Strategic Communication and Algorithmic Advice

Emilio Calvano* Clemens Possnig† Juha Tolvanen‡

March 16, 2024

Abstract

We study a model of communication in which a better-informed sender *learns* to communicate with a receiver who takes an action that affects the welfare of both. Specifically, we model the sender as a machine-learning-based algorithmic recommendation system and the receiver as a rational, best-responding agent that understands how the algorithm works. The results demonstrate robust communication, which either emerges from scratch (i.e., originating from babbling where no common language initially exists) or persists when initialized. We show that the sender's learning hinders communication, limiting the extent of information transmission even when the algorithm's designer's and the receiver's preferences are aligned. We then show that when the two are not aligned, there is a robust pattern where the algorithm plays a cut-off strategy pooling messages when its private information suggests actions in the direction of its preference bias while sending mostly separate signals otherwise.

Keywords: Algorithmic advice, recommendation systems, cheap talk, strategic communication.
Jel codes: D21, D43, D83, L12, L13.

*Università LUISS Guido Carli: emilio.calvano@gmail.com.
†University of Waterloo: cjmpossnig@gmail.com.
‡Università di Roma 'Tor Vergata': juha.k.tolvanen@gmail.com.

1 Introduction

Suppose a learning algorithm conveys messages to a Bayesian decision-maker, whose actions impact both parties' welfare. The algorithm is better informed. However, the parties' objectives can differ, leading to an interaction akin to a strategic communication game. Over time, the algorithm autonomously refines what messages to convey for different states based on feedback from the decision-maker's actions. This paper asks if meaningful communication can emerge and endure in this setting.

This question holds significant economic relevance. Algorithmic advice from AI or machine learning tools has become increasingly important to human decision-makers in a diverse and broadening array of domains from the judiciary system (Ludwig and Mullainathan, 2021) and employee recruitment processes (Hoffman et al., 2018) to recommending pricing for accommodation services (e.g., Huang, 2022; Garcia et al., 2022), airline tickets, and recommendations on platforms such as Amazon, Spotify, and Netflix. These services must not only gather relevant information but also present it to users in a credible and profitable manner. A number of authors have recently argued that the algorithm's designer's and the end user's incentives are often misaligned (e.g. Cowgill and Stevenson, 2020; Huang, 2022; Garcia et al., 2023). For example, Airbnb (an online platform connecting hosts with tenants), offers personalized pricing suggestions to hosts based on a variety of factors. Because Airbnb earns revenue solely when hosts rent their properties, it should ideally prioritize lower prices than those optimal for hosts, who incur variable costs, at any demand level, thus posing a communication challenge. To customize real-time price recommendations for millions of hosts with diverse objectives influenced by individual time varying variable costs, a straightforward and scalable solution involves utilizing a learning algorithm.

But how challenging is it to learn to communicate? This is a difficult question to answer both in theory and in practice. Communicating in strategic settings is a well-known hard coordination problem with the potential for communication breakdowns, often referred to as 'babbling' outcomes. Furthermore, economic theory's descriptive power is

notably limited, primarily due to the severe multiplicity of equilibria, leaving the extent of information transmission in practice unanswered. Finally, the process of learning to communicate is inherently difficult. The fact that the receiver's behavior *adapts* to the sender's policy makes the problem non-stationary, creating a well-known learning challenge. Beforehand, it is unclear whether we should expect any communication to emerge.

Interacting learning algorithms create stochastic dynamic systems of such complexity that analytical results appear out of reach. To make progress, we adopt an experimental approach. We let an AI-powered algorithmic advisor, call it sender, repeatedly interact with a 'Bayesian' receiver in synthetic, computer-simulated interactions. We do so in a 'cheap talk' strategic context, as in Crawford and Sobel (1982), where the sender can send any message without any costs to the receiver. By 'Bayesian,' we mean that the receiver accurately extracts information from messages and chooses the action that maximizes their period's payoff.[1] We focus on the outcome of the learning process rather than the learning rate. Therefore, we focus on relatively simple and transparent learning algorithms intentionally designed for slow, unsupervised learning. These algorithms refine their communication policy through active experimentation. They do so by testing alternative messages for given states, reinforcing those messages that prompt the receiver to choose a more favorable action. Our simulations allow the algorithm to explore extensively and interact repeatedly until its behavior stabilizes.

The results demonstrate robust communication, which either emerges from scratch (i.e., originating from babbling where no common language initially exists) or persists when initialized. As expected, the extent of information transmission decreases with the wedge between the receivers and the sender's preferences. But how much information gets transmitted? We measure the informative content of the learned strategies in two ways. First, we document a significant level of information transmission quantified by the percentage reduction in the average entropy of the state resulting from message

[1] We leave for future research the case in which the receiver foresees the impact of its current choices on the algorithm's future behavior.

observation across a wide range of parameters and initializations and averaged over a large number of independent experiments. Our algorithmic agent systematically learns to communicate. Next, we look at how much information is transmitted in practice relative to the maximum amount that can be transmitted in theory. To address this, we numerically determine the most informative Nash equilibrium of the underlying game finding that our experiments achieve a comparable level of transmission.

We provide a more in-depth analysis studying the anatomy of the communication policy learned and relating it to the underlying algorithm's design. We first consider the simplest case in which preferences are perfectly aligned. We demonstrate that learning hinders communication, bounding the extent of information transmission away from full revelation. To build intuition, consider a simple exploration strategy where the algorithm with some constant small probability sends a message uniformly at random. Then, even if it is truthfully sending information when not exploring, the receiver knows that any message can also result from exploration. This uniform noise implies that the receiver's posterior belief about the algorithm's signal after any message is pulled towards the prior mean of the algorithm's signal distribution. Consequently, the receiver will not necessarily match its actions to the message sent by the algorithm but takes actions closer to the prior mean. The algorithm will learn this bias in actions and will best-respond to it by *exaggerating* its information away from the prior mean. With a compact state space, this will result in the algorithm eventually playing a partitioning strategy where it truthfully communicates only information close to the prior mean and otherwise creates pools of nearby signals, distorting the information transmission. This type of pooling will happen as long as the state space of the algorithm is fine enough relative to its exploration rate. The size of this distortion increases in the extent of exploration. Furthermore, full revelation is not restored even if the algorithm's exploration rate vanished over time: distortions are pinned down by the initial exploration rate.

When the algorithm's designer and the receiver have different preferences as with Airbnb and its hosts, no stable language is learned - the algorithm introduces constant local perturbations in the meaning of messages. To build intuition, notice that if the

algorithm's designer prefers higher actions than the receiver, then this bias goes against the algorithm's incentives to exaggerate low signals because of the mechanism above while increasing its incentives to exaggerate high ones. As a consequence, the algorithm will learn to distort its messages especially when its information is strongly of the same direction as its preference bias. In the Airbnb example, its algorithm would recommend a single low price whenever its information suggest that a price from the low end of possible prices is optimal to the host. Whenever the sender's private information suggests actions that are not in the direction of its bias, the algorithm will cycle over available messages in a way that in any given moment leads to almost complete revelation of the sender's information. In the case of Airbnb this would imply fully revealing or only partially obfuscating recommendations for high prices. We show that this pattern of pooling at the top (or bottom) and near-revelation at the bottom (or top, respectively) is robust to different ways the algorithm can experiment, different "speed" at which the receiver learns the algorithm's current strategy, and lying costs that increase in the distance from the truth (as in Kartik, 2009).

We then turn to analyzing the welfare implications of the pattern we observe. We first show that the behavior learned by the algorithm yields a higher expected welfare to both the sender and the receiver than the most informative equilibrium of the cheap talk game. This is due to the high amount of information revelation implied by the threshold strategy above. In fact, the pattern we find is similar to the equilibrium of the cheap talk game when the receiver can commit ex-ante to how it will map messages to actions as in Alonso and Matouschek (2007). However, the threshold at which the sender pools its messages is determined differentially and hence the receiver's welfare is always lower than in Alonso and Matouschek (2007).

In a paper developed concurrently with this one, Condorelli and Furlan (2023) consider a cheap-talk setting in which both sides of the interaction employ reinforcement learners. The authors find that they can replicate policies of the ex-ante optimal Nash equilibrium of Crawford and Sobel (1982)'s game. Due to the fact that we consider a more sophisticated receiver, our study differs both in the questions we ask, as well as in

the results we obtain.

2 The Model

We consider Crawford and Sobel (1982)'s quadratic payoff setting for our interaction between algorithm and rational agent. There is a sender S, who observes a state variable X which is uniformly distributed on $[0, 1]$. Given a realized state x, the sender's payoff is $U^S(y, x, b) = -(y - x - b)^2$, where y represents the receiver R's action, and $b > 0$ represents the sender's bias. The receiver's payoff given state realization x and action y is $U^R(y, x) = -(y - x)^2$. After observing the state x, S sends a message $m \in M$ to the receiver, who considers the message, updates their belief over the state X, and then plays an action $y \in \mathbb{R}$. The sender has a messaging strategy $\mu(m \mid x)$ which specifies the likelihood of sending message $m \in M$ given state x. The receiver, in turn, has an action strategy $y(m, \mu)$ specifying an action for every possible message sent by S, given S's policy μ.

2.1 Algorithmic Cheap Talk

This is where our study departs from the classical cheap talk setting among rational agents. We consider an interaction in which the sender is not fully rational, but a reinforcement learning algorithm. This algorithm updates its behavior over time, as payoffs from an interaction with a rational receiver are accrued.

Our baseline specification[2] considers a tabular Q-learner as the sender S. Q-learning is a much studied reinforcement learning method, well regarded due to its simplicity and minimal information requirements. For a detailed introduction, consider Sutton and Barto (2018). Q-learning was introduced to find optimal policies for single-agent Markov decision problems (MDPs), when little to no information about payoff functions and state transitions is known to the algorithm's designer. This learning rule has become the focus of attention also of researchers interested in multi-agent learning, again due

[2]We have considered alternative specifications including Q learning with softmax-policies, and contextual exp-3 learning. Qualitatively, the results are analogous to those presented in our baseline setting.

to its minimal modeling requirements, ease of setup, and also since many more involved reinforcement learning schemes retain, at the very least, some of the conceptual intuitions introduced by Q-learning.

Tabular Q-learners require games of finite action and state spaces. We therefore discretize our setting, so that the state $X \in I_K = \{x^{(1)}, \ldots, x^{(k)}, \ldots, x^{(K)}\} \subset [0,1]$, where $x^{(k)}$ are spaced uniformly on the interval $[0,1]$. We fix $x^{(1)} = 0, x^{(K)} = 1$, so that $x^{(k)} - x^{k-1} = \frac{1}{K-1}$ for all $1 < k \leq K$. Correspondingly, the Q-learner can generate messages $m \in M_K = I_K$. We consider message spaces of the same cardinality as the state space. As will become clear shortly, being able send more messages than K will put the algorithm at no advantage.

A Q-learner learns a Q-value function. This function $Q : I_K \times M_K \mapsto \mathbb{R}$ is meant to allow the learner to find an optimal policy mapping from I_K to M_K. Essentially, this function is meant to estimate the expected payoff for any given state and action, allowing to find optimal actions upon realization of a state. The estimator is formed as a simple weighted-averaging rule, and only updated upon realizing a given state-action pair.

We refer to realizations at a period t using subscripts. Upon a realization of state x_t, message m_t, and payoff u_t, the Q-function is updated the following way:

$$Q_{t+1}(x,m) = \begin{cases} Q_t(x,m) + \alpha\Big[u_t + \delta \max_{m' \in M_K} Q_t(x_{t+1}, m') - Q_t(x,m)\Big] & \text{if } x = x_t,\ m = m_t \\ Q_t(x,m) & \text{otherwise} \end{cases}, \tag{1}$$

where $\alpha > 0$ is known as the 'learning rate', and $\delta \in [0,1]$ is a discount factor. Our framework carries no time-dependence among state realizations, as states are sampled every period i.i.d. uniformly from I_K. In our baseline setting, we therefore set $\delta = 0$. Settings of $\delta > 0$ do not affect our results.

Notice that Q-learning does not specify a policy. Convergence results on Q_t give requirements on how often actions are selected over time, but the updating rule is ag-

nostic about how actions a_t are sampled in every period. In our baseline specification, we consider one of the most common sampling rules: ε-greedy. Specifically, given a non-increasing sequence of $\varepsilon_t \in [0,1]$ for every period t, the sender sends the currently believed optimal message (the 'greedy' action), $\arg\max_{m'} Q_t(x_t, m')$, with probability $1-\varepsilon_t$. With probability ε_t, m is sampled uniformly from M_K. Formally this means that if for every period t we define $m_t^*(x^{(k)}) = \arg\max_{m'} Q_t(x^{(k)}, m')$, then the algorithmic sender's policy $\mu_t(m \mid x)$ can be defined as follows:

$$\mu_t(m^{(k)} \mid x_t) = \begin{cases} (1-\varepsilon_t) + \frac{\varepsilon_t}{K} & \text{if } m^{(k)} = m_t^*(x_t) \\ \frac{\varepsilon_t}{K} & \text{otherwise.} \end{cases}. \tag{2}$$

In our baseline scenario, it will help to fix ideas by considering $\varepsilon_t = \varepsilon \in (0,1)$ fixed for all periods. The receiver R is what we consider a 'sophisticated' agent: at every period t, R knows the sender's policy μ_t. This setting can be seen as an extreme version of a receiver who knows that their recommendations are being generated algorithmically and learns the algorithm's strategy faster than the time between the updates to the algorithm's policy. Upon receiving a message m_t, the receiver forms a belief about the true state of the world.

