Bayesian Execution Skill Estimation

Christopher Archibald, Delma Nieves-Rivera
archibald@cse.msstate.edu, din7@msstate.edu
Computer Science and Engineering
Mississippi State University

Abstract

The performance of agents in many domains with continuous action spaces depends not only on their ability to select good actions to execute, but also on their ability to execute planned actions precisely. This ability, which has been called an agent's execution skill, is an important characteristic of an agent which can have a significant impact on their success. In this paper, we address the problem of estimating the execution skill of an agent given observations of that agent acting in a domain. Each observation includes the executed action and a description of the state in which the action was executed and the reward received, but notably excludes the action that the agent intended to execute. We previously introduced this problem and demonstrated that estimating an agent's execution skill is possible under certain conditions. Our previous method focused entirely on the reward that the agent received from executed actions and assumed that the agent was able to select the optimal action for each state. This paper addresses the execution skill estimation problem from an entirely different perspective, focusing instead on the action that was executed. We present a Bayesian framework for reasoning about action observations and show that it is able to outperform previous methods under the same conditions. We also show that the flexibility of this framework allows it to be applied in settings where the previous limiting assumptions are not met. The success of the proposed method is demonstrated experimentally in a toy domain as well as the domain of computational billiards.

1 Introduction

Many real-world physical domains require agents to both plan and execute continuous actions. These planned actions can generally not be executed with perfect precision, resulting in some amount of execution error which will vary from agent to agent. Robust agents in these domains need to plan actions taking into account their execution noise. Continuous domains with these properties include robotics settings, where angles and velocities are selected from continuous ranges; human settings, where a human either moves itself or causes other objects to move, again in continuous space; and computational settings, where abstract toy domains or simulations of games with continuous action spaces are modeled and used as a test-bed for algorithms and approaches. In each of these settings, an important characteristic of an agent is the probability distribution over action execution errors that describe their ability to precisely execute actions.

This paper focuses on the problem of estimating this agent property given observations of the same agent acting in a continuous domain. This problem was first introduced in (Archibald and Nieves-Rivera 2018). We are interested in this problem for the potential impact it has in helping model and analyze agents in adversarial continuous domains, as well as the potential applications for assessing and giving feedback on human skill levels in many real-world settings; including sports, law enforcement, and elderly assistance.

Along with introducing the problem, we demonstrated that it is possible to estimate an agent's execution ability, which we called execution skill, under certain assumptions (Archibald and Nieves-Rivera 2018). The central assumption in that work was that the agent utilized a perfectly rational planning component, which is similar to assumptions made in work on plan recognition in continuous domains (Kaminka, Vered, and Agmon 2018).

In this paper, we introduce a Bayesian approach to the execution skill estimation problem and show that it outperforms the previous approach in the same toy domain. We then demonstrate how this Bayesian approach can be naturally applied to domains where the rationality assumption does not hold and show how it can effectively estimate execution skill in the domain of computational billiards.

The remainder of the paper proceeds as follows, in section 2 relevant related work is discussed. Section 3 presents the definition of the problem and previous results. Section 4 introduces the novel Bayesian approach that is the focus of this work. Sections 5, 6, and 7 present and discuss experimental work in two domains, and section 8 concludes with discussion and future work.

2 Related Work

Many topics related to the execution skill estimation problem have been studied.

The topics of skill in games has been investigated by several researchers (Larkey et al. 1997; Dreef, Borm, and Genugten 2002; Borm and Genugten 2001; Dreef, Borm, and van der Genugten 2004), although skill as these works define it refers to a characteristic of the game itself, and is

meant to indicate how much the outcome of the game relies on the actions of the players as opposed to random chance.

Research in multiple areas has investigated the topic of execution uncertainty. Among them we can find general games (Bowling and Veloso 2004; Archibald and Shoham 2011), where agents cannot execute actions with certainty and security games (Yin et al. 2011; Jiang et al. 2013), where it cannot always be assumed that actions undertaken by the security agents will be executed as planned. Auction settings (Van Valkenhoef et al. 2010), when goods, services, or payments may fail to occur as desired, are an example too.

The notions of execution skill and strategic skill were first introduced and investigated in the domain of computational pool in (Archibald, Altman, and Shoham 2010), where agents with varying strategic and execution skill were evaluated to investigate how the notions interact. Extensive previous work in the computational pool domain exists (Archibald, Altman, and Shoham 2016; Landry, Dussault, and Beaudry 2015; Smith 2007), but these are focused on the problem of selecting a good action, given a skill level. This work started in order to spur the development of methods to plan shots for a pool robot (Greenspan et al. 2008).

