Improving Human Decision-Making with Machine Learning

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

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

Wichinpong Park Sinchaisri
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 the human workers and those taken by the optimal policy in a way that accounts for which actions are consequential for achieving higher performance. We evaluate our approach through a series of randomized controlled 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 further 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 making decisions to optimize the long-term outcomes of their patients (Kleinberg et al. 2015) or drivers on ride-hailing platforms optimizing their long-term profits (Marshall 2020). As a concrete example, physicians

seek to learn good strategies for ordering lab tests, since obtaining the appropriate testing results in a timely fashion is necessary to minimize delays in patient visits. Song et al. (2017) finds that experienced physicians have learned to order these tests early 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 suboptimal 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 on a ride-hailing platform; even retail manager decisions on pricing and inventory management are recorded on a daily basis. This data implicitly encodes the collective knowledge acquired by numerous workers about how to effectively perform their jobs. However, trace data is often extremely noisy, granular, and of tremendous volume, rendering it unreadable to humans. At the same time, recent advances in machine learning have enabled machines to achieve human-level or super-human performance at many artificial intelligence tasks (Mnih et al. 2015, Silver et al. 2016). The knowledge behind their abilities is often hidden within blackbox models such as deep neural networks, making them less amenable to augmenting human understanding. Thus, recent work on interpretable machine learning has strived to learn structured models where the computation is transparent to the user, such as decision trees (Breiman et al. 1984, Bertsimas and Dunn 2017), rule lists (Letham et al. 2015), or generalized additive models (Caruana et al. 2015). Indeed, recent work suggests that there often exist interpretable models that achieve performance similar to their deep neural network counterparts (Rudin 2019, Bravo et al. 2019); instead, the challenge lies in

discovering these high-performing models. Thus, we might hope to leverage these techniques to mine high-volume trace data and automatically discover insights that can help workers improve their performance.

In this paper, we study whether machine learning can be used to infer rules that help improve workers' performance at sequential decision-making tasks. Existing work on interpretable machine learning has largely focused on whether a human can understand the computation performed by a model (Caruana et al. 2015, Doshi-Velez and Kim 2017); in contrast, our goal is to examine whether these models can convey useful insights to humans. To this end, we devise a novel algorithmic framework for inferring simple *tips* (i.e., best practices) that, if adopted, can improve the performance of a human user. We focus on settings where the human must make a sequence of decisions to optimize an outcome in a complex environment. These settings are particularly challenging for humans, since they must trade off short-term and long-term rewards in ways that are often counter-intuitive.

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 high-performance neural network decision-making policy using reinforcement learning (Sutton and Barto 2018), and then training an interpretable policy to approximate this neural network. However, unlike prior work, we are interested in interpreting the *discrepancy* between the human policy and the neural network (rather than the neural network itself). Thus, we encode this discrepancy into a novel objective, and select an interpretable tip that best minimizes the discrepancy. Intuitively, this tip best bridges the gap between the actions taken by the human users and the ones taken by the neural network. Importantly, it does so in a way that accounts for which actions are consequential for achieving higher performance—i.e., it suggests actions that are expected to improve the long-term performance of the human rather than to simply mimic the neural network. While previous work has leveraged machine learning to predict when humans make mistakes compared to the optimal policy (Fudenberg and Liang 2019, McIlroy-Young et al. 2020, Fudenberg et al. 2021), they do so in an uninterpretable way that makes it difficult to convey their insights to humans. We design the search space 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 due to the sequential nature of decision-making problems.

Two criteria are needed for our approach to actually improve human decision-making. First, our algorithm must be able to identify sufficiently useful tips to improve performance. Second, humans must be able to understand our tip and, furthermore, effectively operationalize it by modifying their policy to incorporate the suggested actions. To study these issues, we designed and built a sequential decision-making game where human users manage a virtual kitchen, inspired

by the popular game *Overcooked*. Our primary contribution is a large-scale randomized controlled experiment within this game; Figure 1 illustrates the high-level setup and flow of the game and Section 3 provides a more detailed description. Users must assign subtasks to virtual workers with varying capabilities in a way that optimizes the time it takes to complete a set of food orders. This game is challenging because users 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.

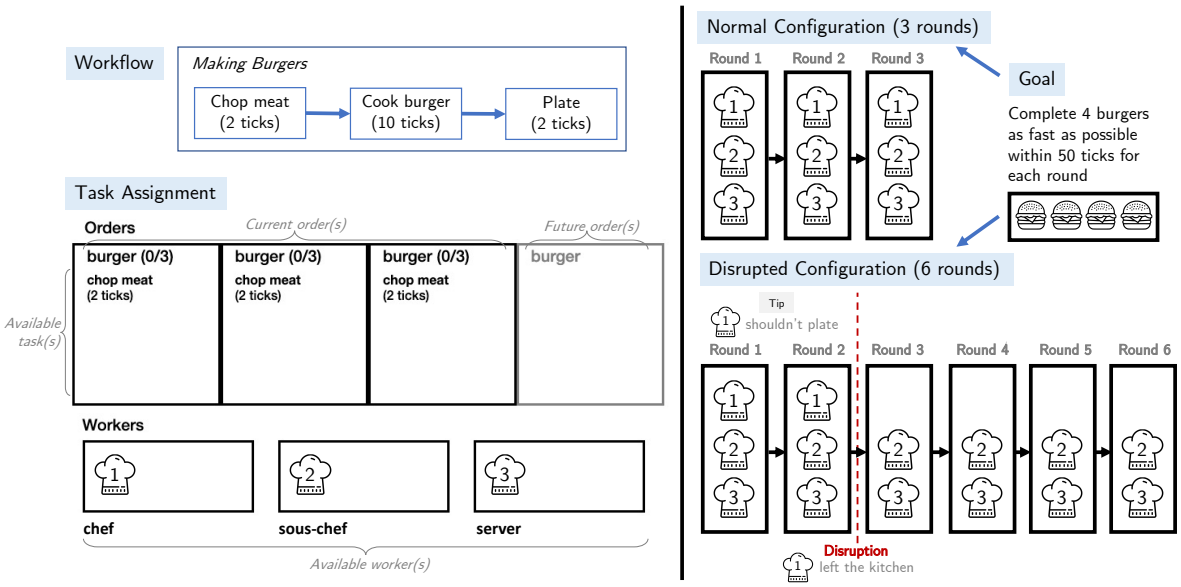


Figure 1 Overview of kitchen management game. The left panel depicts what participants see: (i) the workflow required to complete a burger order, and (ii) the game screen that allows available tasks to be dragged and dropped to one of 3 virtual workers. The right panel depicts the study design: in the 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.