As for the receiver's strategy, we write $y_t(m) = y(m, \mu_t)$ if there is no potential for confusion. The quadratic payoff function implies that, given μ_t and the sender's message m, the receiver's best response is to set $y_t(m)$ to be the expectation of x conditional on (μ_t, m). First, let $\rho(x)$ be the prior probability of $x \in I_K$ being realized. Then, define $\rho_t(x, m)$ as the receiver's belief over x conditional on (μ_t, m):

$$\rho_t(x, m) = \mathbf{P}\big[X = x \mid \mu_t, m\big] = \frac{\rho(x)\mu_t(m \mid x)}{\sum_{k=1}^{K} \rho(x^{(k)})\mu_t(m \mid x^{(k)})}.$$

Then,

$$y_t(m) = \mathbb{E}\big[x \mid \mu_t, m\big] = \sum_{k=1}^{K} x^{(k)} \rho_t(x^{(k)}, m). \tag{3}$$

Note that $\mathbf{P}[m \mid \mu_t] = \sum_{k=1}^{K} \rho(x^{(k)}) \mu_t(m \mid x^{(k)})$. For any $m \in M_K$, let $S_t^*(m) = \{x \in I_K : m = m_t^*(x)\}$, the set of states taking m as the current maximizer of the Q-function, and let $K_t^*(m) = |S_t^*(m)|$. Then we can also write

$$\begin{aligned} y_t(m) &= \mathbf{P}[\text{exploitation } \mid \mu_t, m]\mathbb{E}\Big[x\Big|\ x \in S_t^*(m)\Big] + \mathbf{P}[\text{exploration } \mid \mu_t, m]\mathbb{E}[x] \\ &= \frac{\sum_{x \in S_t^*(m)} \rho(x)(1-\varepsilon)}{\mathbf{P}[m \mid \mu_t]} \mathbb{E}\Big[x\Big|\ x \in S_t^*(m)\Big] + \frac{\sum_{x \in I_K} \rho(x) \frac{\varepsilon}{K}}{\mathbf{P}[m \mid \mu_t]} \mathbb{E}[x]. \end{aligned}$$

The receiver's optimal action can thus be seen as a convex combination of the expected state conditional on the sender exploiting (i.e. choosing the 'greedy' action which maximizes Q_t), and exploring, in which case the conditional expected state corresponds to the prior.

We consider a uniform distribution over states, so we can simplify:

$$y_t(m) = \frac{K_t^*(m)(1-\varepsilon)}{K_t^*(m)(1-\varepsilon) + \varepsilon} \mathbb{E}\Big[x\Big|\ x \in S_t^*(m)\Big] + \frac{\varepsilon}{K_t^*(m)(1-\varepsilon) + \varepsilon} \mathbb{E}[x]. \tag{4}$$

To make this learning dynamic more concrete, in the example of Airbnb's price recommendations, a host may have an idea of what prices are optimal for the platform to set given the demand conditions that the platform observes. They may also have an idea how Airbnb is using its superior information in its recommendations to try to influence the hosts's pricing decisions. However, the only thing we really require is that the receiver has enough interactions with Airbnb's recommendations to have correct beliefs about how Airbnb is currently mapping its private information to recommendations. We also require that the host understands that Airbnb is constantly updating its recommendation

algorithm and exploration of different communication strategies is part of this process. Based on these ingredients the host then forms a posterior belief over what different price recommendations can mean and selects final prices for their apartment optimally given these beliefs.

3 Results

In the following, we discuss simulation results and related theoretical insights. Our baseline setup contains 960 simulation runs, consisting of $T = 10$ Million periods each. We set $\alpha = 0.1, \varepsilon = 0.15$, and number of states and messages $K = 15$. We begin by considering the no-bias case ($b = 0$).

Notice that every period, the receiver changes how they react to the sender's messages, based on the messaging policy. This makes for a highly non-stationary learning problem for the sender. Furthermore in this model, messages have no intrinsic meaning known to the sender. Can we expect any kind of information transmission to be learned at all? It turns out that, indeed, in all our experiments, the sender learns to communicate to some extent. We consider the long run policy μ_∞ learned by the sender, and judge it by the quality of the information it reveals about the true state. To do so, we follow the approach by Condorelli and Furlan (2023) and consider the mutual information between the state and generated messaging policy, normalized by the entropy of the state. This is a useful measure as it can be interpreted as the percentage reduction in the entropy of the state resulting from observing a message. In our notation,

$$I = \left(\sum_{x \in X_K} \rho(x) \log \left(\frac{1}{\rho(x)} \right) \right)^{-1} \sum_{x \in X_K} \sum_{m \in M_K} \mu_\infty(m \mid x) \log \left(\frac{\mu_\infty(m \mid x)}{\sum_{x' \in X_K} \mu_\infty(m \mid x') \rho(x')} \right). \tag{5}$$

The heatmap in figure 1 shows the information I of the final policy, averaged over simulation experiments, for a grid of differing exploration rates and biases (ε, b):

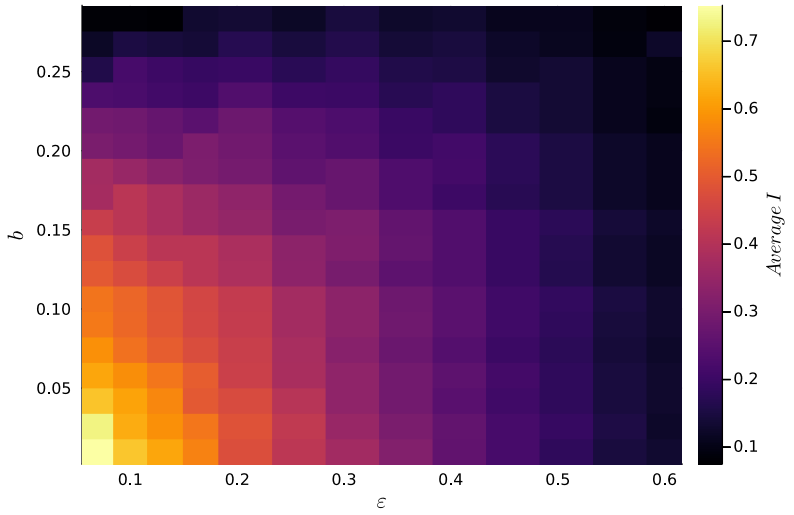

Figure 1: Average I over 960 simulation experiments per grid point, for fixed exploration rate ε. The information transmitted is substantial for smaller values of the parameters, and only reaches 0 for extreme values of b, under which only babbling survives.

While it is clear from this figure that our learning agent can learn to communicate, it remains unclear what kinds of policies might survive in the long run, and whether some of those policies can be understood as equilibria of an underlying game. Furthermore, it is not obvious whether the learning process of the sender converges, and if it does, what behaviors it might converge to. Interestingly, the case of $b = 0$ is special here in that all our simulation experiments feature convergence of the sender's policy μ_t. As will be discussed in later sections, the policies converge to Nash equilibria of a noisy version of Crawford and Sobel (1982)'s cheap talk game. We observe that the sender consistently learns to play policies that are close in informativeness to the best possible equilibria of that noisy cheap talk game, as will be discussed. For $b > 0$ on the other hand, our experiments lead to cycles, with a common feature of top-pooling as will be discussed later on.

In the following sections, we delve into the anatomy of the learning process and the policies learned in the long run in this interaction.

3.1 No Bias: The Pure Communication Problem

We start by considering what happens when the sender's and receiver's payoff functions turn out to be fully aligned. We consider this pure communication problem from the two extreme perspectives a sender might take when initializing their algorithm. Recall that S does not initially know the incentives of the receiver. Furthermore, in some applications, there might not even be a natural language in which to communicate their private information. Given all of this uncertainty, there are two extreme stances a designer of an algorithm might take when initializing their learning algorithm: zero revelation ('babbling'), or full revelation (when messages have some expected meaning). As we show below, neither option is robust to learning: a sender who initially fully reveals the state, will eventually play a less informative strategy, that still reveals some amount of information. On the other hand, babbling is not robust to learning either.

3.1.1 Full Revelation

We first consider the case in which the algorithm is initialized to fully reveal the state. Note that there are multiple options when initializing Q values that lead to a fully revealing policy, since any messaging strategy that is one-to-one will be fully revealing. Generically, initializing Q values randomly from the payoff range $[-1, 0]$ will lead to a fully revealing policy, as the probability of having two identical Q values is zero. Alternatively, one can consider what we call a 'diagonal' initial policy, where values are set in any such way that the optimal policy satisfies $m_0^*(x) = x$ for all $x \in X_K$. For example, a naive initialization would set $Q_0(m, x) = 0$ if $m = x$, and draw uniformly randomly from $[-1, 0]$ for all other pairs. Here we discuss the case of diagonal initial policy, since intuitions and results will qualitatively carry over to alternative initializations of full revelation, which are relegated to Online Appendix B.

When exploration $\varepsilon_t = \varepsilon$ remains fixed over time, we focus attention on the exploitative part of a final policy, $m_\infty^*(x)$ for $x \in I_K$, which we sometimes refer to as 'argmax-policy'. When there is little potential for confusion, we will refer to $m^*(x)$ as 'policy', understanding that the actual policy used by the algorithm to realize actions is

in fact μ.

Figure 2 shows the mean final policy, averaged over simulations, when the bias $b = 0$. On the x-axis, we show the index k of any given state $x^{(k)}$, and on the y-axis, the message index associated with each state according to the mean policy. Notice that, since messages are not payoff-relevant in our baseline scenario, a fully-revealing message for a given state $x^{(k)}$ could be any value in M_K, as long as it is assigned to be fully-revealing through policy μ. For ease of exposition, in each simulation run, messages of the final policy have been re-labeled to be monotone in state index.

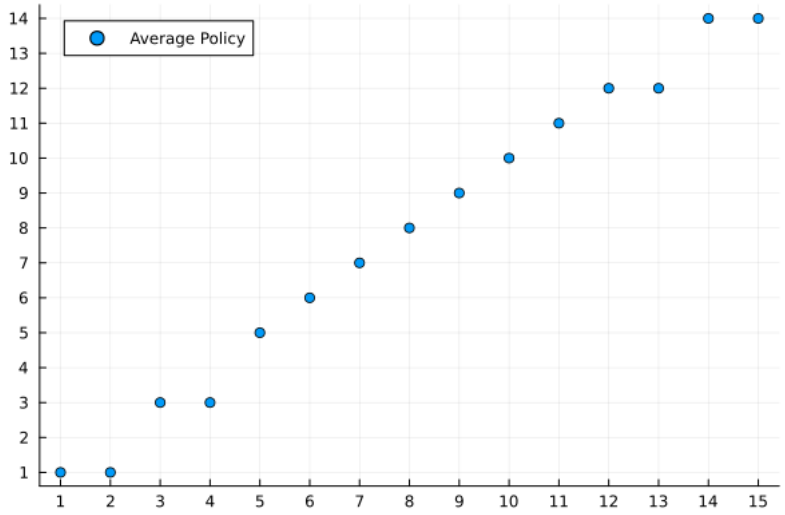

Figure 2: Long run policy averaged over simulation runs. We observe 3 different final policies, 99% of simulation experiments coincide with the above. The remainder differs from the above only in the message assignment of one state.

All our simulation runs converge in the sense that the argmax-policy $m_t^*(\cdot)$ of the algorithm remains constant for 500,000 periods. In fact, convergence occurs to policy profiles that are Nash equilibria of a game related to our cheap talk game, with the added difficulty that the sender's message gets lost with a probability ε:

Definition 1. *Let $\Gamma_K^{\varepsilon}(b)$ be the K-discrete, 'noisy cheap talk game'. In this game, senders and receivers are rational, with payoffs $U^S(y, x, b), U^R(y, x)$ states and messages are discrete $X_K = M_K \subset [0, 1]$, and importantly, the sender does not have full control*

over their messaging channel. The sender chooses a messaging policy $m^* : X_K \mapsto M_K$. *With probability* $1-\varepsilon$, *messages are implemented according to* m^*. *With probability* ε, *a message is instead generated uniformly from* M_K. *The receiver is unconstrained and best responds to how messages are generated, .*

Note that $\Gamma_K^0(b)$ coincides with our discretized cheap talk game. For general $\varepsilon > 0$, this noisy cheap talk game is a discretized version of the game analysed in Blume et al. (2007). The next result specifies the connection between long-run policies of our algorithm and Nash equilibria of $\Gamma_K^\varepsilon(0)$:

Proposition 1. *Suppose exploration satisfies* $\sum_{t\geq 0} \varepsilon_t = \infty$, *and define* $\bar{\varepsilon} = \lim_{t\to\infty} \varepsilon_t$. *Suppose the sender's argmax-policy* m_t^* *and receiver best response* $y_t(m)$ *converge to a point* $\big(\bar{m}, \bar{y}(m)\big) \in M_K^K \times \mathbb{R}^K$. *Then* $\big(\bar{m}, \bar{y}(m)\big)$ *is a Nash equilibrium of* $\Gamma_K^{\bar{\varepsilon}}(b)$.