Another recent domain that has the same continuous action and state space characteristics as billiards is curling, which has been investigated again from the action selection perspective (Yee, Lisỳ, and Bowling 2016; Ahmad, Holte, and Bowling 2016).

Much work also exists in the area of estimating properties of an opponent, or opponent modeling. The majority of this work has been done in imperfect information games like poker (Billings et al. 1998; Bard et al. 2015; 2013; Davis, Burch, and Bowling 2014), but this work focuses on strategic characteristics and limitations of the opponents and there is no execution uncertainty. Additional work exists in general multi-agent systems (Carmel and Markovitch 1995), real-time strategy games (Schadd, Bakkes, and Spronck 2007), and in n-player games (Sturtevant, Zinkevich, and Bowling 2006).

The Bayesian reasoning techniques we apply to this problem have been well established in artificial intelligence, and can be found in many textbooks, notably (Russell and Norvig 2009) and (Thrun, Burgard, and Fox 2005).

3 Problem Definition and Background

The *execution skill estimation problem* was introduced in (Archibald and Nieves-Rivera 2018). This problem consists of estimating parameters of an agent's execution noise distribution, given only observations of the actions taken by the agent in an environment. In this section this problem will be precisely defined and the previous work introduced.

Setting

The execution skill estimation problem focuses on domains that can be modeled as traditional Markov Decision Processes (MDPs) with continuous action spaces. These domains can be described by the tuple $\langle S, A, R, P \rangle$, which is composed of a set of states S, a continuous set of possible actions $A \subset \mathbb{R}^m$, a rewards function R mapping a state and action combination to a real-valued reward, and a transition function P which specifies for any state and action combination a distribution over next states.

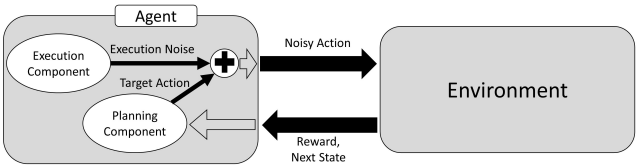

Figure 1: Agent with Execution Skill Interacting with Environment

An *agent* is an entity that executes actions in an MDP and receives rewards. The main focus of the current work is on two components of an agent, their strategic skill and their execution skill. Strategic skill refers to the method that the agent uses to select actions in a given state. This is represented as a policy $\pi : S \mapsto A$ which specifies, for any state $s \in S$ the action to perform in that state. $\pi(s)$ will be referred to as the *planned*, *intended*, or *target* action.

An agent's execution skill is represented by χ, a distribution over random perturbations $\epsilon \sim \chi$. A sample from χ is added to each attempted action before it is executed in the current state. The entire model is shown in Figure 1. At present we assume that an agent's execution skill distribution χ is independent of both the current state and the planned action, as this noise is meant to model uncertainties and imperfections in the agent over which the agent has no control. We additionally assume that each dimension of χ is independent of the others. These assumptions can be relaxed in future work, but they are utilized for the present work.

Every time an action is executed a sample is drawn from χ and the resulting ϵ value is added to the planned action. This perturbed action $\tilde{a} = a + \epsilon$ is then executed in the environment and the reward and next state are drawn randomly from the distributions $R(\cdot \mid s, \tilde{a})$ and $P(\cdot \mid s, \tilde{a})$ respectively.

At present, it is assumed that the distribution χ is fully known to the planning component of an agent. This allows χ to be considered zero-mean without loss of generality since for any other mean, the strategic component of the agent can compensate by making adjustments to the intended action so that the resulting distribution over executed actions is centered on the desired action. With these assumptions, the main property of interest describing an agent's execution skill is the standard deviation of χ in each dimension, which will be referred to as σ.

Problem Definition

An observation consists of a tuple (s, a, r, s') which specifies the state s that the agent was in, the action a that was *actually executed*, and the subsequent reward r and next state s' that resulted from the execution of a in state s. The *execution skill estimation problem* can now be defined. Is it possible to identify the execution skill parameter σ for an agent given only observations of its interactions with the environment? The main difficulty arises due to the fact that the only action that is observed is the executed action and

not the intended action. The interaction between the strategic skill and execution skill makes it challenging in general to estimate each component independently.