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 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 pre-conceived 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. Interestingly, 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 machine learning can effectively leverage 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

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. For instance, as mentioned earlier, Ramdas et al. (2017) finds that surgery duration increases substantially when the surgeon is using a new surgical device, hurting both service quality and productivity. Similarly, unexpected critical medical incidents slow down ambulance activation among paramedics (Bavafa and Jónasson 2020). 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. 2019). 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, offering best practices (Nonaka and Takeuchi 1995, Pfeffer et al. 2000, Spear 2005). This involves first creating standards, and then communicating them. Creating standards can be challenging (e.g., since knowledge doesn't transfer across organizational borders, Szulanski 1996, Argote 2012) and is known to be time-consuming (Nonaka and Takeuchi 1995). A rich literature in operations management and organizational behavior has studied how various experiences can improve individuals' productivity and performance. For example, professional web developers frequently learn new concepts and strategies by trial and error (Dorn and Guzdial 2010). Past experience on the same or related tasks play an important role in performance, sometimes making

it challenging for workers to identify best practices (Huckman and Pisano 2006, Kc and Staats 2012) but also beneficially reducing variance in performance (Bavafa and Jónasson 2021). Social interaction is another common way to learn, e.g., therapy workers learn from client feedback to adjust their treatment process (Brattland et al. 2018). Workers also learn significantly from their colleagues, particularly those with a high level of knowledge or valuable skills (Herkenhoff et al. 2018, Jarosch et al. 2019). Song et al. (2017) shows that by publicly disclosing relative performance feedback, physicians can better identify their top-performing co-workers, enabling the identification and validation of best practices. Working alongside experienced peers has been shown to improve worker performance (Chan et al. 2014, Tan and Netessine 2019). To this end, familiarity between workers and prior experience working with these peers on the same tasks has been associated with improving both team and individual performance (Akşin et al. 2020, Kim et al. 2020). In summary, prior work has shown that the availability of experts and knowledge of best practices, if deployed effectively, can significantly improve worker learning. 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.

Once standards or best practices are identified, effectively sharing and encouraging workers to adopt them is also known to be challenging (Tucker et al. 2007). One way to improve such knowledge transfer is to structure it as a simple rule. The clarity of a simple rule allows workers to gain a deeper understanding of the environment and the potential improvement to be obtained (Sull and Eisenhardt 2015, Gleicher 2016). Similarly, simple training interventions have been shown to improve decision-making by persistently reducing cognitive biases (Morewedge et al. 2015, Sellier et al. 2019). Thanks to the fast-growing advancement of artificial intelligence, machine learning models have demonstrated great success in learning complex systems and making predictions that help guide high-stakes decision-making in various domains, from healthcare to criminal justice (Bastani et al. 2022, Letham et al. 2015). However, most commonly used black-box models do not provide users with transparency, accountability, or explanations. The lack of human understanding of how algorithms work can pose serious problems to society (Rudin 2019) and, furthermore, can lead to aversion from adopting these tools (Dawes et al. 1989, Dietvorst et al. 2015). In recent years, significant effort has been dedicated towards the development of models that are inherently interpretable (see, for e.g., Murdoch et al. 2019 for an in-depth review of methods and applications of interpretable machine learning). Relatedly, recent work has sought to incorporate human domain knowledge into algorithms (Arvan et al. 2019, Ibrahim et al. 2021), and to leverage AI to augment decision-making and service offerings (Snyder et al. 2022, Choi et al. 2022, Hu et al. 2022). Our work contributes to this stream of literature in two ways. First, we show that a simple

intervention—providing a simple tip—can help improve worker performance over time and speed up their learning process; we further find that workers obtain additional performance improvements by augmenting the surface-level meaning of the tip with their own experience. Second, we develop a novel algorithm that leverages the largely untapped potential of worker trace data to complement existing training programs and learning among workers.

2. 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.

Preliminaries: 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 § 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 § 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

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

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

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

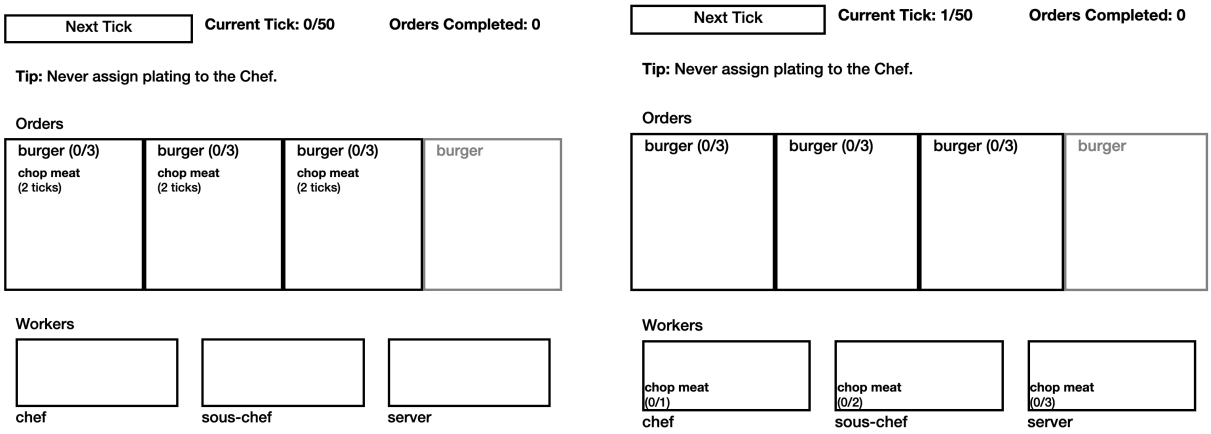
We optimize Eq. (4) by enumerating through candidate tips ρ, evaluating the objective, and selecting the tip $\hat{\rho}$ with the highest objective value.

3. Case Study: Virtual Kitchen-Management Game

Inspired by the popular game *Overcooked*, we developed a sequential decision-making task in the form of a virtual kitchen-management game that can be played by individual human users (see Figure 1). We deliberately chose a task 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 this game, the user takes the role of a manager of several virtual workers—namely, chef, sous-chef, and server—serving burgers in a virtual kitchen. Each burger consists of a fixed set of subtasks 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 user 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 user completes the game once four burger orders are completed. The user'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 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 A.1 in Appendix B for details. Ideally, one would match workers to tasks that they are skilled at to reduce completion time. Thus, the user faces the following dilemma. When a worker becomes available but is not skilled at any of the currently available subtasks, then the user must decide between (i) assigning a suboptimal subtask to that worker, potentially creating a bottleneck, or (ii) leaving the worker idle until a more



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

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

Figure 2 Example screenshots from the game.

suitable subtask becomes available. For instance, if the server is idle but all available subtasks are "cook burger", then the user 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, users 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 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 user has access to all three virtual workers (chef, sous-chef, and server). In the second scenario, the human user faces a disruption and the kitchen becomes *understaffed*, with only two virtual workers (sous-chef and server).

We describe how this decision-making problem can be formulated as a MDP and the resulting optimal policies in Appendix A.

4. Experimental Design

We investigate how humans interpret and follow the tips inferred by our algorithm in pre-registered behavioral experiments involving Amazon Mechanical Turk (AMT) users.[1] Participants are compensated a flat rate for completion of the study, and a relatively larger performance-based bonus determined by how quickly they complete each round of the game; see Experimental Design Details in the Appendix for payment details and game screenshots.

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

Hypotheses: First and foremost, we study whether participants shown our tip ("algorithm" treatment condition) outperform

1. participants not shown any tips (control group),
2. participants shown the tip most frequently suggested by previous human participants who played the same scenario multiple times ("human" condition), and
3. participants shown a tip derived by a baseline algorithm that identifies the state-action pair where human participants and the optimal policy most frequently differ ("baseline" condition).

Second, we examine participant behaviors in response to different tips, particularly their compliance and how they learn to improve decision-making beyond the provided tips.

We chose these baselines to illustrate how our algorithmic approach compares to and complements worker learning in practice. The control condition represents settings where expert advice and knowledge of best practices are not readily available, and so workers learn over time based on their own experience; indeed, we observe that performance improves over time without any tips. 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). Finally, the baseline condition illustrates a naïve use of 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.