Proof. All proofs can be found in the appendix. □

Proposition 1 suggests that the absence of full revelation in Figure 2 indicates that full revelation is *not* a Nash equilibrium of $\Gamma_K^\varepsilon(0)$. To see why this is the case, recall from (4) that the receiver's optimal action $y_t(m)$ is always biased towards the prior $\mathbb{E}[X]$ by ε. Now consider the following example: suppose that at some period t, we have $m_t^*(x^{(1)}) = m^{(1)}$, and $m_t^*(x^{(2)}) = m^{(2)}$, with $K_t^*(m^{(1)}) = K_t^*(m^{(2)}) = 1$, i.e. $m^{(1)}, m^{(2)}$ each are only considered optimal for one state. In that case, we can calculate when S will receive higher payoff streams from "deviating" to sending $m^{(1)}$ if $x^{(2)}$ is realized instead of sending $m^{(2)}$:

$$
\begin{aligned}
U^S(y_t(m^{(2)}), x^{(2)}, 0) &= -\left(-\frac{\varepsilon}{K_t^*(m)(1-\varepsilon)+\varepsilon} x^{(2)} + \frac{\varepsilon}{K_t^*(m)(1-\varepsilon)+\varepsilon}\frac{1}{2} \right)^2 \\
&= -\left(\varepsilon\frac{1}{2} - \varepsilon x^{(2)} \right)^2, \\
U^S(y_t(m^{(1)}), x^{(2)}, 0) &= -\left((1-\varepsilon)x^{(1)} + \varepsilon\frac{1}{2} - x^{(2)} \right)^2 \\
&= -\left(\varepsilon\frac{1}{2} - \varepsilon x^{(2)} - (1-\varepsilon)(x^{(2)} - x^{(1)}) \right)^2
\end{aligned}
$$

Recall that we have $x^{(1)} = 0$, $x^{(2)} - x^{(1)} = \frac{1}{K-1}$, so that we arrive at the following:

$$
U^S(y_t(m^{(2)}), x^{(2)}, 0) < U^S(y_t(m^{(1)}), x^{(2)}, 0) \Leftrightarrow \left|\varepsilon\frac{1}{2} - \varepsilon\frac{1}{K-1}\right| > \left|\varepsilon\frac{1}{2} - \frac{1}{K-1}\right|. \quad (6)
$$

We have $K - 1 > 0.5$, so the argument of the left hand side must be positive. We have two cases:

1.) $\varepsilon\frac{1}{2} - \frac{1}{K-1} > 0$.

Then, (6) simplifies to $(1-\varepsilon)\frac{1}{K-1} > 0$, which always holds. For this case to be true, we get the condition $\varepsilon > \frac{2}{K-1}$.

2.) $\varepsilon\frac{1}{2} - \frac{1}{K-1} < 0$.

Then, (6) requires $\varepsilon > \frac{1-\varepsilon}{K-1}$, so that $\varepsilon > \frac{1}{K}$ must hold. Thus here, one must have $\frac{1}{K} < \varepsilon < \frac{2}{K-1}$.

In the above example, we attain two possible relationships between exploration probability and discreteness of the state-space that are sufficient for a Q-learner to never learn to separate states $x^{(1)}, x^{(2)}$. Note that case (1) can be true for arbitrarily fine grids, and thus this result is not implied by our considering a discretized problem. Note also that the simulations generating Figure 2 are done under the baseline parameters

$\varepsilon = 0.15, K = 15$. We get that here $\frac{2}{K-1} \sim 0.142 < \varepsilon$, so that case (1) is satisfied.

As mentioned above, the algorithm learns to pool states, because the receiver knows that any message can come from exploration and hence biases its actions towards the prior mean. Therefore pooling two states yields higher gains for states that are far away from the prior mean. This explains why in our simulations the pools form both for top and bottom states but the algorithm still fully reveals the state close to the middle of the state space. Notice further that the algorithm pools states by inflating language by associating a lower state to a message that was previously sent only at a more extreme state. Consequently, the receiver will learn to best respond to this message by taking actions that are even closer to the prior mean than before. This, in turn, may lead to the algorithm finding it optimal to pool even less extreme states to the same pool. In other words, information distortions caused by the baseline exploration can be magnified by further strategic interaction between the algorithm and the receiver.

Given that the policies converge, it seems wasteful for the algorithm to keep exploring at a constant rate ε. One might expect that a decaying exploration rate should counteract the forces described above, and lead to full revelation. To address this question, we run the following experiment: We first run the simulation under a given fixed ε_0 until convergence. Once convergence is achieved, we let ε_t decay to zero while the algorithm continues to run. We refer to this as the 'ε-decay' setting. We let $\varepsilon_t = \varepsilon_0 exp(-t\beta)$, where $\beta = 10^{-5}$ in our baseline setting, a slow decay rate ensuring $\sum_{t\geq 0} \varepsilon_t = \infty$.[3] To match the previous simulation experiment, we initialize $\varepsilon_0 = 0.15$ and plot the results in Figure 3 below.

[3] See e.g. Calvano et al. (2020).

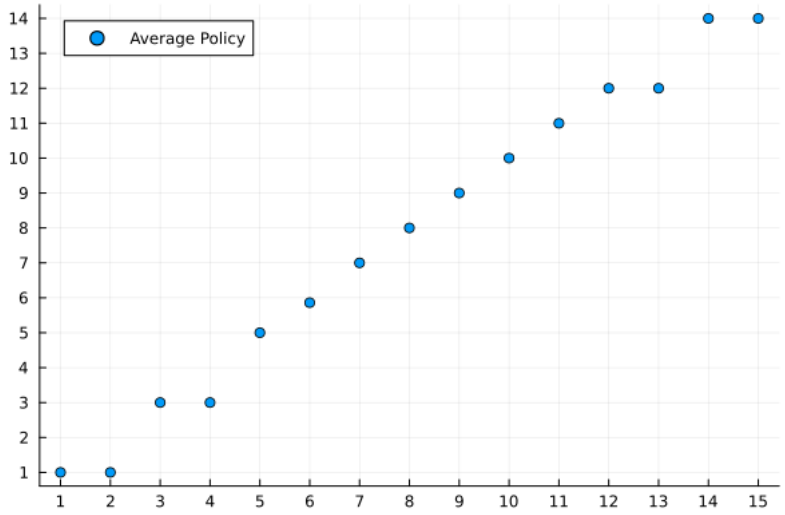

Figure 3: Long run policy averaged over simulation runs. We observe a set of 4 final policies. 96% of simulation experiments coincide with policy shown in Figure 2.

Again, all simulation runs converge to equilibria - in this case, since exploration decays to zero, the equilibria recovered correspond to Nash equilibria of the game without noise. None of the long run policies feature full revelation. The algorithms find policies that are equilibria of the noisy game while $\varepsilon_t > 0$. The issue is, that these equilibria *remain equilibria* for smaller $\varepsilon < \varepsilon_t$. Since the equilibrium conditions remain satisfied while ε_t continues to decay, the policy remains unadjusted and learning stops. We can define a class of equilibria that we observe in the majority ($> 99\%$) of our experiments under full revelation, for which there is a simple mechanism to their robustness to decaying ε_t.

First, some definitions will be helpful:

Definition 2. *Say that an argmax-policy $m^*(\cdot)$ 'has a connected pool of size n' if there is a set of states $P = \{x^{(k_1)}, \ldots, x^{(k_1+1)}, \ldots, x^{(k_n)}\} \subseteq M_K$ s.t.*

1. $k_1 \leq k \leq k_n \Rightarrow x_k \in P$,
2. $m^*(x) = m$ *if and only if* $x \in P$.

Definition 3. *For any $K < \infty$, $\varepsilon \in [0, 1]$ say a Nash equilibrium (m^*, y^*) of $\Gamma_K^\varepsilon(0)$ is a connected-pool (CP) equilibrium if the policy m^* consists of connected pools (possibly of*

size 1*) only.*

Existence is guaranteed, since babbling is a CP-equilibrium. Connected-pool equilibria have a simple structure with important implications on incentives: take a CP-equilibrium for ε small enough. Then as ε is decreases, incentives for the sender are unaffected by changes in the receiver's posterior. As a result, the equilibrium continues to exist also for lower levels of ε.

Lemma 1. *For all $K < \infty$ there exists $\bar{\varepsilon}_K > 0$ small enough s.t. all CP-Nash equilibria of $\Gamma_K^{\bar{\varepsilon}_K}(0)$ remain Nash equilibria of $\Gamma_K^{\varepsilon}(0)$ for all $\varepsilon < \bar{\varepsilon}_K$.*

Indeed, in our simulation experiment under decaying ε_t and diagonal initial policy as shown in figure 3, all 4 final policies are CP-equilibria. Moreover, for our remaining experiments in the no-bias setting, all final policies are CP-equilibria.

Given this fact about convergence in this setting, it is natural to ask whether more can be said about selection, i.e. whether some CP-equilibria are favoured by the algorithmic learning process. We computationally answer this question by computing, for every fixed $\varepsilon \geq 0$, the most informative CP-Nash equilibrium of $\Gamma_K^{\varepsilon}(0)$, given $K = 15$.[4] We refer to this as μ_ε^*. We then compute the percentage reduction in uncertainty over the state of that $\bar{\mu}_\varepsilon$ as measured by I (5), which we refer to as $\bar{I}(\varepsilon)$. The following Figure 4 shows $R_I = \frac{I}{\bar{I}(\varepsilon)}$, the ratio of average I observed by final policies in our experiments to $\bar{I}(\varepsilon)$.

[4]This is done by sorting candidate policies according to their informativeness I, and then searching over all policies for a given I until an equilibrium is found. Notice that due to the uniform prior ρ_0, policies that have the same number of pools of the same size will have equal I. The search then amounts to iterating over all possible combinations of pools of a given number and size, in decreasing order of I.

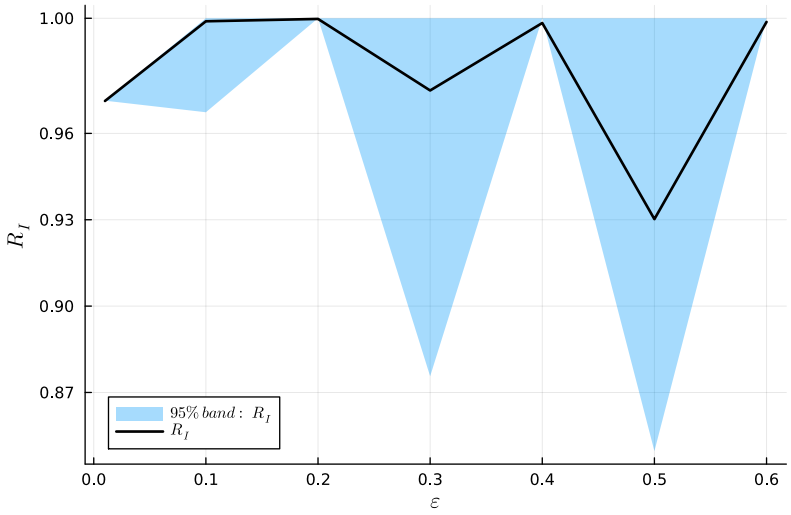

Figure 4: R_I, plotted over fixed ε.

The figure shows how, with little variation, the sender's policy converges to policies close in information quality to the best possible information conveyed among all CP-Nash equilibria of $\Gamma_K^{\varepsilon}(0)$. This is likely due to the full-revelation initialization of the sender: starting from there, the rest point involving the fewest adjustments to m^* in the learning process would of course be one of the most informative equilibria.

Regarding convergence speed, we observe that larger ε leads (usually) to faster convergence speed. We refer to the time period at which convergence was achieved as T_C and plot in Figure 5 below. In this initialization, we find a few large outliers that strongly influence the average convergence speed; however, for most experiments, larger ε implies faster convergence.

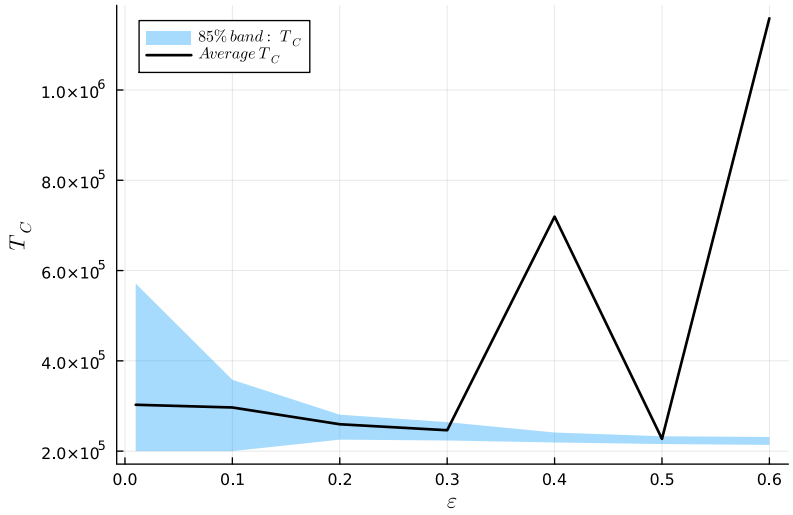

Figure 5: T_C, plotted over fixed ε.

Taken together, figures 4 and 5 imply a tradeoff in terms of learning outcomes: while lower exploration is more likely to lead to more informative policies relative to the best equilibrium, higher exploration will lead to faster convergence.

As can be expected based on Proposition 1, both the value of ε if it is constant, and the initial value ε_0 if it decays, affect our simulation experiments. To gain an overview of the learned policies as ε is changed, we analyze long-run policies through the information content they transmit to the receiver. The following Figures **??** and 7 show the relationship between I and varying (initial) ε when exploration is constant (decaying in Figure 7, respectively), and a diagonal initial policy is used.

Notice first that by definition, when exploration is fixed, the information content measured by I will include the fixed exploration rate ε, which necessarily leads to a degradation in information transmission of the policy μ_∞. For the case where ε remains fixed over time, we therefore also introduce the alternative measure I^*, which measures the information content of the argmax policy m^*_∞ alone:

$$I^* = \left(\sum_{x \in X_K} \rho(x) \log \left(\frac{1}{\rho(x)} \right) \right)^{-1} \sum_{x \in X_K} \sum_{m \in M_K} m^*_\infty(x) \log \left(\frac{m^*_\infty(x)}{\sum_{x' \in X_K} m^*_\infty(x') \rho(x')} \right).$$

The following figure shows the result including both I and I^*:

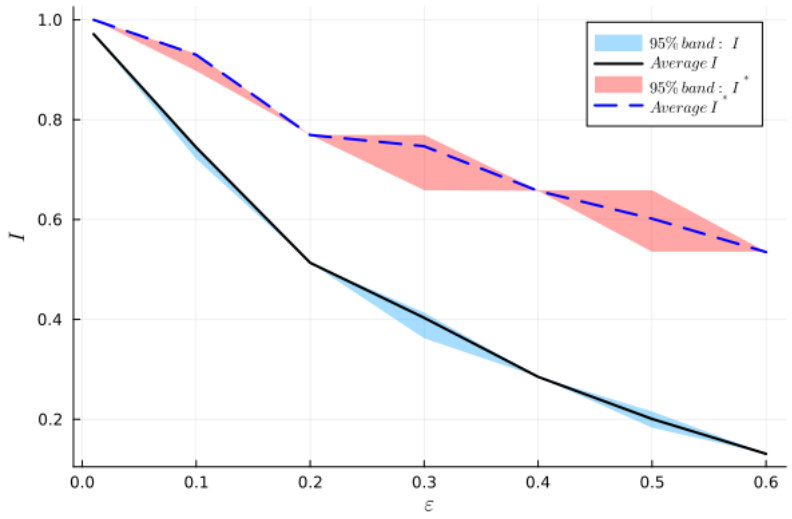

Figure 6: I of long run policy averaged over simulation runs. ε constant $\in \{0.01, 0.1, 0.2, \ldots, 0.6\}$, Information replaced by I^*.

