Modeling Boundedly Rational Agents with Latent Inference Budgets

Athul Paul Jacob
Massachusetts Institute of Technology
apjacob@mit.edu

Abhishek Gupta
University of Washington
abhgupta@cs.washington.edu

Jacob Andreas
Massachusetts Institute of Technology
jda@mit.edu

Abstract

We study the problem of modeling a population of agents pursuing unknown goals subject to unknown computational constraints. In standard models of bounded rationality, sub-optimal decision-making is simulated by adding homoscedastic noise to optimal decisions rather than actually simulating constrained inference. In this work, we introduce a *latent inference budget model (L-IBM)* that models these constraints explicitly, via a latent variable (inferred jointly with a model of agents' goals) that controls the runtime of an iterative inference algorithm. L-IBMs make it possible to learn agent models using data from diverse populations of suboptimal actors. In three modeling tasks—inferring navigation goals from routes, inferring communicative intents from human utterances, and predicting next moves in human chess games—we show that L-IBMs match or outperform Boltzmann models of decision-making under uncertainty. Moreover, the inferred inference budgets are themselves meaningful, efficient to compute, and correlated with measures of player skill, partner skill and task difficulty.

1 Introduction

Building effective models for multi-agent decision-making—whether cooperative or adversarial—requires understanding other agents' goals and plans. To help a friend navigate in a new environment, we must first understand where they want to go; to beat an opponent at chess, we must be able to predict their likely next moves. But decision-making, in humans and machines, is subject to computational constraints. Decision-makers often act suboptimally, relying on heuristics and approximations to choose their actions. Techniques that do not account for this suboptimality carefully may invariably attribute behavior to differing intentions rather than different inference procedures.

How should we interact with agents seeking to accomplish unknown goals subject to unknown computational constraints? In this paper, we describe a simple approach for building models of agents given traces of their behavior in a way that accounts for agent-specific suboptimality. In standard models of bounded rationality [20], sub-optimal decision-making is simulated by adding noise to optimal decisions rather than actually simulating constrained inference. This results in inference methods that treat agents as uniformly suboptimal in a way that fails to account for sub-optimal inference *algorithms* or for *non-homogenous* suboptimality. In this work, by contrast, we look to explicitly model an agent's "inference budget", via a latent variable that controls the runtime of an iterative inference algorithm, inferred jointly with a model of an agent's goals. This provides the ability to perform accurate intent inference from a mixture of agents across a range of skill levels. We show that a diverse set of multi-agent decision-making procedures—including graph-based

NeurIPS 2023 Workshop on Generalization in Planning (GenPlan 2023)

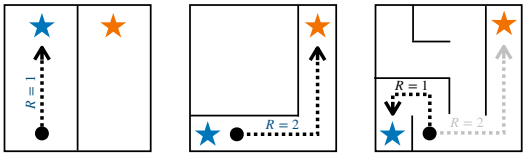
planning algorithms, recursive-rational models of human language production, and Monte Carlo tree search—admit efficient imputation of inference budgets in this framework.

In three diverse agent modeling tasks—inferring navigation goals from routes, inferring communicative intents from human utterances, and predicting subsequent moves in human–human chess matches—we show that our approach matches or outperforms Boltzmann models of decision-making under uncertainty. Moreover, inferred inference budgets are themselves meaningful, correlating with measures of player skill, partner skill, and task difficulty. Our results show that sub-optimal human decision-making can be efficiently modeled with computationally constrained versions of standard search algorithms. In doing so, we obtain both accurate models of human behavior and informative measures of inferential capacity.

2 Background and problem formulation

We study the problem of modeling one or more agents given given traces of their behavior. In particular, we assume that we observe a collection of trajectories (state–action sequences) produced by agents $\pi^* : s \mapsto a$ acting in a Markov decision process to maximize some reward function $R^*(\tau)$. Even when $R^*(\tau)$ is known to agents, inferring optimal actions is often intractable, so agents in the real world will in general *approximate* optimal behavior subject to some (unknown) computational constraints (which may differ from agent to agent). From this data, we seek to infer **agent models** π defined in terms of (1) estimates R of reward function R^* (known to agents but not modelers), and (2) descriptions of the computational limitations that govern agents' choice of actions. In other words, we seek to model both *what agents wish to do* and *what agents will actually do* in any given state. Fig. 1 shows an example: the three trajectories depicted there cannot be generated by the optimal policy for any cost function, but can be explained by model that is only look ahead to a limited number of positions in the maze.

Throuhgout this paper, we will model agent actions as arising from an **approximate inference procedure** $\pi(a \mid s; R, \beta)$ that takes as input a reward function and a **computational budget** β; we may then model the agent by inferring values of R and β given the actually executed trajectories τ_i.



(a) (b) (c)

Figure 1: Inferring rewards from boundedly-rational trajectories. The agent will move to the blue star (a), but prefers to move toward the orange star when both are available (b). When locating the orange star requires solving a harder search problem, however, the agent seeks the blue star instead, indicating that its search abilities are limited. Knowing this, we could perhaps assist this agent by providing a targeted hint (*move right*) at the beginning of its trajectory.

The ability to infer goals from suboptimal (and even completely unsuccessful) plans is a key human skill, present in children as young as 18 months [24]. Computational models of bounded rationality thus have a long history in artificial intelligence, cognitive science, and behavioral economics. But what does this suboptimality look like in practice, and how should we model and infer the inference budget β simply from observations of behavior?