Experimental Flow: Our experiments proceed in two phases. In Phase I, we collect trace and survey data for learning tips (without showing tips to any participants). This phase is intended to collect historical data that would normally already be available for an existing decision-making task. In Phase II, we randomize participants to the four different conditions (where they are shown varying tips) and evaluate their performance. In both phases, each participant plays a sequence of three to six rounds of the game, which we call a *configuration*; this approach enables us to study both how performance varies with the tip they are shown, as well as how it evolves across games as participants gain experience.

We consider two configurations. In the *normal* configuration, each participant simply plays three rounds of the fully-staffed scenario. In the *disrupted* configuration, each participant plays two rounds of the fully-staffed scenario, followed by four rounds of the understaffed scenario. 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 set of participants across all four configuration-phase pairs is mutually exclusive—i.e., no participant has prior experience with any version of the game.

Figure 3 summarizes the experimental flow. We further detail each phase below.

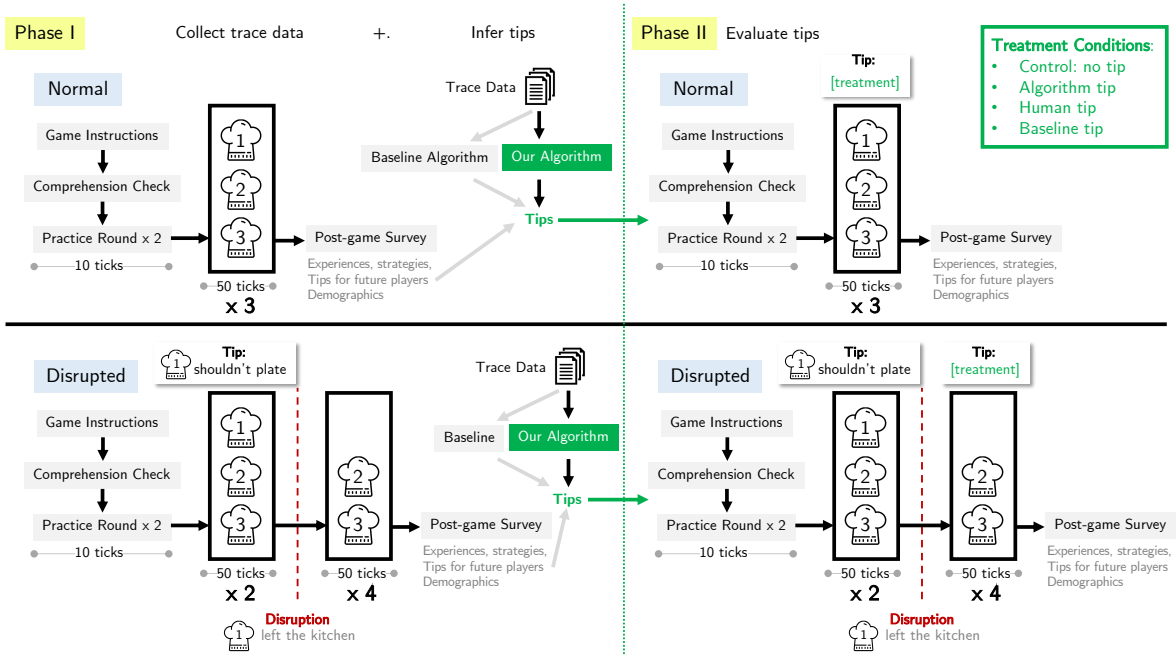


Figure 3 Overview of experimental flow. The top two panels depict Phase I (left) and II (right) for the normal configuration, where each participant plays three fully-staffed scenarios. The bottom two panels depict Phase I (left) and II (right) for the disrupted configuration, where each participant plays two fully-staffed and four understaffed scenarios. Participants in Phase II 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.

Phase I Details: In Phase I, we have a small sample of human participants (normal: $N = 183$ participants; disrupted: $N = 172$ participants) play one of the two configuration without tips, and collect data on the actions they take. Then, we use our tip inference algorithm, given by Eq. (4), in conjunction with the trace data to generate the baseline and algorithm tips. Details are provided in Appendix A.

Next, each participant is also shown a comprehensive list of candidate tips at the end of Phase I, 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 user 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").

Note that the optimal 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.

Lastly, we compute the baseline tip, which simply attempts to best match the human user's actions with those of the optimal policy, without regard to which actions are consequential 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) $\zeta_1, ..., \zeta_k \sim D^{(\pi_H)}$, we select 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, $C^*(s,a)$ encodes the frequency with which the optimal policy reaches state s and takes action a, so 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.

Phase II Details: In Phase II, we randomly assign a large number of new participants (normal: $N = 1,317$ participants; disrupted: $N = 1,011$ participants) into one of four conditions: control (no tip), algorithm (shown the tip $\hat{\rho}$ inferred by our algorithm in Eq. (4) based on Phase I data), human (shown the tip most frequently chosen by Phase I participants), or baseline (shown the tip $\hat{\rho}_{\text{bl}}$ inferred by the baseline algorithm in Eq. (5) based on Phase I data). In the normal configuration, Phase II participants are shown the tip designated by their 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 2 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 the next section, 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.

Finally, we assess performance in each condition both through the overall completion time in the final round, as well as the fraction of participants who ultimately learn the optimal policy. Note that the optimal policy would complete four burgers in 20 and 34 time steps for the fully-staffed and understaffed scenarios, respectively.

Details on the experiment, inferred tips, participant demographics, performance-based pay, and game screenshots are provided in Appendices B and C.

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*

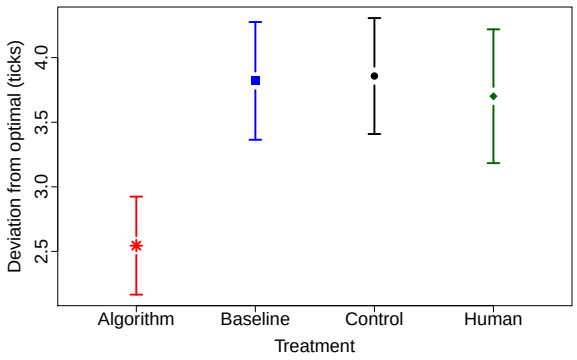
Figure 4 shows performance results across all four conditions and both configurations. Figure 4b & 4c show participant performance in the final round of our game, Figure 4d & 4e show how performance improves across rounds, and Figure 4f & 4g show the fraction of participants achieving optimal performance across rounds. We discuss these results below.

The normal configuration is relatively easy—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.