Consider an agent with minimal strategic skill that selects actions uniformly at random. In this situation it won't be possible to determine how much noise is product of the agent's imperfect ability to execute actions and how much is due to the random aiming.

Motivated by this observation, we originally focused on a constrained version of the execution skill estimation problem where all agents were perfectly rational and introduced a successful approach under these assumptions (Archibald and Nieves-Rivera 2018). A perfectly rational agent always selects the action that maximizes expected value.

Baseline Estimation Method: True Noise

If the observations included the *intended* actions along with the executed ones, identifying the execution skill level of an agent becomes trivial. For each one of the observations, the difference between the intended action and the executed one can be computed. The sample standard deviation of those differences will provide an estimate at any point in time. This method, called the *true noise* method (TN), was introduced as a baseline for comparison (Archibald and Nieves-Rivera 2018), and it will be used in like manner in this work. For real applications, it is not a feasible approach because we typically don't have access to the internal state and intentions of other agents.

Previous Estimation Method: Observed Reward

The method introduced in (Archibald and Nieves-Rivera 2018) focuses on the rewards that an agent receives from the environment in the observations, and is called the *observed reward* (OR) method. The method calculates, for each state observed, what the expected reward would be for a perfectly rational agent with each of a set of hypothesis execution skill levels. The mean expected reward is tracked for each hypothesis over all the observations. The execution skill estimate is produced by interpolating the observed mean reward of the agent into the set of mean expected rewards to produce an execution skill level that would be most likely to produce the observed rewards.

The main limitation of this method is that it assumes that the planning component of the agent is perfectly rational, or at least can be perfectly predicted, given an execution skill level. It also completely ignores the specific actions that were executed, focusing entirely on the rewards. The proposed Bayesian method is designed to address these limitations.

4 Bayesian Execution Skill Estimation

The Bayesian approach to execution skill estimation was initially motivated by the following question: Can estimation be done by focusing on the agent's executed actions, instead of simply the obtained rewards? The proposed Bayesian method is general enough to be applied both to settings with assumptions of perfect rationality, as in (Archibald and Nieves-Rivera 2018), as well as settings where this assumption does not hold.

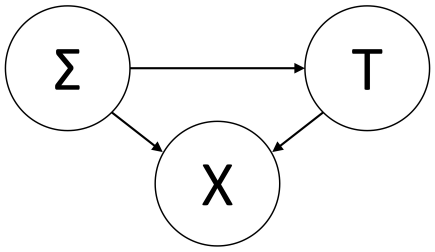

Figure 2: Action Planning and Execution Bayesian Network

The Bayesian approach (TBA) can be seen as performing probabilistic inference in the continuous Bayesian network shown in Figure 2, where Σ is a random variable corresponding to the agent's execution skill level, T is a random variable for the target, or intended, action, and X is a random variable that represents the executed action. The arrows indicate the probabilistic influence that these different random variables have on each other. The execution skill level Σ influences the target action T, and the final executed action X depends on both Σ and T.

The Bayesian Approach Derivation

The main goal of TBA is to estimate the probability of an execution skill level σ given an observed executed action x, using Bayes rule as follows:

$$P(\sigma \mid x) \propto P(x \mid \sigma)P(\sigma)$$

A single update will be derived, and this update will be used repeatedly, in an online manner, using the posterior $P(\sigma \mid x)$ after each update as the prior $P(\sigma)$ for the next. For notational simplicity the different time steps are not distinguished, as they are each processed sequentially. We also focus on a single dimension of action space for simplicity, but the method could be used for each dimension independently in the case of multiple dimensions.

The initial conditional distribution $P(x \mid \sigma)$ does not correspond conceptually to any components of the model and setting as described. Using the dependencies described in the Bayesian network of Figure 2 this can be rewritten in terms of modeled concepts as:

$$P(x \mid \sigma) = \int_{t \in T} P(t \mid \sigma)P(x \mid t, \sigma)$$

Here the first component $P(t \mid \sigma)$ corresponds to the planning component of the agent, and represents the probability that the agent will select t as a target action given its execution skill level is σ. $P(x \mid t, \sigma)$ reflects the true conditional distribution over executed actions given an intended action t and an execution skill level σ, and will directly correspond to the execution noise distribution χ.

The question now becomes, how can the planning distribution $P(t \mid \sigma)$ be represented and computed? Different assumptions about the setting will lead to different extensions of this approach.