One of the most widely used models of boundedly rational decision-making is the so-called **Boltzmann** model [20], in which agents take actions according to

$$\pi(a \mid s; R, \beta) \propto \exp\{\beta \cdot R(s, a)\} \tag{1}$$

This equation has a number of appealing interpretations, e.g. as the solution to the maximum-entropy policy search problem for an agent aiming to maximize rewards subject to an entropy constraint. It has been used to model not just the selection of actions, but also trajectories, preferences, corrections, and more—see [17] for a recent survey. More elaborate approaches in this family also predict β conditioned on the current state or action history, making it possible to model state-dependent skill [2].

However, Boltzmann models have a significant limitation: the probability of generating an action in Eq. (1) depends only on the true value of that action, and not on the cost of acquiring a high-quality value estimate in the first place. To see why this might be a problem, consider again the trajectories depicted in Fig. 1(b–c), which differ *only* in the difficulty of the search problem, and not in the cost of

the optimal trajectory at all. A model of boundedly rational decision-making with the form of Eq. (1) cannot account for this difference.

There is a large body of other approaches on modeling human planning under resource constraints in psychology, economics and in classical/symbolic AI [4, 27, 15, 14, 5, 11, 3, 30, inter alia]. However, these approaches make strong assumptions about how planning is performed, limiting their applicability to real-world data. Here, we seek to develop a general framework that avoids strong assumptions about either the functional form of the reward model or the algorithmic form of the planning procedure. As a result, we can apply this single framework to real-world behavior in tasks as diverse as language generation and chess gameplay.

3 Inferring Rewards and Inference Budgets from Behavior

As motivated in Section 2, our goal is to model agents acting to optimize an unknown value function subject to an unknown computational constraint. In practice, we often want to model populations comprising multiple agents or agent sub-populations $(\pi_1^*, \pi_2^*, \dots \pi_N^*)$ with a shared reward functions R^* (e.g. winning at chess) but differing computational constraints.

To do so, we assume we have access to a collection of trajectories $\{\tau\}_i = \{\tau_i^1, \tau_i^2, \dots \tau_i^{M_i}\}$, with each collection of trajectories $\{\tau\}_i$ generated by a different agent or sub-population i. We model these trajectories as drawn from the following generative process:

1. at each timestep, agent i draws a budget β from an agent-specific prior $p_{\text{budget}}(\beta \mid \eta_i)$
2. π_i^* chooses actions according to a budget-constrained inference procedure $\pi_i^*(a \mid s; R^*, \beta)$

Because budgets may vary between trajectories, learning a model of these agents ultimately learning reward parameters θ and agent-specific budget-generating parameters η_i while *marginalizing* over latent budgets themselves. We do so via maximum *a posteriori* inference, optimizing:

$$\arg\max_{\theta,\eta} \sum_{\substack{i \\ \tau \in \{\tau\}_i \\ (s,a) \in \tau}} \log \pi(a \mid s; \theta, \eta) = \arg\max_{\theta,\eta} \sum_{\substack{i \\ \tau \in \{\tau\}_i \\ (s,a) \in \tau}} \log \sum_{\beta} p_{\text{budget}}(\beta \mid \eta_i) \cdot \pi(a \mid s; R_\theta, \beta) \tag{2}$$

If $\pi(a \mid s; R^*, \beta)$ is an arbitrary inference algorithm, Eq. (2) might present a challenge: this inference procedure must be run for all possible values of β, which will in general be intractable. Under what circumstances can we optimize this equation efficiently? The key observation in this paper is that if π is an *anytime inference algorithm* [6], we can evaluate n values of β as quickly as we can evaluate one, making this optimization tractable.

Definition 1. *An anytime algorithm π is one that runs for t timesteps and produces a sequence of inference states $(f_1, f_2, \dots f_t)$, where every f_i can be computed from f_{i-1} in $\mathcal{O}(1)$ time, and f_i can be used to select an action according to some $\pi(a \mid s; R, f_i)$.*

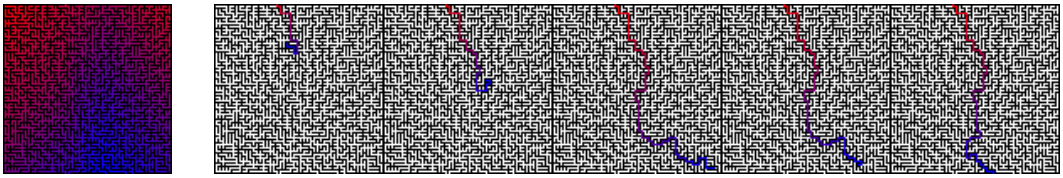
As we will see shortly, many canonical inference algorithms used in single- and multi-agent decision-making scenarios have this from. In these cases, rather than letting the budget parameter β determine noise or suboptimality, we may use it to parameterize the runtime of the agent's inference procedure itself, writing:

$$\log \pi(a \mid s; \theta, \eta_i) = \log \sum_{\beta} p_{\text{budget}}(\beta_{\text{runtime}} \mid \eta_i) \cdot \pi(a \mid s; R_\theta, f_{\beta_{\text{runtime}}}) \tag{3}$$

where we have denoted the budget β_{runtime} to indicate that it parameterizes the runtime of the anytime inference algorithm. This can be computed, up to some maximum β, as efficiently as computing only the final term in the sum. The remainder of this paper looks at instantiations of this basic modeling framework in three different domains. In Section 4, we study the problem of inferring navigation goals from maze domain using a truncated graph search algorithm. In Section 5, we study rational speech acts (RSA) for inferring communicative intents from human utterances. Finally, in Section 6, we model human action prediction in chess using Monte-Carlo tree search (MCTS).