The figure clearly illustrates how higher exploration rates lead to more equilibrium pooling and hence less information being transmitted. This finding holds also when we run this experiment under the condition that ε_t decays over time, as the CP-equilibria that are first found correspond to higher exploration rates:

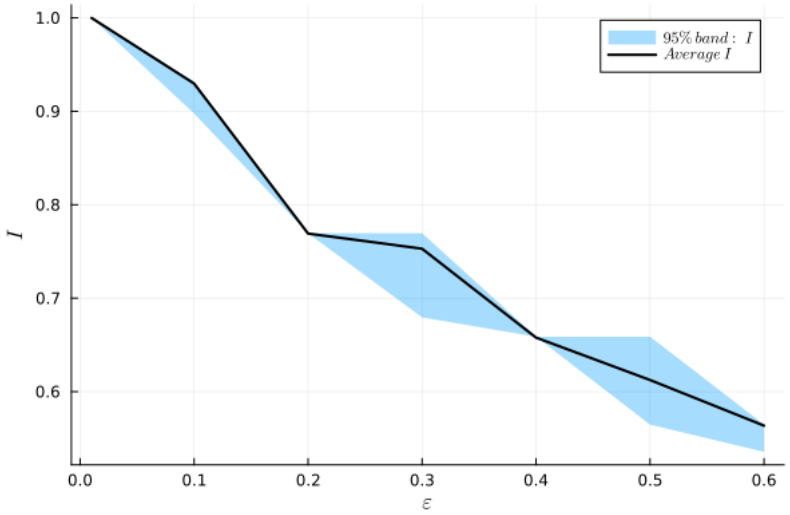

Figure 7: I of long run policy averaged over simulation runs. ε-decay at $\beta = 10^{-5}$. X-axis signifies initial $\varepsilon_0 \in \{0.01, 0.1, 0.2, \ldots, 0.6\}$.

3.1.2 Babbling

In this section we show that when algorithms are initialized with Q values corresponding to a babbling strategy, they generically learn a more informative policy. We show that this property is due to an inherent instability of the babbling policy. Indeed, should values be initialized at the correct payoffs to babbling (including the correct best response of the receiver (4)), the learner is stuck at babbling. However, initializing in a small neighborhood of those correct payoffs leads the learner away from babbling. Experiments investigating this further are found in Online Appendix B. Again, all our experiments converge to a Nash equilibrium, as proposition 1 still applies.

First, we initialize babbling with $Q_0(m, x) = v$ for all m, x, for $v \sim U[-1, 0]$ and plot the results in Figures 8 and 9.

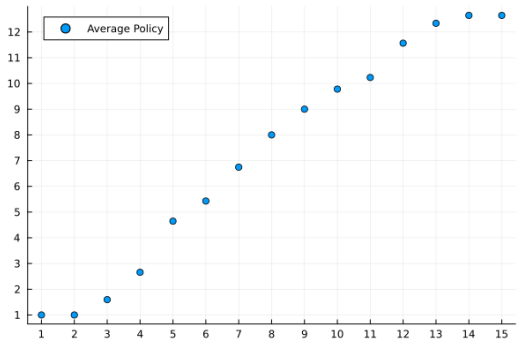

Figure 8: Long run policy averaged over simulation runs, decaying ε_t. Here, we observe a large set (67) runs, constant $\varepsilon = 0.15$. Here, we observe a large set of final policies, all of which are CP-equilibria. 56 (73) of final policies, all of which are CP-equilibria. of these (84%) are also present in the experiment above, of figure 8.

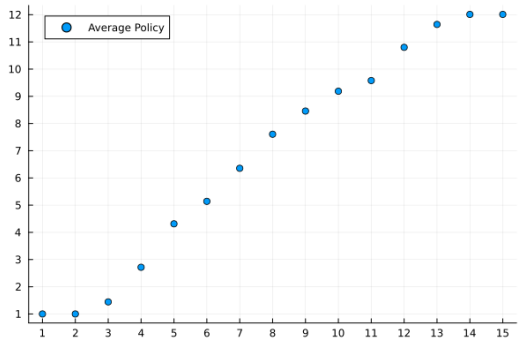

Figure 9: Long run policy averaged over simulation

Despite the algorithm and the receiver not having an initial common language, the algorithm learns to communicate surprisingly close to the same level as when initialized from perfect communication. Especially states close to the prior of the receiver are communicated nearly perfectly. This is also reflected in the high informativeness of communication when we plot both I, I^* together in Figure 10 below.

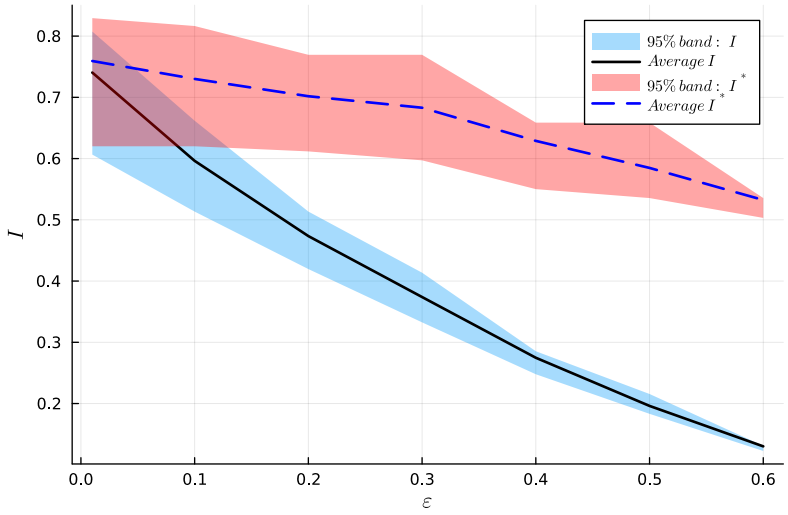

Figure 10: I, I^* of long run policy averaged over simulation runs. $\varepsilon \in \{0.01, 0.1, 0.2, \ldots, 0.6\}$ constant.

Next, in Figure 11, we consider R_I, the ratio of average information I to the best theoretical equilibrium information $\bar{I}(\varepsilon)$ under fixed ε.

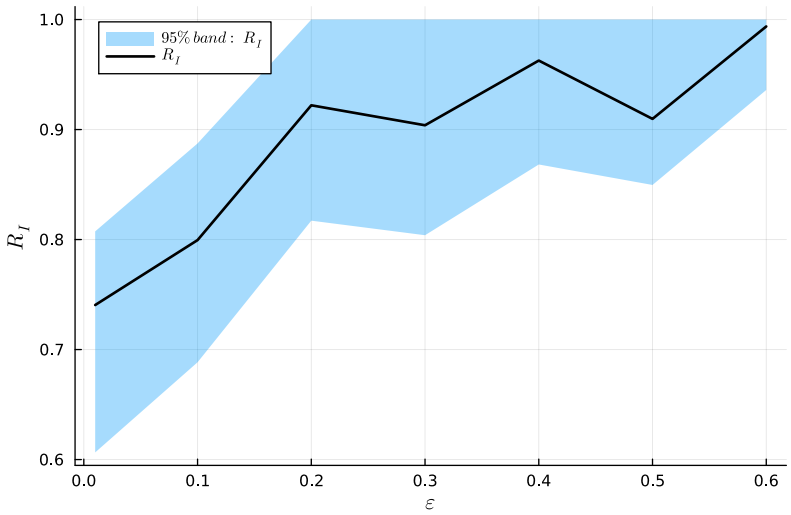

Figure 11: R_I, plotted over fixed ε.

Interestingly, even when initializing at the worst possible information quality, the final policy's information quality on average is no less than 75% of the best information quality among all CP equilibria of $\Gamma^{\varepsilon}_{K}(0)$. Furthermore, increasing exploration leads to information quality closer to the best equilibrium. The result may suggest a practical trade-off when choosing the exploration rate in situations where the sender and the receiver do not have a common language: A higher exploration rate leads to learning equilibria that are closer to the best available equilibrium. However, at the same time the best equilibrium that can be learned becomes worse due to exploration distorting communication at extreme states. Figure 10 suggests that as long as we care only about the informativeness of the final outcome, lower initial exploration rates lead to better outcomes.

Contrary to the situation of full revelation initialization, here we observe that, while larger exploration ε both leads to better final policies in terms of as well as a faster convergence speed.

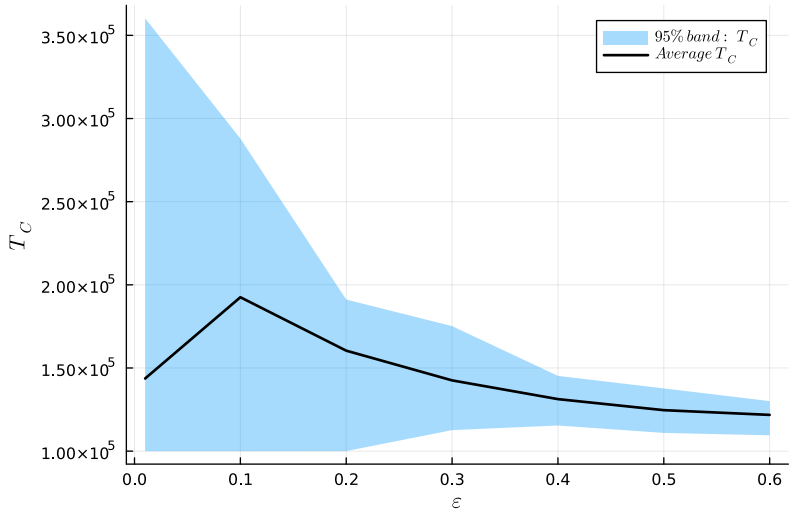

Figure 12: T_C, plotted over fixed ε.

Last, we consider a decaying exploration rate as before. Figures 13 below plots the informativeness of the learned outcome I as a function of the initial exploration rate. As can be seen from the figure, there is large downward variation in the informativeness of the learned outcome for low levels of ε which almost completely vanishes for exploration rates higher than 0.3. Moderate initial exploration rates help learning equilibria before the exploration rate die out. However, as we know from above, it also adversely affects the best equilibrium that can be learned. This is also evident in the fact that the highest levels of informativeness in Figure 13 are achieved at exploration rates around 0.1.

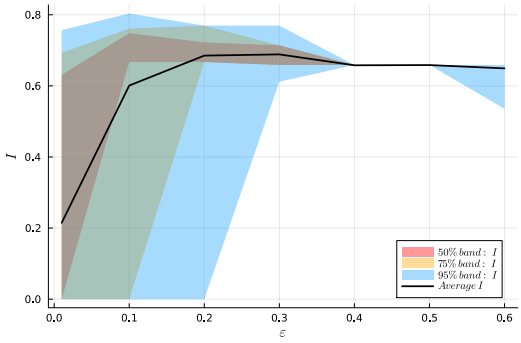

Figure 13: I of long run policy averaged over simulation runs. ε decay at $\beta = 10^{-5}$.

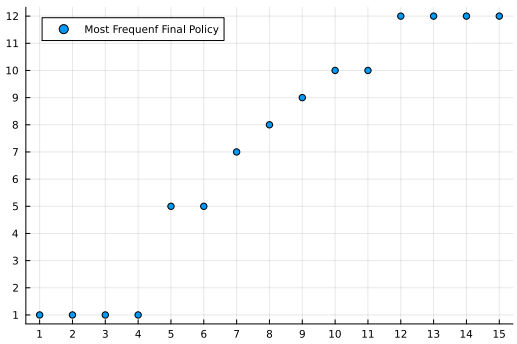

Figure 14: Most frequent final policy for $\varepsilon_0 \geq 0.3$

As the average I appears to stabilize for $\varepsilon_0 \geq 0.3$, we plot the most commonly learned policy for this range of exploration rates in Figure 14. In line with our findings above, the most frequent final policy learned is the same, which is shown in figure 14. For $\varepsilon_0 = 0.3$, this policy is learned in $\sim 29\%$ of the experiments, while for all higher $\varepsilon_0 > 0.3$, it is learned in $> 90\%$ of the simulation experiments. This suggests that when no common language exists, losses from exploration with a decaying exploration rate are bounded from above and the algorithm "gets stuck" only at relatively informative equilibria of the underlying cheap talk game without noise.