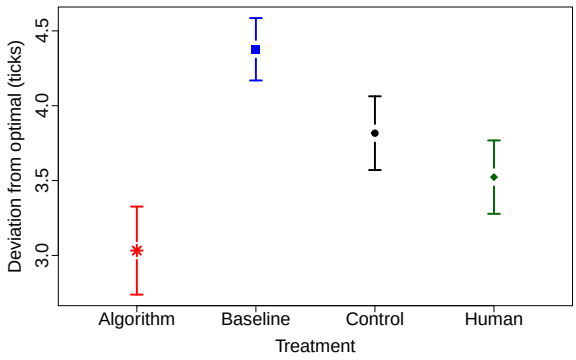
The disrupted configuration is substantially harder, since participants must adapt to the more counter-intuitive understaffed scenario. Perhaps as a consequence, participants benefit much more from tips: those in the control group took four rounds to achieve the same level of performance as those shown 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 tip from the baseline algorithm ($t(246) = -7.348,\ p < 10^{-4}$). In the disrupted configuration, the baseline tip actually *reduces* participant performance, likely by misleading participants. 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

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

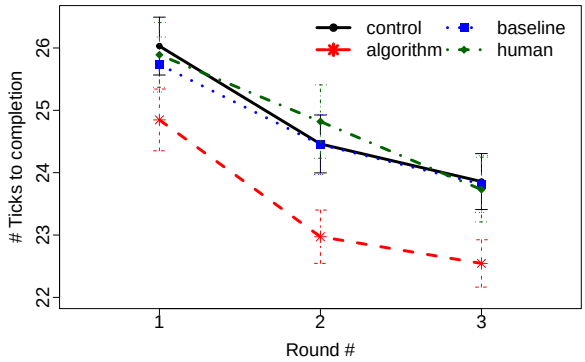
(a) Tips for each condition and configuration



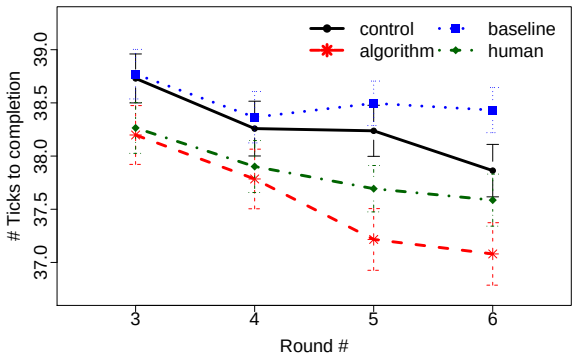
(b) Final Round Performance (Normal)



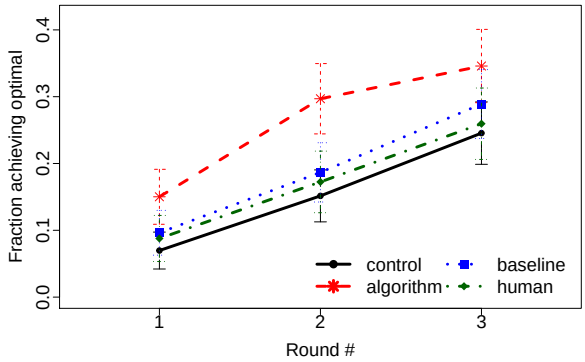
(c) Final Round Performance (Disrupted)



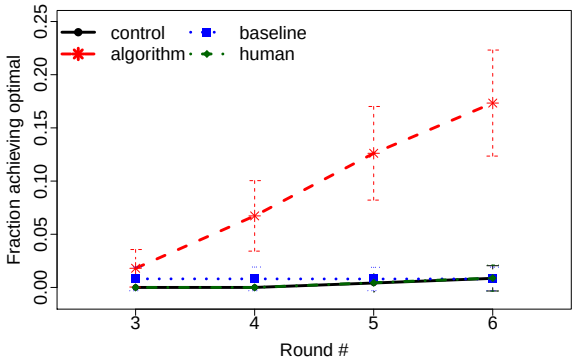
(d) Performance Over Time (Normal)



(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.

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 alternative tips: Next, we discuss potential reasons for the success of our algorithmic tips compared to other tips. First, the baseline tips likely perform poorly since it blindly tries to mimic the optimal policy rather than focusing on consequential actions. Specifically, in the disrupted configuration, the baseline tip "Sous-chef should plate twice" suggests actions that occur at the end of the game (after it is too late for participants to significantly improve their performance) and does not focus on the critical performance bottleneck (cooking). In contrast, our algorithm focuses on mimicking decisions made by the optimal policy that have long-term benefits—e.g., our tip "Server should cook twice" frees the sous-chef to plate later in the game.

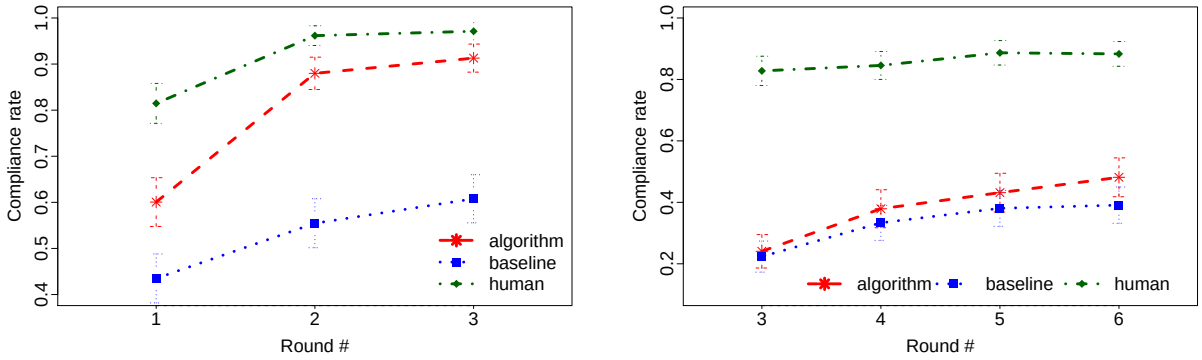
While the human-suggested tips consistently improve performance compared to the control group, they can be overly general or incorrect. In the normal configuration, Phase I participants were unable to translate their strategy into a specific tip—i.e., their suggested tip "Strategically leave some workers 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, but at an insufficient magnitude.

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. Here, we examine compliance with the tips; we examine learning strategies beyond the tips below. Note that participants were not informed of the source of the tip (i.e., algorithm or human suggestions), so any variation in compliance is due to the content of the tip, rather than inferences based on the source of the tip (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. 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 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 algorithmic tip, particularly in the disrupted configuration. Based on participants' post-game feedback, we found that this is likely because the human-suggested tip



(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.

better matches human intuition (since it is devised by humans). The disrupted configuration is illustrative. Although the algorithmic tip is correct (unlike the human-suggested tip), it is highly counter-intuitive, hurting adoption. For example, in the disrupted scenario, our tip "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 the algorithmic tip (i.e., compliance with the algorithmic 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 § 5.4). Thus, compliance is not only a function of the syntactic structure of the tip (which is unchanged across conditions), but also its semantic content. When the optimal strategy is counter-intuitive, we observe an intrinsic trade-off between the optimality of the tip and compliance to the tip. Even in the disrupted configuration, our 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 algorithmic 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 be effective at improving human compliance with recommendations or guidelines; 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).

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

Next, we examine if humans learn performance-improving strategies that goes beyond the specific tips they were shown. To this end, we examine *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 (assuming that it does not overlap with the tip they were shown). Thus, any cross-compliance beyond that of the control group measures how a tip enables participants to learn strategies beyond the stated tip. We focus on the disrupted configuration since it is more challenging for participants, leading to more interesting cross-compliance patterns across conditions.[3] Figure 6 shows the cross-compliance of participants in each condition with the different tips (algorithm, baseline, human), as well as a new rule ("Server chops once") that is a necessary part of the optimal policy but was not explicitly shown to any participants.