4 Solving Mazes with Truncated Depth-First Search

We begin with a pedagogical example of latent inference budget model applied to a simple, single-agent decision-making task: maze navigation. Agents are placed at a random position in a maze with



(a) (b) (c) (d) (e) (f)

Figure 2: Examples of the maze task. (a) Example of the value function heuristic applied to each state in the maze. Red indicates low value states and blue indicates high value states. (b)-(f) depicts the example trajectories of agents with depth budgets of 1, 2, 5, 10 and 20.

five exits. Each exit is a state e_i associated with a reward R_i. Agents attempt to navigate toward the highest scoring exit by taking navigation actions (`north`, `east`, `south`, `west`). Here our goal is to recover the rewards R_i that a single agent associates with each exit, along with agent budget parameters η, given observations of the agent's behavior.

4.1 Agent Model

We assume that agents select navigation actions using a heuristic with a known functional form, in which the value of a state s is approximated as:

$$V(s) = \frac{\sum_i R_i e^{-\|s-e_i\|_1 \cdot R_i}}{\sum_i e^{-\|s-e_i\|_1 \cdot R_i}} \tag{4}$$

where $\|s - s'\|_1$ measures the Manhattan distance between a pair of states (i.e. maze positions). Intuitively, we model agents as "attending" to each exit in proportion to both their distance and rewardingness. We assume that agents use this heuristic to perform **truncated breadth-first search**. In a state s, agents first estimate the value of each action a by computing the value of the best state reachable in β_{runtime} actions, starting with a. Formally:

$$Q_{\text{depth}}(a \mid s) = \max_{\tau:\tau_0=a, |\tau|=\beta_{\text{runtime}}} V(\tau_{\beta_{\text{runtime}}}) \tag{5}$$

where τ_0 and $\tau_{\beta_{\text{runtime}}}$ respectively denote the first and last actions in the trajectory τ. Finally, agents select actions in proportion to these Q-values [13]:

$$\pi(a \mid s; \beta_{\text{runtime}}, R) \propto e^{Q(a|s)} \tag{6}$$

With this agent parameterization, Eq. (2) can be computed efficiently:

Proposition 1. *Truncated breadth-first Search (TBFS) is an anytime inference algorithm. (Represent each inference state f_β as the set of frontier states and values reachable from each starting action. To compute $f_{\beta+1}$, add the unexplored children of these states to the set.)*

4.2 Data

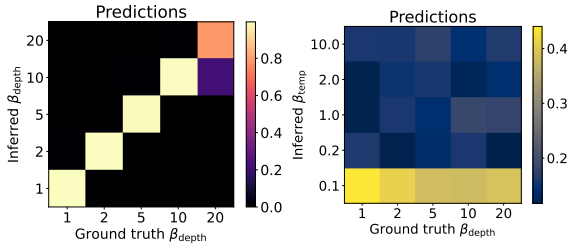
In this pedagogical example, we treat the agent model in Section 4.1 as the true data-generating process. We fix a set of parameters R_i and β_{runtime}, generate a collection of synthetic trajectories using Eq. (6), then attempt to recover these parameters using Eq. (2). (This allows us to validate the feasibility of our approach under ideal conditions—later sections will apply it to real datasets of human-generated actions). In particluar, we generate 5 agents with depth budgets of 1, 2, 5, 10, and 20 respectively. Example trajectories from each of these agents are depicted in Fig. 2.

4.3 Evaluation

We compare L-IBMs with a Boltzmann model in which agents select actions according to:

$$Q_{\text{temp}}(a \mid s) = \beta_{\text{temp}} \cdot \max_{\tau:\tau_0=a} R(\tau) \tag{7}$$

where $R(\tau)$ denotes the *final* reward obtained along the complete trajectory τ (i.e. upon reaching some exit R_i). We also compare to simple baselines in which the agent performs truncated search up to a constant (not inferred) depth. We evaluate these models in two ways:



(a) (b)

Figure 3: Inferred parameters η_i (distributions over β). (a) L-IBM almost perfectly recovers these parameters, while (b) the Boltzmann model shows no significant differences across inferred β_{temp}.

Approach	Accuracy
Fixed $\beta_{\text{runtime}} = 0$	5
Fixed $\beta_{\text{runtime}} = 20$	16
Inferred β_{temp}	20
Inferred β_{runtime}(L-IBM)	**44**

Table 1: Agent action prediction accuracies. L-IBM significantly outperforms baselines.

Predicting actions. In held-out states, we evaluate models' **exact-match** accuracy in predicting an agent's next action. Results are shown in Table 1. Models that assume a constant depth perform worst. While Boltzmann models are better able to predict agents' next actions than these fixed-budget models, they are significantly outperformed by L-IBM.

Predicting rewards. We also evaluate whether inferred prior distributions over β recover the true values used to generate the data. Results for L-IBM and the Boltzmann model are shown in Fig. 3b. It can be seen that L-IBM almost perfectly recovers these parameters (suggesting that prediction errors in Table 1 result entirely from errors in the inferred reward parameters R_i). Meanwhile, the Boltzmann model shows no significant differences in inferred β_{temp} across depth budgets, emphasizing the discrepancy between the two mdoels of suboptimality.

Together, these results show that L-IBM is computationally tractable and capable of making accurate predictions and inferring meaningful parameters in simple search problems. In the remainder of this paper, we apply it to modeling real human behavior in more complex decision-making tasks.

5 Pragmatic Language Understanding with Rational Speech Acts

The next task we consider focuses on **pragmatic language understanding**—inferring speakers' communicative intents from their utterances. Humans readily produce and understand language in ways that is incompatible with their literal meanings. In Table 2, for example, a color that would be described on its own by most speakers *purple* is instead labeled *blue* in some contexts.