3.2 Positive Bias

When the sender prefers higher actions than the receiver, i.e. when $b > 0$, the pattern we observe is consistent and robust: full revelation for low states, and pooling for high states. In fact, even the cutoff state at which the algorithm starts to pool messages is robust and commonly fixed at a single value, varying only in the bias and exploration rate. The following figures show mean final policies for varying biases, when Q-values are initialized at the correct payoffs for full revelation, and $\varepsilon = 0.15$ fixed:

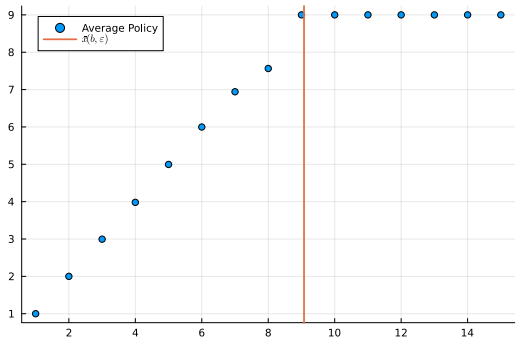

Figure 15: $b = 0.1$

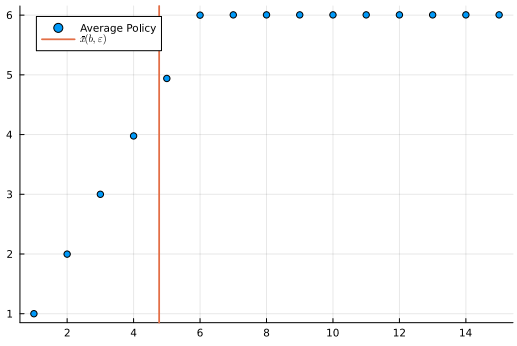

Figure 16: $b = 0.2$

We can characterize the cutoff to the top pool, which for large enough K takes a simple form we refer to as $\bar{x}(b, \varepsilon)$ in the figures above.

Lemma 2. *When* $0 < b < \frac{1+\varepsilon}{4}$, *and* $\varepsilon \leq \min\{2b, \frac{1-2b}{1-b}\}$, *let the largest state below the top pool be* x_{k^*}. *Then* k^* *is pinned down as* $\lfloor k_1 \rfloor$, *where* k_1 *is the lower root of the following quadratic polynomial* $a_2k^2 + a_1k + a_0 = 0$, *with*

$$\begin{aligned}
a_2 &= (1-\varepsilon)(1+\varepsilon) \\
a_1 &= -(1-\varepsilon)\big(K(2+\varepsilon-4b)+1+\varepsilon+4b\big) - 2\varepsilon\big(1+\varepsilon+K(1-\varepsilon)\big) \\
a_0 &= \big(\varepsilon+K(1-\varepsilon)\big)\big(K(1+\varepsilon-4b)+1+\varepsilon+4b\big).
\end{aligned}$$

Furthermore,

$$\bar{x}(b,\varepsilon) = \lim_{K\to\infty} \frac{k^*}{K} = \frac{1+\varepsilon-4b}{1+2\varepsilon}.$$

Thus, the limit cutoff $\bar{x}$ falls in b, with only babbling remaining an outcome for large enough b. The cutoff falls (rises) in ε if b is smaller (larger) than $\frac{1}{8}$. Notice that we display policies *averaged* over the simulation runs; average messages being whole numbers therefore implies that the same message was used for the same state in all simulation runs.

Similar behavior emerges under our experiments for decaying ε_t:

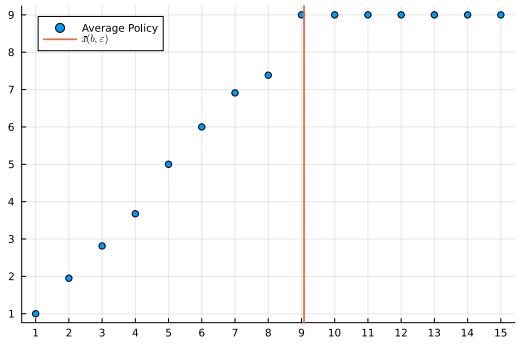

Figure 17: $b = 0.1$, ε_t decay with $\varepsilon_0 = 0.15$

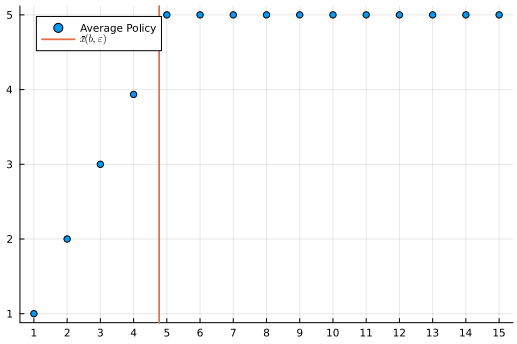

Figure 18: $b = 0.2$, ε_t decay with $\varepsilon_0 = 0.15$

Plotting I, I^* together:

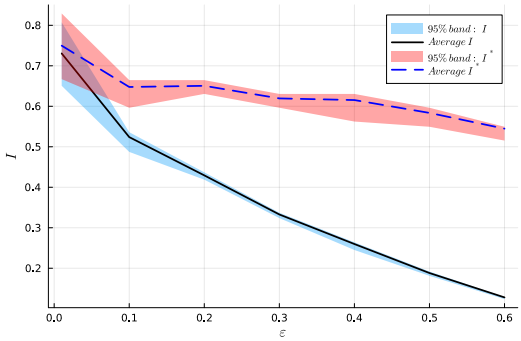

Figure 19: I, I^* of long run policy averaged over simulation runs. $b = 0.1$, $\varepsilon \in \{0.01, 0.1, 0.2, \ldots, 0.6\}$ constant.

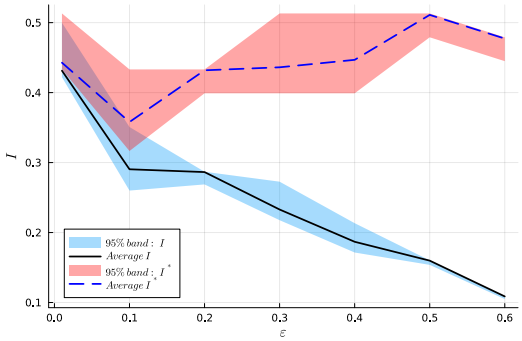

Figure 20: I, I^* of long run policy averaged over simulation runs. $b = 0.2$, $\varepsilon \in \{0.01, 0.1, 0.2, \ldots, 0.6\}$ constant.

Figures 19 and 20 exemplify the non-monotone dependence of cutoff $\bar{x}(b, \varepsilon)$ on ε. Notice that average I^* decreases in ε for $b = 0.1$, but increases over a range $\varepsilon \in [0.1, 0.5]$ for $b = 0.2$. As noted below lemma 2, this is in line with how the cutoff behaves, as $0.1 < \frac{1}{8} < 0.2$. Of course, $\bar{x}(b, \varepsilon)$ is an approximation of the true cutoff at $K = 15$, but apparently the behavior as a function of b, ε carries over to that finite K.

The pattern changes considerably when we consider a decaying ε_t in Figure 21. The average information remains quite stable for $\varepsilon_0 \geq 0.1$. We find that since ε_t decays to zero, the final policies for $\varepsilon_0 \geq 0.1$ all involve the same cutoff to the top pool, which is

close to $\bar{x}(b,0) = 1 - 4b$, the cutoff in Lemma 2 when $\varepsilon \to 0$, and therefore *independent* of ε_0. For $\varepsilon_0 < 0.1$, exploration is initially too small to lead to the top-pool pattern we otherwise observe.

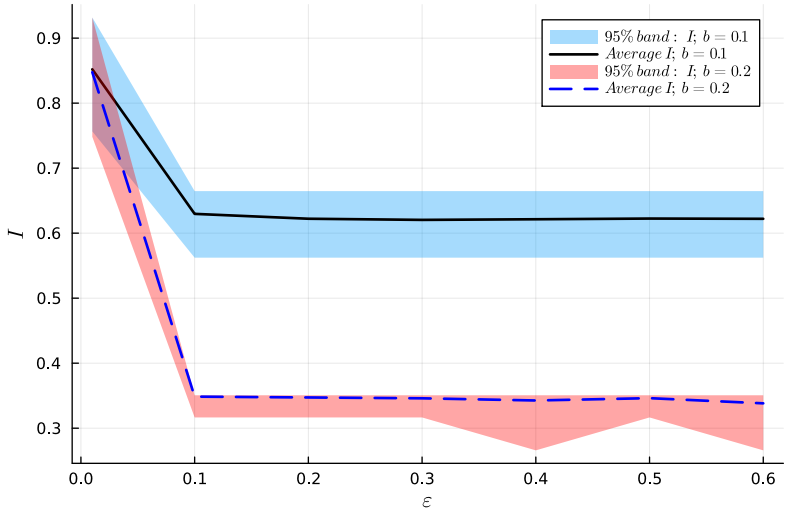

Figure 21: I of long run policy averaged over simulation runs, case of $b = 0.1, b = 0.2$ plotted together. ε decay at $\beta = 10^{-5}$. X-axis signifies initial $\varepsilon_0 \in \{0.01, 0.1, 0.2, \ldots, 0.6\}$.

Our simulations do not converge to a stationary policy. Instead, they appear to be cycling through full revelation below the cutoff and sometimes having a small number of pools of 2-4 states. The pool above the cutoff is robust, however. By inspecting the policies over time, one can see that most of the time, the algorithm spends playing full revelation at states below the cutoff. To see this, we generate 100 snapshots of policies for each simulation run, starting at period $3,333,333$, and equally spaced until the final simulation period at $10,000,000$, to ensure that Q-functions can be updated sufficiently between the snapshots. The following figure shows the fraction of the time policies were fully revealing during those snapshots in a histogram over all simulations, under constant exploration $\varepsilon = 0.15$.

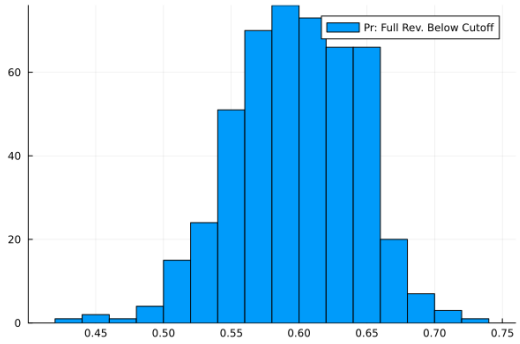

Figure 22: $b = 0.1$

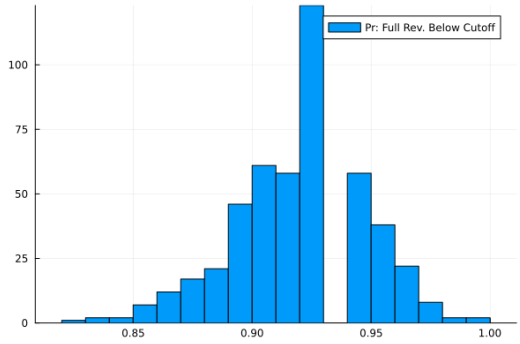

Figure 23: $b = 0.2$

How does this pattern emerge? Starting from the initial full revelation, given their bias, all sender types would benefit from pooling with the type just above them. The only type that does not have this option is the highest type. Hence, lower types start pooling with it, making the mean of the pool progressively lower and hence potentially enabling even more types to join. Furthermore, it is important to note that compared to standard perfect Bayesian equilibrium, the exploration strategy of the sender fully pins down the receiver's beliefs whenever the receiver observes something that is not part of the sender's current optimal exploitative strategy. Since states are uniformly distributed, and S's exploration strategy is to choose any action with uniform probability, the posterior belief after a message that currently does not maximize the Q_t matrix for any state must be uniform across all states. As a result, whenever such an 'unused' message m' is sent, R will play $y_t(m') = \mathbb{E}[x] = 0.5$. This means that any sender type always has as an outside option: the choice of playing an unused message that will be associated with the midpoint policy. Hence, the cutpoint for the top pool is pinned down by the first type who prefers either fully revealing their type or the midpoint of the state space to the payoff associated with the top pool. Now, types below the cutoff type all have a strict incentive to join the type just above them. This leads to a number of them packing together with the cutoff type, until that type prefers to switch to an unused action and hence escape the pool. This creates a constant cycling where each type below the cutoff type joins a type above them until enough of them pool with the

cutoff type who then escapes the pool by selecting an unused action.[5]

Accordingly, one would expect that, if there is pooling among states below the cutoff, most frequently such pools should appear for high states below the cutoff. Figures 24 and 25 show the frequency of having a pool below the cutoff on states *larger* than the average state below the cutoff, conditional on having a pool below the cutoff at all. The figures clearly confirm the intuition.

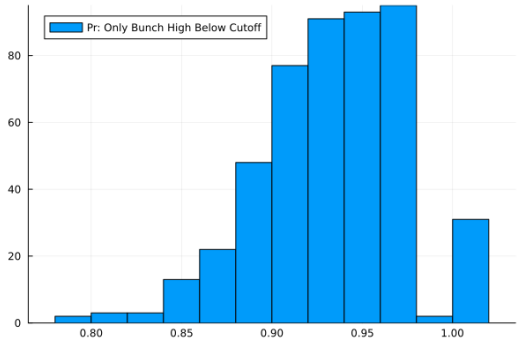

Figure 24: $b = 0.1$

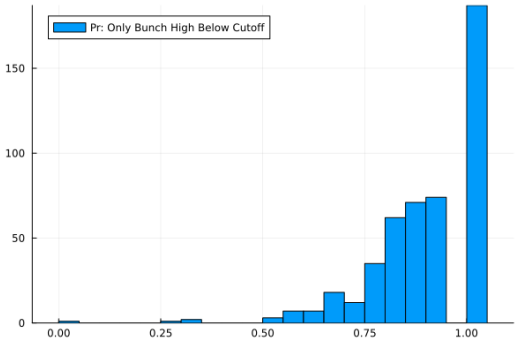

Figure 25: $b = 0.2$

The overall pattern we observe in this section is very close to the one found theoretically in Kartik (2009). The biggest differences are the lack of language inflation and the instability of the final outcome. This is not very surprising, since our setup does not include a cost for lying and hence the message space does not have a natural initial meaning. One may wonder what happens when messages are initially meaningful and deviating from that meaning is costly, for example, because of some reputational concerns. In Appendix Online Appendix B.3 we re-apply our methodology to a version of Kartik's model. Our findings confirm most of Kartik's predictions, including a fairly consistent language inflation and a large pool at the top.

