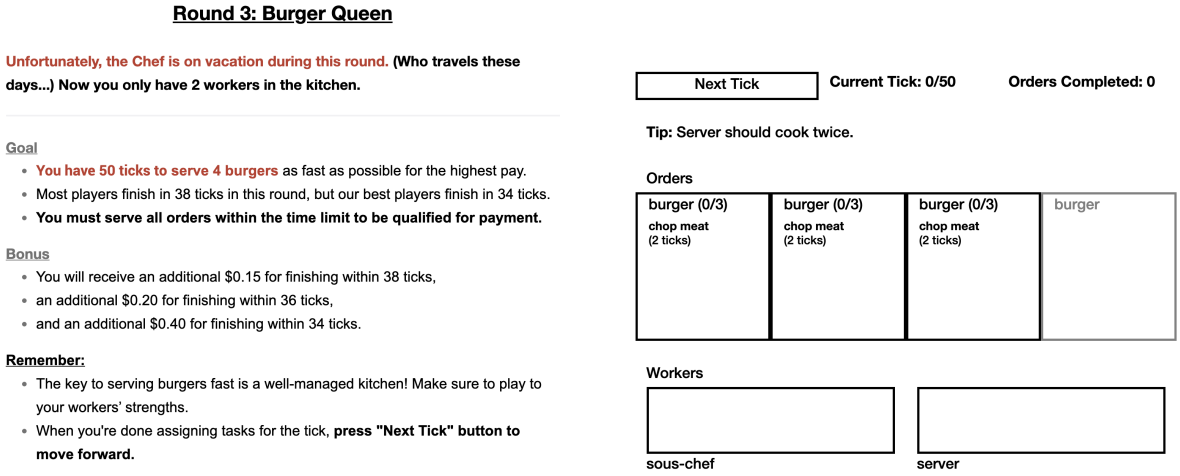
Figure D.2 Screenshots of the game introduction (continued).



(a) Burger's subtasks and available workers

(b) Goal, incentives, and reminder

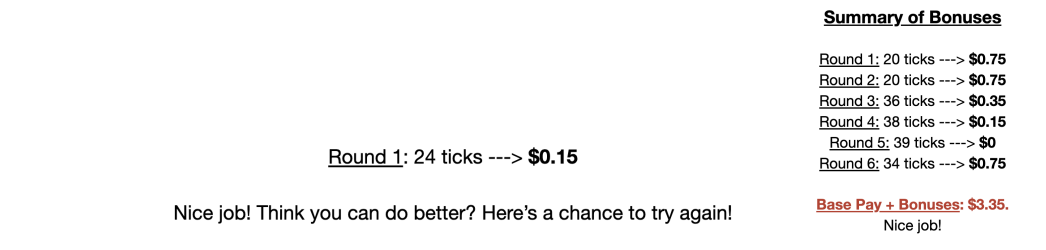
Figure D.3 Screenshots of the instructions for the fully-staffed scenario.



(a) Updated instructions following the in-game disruption

(b) Game interface (with the algorithm tip)

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



(a) Individual round pay information

(b) Summary of total pay

Figure D.5 Screenshots of the pay information.