A large body of work in cognitive science models this kind of context-based language understanding as the result of an iterative inference process [8, 9]: for example, in Row 2 of Table 2, a speaker might choose to describe the highlighted color as *blue* by reasoning that a naïve listener might resolve *purple* to the second color in the row. A more sophisticated listener, in turn, can predict this speaker behavior, and successfully infer the intended meaning. But this kind of recursive reasoning about other agents can be computationally demanding, and requires sophisticated internal models of other language users. Experimental evidence suggests that is deployed selectively, and to different degrees by different language users [10].

Our experiments focus on a **reference game** of exactly the kind depicted in Table 2 [25]. Reference games are a staple of research on pragmatic language use. They are played between a listener and a speaker. Both the listener and speaker observe a set of candidate referents (e.g. colors). The speaker is privately given one of the colors as a target; they must then produce a natural language utterance for the listener. Finally, the listener selects a color patch, and both players win if they agreed on the target.

By fitting an L-IBM to utterances and choices in human reference games, we may infer (1) whether we can infer whether humans are engaged in pragmatic reasoning from behavior alone, (2) whether there are differences between players in their ability to reason about their interlocutors, and (3) whether these differences actually predict

	Context	Utterance
1.		*purple*
2.		*blue*
3.		*blue*

Table 2: Example of the reference color (within the black box) and the two distractor colors, along with the utterance produced by a speaker from the colors in context task [25]. Notice how the context affects the utterance, even as the reference color remains fixed.

communicative success (i.e. whether players with greater inference budgets are better at making themselves understood).

5.1 Agent Model

We build on the Rational Speech Acts (RSA) model of Frank & Goodman [8]. This model frames communication as one in which Bayesian listeners and speakers reason recursively about each others' beliefs in order to select utterances and actions. The starting point of RSA is a **literal listener** π_L^0 that maps utterances u to actions according to their non-contextual meanings. (In Table 2, a literal listener hearing the word *purple* might choose randomly between the first two colors in the second row, as both would be reasonably described as purple out of context.) The literal listener may be implemented by any model (e.g. a lookup table or a neural network; 1) with parameters θ. Next, given a reference target t, a **pragmatic speaker** π_S chooses an utterance in proportion to the probability that it will cause a literal listener to take the right action:

$$\pi_S^1(u \mid t) \propto p(\pi_L^0 \text{ selects } t \text{ upon hearing } u) = \pi_L^0(t \mid u) \tag{8}$$

(RSA speakers are standardly parameterized with an additional Boltzmann rationality parameter, which we will discuss momentarily.) Finally, **pragmatic listeners** observe speaker utterances u, and reason about which reference targets were most likely to have produced those utterances:

$$\pi_L^1(t \mid u) = p(\pi_S^1 \text{ intends to signal } t \mid u) \propto \pi_S^1(u \mid t)\, p(t) \tag{9}$$

Crucially, this process may be repeated, with speakers π_S^i reasoning about ever-more-sophisticated speakers π_L^{i-1}, etc. But how many rounds of iteration actually explain human behavior? In the latent inference budget model framework, we may model this by embedding RSA inside an L-IBM, with the budget β parameterizing the number RSA iterations performed by each agent:

$$\pi_S(u \mid t; \theta, \eta) = \sum_{\beta} \beta_{\text{runtime}}(\beta \mid \eta) \pi_S(u \mid t; \theta, \beta) \tag{10}$$

$$\pi_S(u \mid t; \theta, \beta) = \pi_S^{\beta}(u \mid t) \tag{11}$$

(and analogously for π_L.)

Proposition 2. *Rational Speech Acts (RSA) is an anytime inference algorithm. (Each inference state f_β is π_S^β or π_L^β. Each of these can be derived from the other in constant time via Eqs.8–9.)*

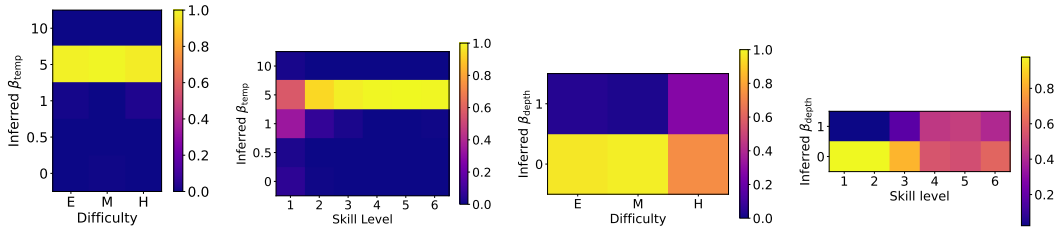
5.2 Data

For this task, we use the data collected by [25]. Each game consists of roughly 50 rounds played between a human speaker and a human listener. In each round, the speaker observes a target color along with two distractors. The speaker produces an utterance and the listener has to click on one of the colors. The dataset consists of 46,994 rounds across 948 games. We create a 80/10/10 split across train, valid and test sets. Monroe et al. stratify the dataset into three difficulties (easy, medium and difficult) based on perceptual similarity between colors and distractors. Because each game is annotated with a unique identifier for both the speaker and the listener, we may further stratify the dataset according to *player skill*: we compute the fraction of games won by each (speaker, listener) pair, then group these pairs into six buckets according to their win rate percentile relative to other players. This allows us to examine the relationship between inference budget and both task difficulty and communicative success.

5.3 Models

Following [25], we implement the literal listener π_L^0 using a transformer model that receives all three colors (represented as HSL vectors) and a natural language utterance as input, and predicts the index of the target color as output. We embed this listener model within the speaker–listener recursion defined by Eq. (9), then train it end-to-end (with budget parameters η_i) on the Colors in Context data using Eq. (2).