[5]Notice that the highest type below the top pool may also join the top pool to avoid pooling with lower types. If they prefer that to an unused action, they are not the cutoff type, since after joining the pool their only option to leave it is through an unused action or by pooling with a lower type which they already revealed to dislike relative to the top pool. This is the exact way we pin down the cutoff type.

3.3 Welfare

An important question we face when looking into the effects of the introduction of algorithmic learning is whether participants of the interaction can be shown to be better off in comparison to a non-algorithmic alternative. Here we are interested in asking whether the patterns learned by our recommendation algorithm can be ranked in terms of the sender's payoff, receiver's payoff, and the best equilibrium of the alternative Crawford and Sobel (1982) rational player interaction. Here we consider the best equilibrium of the game of continuous states and actions as outlined in Crawford and Sobel (1982), since the equilibrium computation method we employed in the $b = 0$ scenario remained unsuccessful within reasonable computation time.

As can be seen below, we find a robust pattern indicating that expected welfare is increased for all preference misalignment values, except for perfect and almost perfect alignment of preferences.

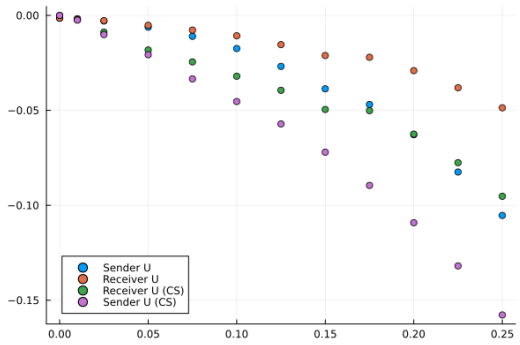

Figure 26: Average Payoffs

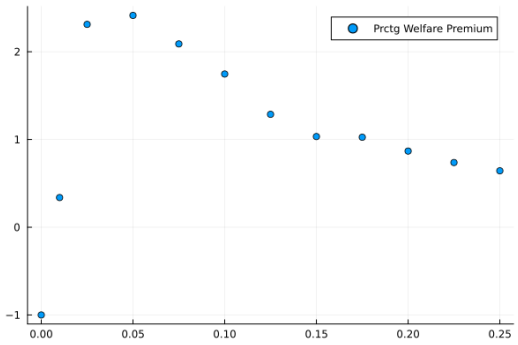

Figure 27: Welfare Premium

Figure 26 shows expected payoffs on the vertical, plotted against fixed bias values on the horizontal axis. We plot expected payoffs for sender, receiver in terms of the final policies learned and averaged over all 480 simulation experiments, and in terms of the best CS equilibrium given the fixed bias. A clear ordering can be observed: when $b > 0$, then both sender and receiver are better off under our learned pattern than under the best CS equilibrium. Under $b = 0$, the best CS equilibrium is fully revealing and leads to

a payoff of zero, which can not be bested by the learning outcomes. Figure 27 plots the percentage premium that average welfare from our simulation experiment has over the best CS equilibrium's welfare (sum of sender and receiver payoffs). For very low biases, it makes sense that the CS equilibrium welfare is higher than our learning results, owing to the full or close to full revelation equilibrium that is feasible to rational players then. However, already biases above 0.01 lead to welfare gains in our learning setting.

4 Conclusions

Algorithmic advise is becoming increasingly ubiquitous especially with the recent advancements in generative artificial intelligence and concerted industry effort in making it easily accessible in professions outside of computer science. Typical to these applications is the algorithm's superior ability to aggregate and process information relative to the ultimate decision maker. This paper highlights new sources of possible frictions in communication between recipients of algorithmic advice and the algorithmic advisors.

We argue that basic algorithms learn to communicate to some degree with decision makers even when no clear language between the algorithm and the decision maker exist ex-ante. Nevertheless, this communication will typically remain imperfect due to a strategic interaction between the algorithm's need to explore to learn to communicate and the consecutive imperfect responses from the decision maker. For designers of algorithms our paper allows identifying patterns that might be indicative communication frictions due to algorithmic exploration, such as pooling of messages at both extremes of the outcome space. Once identified, there is hope for correcting for such frictions with reprogramming the algorithm especially if there is no more need for the algorithm to continue exploring.

On a positive note, we also show that when the algorithm's designer and the receiver of the advice have different preferences, the algorithm leads to a welfare improvement relative to the prediction from the workhorse Crawford and Sobel (1982) model of cheap talk. The welfare improvements result from the algorithm being unable to find a perfect

Bayesian equilibrium and keeps cycling over policies. These policies are on average more informative than the equilibrium policies of Crawford and Sobel (1982). As both the algorithm's designer and the end-user benefit from this instability, the designer may not have incentives to fix it and hence algorithms may help in communication problems where traditional strategic communication frictions are sizable.

References

Alonso, Ricardo and Niko Matouschek, "Relational delegation," *The RAND Journal of Economics*, 2007, *38* (4), 1070–1089.

Blume, Andreas, Oliver J Board, and Kohei Kawamura, "Noisy talk," *Theoretical Economics*, 2007, *2* (4), 395–440.

Calvano, Emilio, Giacomo Calzolari, Vincenzo Denicolo, and Sergio Pastorello, "Artificial intelligence, algorithmic pricing, and collusion," *American Economic Review*, 2020, *110* (10), 3267–3297.

Condorelli, Daniele and Massimiliano Furlan, "Cheap Talking Algorithms," *arXiv preprint arXiv:2310.07867*, 2023.

Cowgill, Bo and Megan T Stevenson, "Algorithmic social engineering," in "AEA Papers and Proceedings," Vol. 110 American Economic Association 2014 Broadway, Suite 305, Nashville, TN 37203 2020, pp. 96–100.

Crawford, Vincent P and Joel Sobel, "Strategic information transmission," *Econometrica: Journal of the Econometric Society*, 1982, pp. 1431–1451.

Garcia, Daniel, Juha Tolvanen, and Alexander K Wagner, "Demand estimation using managerial responses to automated price recommendations," *Management Science*, 2022, *68* (11), 7918–7939.

_ , _ , and _ , "Strategic Responses to Algorithmic Recommendations: Evidence from Hotel Pricing," *Working Paper*, 2023.

Hoffman, Mitchell, Lisa B Kahn, and Danielle Li, "Discretion in hiring," *The Quarterly Journal of Economics*, 2018, *133* (2), 765–800.

Huang, Yufeng, "Pricing frictions and platform remedies: the case of Airbnb," *Available at SSRN 3767103*, 2022.

Kartik, Navin, "Strategic communication with lying costs," *The Review of Economic Studies*, 2009, *76* (4), 1359–1395.

Ludwig, Jens and Sendhil Mullainathan, "Fragile algorithms and fallible decision-makers: lessons from the justice system," *Journal of Economic Perspectives*, 2021, *35* (4), 71–96.

Sutton, Richard S and Andrew G Barto, *Reinforcement learning: An introduction*, MIT press, 2018.

Watkins, Christopher JCH and Peter Dayan, "Q-learning," *Machine learning*, 1992, *8*, 279–292.

Appendix

For Online Publication

Strategic Communication and Algorithmic Advice

Emilio Calvano Clemens Possnig Juha Tolvanen

Online Appendix A Proofs Omitted from the Main Text

Online Appendix A.1 Proof of Lemma 1

Fix K, pick an CP-Nash equilibrium (ω, y) of $\Gamma_K^{\varepsilon}(0)$ for some ε. Suppose first that there are at least two pools with upper bound below $\frac{1}{2}$. Let $x^{(k)}$ be the cutoff- type for two successive pools, i.e. the lowest type of the upper pool, and just above the largest type of the lower pool. Let n_1, n_2 be the sizes of the lower and upper pool, respectively. By the equilibrium property, the following IC constraint must hold:

$$\begin{aligned} \Big(y(m^{(2)}) - x^{(k)}\Big)^2 &\leq \Big(y(m^{(1)}) - x^{(k)}\Big)^2 \\ \Leftrightarrow \Big(\bar{x}^{(2)} - x^{(k)} + \alpha(2,\varepsilon)\big(\frac{1}{2} - \bar{x}^{(2)}\big)\Big)^2 &\leq \Big(\bar{x}^{(1)} - x^{(k)} + \alpha(1,\varepsilon)\big(\frac{1}{2} - \bar{x}^{(1)}\big)\Big)^2 \\ \Leftrightarrow L(k,\varepsilon) &\leq R(k,\varepsilon), \end{aligned}$$

where we follow equation (4), and define $\bar{x}^{(i)}$ as the mean state of pool i, and $\alpha(i,\varepsilon) = \frac{\varepsilon}{n_i(1-\varepsilon)+\varepsilon}$. First, note that for any $\varepsilon \geq 0$,

$$\bar{x}^{(2)} - x^{(k)} + \alpha(2,\varepsilon)\big(\frac{1}{2} - \bar{x}^{(2)}\big) \geq 0,$$

since $\bar{x}^{(2)} \geq x^{(k)}$, and $\frac{1}{2} \geq \bar{x}^{(2)}$ by assumption. Also, for any $\varepsilon \geq 0$ small enough,

$$\bar{x}^{(1)} - x^{(k)} + \alpha(1,\varepsilon)\big(\frac{1}{2} - \bar{x}^{(1)}\big) \leq 0,$$

since $\bar{x}^{(1)} - x^{(k)} \leq 0$. From now on, let $\bar{x}^{(1)} - x^{(k)} < 0$ (otherwise there is nothing to prove), and pick $\bar{\varepsilon}$ such that

$$\bar{x}^{(1)} - x^{(k)} + \alpha(1,\bar{\varepsilon})\big(\frac{1}{2} - \bar{x}^{(1)}\big) = 0.$$

Now, $L(k,\varepsilon)$ must increase in ε, while $R(k,\varepsilon)$ *decreases* in ε for all $\varepsilon < \bar{\varepsilon}$. Thus, the IC constraint continues to hold as ε is decreased from $\bar{\varepsilon}$. Due to the fact that all pools are

connected, these IC constraints holding imply all other ICs hold as well.

Now suppose there is exactly one pool below and above $\frac{1}{2}$, in which case the relevant cutoff-type $x^{(k)}$ must satisfy $x_{k-1} \leq \frac{1}{2} \leq x^{(k)}$. In this case, $\bar{x}^{(2)} - x^{(k)}$ holds for all $K > 3$ and we have that

$$\bar{x}^{(2)} - x^{(k)} + \alpha(2, \varepsilon)\big(\frac{1}{2} - \bar{x}^{(2)}\big) \geq 0,$$

holds for all ε small enough. Similarly,

$$\bar{x}^{(1)} - x^{(k)} + \alpha(1, \varepsilon)\big(\frac{1}{2} - \bar{x}^{(1)}\big) < 0,$$

holds for all $\varepsilon \geq 0$ small enough. Thus, an argument analogous to the case before can be applied here also.

Finally, the consideration of cutoff types for pools above $\frac{1}{2}$ can be done analogously as above, and so the conclusion follows: the equilibrium is sustained by the incentive constraints holding for all cutoff types; we have shown that these constraints remain valid for all $\varepsilon \geq 0$ small enough, which finishes the proof. ■

Online Appendix A.2 Proof of Lemma 2

To characterize the cutoff under discreteness and $b > 0$, first define upper and lower incentive constraints as the constraints faced by two neighboring types:
ICU:

$$U^S(y(m^*(x^{(k)}), x^{(k)}, b) \geq U^S(y(m^*(x_{k+1}), x^{(k)}, b),$$

ICD:

$$U^S(y(m^*(x_{k+1}), x_{k+1}, b) \geq U^S(y(m^*(x_k), x_{k+1}, b).$$

Lemma 3. *Let $b < \frac{1}{2}$, and let $x^{(k)}$ be the lowest type below the cutoff to the top*

pool. A binding upper incentive constraint implies the lower incentive constraint if $\varepsilon \leq \min\{2b, \frac{1-2b}{1-b}\}$.

Proof. First, note that

$$\begin{aligned} U^S(y(m^*(x^{(k)}), x^{(k)}, b) &> U^S(y(m^*(x_k), x_{k+1}, b) \\ \Leftrightarrow (\varepsilon\frac{1}{2} - \varepsilon x^{(k)} - b)^2 &< (\varepsilon\frac{1}{2} - \varepsilon x^{(k)} - b - \frac{1}{K-1})^2 \\ \Leftrightarrow |a| &< |a - \frac{1}{K-1}|, \end{aligned}$$

where $a = \varepsilon\frac{1}{2} - \varepsilon x^{(k)} - b$. This holds if $a < 0$, which holds if $\varepsilon < 2b$.