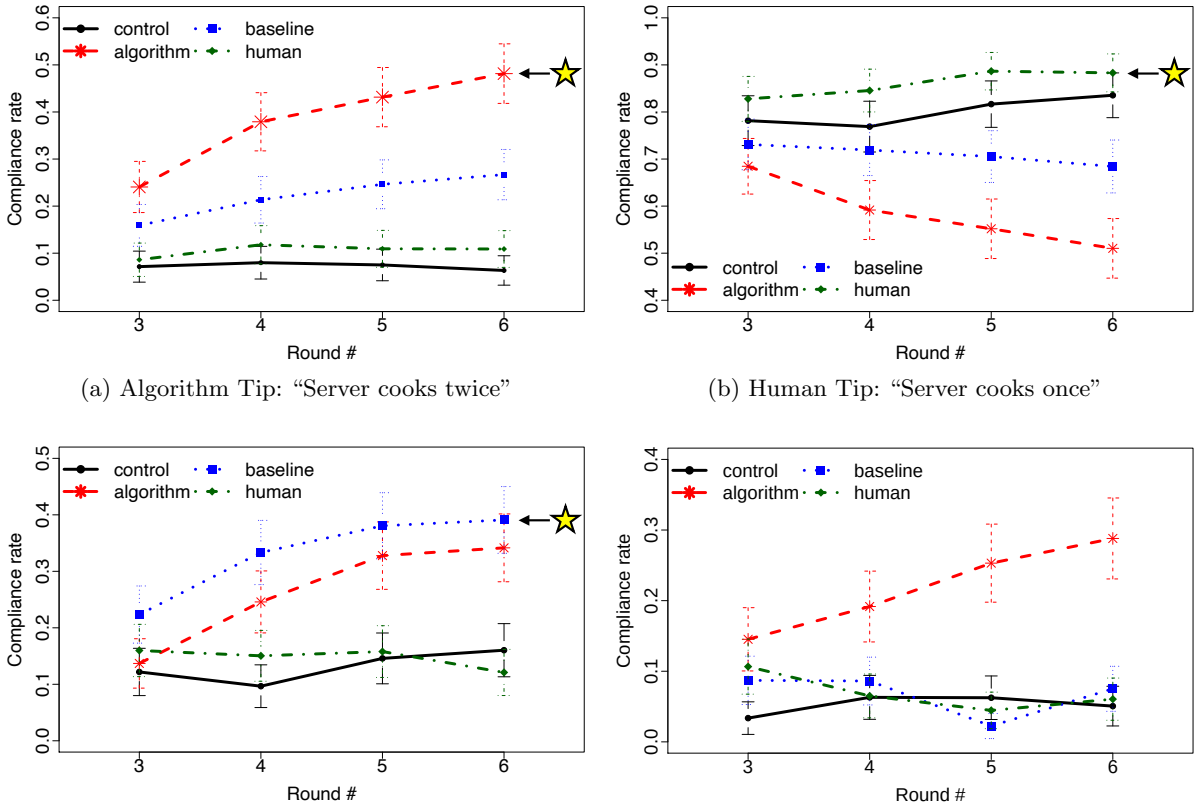
Participants in the human and control groups only comply with the human tip; indeed, the human-suggested tip actually contradicts the optimal policy, thereby preventing participants from learning the other tips which are part of the optimal policy.[4] Similarly, participants shown the baseline tip only have high compliance with the baseline tip, indicating that the baseline tip could not help participants uncover the optimal policy. In contrast, participants who received our tip have high cross-compliance with *all* parts of the optimal policy (i.e., the baseline and unshown tips); 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 beyond what is stated in the tip.

5.4. Participant Comments on The Provided Tips

"What did you think about the tip for these last [three/four] rounds and how did you incorporate it in your strategy?"

[3] 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 tip (see Appendix C.1).

[4] 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.



(a) Algorithm Tip: "Server cooks twice"

(b) Human Tip: "Server cooks once"

(c) Baseline Tip: "Sous-chef plates twice"

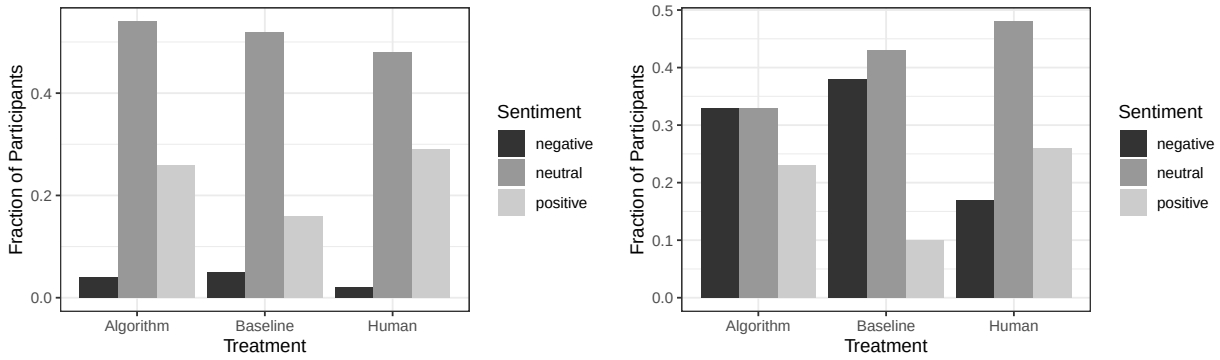
(d) Unshown Tip: "Server chops once"

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; results from participants in the same condition are marked with yellow stars. 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.

We manually code participants' sentiment towards the tip they received—positive, neutral, or negative—based on their post-game survey responses. Figure 7 exhibits the breakdown of responses for participants in each condition and configuration.[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, 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

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



(a) Normal Configuration (b) Disrupted Configuration

Figure 7 Participant sentiment on the provided tips in post-game survey.

incorporate these tips into their workflow and execute an optimal strategy. This is supported by selected excerpts from participant comments presented in Table 1.

Disrupted Configuration	Algorithmic 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).

As can be seen, many participants felt that the human tip was more accurate since it more closely matched their intuition, and they disagreed with the tips that they found counter-intuitive. Interestingly, some participants even found the human tip to be counter-intuitive since it did not match their own intuition from the fully-staffed scenario in prior rounds (i.e., it asks the server to cook once instead of not at all). As a consequence, compliance and performance suffered. Importantly, we also observe that even participants who successfully understood the algorithmic 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 machine learning algorithm for automatically identifying interpretable tips designed to help improve human 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 machine learning can be used to improve human 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 to identify good strategies. In recent years, a growing number of organizations have adopted a gig economy employment model or allowed for remote work in response to worker preferences for flexibility and independence. To compensate for the

lack of interactions among workers, firms can employ our algorithm to learn best practices from high-performing workers, and provide these as tips to help less experienced workers improve. Our approach is also relevant to safety-critical applications where algorithms may not be as reliable as humans (e.g., driving, healthcare); in these cases, the optimal policy can be replaced with a policy estimated on trace data from top human workers to allow transfer of knowledge from experienced (high-performing) to novice workers. There are also sensitive applications (e.g., customer service) where human decision-makers may naturally be preferred over algorithms (Buell 2018).

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 decision-making.

References

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

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

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

Bastani H, Drakopoulos K, Gupta V, Vlachogiannis J, Hadjicristodoulou C, Lagiou P, Magiorkinis G, Paraskevis D, Tsiodras S (2022) Interpretable operations research for high-stakes decisions: Designing the greek covid-19 testing system. *INFORMS Journal on Applied Analytics (Forthcoming)* .

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

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

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

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.

Bravo F, Rudin C, Shaposhnik Y, Yuan Y (2019) Simple rules for predicting congestion risk in queueing systems: Application to icus. *Available at SSRN 3384148* .

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.

Buell RW (2018) The parts of customer service that should never be automated. *Harvard Business Review* .

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

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

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.

Choi S, Kim N, Kim J, Kang H (2022) How does artificial intelligence improve human decision-making? evidence from the ai-powered go program .