The constant of proportionality in Eq. (8) involves a sum over all natural language strings, which is cannot be computed efficiently. Here, also following [25], we perform a sampling-based approximation: we train a *literal speaker* model to generate plausible utterances, then sum over a finite number of such samples to obtain a distribution over strings. See McDowell & Goodman [21] for



(a) (b) (c) (d)

Table 3: Inferred distributions over β in RSA. X-axis indicates the difficulty level (**E**asy, **M**edium, **H**ard) or the player skill level (between 1 and 6, 6 being the most skilled players). The inferred β_{temp} across difficulty in a) and player skill in b) is not as meaningful as it is for β_{runtime} in c) and d). c) When separating games by difficulty, L-IBM infers that the non-literal speaker is employed only for the hardest condition. d) When separating games by player skill, we infer that the weakest players can be modeled exclusively as literal speakers, while stronger players can be modeled as a mix of literal and pragmatic speakers.

more details. The literal speaker is parameterized identically to the literal listener, but outputs strings rather than color indices.

In experiments investigating the relationship between task difficulty and inference budget, we fit one η_i *per condition* (easy, medium, hard). In experiments investigating the relationship between communicative success and inference budget, we fit one η_i *per skill level* (between 1 and 6).

5.4 Evaluation

Standard implementations of RSA modifies Eq. (8) to include a Boltzmann parameter for speakers:

$$\pi_S^i(u \mid t; \beta) \propto \exp\{\beta_{\text{temp}} \log \pi_L^{i-1}(t \mid u)\} \tag{12}$$

Like our β_{runtime}, this parameter is intended to model possibly sub-optimal behavior on the part of speakers and listeners. We compare an L-IBM to a model of this form. In particular, we fix the number of RSA iterations to one, use the same data as above to estimate literal listener parameters jointly with a prior distribution over β_{temp}:

$$\pi_S^1(u \mid t; \theta, \eta) = \sum_{\beta} p_{\text{temp}}(\beta \mid \eta) \pi_S^1(u \mid t; \beta) \tag{13}$$

where π_S^1 is defined as in Eq. (12).

Table 4 shows different models' ability to predict the target referent given human speaker utterances. Consistent with the findings of [25], because even literal models have access to all three referents, all model variants can achieve good task performance. When we look at inferred values for β_{runtime} and β_{temp}, however, we begin to see significant differences between models. When stratifying games by *difficulty*, we infer that the non-literal speaker is employed only for the hardest conditions. When stratifying games by *player skill*, we infer that the weakest players can be modeled exclusively as literal speakers, while stronger players can be modeled as a mix of literal and pragmatic speakers. To the best of our knowledge, this is the first example of an RSA-type model being used to infer individual differences in pragmatic language use within a speaker population; we expect that these tools may be of independent interest to the cognitive science community. Additional experiments, predicting the object that the listener picked instead of the one the speaker is presented can be found in Appendix C.

6 Playing Chess with Monte-Carlo Tree Search

Finally, we turn from cooperative to adversarial decision-making tasks. We focus on chess, a popular two-player sequential game widely used as a benchmark for AI systems. Here, we are interested in modeling human chess play—specifically, observing data from a population of sub-optimal agents with a common reward function (winning the game) and attempting to infer those agents' computational constraints. In human human play, there can be numerous sources of such constraints: a player paired against a strong opponent will likely to plan for longer than against a weaker opponent; some variants (like blitz chess) deliberately limit players' time-per-move (and, we might expect, the

Model	Type	Accuracy
$\beta_{\text{runtime}} = 0$ (Literal listener)	-	83.3
$\beta_{\text{runtime}} = 1$	-	83.0
Inferred β_{temp}	player skill	83.9
Inferred β_{runtime} (L-IBM)	player skill	**84.0**
Inferred β_{temp}	difficulty	83.5
Inferred β_{runtime} (L-IBM)	difficulty	82.7

Table 4: Performance of different RSA models in predicting the speaker target. All models (including literal models and fixed-depth RSA models) achieve similar predictive performance—because even literal models have access to all three referents, all model variants can achieve good task performance.

Model	Type	Accuracy
IL	-	42.06
MCTS	-	43.64
L-IBM	Active Elo	**44.17**
Inferred β_{puct}	Active Elo	43.77
L-IBM	Opponent Elo	**44.17**
Inferred β_{puct}	Opponent Elo	43.84
L-IBM	Time Control	**44.15**
Inferred β_{puct}	Time Control	43.61

Figure 5: Accuracy of predicting an agent's next action in chess. Models with MCTS outperform the depth-0 (imitation learning) baseline. Learning sub-population-specific β enhances performance, with L-IBM-based learning of β_{runtime} consistently outperforming β_{puct} by a slight margin.

quality of their plans). Given a dataset of human games played under different time constraints and player strengths, can we use L-IBM to model variability in players' decisions across game states?