Next, let $K^* = K - k$, the number of types in the top pool. Write $\bar{y} = y(m^*(x_{k+1}$.

$$\begin{aligned} U^S(y(m^*(x_{k+1}), x_{k+1}, b) &> U^S(y(m^*(x_{k+1}), x_k, b) \\ \Leftrightarrow (\bar{y} - b - x_{k+1})^2 &< (\bar{y} - b - x_{k+1} + \frac{1}{K-1})^2, \end{aligned}$$

which is true when $\bar{y} - b - x_{k+1} > 0$. Note that

$$\bar{y} - x_{k+1} = \frac{K^*(1-\varepsilon)}{K^*(1-\varepsilon)+\varepsilon}\frac{1 - x_{k+1}}{2} + \frac{\varepsilon}{K^*(1-\varepsilon)+\varepsilon}\big(\frac{1}{2} - x_{k+1}\big),$$

which is decreasing in k. Setting $k = K - 2$, we get $K^* = 2$ and

$$\begin{aligned} \bar{y} - x_{k+1} &= \beta\frac{1}{2(K-1)} + (1-\beta)\big(\frac{1}{K-1} - \frac{1}{2}\big) \\ &= \beta\frac{1}{2} + \frac{1}{K-1}(1 - \beta\frac{1}{2}) > \beta\frac{1}{2}, \end{aligned}$$

where $\beta = \frac{2(1-\varepsilon)}{2(1-\varepsilon)+\varepsilon}$. We arrive at the sufficient condition

$$\begin{aligned} \beta\frac{1}{2} > b \Leftrightarrow (1-\varepsilon) &> b(2(1-\varepsilon)+\varepsilon) \\ \Leftrightarrow \frac{1-2b}{1-b} &> \varepsilon. \end{aligned}$$

Finally, note that when ICU binds, $U^S(y(m^*(x^{(k)}), x^{(k)}, b) > U^S(y(m^*(x_k), x_{k+1}, b)$ and

$U^S(y(m^*(x_{k+1}), x_{k+1}, b) > U^S(y(m^*(x_{k+1}), x_k, b)$ together imply that ICD must hold also. The result follows. □

Now, using Lemma 3, we pin down k_1, k_2 by taking the ICU bind:

$$U^S(y(m^*(x_{k^*}), x_{k^*}, b) \geq U^S(y(m^*(x_{k^*+1}), x_{k^*}, b),$$

where some tedious algebra leads to the quadratic equation in the statement of the Lemma, with

$$a_2 k^2 + a_1 k + a_0 \geq 0.$$

Since $a_2 > 0$, the LHS is convex and therefore the two roots $k_1 \leq k_2$ are cutoffs such that the ICU is satisfied for $k \leq k_1$ and $k \geq k_2$. Note also that the ICU must be violated at $k = K - 1$, since then the top 'pool' consists of the top state $x^{(K)}$ only, and due to the bias $b > 0$, x_{K-1} would always be better off pretending to be $x^{(K)}$. Thus, $k_2 \geq K$ must hold, and is therefore not available as a solution. $\lfloor k_1 \rfloor$ is then the largest integer under which ICU is still satisfied. ■

Corollary 1.

$$\lim_{K \to \infty} \frac{k^*}{K} = \frac{1 + \varepsilon - 4b}{1 + 2\varepsilon}.$$

Online Appendix A.3 Proof of Proposition 1

Lemma 4. *For any sequence of policies $\{m_t(\cdot)\}_{t \geq 0}$, every action-state pair (m, x) is visited infinitely often with probability 1, if $\sum_{t \geq 0} \varepsilon_t = \infty$.*

Proof. Let m_t be the realized message in period t. Notice that, no matter the policy, for any pair (m, x):

$$\mathbf{P}\Big(x_t = x, m_t = m\Big) \geq \frac{\varepsilon_t}{K^2},$$

as per the exploration part of the epsilon-greedy action selection policy. Let $D_t(m,x)$ be the event that x, m are realized in period t through exploration. By definition, the sequence $\{D_t(m,x)\}_{t\geq 0}$ is a sequence of independent events. Thus,

$$\mathbf{P}\Big(\{D_t(m,x)\}_{t\geq 0}\Big) = \frac{1}{K^2}\sum_{t\geq 0}\varepsilon_t = \infty,$$

as by assumption of the lemma. By the second Borel-Cantelli lemma, (x,m) must be visited infinitely often with probability 1. □

Proof of Proposition 1. It is well known (Watkins and Dayan (1992)) that if $y_t(m) = \bar{y}(m)$ constant for all t large enough, all $Q_t(m,x)$ values will converge to the correct estimates $U^S(y(m),x)$, with probability one. Here, since states x are i.i.d., this is an application of the law of large numbers once all states and actions are observed infinitely often, as ensured by lemma 4. Given that in fact, policies are changing, continuity is all that's needed to conclude this proof.

Pick any converging sequence of argmax policies $\{m_t^*\}_{t\geq 0}$. Notice that the receiver's best response y_t depends on time t only through μ_t, the sender's policy. Define

$$n_T(m,x) = \sum_{t=1}^{T} \mathbf{1}\{m_t = m, x_t = x\}$$

as the number of times up to T at which (m,x) were realized. Fix m, x, and write $n_T = n_T(m,x)$ for ease of notation. Let h_T be a sequence s.t. $h_T \leq n_T$, $h_T \to_{n_T\to\infty} \infty$, and $n_T - h_T \to_{n_T\to\infty} \infty$. Using the recursion (1), we can write

$$Q_{T+1}(m,x) = \alpha \sum_{h=h_T+1}^{n_T} (1-\alpha)^{n_t-h} U^S(y_h(m),x) + (1-\alpha)^{n_T-h_T} Q_{h_T}(m,x).$$

Letting $(\bar{m},\bar{y})$ be the limit of (m_t^*, y_t), we have

$$\left|Q_{T+1}(m,x) - U^S(\bar{y}(m),x)\right| \leq \alpha \sum_{h=h_T+1}^{n_T} (1-\alpha)^{n_T-h}\left|U^S(y_h(m),x) - U^S(\bar{y}(m),x)\right| + (1-\alpha)^{n_T-h_T}\left|Q_{h_T}(m,x) - U^S(\bar{y}(m),x)\right|.$$

By lemma 4, for all (m,x), $n_T(m,x) \to \infty$ with probability one. By continuity, $\left|U^S(y_t(m),x) - U^S(\bar{y}(m),x)\right| \to 0$ as $t \to \infty$. The second term converges to zero as $n_T - h_T \to \infty$. For the first term, we can write

$$\alpha \sum_{h=h_T+1}^{n_T} (1-\alpha)^{n_T-h}\left|U^S(y_h(m),x) - U^S(\bar{y}(m),x)\right| \leq \left((1-\alpha)^{h_T+1} - (1-\alpha)^{n_T}\right) \sup_{h_T+1\leq t\leq n_T} \left|U^S(y_t(m),x) - U^S(\bar{y}(m),x)\right|,$$

which also converges to zero as $n_T, h_T \to \infty$, by the previous observation of continuity. The result follows: for any sequence m_t^* that converges, it must be true that, with probability one, $Q_t(m,x)$ converges to $U^S(\bar{y}(m),x)$ for all (m,x). Since the problem of finding the argmax action for the sender given $Q_t(m,x)$ is discrete and unconstrained, the argmax-correspondence is upper hemicontinuous in Q_t-values. Thus, as Q_t converges, it must be true that $\bar{m}(x)$ satisfies $\bar{m}(x) \in \arg\max_m U^S(\bar{y}(m),x)$. By definition, $\bar{y}$ is the receiver's best response to $\bar{\mu}(m|x)$, defined following (2), by replacing ε_t with $\bar{\varepsilon}$. Thus, $\bar{m}, \bar{y}$ must constitute a Nash equilibrium of $\Gamma_K^{\bar{\varepsilon}}(b)$. ■ □

Online Appendix B Alternative Specifications

Online Appendix B.1 Full Revelation

Here we run the same experiment as in the main text in 3.1.1 under a full revelation policy initialized through Q-values randomly generated from $U[-1,0]$. We do observe a larger variety of long-run policies, but the flavor of our result does not change.

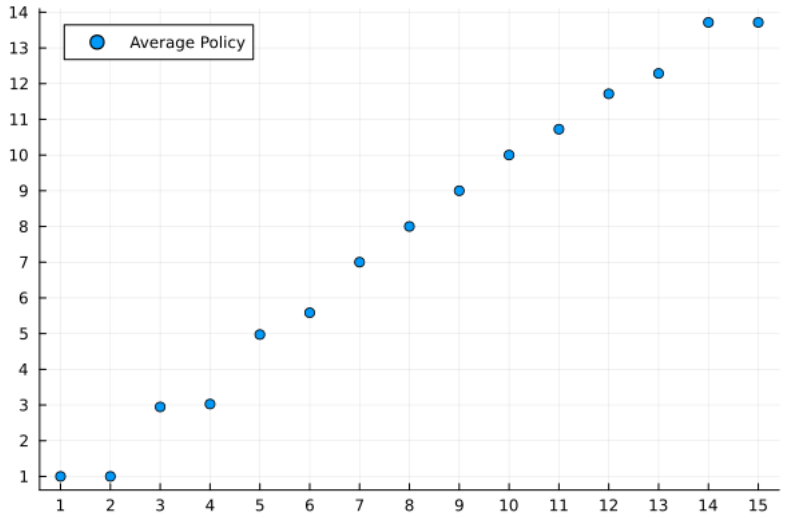

Figure OA.1: Long run policy averaged over simulation runs, constant $\varepsilon = 0.15$. Here, we observe a larger set (41) of final policies than in the diagonal initialization. The largest share ($\sim 24\%$) coincides with the majority final policy in figure 3.

The behavior of I, I^* is also similar to what we observed in the case of diagonal initial policy with the information distortion increasing with the exploration rate:

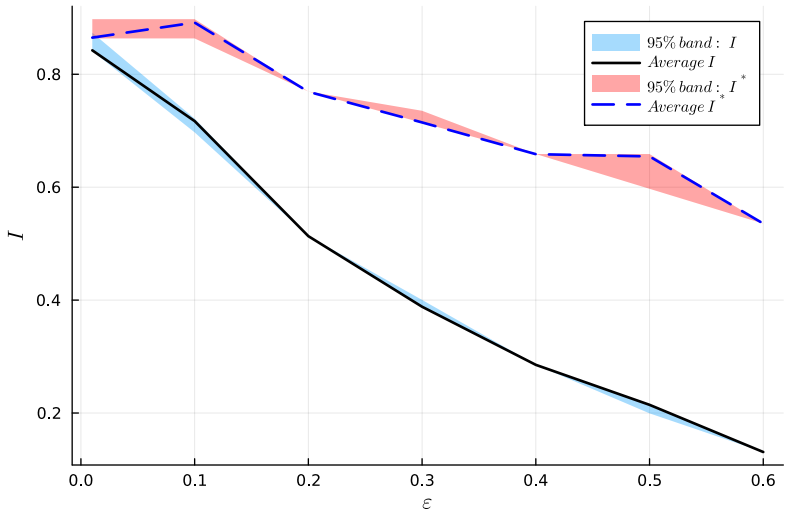

Figure OA.2: I, I^* of long run policy averaged over simulation runs. ε constant $\in \{0.01, 0.1, 0.2, \ldots, 0.6\}$, full revelation initialized from uniformly randomly.

Similarly, in the case of decaying ε_t, we observe a larger variation in final policies than in the diagonal initial policy case (22 in the former, 4 in the latter), all of which are CP-equilibria. However, extreme states are still robustly pooled.:

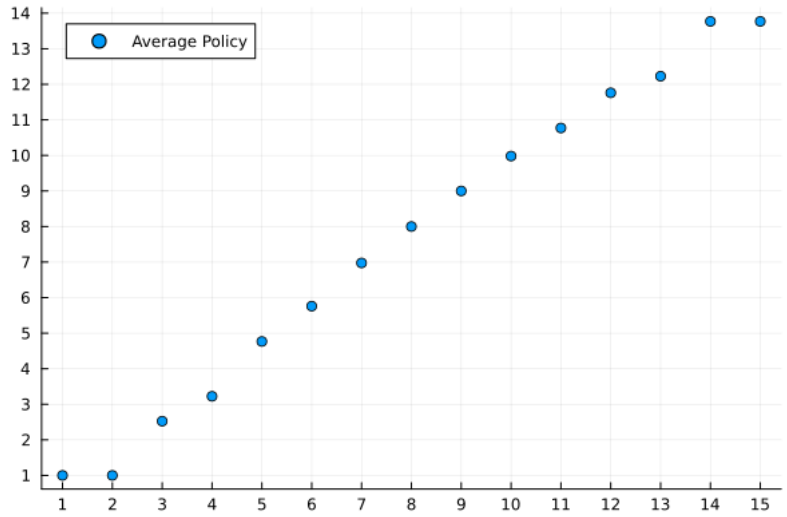

Figure OA.3: Long run policy averaged over simulation runs, decaying $\varepsilon = 0.15$.

Also, the information distortion is robustly increasing in the initial exploration rate:

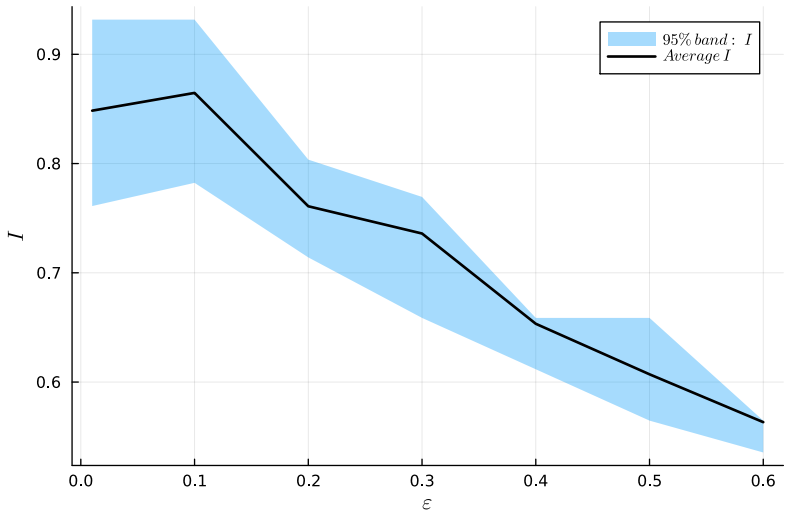

Figure OA.4: I of long run policy averaged over simulation runs. ε decay at $\beta = 10^{-5}$. X-axis signifies initial $\varepsilon_0 \in \{0.01, 0.1, 0.2, \ldots, 0.6\}$. Full revelation initialized uniformly randomly.

Online Appendix B.2 Babbling

This section strengthens the results of 3.1.2. To emphasize the instability of babbling, we initialize Q-values the following way: Let $m_B()$ be the babbling policy, so that $m_B(x) = m$ for all $x \in X_K$. Then $Q_0(x, m) = U^S(y(m_B(x)), x, 0) + v_{x,m}$, where $v_{x,m} \sim U[-0.05, 0.05]$ drawn i.i.d for every (x, m). None of the resulting final policies remain at babbling:

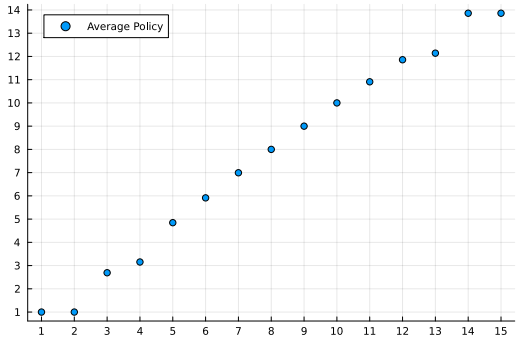

Figure OA.5: Long run policy averaged over simulation runs, constant $\varepsilon = 0.15$.