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

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

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

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

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

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

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

Fudenberg D, Kleinberg J, Liang A, Mullainathan S (2021) Measuring the completeness of economic models. Technical report.

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

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.

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

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

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

Hu S, Zhang J, Zhu Y (2022) Zero to one: Sales prospecting with augmented recommendation. *Available at SSRN 4006841* .

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

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

Ibrahim R, Kim SH, Tong J (2021) Eliciting human judgment for prediction algorithms. *Management Science* 67(4):2314–2325.

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. *Available at SSRN 3937554* .

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

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

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

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

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.

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.

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.

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

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

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

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

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

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

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

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.

Snyder C, Keppler S, Leider S (2022) Algorithm reliance under pressure: The effect of customer load on service workers. *Available at SSRN 4066823* .

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

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

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

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

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

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

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

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

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

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

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

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.[6] 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

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

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

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 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

[6] 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.

goal is to help human users improve their performance, we restrict the training dataset to the bottom 25% performing human users. In addition, we apply two post-processing steps to the set of candidate tips. First, we eliminate tips that apply in less than 10% of the (featurized) states that occur in the human dataset. This step eliminates high-variance tips that may have large benefit, but are useful only a small fraction of the time; we omit such tips since our estimates of their quality tend to have very high variance. Second, we eliminate tips that disagree with the expert policy more than 50% of the time—i.e., for a tip (ψ, a), we have $\psi(s) = 1$ and $a \neq \pi^*(s)$ for more than 50% of state-action pairs in the human dataset. This step 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.

Appendix B: 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 A.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.

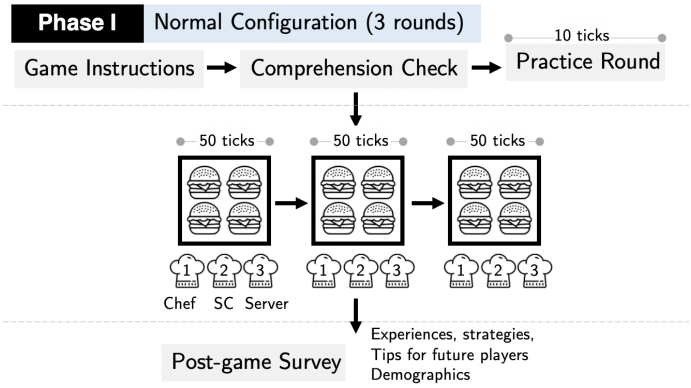
	Chopping meat	Cooking burger	Plating burger
Chef	1	4	6
Souf-chef	2	8	2
Server	3	12	1

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

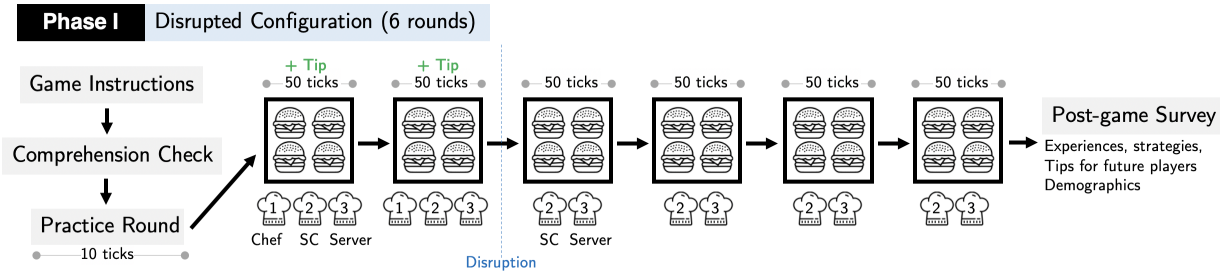
After completing all scenarios, we give each participant a post-game survey regarding their experience with the game. Each participant receives a participation fee of \$0.10 for each round they complete; they also receive a performance-based bonus based on the number of time steps taken to complete each 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



(a) Normal configuration



(b) Disrupted configuration

Figure A.1 Study flow for Phase I.

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.

B.2. Inferred tips

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

For the normal configuration, 183 participants[7] successfully completed the game. The top three tips inferred from each of the sources are reported in Table A.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.

[7] 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 A.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%)

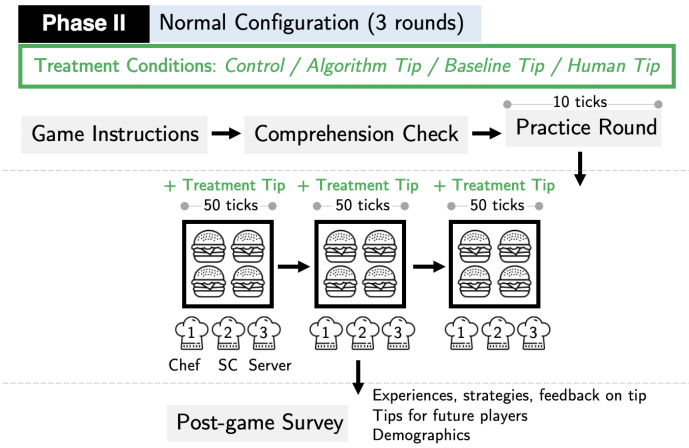
Table A.3 Top three tips inferred from different sources for the disrupted configuration.

For the disrupted configuration, 172 participants[8] successfully completed the game. Table A.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.

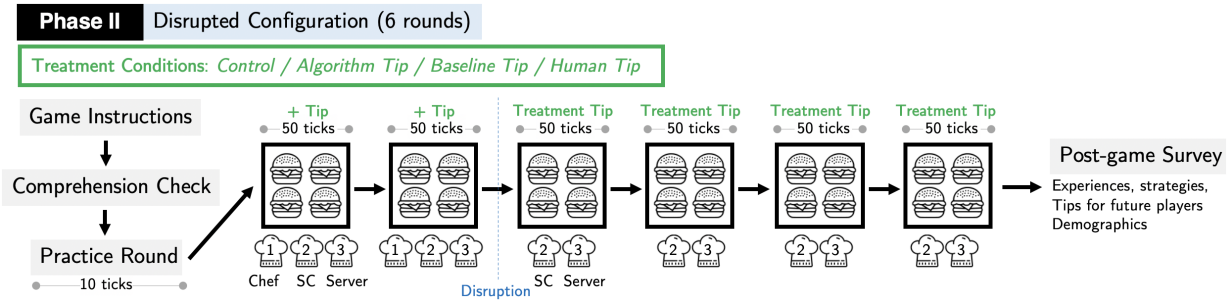
B.3. Phase II

Next, we evaluate the effectiveness of each of the inferred 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.

[8] 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 A.2 Study flow for Phase II.

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., 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 \$2,149.7 (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

Table A.4 exhibits the demographic and gameplay information of the participants 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 A.5 and A.6 show the average performance within each round across two phases and treatment conditions for normal and disrupted configurations, respectively.

	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 A.4 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

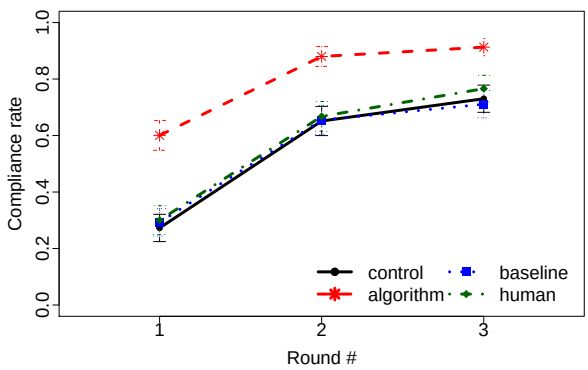
Table A.5 Average performance by treatment condition and round (normal configuration).