6.1 Agent Model

In this work, we model chess players as selecting actions using **Monte Carlo tree search** (MCTS). Recent work [16] has shown that MCTS is a good model of strong human players. Here, following [29, 28, 16, 12], we implement one of the most common modern forms of MCTS, which uses a value function V predicting the expected total future reward and a policy prior π^0 to guide exploration. At a high level, MCTS operates by incrementally growing a game tree starting at the root node, repeatedly picking some path to explore down the tree, performing a value function evaluation and then walking back up the tree updating all the value estimates based on that result. At each node, MCTS treats action selection as a multi-armed bandit problem. We use a standard exploration policy [19]: during inference at each node of the search tree, we choose actions according to:

$$\arg\max_a \; Q_t(a \mid s) + \beta_{\text{puct}} \pi^0(a \mid s) \frac{\sqrt{\sum_b N(s,b)}}{N(s,a)+1} \tag{14}$$

where $Q_t(s,a)$ is the estimated expected future reward for i from playing action a in state s at iteration t, the visit count $N(s,a)$ is the number of times a has been explored from s, $\pi^0(a \mid s)$ is an "anchor" policy, and β_{puct} is a tunable parameter trading off exploration versus exploitation. After expanding β_{runtime} nodes of this tree, an agent's final action is sampled from a distribution:

$$\pi(a \mid s; \beta_{\text{runtime}}) = \beta_{\text{puct}} \frac{\sqrt{\beta_{\text{runtime}}}}{N(s,a)+1} \frac{\pi^0(a|s)}{\gamma - Q_{\beta_{\text{runtime}}}(a \mid s)} \tag{15}$$

where γ is chosen such that π forms a proper probability distribution.

Proposition 3. *Monte-Carlo tree search (MCTS) is an anytime inference algorithm. (Let each inference state f_β be the tree of nodes and visitation counts after β evaluations. This tree is refined by evaluating Eq.* (15) *once.)*

With $\pi(a \mid s; \beta_{\text{runtime}})$ as defined above, we may instantiate an L-IBM for MCTS:

$$\pi^{\text{depth}}(t|u;\eta,\theta) = \sum_{\beta_{\text{runtime}}} p_{\text{budget}}(\beta_{\text{runtime}} \mid \eta_i) \cdot \pi(a; s, \beta_{\text{runtime}}) \tag{16}$$

6.2 Data

We use similar data to previous models of human chess play by McIlroy-Young et al. [22], Jacob et al. [16], McIlroy-Young et al. [23]. Our experiments use two different datasets. First, a dataset D_{large} containing roughly 6 million moves; second, a dataset D_{small} containing roughly 75,000 moves. D_{small} includes metadata describing players' Elo ratings (a measure of strength) and game formats (the amount of time players had to select moves). See Appendix B for details.

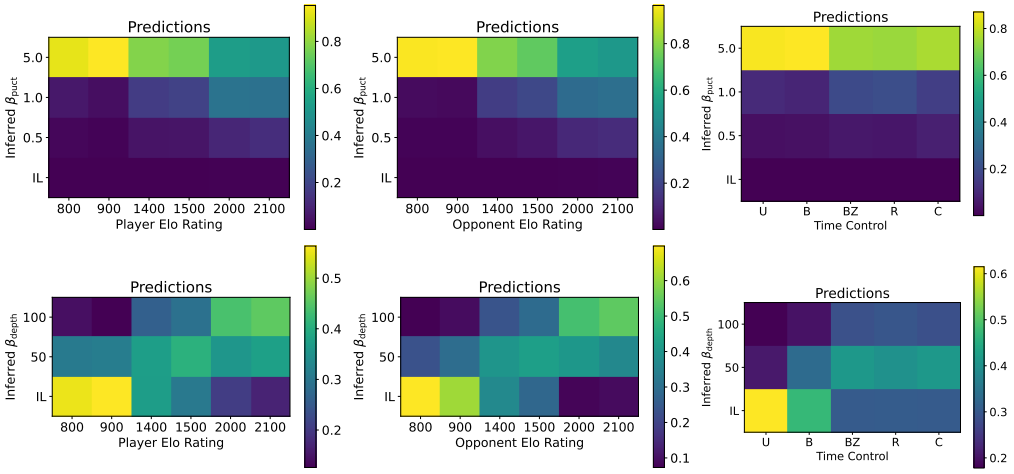


Figure 6: Inferred distributions over β in Chess using MCTS. X-axis indicates the player Elo rating, opponent elo rating buckets and time control: Ultra Bullet (U), Bullet (B), Blitz (BZ), Rapid (R) and Classical (C). The top row depicts the distributions for β_{puct} and the bottom row depicts the distributions for β_{runtime}. When the player's or opponent's strength increases, β_{runtime} infers greater depths. This pattern also holds true as the time control extends. β_{puct} displays a similar pattern, as the agents or opponents get stronger, or as the time control extends, β_{puct} suggests lower values, placing greater reliance on the search Q-values.

6.3 Modeling details

We train the base initial policy π_0 and a value model $\tilde{v}_0$ as two different output heads of a deep neural network using imitation learning on the large dataset split D_{large}. Our architecture is a 4-block residual network similar to those used in prior work [22, 16, 23]. Unlike previous sections, we do not learn the value functions jointly with p_{budget}. Instead, we first learn a single value function from D_{large}, then fit $p_{\text{budget}}(\beta_{\text{puct}} \mid \eta_i)$ and $p_{\text{budget}}(\beta_{\text{runtime}} \mid \eta_i)$. We investigate three ways of stratifying players into sub-populations: player Elo (a proxy for player skill), and opponent Elo and time control (both proxies for task difficulty). As in Section 5, we estimate a separate η_i for each group within each stratified dataset.

6.4 Evaluation

Unlike in the two domains studied above, there is already an established literature on modeling sub-optimal behavior via MCTS outside the Boltzmann framework. The most successful current approach models individual differences in play [16] by fitting β_{puct}. We thus compare to a baseline in which η_i parameterizes a distribution over values of β_{puct} rather than tree expansions.

Accuracy (in terms of top-one predictions and negative log-likelihood) is reported in Fig. 5. As in past work, we find that models that with explicit search outperform depth-0 (imitation-learning) baseline. Learning sub-population specific β improves the performance even further, with L-IBM-based learning of β_{runtime} consistently outperforming β_{puct} by a small margin.