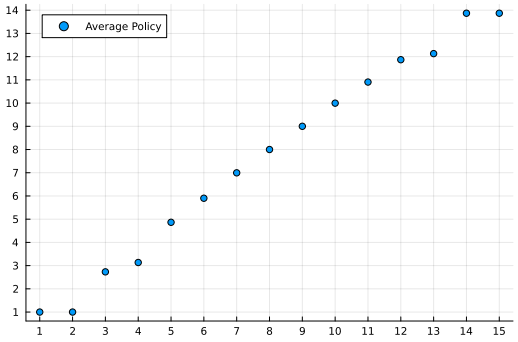

Figure OA.6: Long run policy averaged over simulation runs, ε decay at $\beta = 10^{-5}$.

As expected, I, I^* also behave similarly to our baseline babbling experiment:

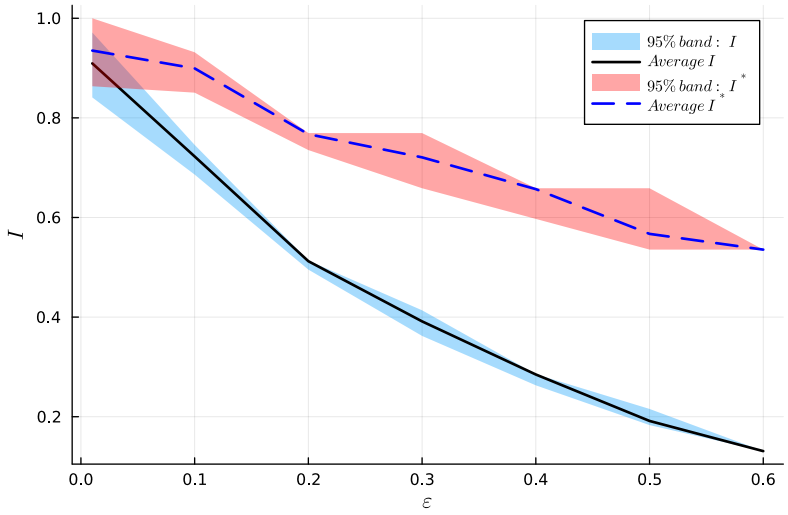

Figure OA.7: I, I^* of long run policy averaged over simulation runs. $\varepsilon \in \{0.01, 0.1, 0.2, \ldots, 0.6\}$ constant.

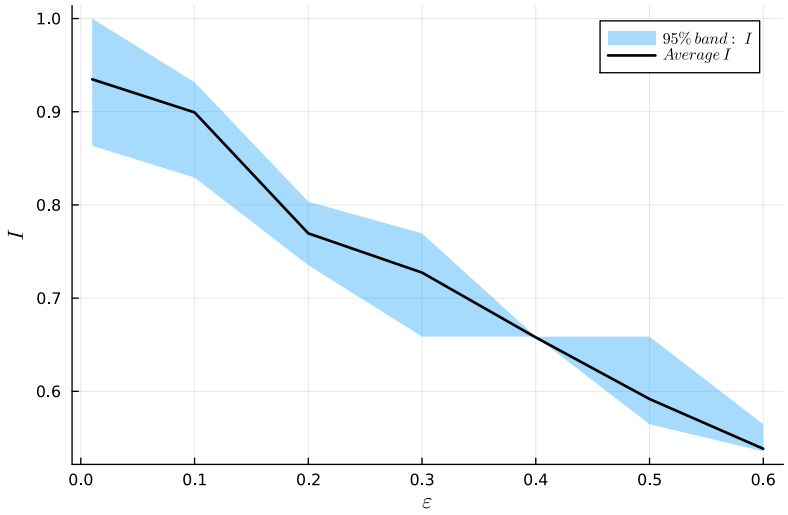

Figure OA.8: I of long run policy averaged over simulation runs. ε decay at $\beta = 10^{-5}$. X-axis signifies initial $\varepsilon_0 \in \{0.01, 0.1, 0.2, \ldots, 0.6\}$.

Online Appendix B.3 Lying Costs

An extension to our baseline interaction is one in which, contrary to the standard cheap talk literature, messages have meaning. Here we consider the situation where the sender faces some cost if they were misrepresenting their private information about the state. One can imagine a situation where receivers have the possibility of verifying ex post what has been claimed by the sender, and would punish deviations from the truth. This setting is related to the one studied in Kartik (2009), where costs are incurred in terms of the distance of the message sent from the message associated with the true state. In the case of our learners, we have message and state spaces equal to $I_K = M_K \subset [0, 1]$. We then model lying costs of message m given true state x as a function $c(m, x) = c_0(m-x)^2$, for varying marginal costs $c_0 > 0$.

We keep all baseline parameters of the Q- learning agent as they have been in our main study above. We run experiments over $c_0 \in \{0.1, 0.5, 1.0, 2.0\}$, and for bias values of $b \in \{0.0, 0.1, 0.2\}$.

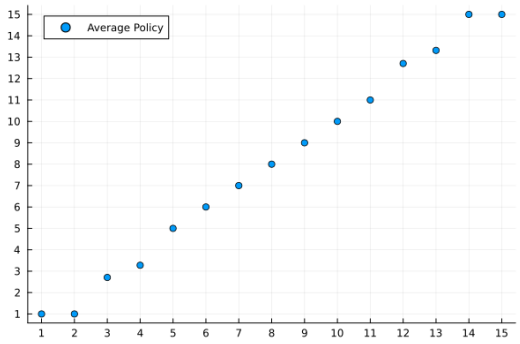

Figure OA.9: $b = 0, c_0 = 0.1$

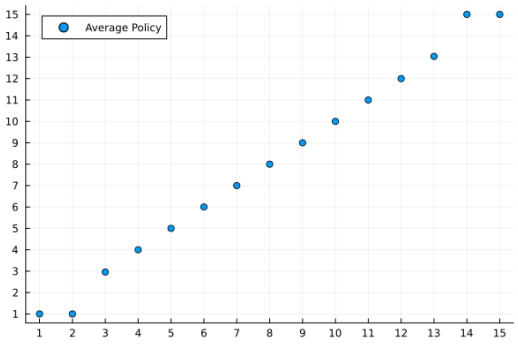

Figure OA.10: $b = 0, c_0 = 0.5$

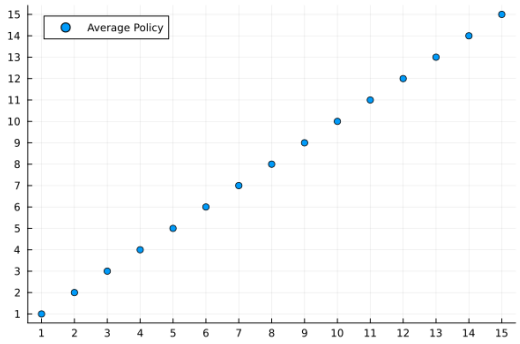

Figure OA.11: $b = 0, c_0 = 1.0$

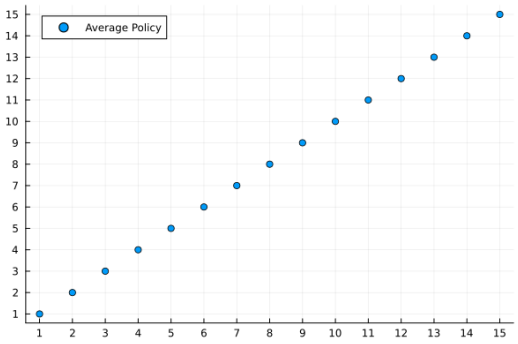

Figure OA.12: $b = 0, c_0 = 2.0$

As can be intuitively expected, under $b = 0$ we arrive at a situation where the otherwise emerging small pools below and above 0.5, as observed in 2, vanish. The lying cost makes supporting pooled actions more and more unprofitable, and full revelation is restored in this case.

Furthermore, in line with Kartik (2009), when lying costs are not too high, we observe language inflation where types below the cutoff choose messages higher than their true type despite these messages being fully revealing.

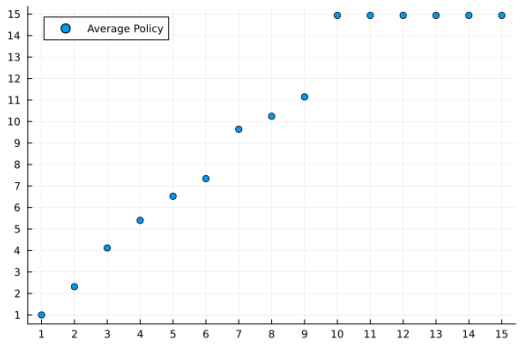

Figure OA.13: $b = 0.1, c_0 = 0.1$

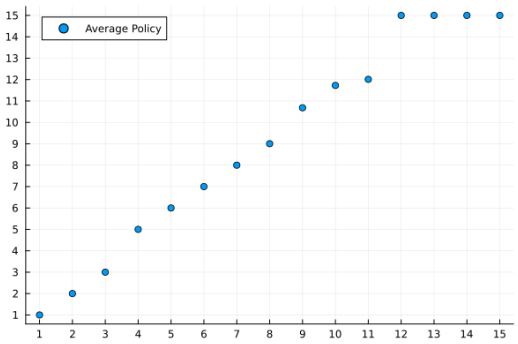

Figure OA.14: $b = 0.1, c_0 = 0.5$

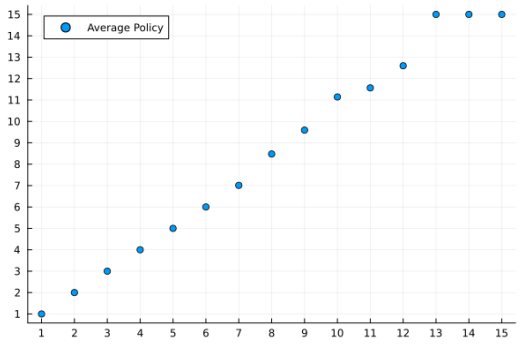

Figure OA.15: $b = 0.1, c_0 = 1.0$

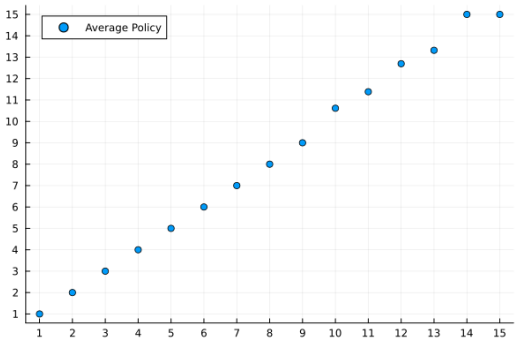

Figure OA.16: $b = 0.1, c_0 = 2.0$

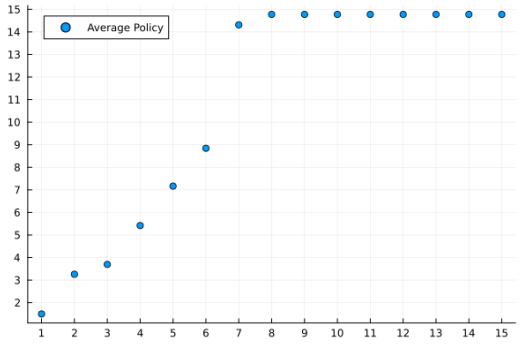

Figure OA.17: $b = 0.2, c_0 = 0.1$

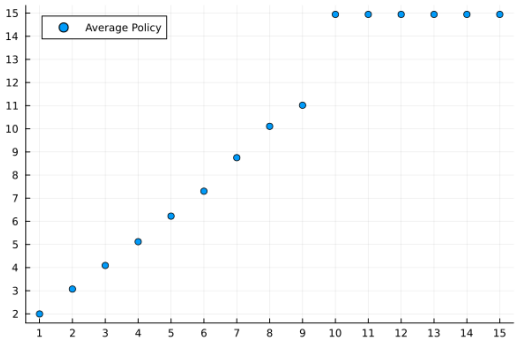

Figure OA.18: $b = 0.2, c_0 = 0.5$

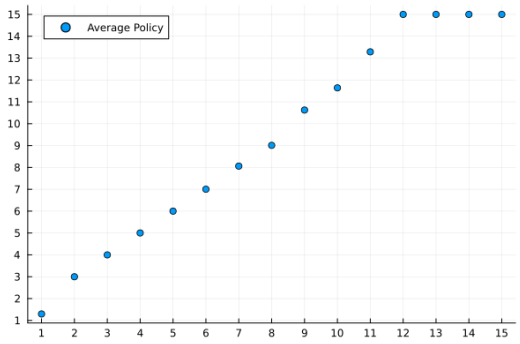

Figure OA.19: $b = 0.2, c_0 = 1.0$

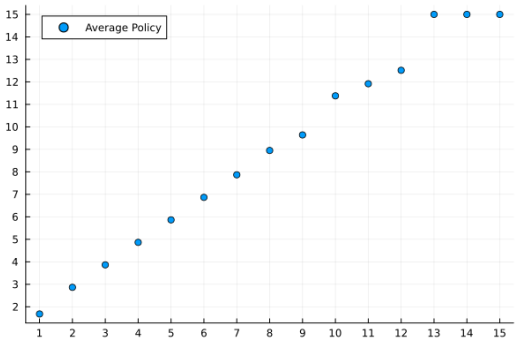

Figure OA.20: $b = 0.2, c_0 = 2.0$