	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

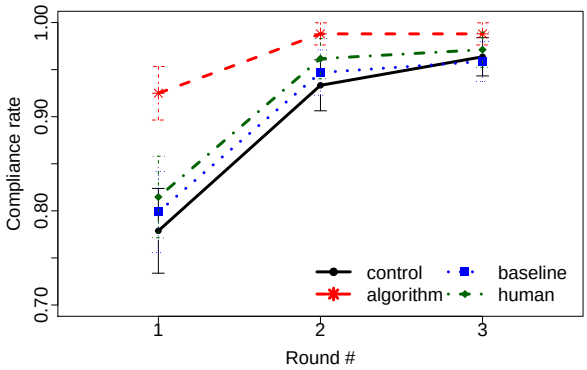
Table A.6 Average performance by treatment condition and round (disrupted configuration).

C.1. Learning beyond tips under the normal configuration

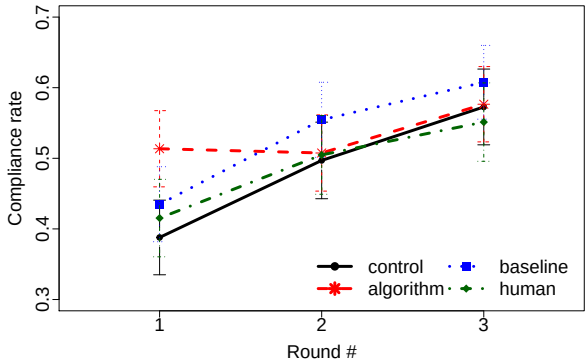
Compared to the disrupted configuration, the normal configuration is much easier. We find that participants across all conditions cross-comply with all other tips: not to assign plating to the chef (Figure A.3a), strategically leave some virtual workers idle (Figure A.3b), and let the chef chop only once (Figure A.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 statistically different among the arms (see Figure A.3d). This result suggests that participants can uncover this unshown rule by themselves across all conditions.



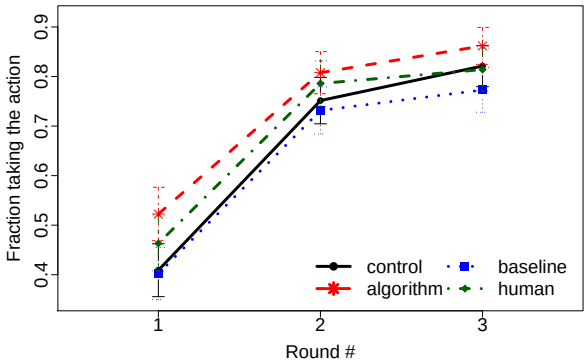
(a) Algorithm Tip: "Chef shouldn't plate"



(b) Human Tip: "Strategically leave some workers idle"



(c) Baseline Tip: "Chef should chop once"



(d) Unshown Tip: "Server shouldn't cook"

Figure A.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.2. Screenshots from the kitchen-management game

Finally, we provide screenshots to illustrate our experimental design. Figures A.4 & A.5 show the introduction to the task shown to participants explaining various concepts in the game. Figures A.6 & A.7 show instructions for the fully-staffed and understaffed scenarios, respectively, shown to participants. Finally, Figure A.8 shows the payment information shown to the participants.

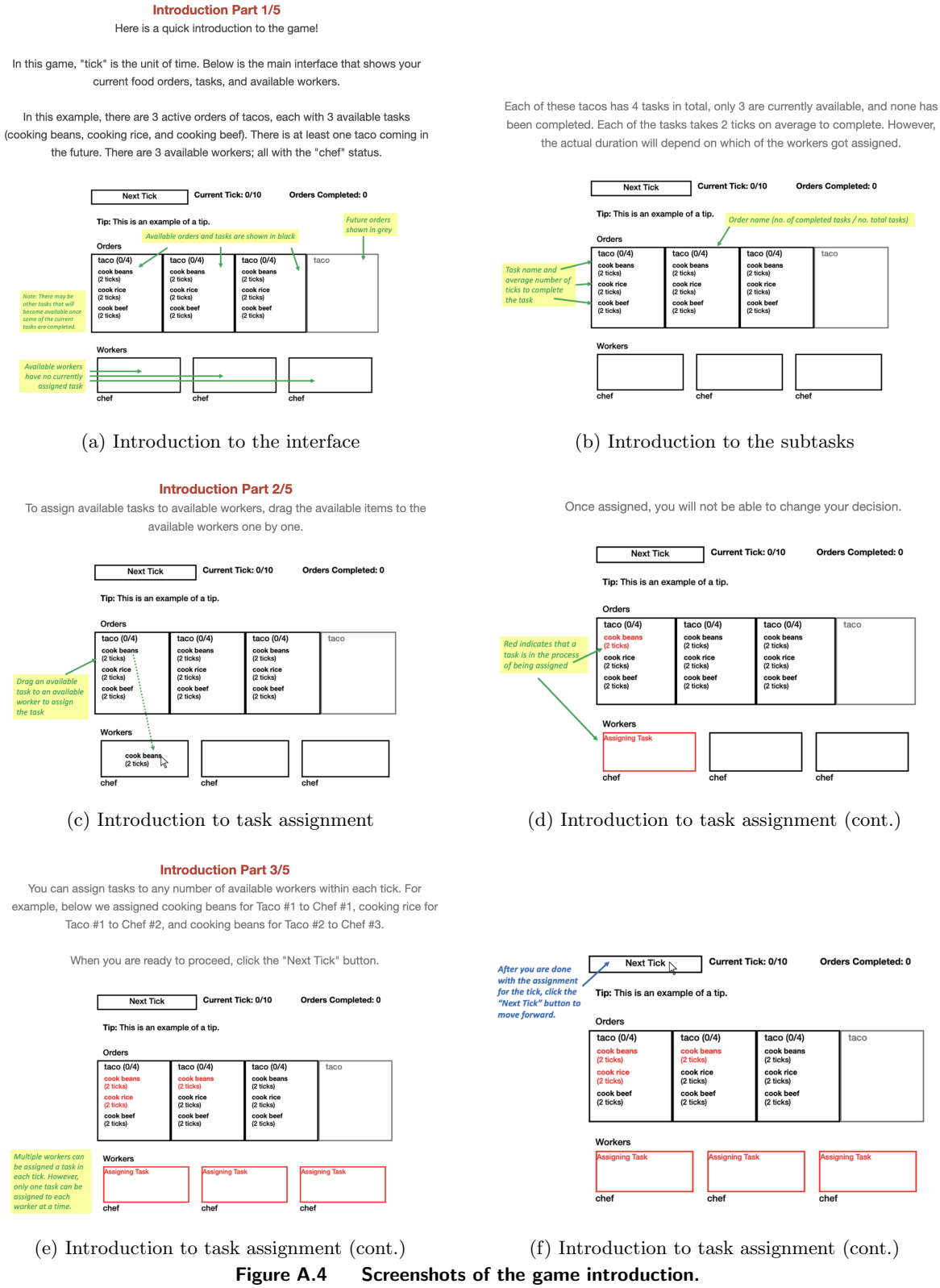
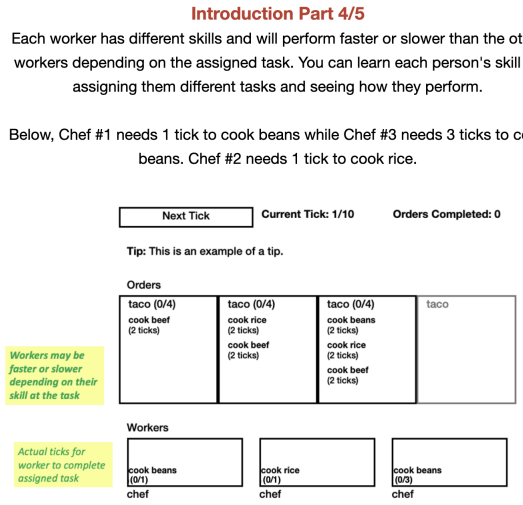
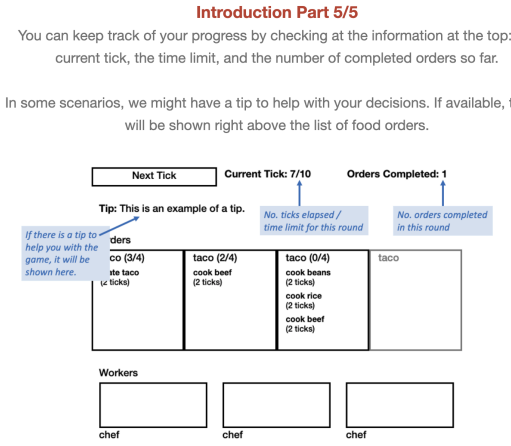