Inferred budget parameters are shown in Fig. 6. Here, we observe that as the player strength or the opponent strength increases as measured by the Elo ratings, β_{runtime} infers higher depths. We also observe the same as the time control increases: β_{runtime} infers higher depths as the duration of each move of the game increases. β_{puct} shows a weaker, but similar trend: as the agents or opponents get stronger, or as the time control increases, β_{puct} infers lower values of β_{puct}, indicating that players are deviating from the prior and are relying more on the search Q-values.

7 Conclusion

We have described latent inference budget models, a family of approaches for modeling agents acting to achieve unknown goals subject to unknown constraints on their inferential capabilities. Instead of assuming either global optimality of decision-making or uniform suboptimality, our approach explicitly infers the runtime that agents devote to approximate inference. This paradigm is applicable to all anytime inference algorithms. In three domains—maze navigation, pragmatic

language understanding, and playing chess—we demonstrated that it can outperform classical models of bounded rationality while imputing meaningful measures of human skill and task difficulty.

References

[1] Jacob Andreas and Dan Klein. Reasoning about pragmatics with neural listeners and speakers. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1173–1182, 2016.

[2] Mark Beliaev, Andy Shih, Stefano Ermon, Dorsa Sadigh, and Ramtin Pedarsani. Imitation learning by estimating expertise of demonstrators. In *International Conference on Machine Learning*, pp. 1732–1748. PMLR, 2022.

[3] Mark Boddy and Thomas L Dean. *Solving time-dependent planning problems*. Brown University, Department of Computer Science, 1989.

[4] Frederick Callaway, Bas van Opheusden, Sayan Gul, Priyam Das, Paul M Krueger, Thomas L Griffiths, and Falk Lieder. Rational use of cognitive resources in human planning. *Nature Human Behaviour*, 6(8):1112–1125, 2022.

[5] Colin F Camerer, Teck-Hua Ho, and Juin-Kuan Chong. A cognitive hierarchy model of games. *The Quarterly Journal of Economics*, 119(3):861–898, 2004.

[6] Thomas L Dean and Mark S Boddy. An analysis of time-dependent planning. In *AAAI*, volume 88, pp. 49–54, 1988.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[8] Michael C Frank and Noah D Goodman. Predicting pragmatic reasoning in language games. *Science*, 336(6084):998–998, 2012.

[9] Michael Franke. Game theoretic pragmatics. *Philosophy Compass*, 8(3):269–284, 2013.

[10] Michael Franke and Judith Degen. Reasoning in reference games: Individual-vs. population-level probabilistic modeling. *PloS one*, 11(5):e0154854, 2016.

[11] Thomas L Griffiths, Frederick Callaway, Michael B Chang, Erin Grant, Paul M Krueger, and Falk Lieder. Doing more with less: meta-reasoning and meta-learning in humans and machines. *Current Opinion in Behavioral Sciences*, 29:24–30, 2019.

[12] Jean-Bastien Grill, Florent Altché, Yunhao Tang, Thomas Hubert, Michal Valko, Ioannis Antonoglou, and Rémi Munos. Monte-carlo tree search as regularized policy optimization. In *International Conference on Machine Learning*, pp. 3769–3778. PMLR, 2020.

[13] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *International conference on machine learning*, pp. 1352–1361. PMLR, 2017.

[14] Quentin JM Huys, Neir Eshel, Elizabeth O'Nions, Luke Sheridan, Peter Dayan, and Jonathan P Roiser. Bonsai trees in your head: how the pavlovian system sculpts goal-directed choices by pruning decision trees. *PLoS computational biology*, 8(3):e1002410, 2012.

[15] Quentin JM Huys, Níall Lally, Paul Faulkner, Neir Eshel, Erich Seifritz, Samuel J Gershman, Peter Dayan, and Jonathan P Roiser. Interplay of approximate planning strategies. *Proceedings of the National Academy of Sciences*, 112(10):3098–3103, 2015.

[16] Athul Paul Jacob, David J Wu, Gabriele Farina, Adam Lerer, Hengyuan Hu, Anton Bakhtin, Jacob Andreas, and Noam Brown. Modeling strong and human-like gameplay with kl-regularized search. In *International Conference on Machine Learning*, pp. 9695–9728. PMLR, 2022.

[17] Hong Jun Jeon, Smitha Milli, and Anca Dragan. Reward-rational (implicit) choice: A unifying formalism for reward learning. *Advances in Neural Information Processing Systems*, 33: 4415–4426, 2020.

[18] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[19] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *European conference on machine learning*, pp. 282–293. Springer, 2006.

[20] R Duncan Luce. *Individual choice behavior: A theoretical analysis*. Courier Corporation, 2012.

[21] Bill McDowell and Noah Goodman. Learning from omission. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 619–628, 2019.

[22] Reid McIlroy-Young, Siddhartha Sen, Jon Kleinberg, and Ashton Anderson. Aligning superhuman ai with human behavior: Chess as a model system. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1677–1687, 2020.

[23] Reid McIlroy-Young, Russell Wang, Siddhartha Sen, Jon Kleinberg, and Ashton Anderson. Learning models of individual behavior in chess. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 1253–1263, 2022.

[24] Andrew N Meltzoff. Understanding the intentions of others: Re-enactment of intended acts by 18-month-old children. *Developmental psychology*, 31(5):838, 1995.

[25] Will Monroe, Robert XD Hawkins, Noah D Goodman, and Christopher Potts. Colors in context: A pragmatic neural model for grounded language understanding. *Transactions of the Association for Computational Linguistics*, 5:325–338, 2017.

[26] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

[27] Stuart Russell and Eric Wefald. Principles of metareasoning. *Artificial intelligence*, 49(1-3): 361–395, 1991.

[28] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[29] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

[30] Bas van Opheusden, Ionatan Kuperwajs, Gianni Galbiati, Zahy Bnaya, Yunqi Li, and Wei Ji Ma. Expertise increases planning depth in human gameplay. *Nature*, pp. 1–6, 2023.

A Training hyperparameters

We will detail the training hyperparameter details in this section.

A.1 Maze

All models in Section 4 were trained using the Adam optimizer [18], where the learning rates were sweeped across the following values $[1.0, 0.5, 1e{-}1, 0.05, 1e{-}2, 5e{-}3, 1e{-}3, 5e{-}4, 1e{-}4, 5e{-}5]$ for 50 epochs. The values presented in Table 1 were picked from the model with the best validation accuracy across the learning rates.

A.2 Colors in context

The models trained in Section 5 are based on the transformer architecture and trained from scratch. The speaker model was trained based on the T5 model [26] with the following hyperparameters described in Table 5. The speaker was trained with a batch size of 64 using the Adam optimizer with learning rate $1e-4$ for 25 epochs.

Parameter	Value
Number of Layers	4
Number of Heads	4
Model Dimension	32
Key-Value Dimension	16
Feedforward Dimension	32

Table 5: Hyperparameter configuration of the speaker model based on T5 [26].

Parameter	Value
Hidden Size	64
Number of Hidden Layers	4
Number of Attention Heads	4
Intermediate Hidden Size	256

Table 6: Hyperparameter configuration of the listener model based on BERT.

All the listener models were based on the BERT [7] model with the configuration described in Table 6. The listener models were trained using Adam and the learning rates were sweeped across the following values $[1e-3, 5e-4, 1e-4, 5e-5]$ for upto 50 epochs. The values presented in Table 8 and Table 4 were picked from the model with the best validation accuracy across the learning rates.

A.3 Chess

The value and policy network used in Section 6 are based on an architecture that is a 4-block residual network similar to those used in prior work [22, 16, 23]. The policy and value network was trained using Adam with a learning rate of 0.001, a batch size of 4096 and for upto 30 epochs. The epoch used in the rest of the section was picked based on the validation accuracy.

In the second set of fine-tuning experiments, for every set of conditioning type, a simple feedforward network was trained using Adam with a batch size of 512. The models in **??** were picked by selecting the learning rates between $1e-3, 5e-4, 1e-4, 5e-5$ with the best validation accuracy.

B Chess Data

D_{large} consists of 5,974,872 moves in the training split, 60,968 in the validation split and 60,969 moves in the test set. These data points were randomly sampled from the January, 2019 database release of a chess website (lichess). D_{small} consists of 50,000 moves in the training split, 12,041 moves in the validation split and 12,040 moves in the test split. These data points were randomly sampled from the February, 2019 lichess database release but filtering such that only those players with Elo ratings in the following buckets were considered: [800-1000], [1400-1600] and [2000-2200].

The dataset contains 5 different types of time control. In increasing duration, they are **Ultra Bullet**, **Bullet**, **Blitz**, **Rapid** and **Classical** (see Table 7).

The time controls used in our work have estimated durations that are defined in Table 7:

Time control	Estimated Duration (seconds)
UltraBullet	< 29
Bullet	< 179
Blitz	< 479
Rapid	< 1499
Classical	≥ 1500

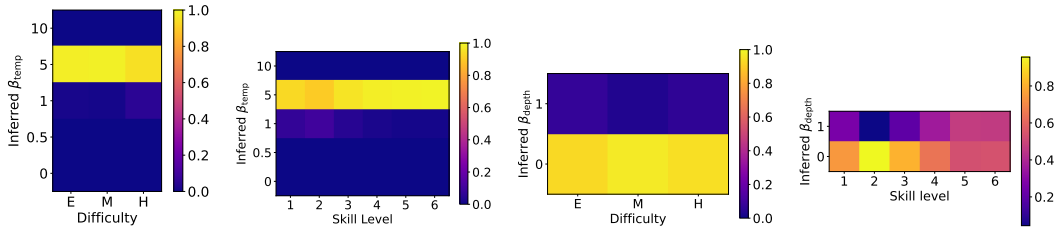
Table 7: Estimated game durations across different time controls.

C Additional Experiments: Colors in context

In this section, we include additional experiments for the pragmatics domain where we train the models to predict the object that the listener picks. We present the results of a similar set of experiments as in Section 5 in Table 8 and Fig. 8. We specifically note that the inference based approaches outperform the baselines in this setting.

Model	Type	Accuracy
$\beta_{\text{runtime}} = 0$ (Literal listener)	-	80.4
$\beta_{\text{runtime}} = 1$	-	81.8
Inferred β_{temp}	player skill	82.3
Inferred β_{runtime} (L-IBM)	player skill	**83.1**
Inferred β_{temp}	difficulty	82.7
Inferred β_{runtime} (L-IBM)	difficulty	82.1

Table 8: Performance of different RSA models in predicting the speaker target. The β based models outperform the baseline models: literal models and fixed-depth RSA models.



(a) (b) (c) (d)

Figure 8: Inferred distributions over β in RSA, with the listener target. X-axis indicates the difficulty level (**E**asy, **M**edium, **H**ard) or the player skill level (1 - 6, 6 being the most skilled players). The inferred β_{temp} across difficulty in a) and player skill in b) is not as meaningful as it is for β_{runtime} in d). When separating games by player skill, we infer that the weakest players can be modelled with a smaller mix towards pragmatic speakers compared to stronger players