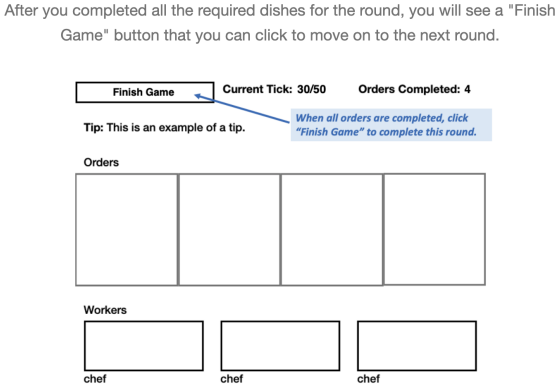
Figure A.4 Screenshots of the game introduction.



(a) Introduction to workers' skill levels

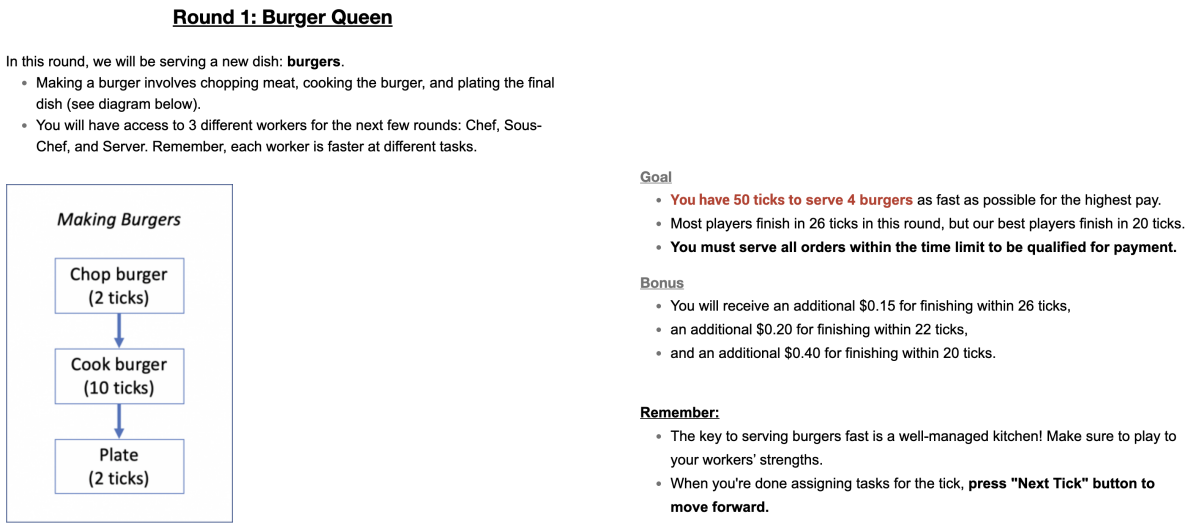


(b) Introduction to the tip



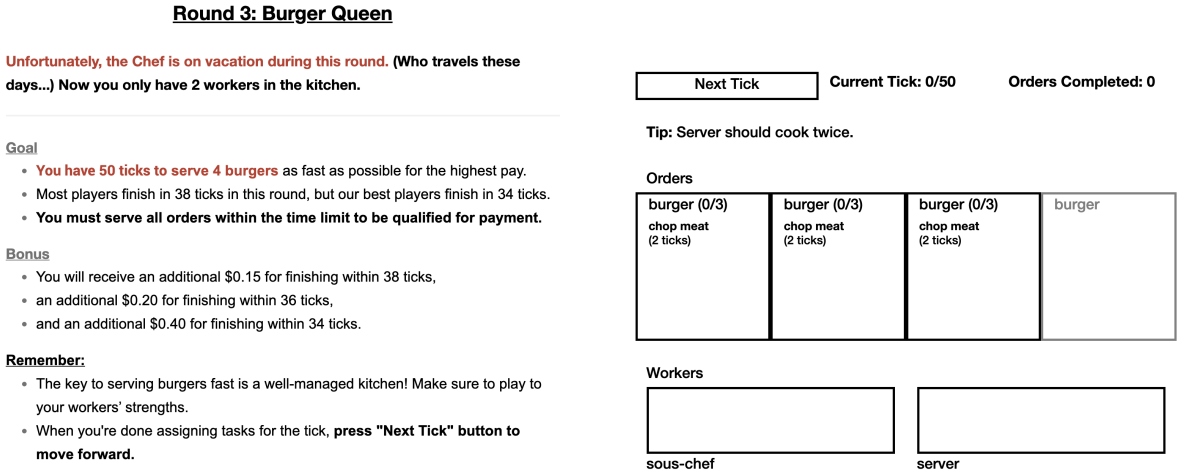
(c) Introduction to round completion

Figure A.5 Screenshots of the game introduction (continued).



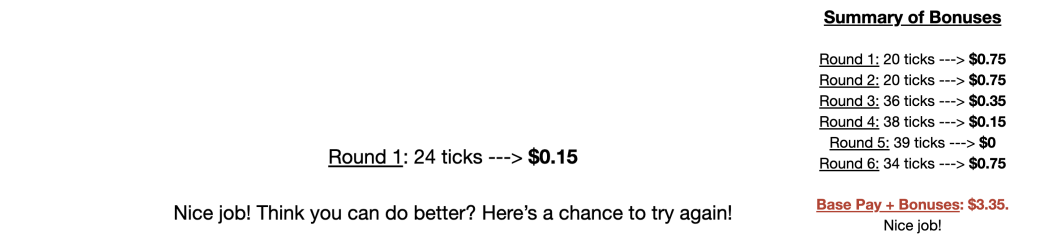
(a) Burger's subtasks and available workers (b) Goal, incentives, and reminder

Figure A.6 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 A.7 Screenshots of the instructions for the understaffed scenario.



(a) Individual round pay information (b) Summary of total pay

Figure A.8 Screenshots of the pay information.