Rational TBA Variant

In the constrained setting investigated in (Archibald and Nieves-Rivera 2018) the planning component was assumed to be perfectly rational. That can be represented within TBA by having $P(t \mid \sigma)$ be a distribution with all of the probability mass on a single target action for each σ. This single target action, which can be denoted by t_σ, will be the optimal action given an execution noise level of σ. This results in $P(x \mid \sigma) = P(x \mid t_\sigma, \sigma)$.

Beyond Rationality

How can TBA be moved beyond the assumption that the agent being observed has a completely rational planning component? Two extensions of TBA that broaden its applicability will now be presented.

First, instead of assuming that the optimal action for a given state and execution skill level can be calculated, we will assume that for a given state there are a set of focal actions which correspond to actions that are expected to generate high reward in the state. It is assumed that this set of actions can be quickly generated for a given state, without necessarily taking into account their robustness with respect to execution noise. The planning component can then be assumed to select one of these focal actions. The weakest assumption on how the planning component makes use of these focal actions is that it selects among them uniformly at random, and this is the variant that of TBA that is explored in this paper. Given that the number of focal actions in the set T is m, the $P(x \mid \sigma)$ distribution of TBA can now be rewritten as $P(x \mid \sigma) = \frac{1}{m} \sum_{t \in T} P(x \mid t, \sigma)$

The second extension addresses the possibility that an agent with imperfect planning skill might not even select one of the focal actions. In this case the distribution $P(x \mid \sigma)$ can be constructed as a mixture of β times the previous distribution ($\frac{1}{m} \sum_{t \in T} P(x \mid t, \sigma)$) and $(1 - \beta)$ times a uniform distribution over all actions. If W is the width of the action space then $P(x \mid \sigma)$ can be rewritten as

$$P(x \mid \sigma) = \beta \left(\frac{1}{m} \sum_{t \in T} P(x \mid t, \sigma) \right) + \frac{(1 - \beta)}{W}$$

Both of these extensions move the estimator away from relying on the assumption that the agent's target action selections can be perfectly predicted, given the state and execution skill level. As will be demonstrated, full TBA, or TBA with both of these extensions, can be applied to more interesting domains than the observed reward method, and on less rational agents.

Representation and Predictions

The estimates $P(\sigma \mid x)$ are represented using a discrete distribution over a set of hypothesis execution skill levels. Two different methods are investigated and compared for producing a single numerical execution skill level estimate from $P(\sigma \mid x)$ after each observation.

The first method is the *maximum a posteriori* (MAP) estimate $\tilde{\sigma}_{MAP}$, which selects the execution skill hypothesis with the highest posterior probability.

$$\tilde{\sigma}_{MAP} = \arg\max_{\sigma^i} P(\sigma^i \mid x)$$

The second method is the *expected execution skill* (EES) estimate $\tilde{\sigma}_{EES}$, which takes into account each hypothesis and its corresponding posterior probability. It computes the final estimate as follows:

$$\tilde{\sigma}_{EES} = \sum_i \sigma^i P(\sigma^i \mid x)$$

The experimental analysis of TBA will proceed as follows. First, in section 5 the rational TBA variant will be compared to the observed reward method in the constrained toy domain used in (Archibald and Nieves-Rivera 2018). Then, in section 6, full TBA method will be applied to the domain of computational billiards, a continuous domain with execution skill for which many computer agents have been designed.

5 Experimental Domain: 1D-Darts

The *one-dimensional darts*, or 1D-Darts, domain was introduced in (Archibald and Nieves-Rivera 2018) as a simple to domain to validate the OR method. Rational TBA will be compared to OR in a reproduction of this domain to compare the performance of the two methods.

The 1D-Darts domain consists of a single state and the one-dimensional continuous action space of $[-10, 10]$. The reward function is defined on the action space and alternates between having a value of 1 and 0. Given a 1D-Darts reward function, an agent selects an action, which then has noise from the agent's execution skill distribution added to it before execution. The reward is determined by where the executed action falls on the reward function.

The OR and TBA methods both used the same 17 hypothesis execution skill levels. In each state and for each execution skill level, the optimal action was computed by convolution of the reward function with the execution noise distribution using a resolution of 0.01. The OR method utilizes the expected reward obtained by this optimal action, while TBA uses the actual optimal action. For each experiment, the agent's true execution noise distribution was a zero-mean Gaussian with a standard deviation generated uniformly at random from the interval [0.5, 4.5]. Each experiment ran for 5,000 observations, and the mean squared error of the different methods across all 10,000 experiments are shown in Figure 3. The figure shows that both variants of TBA have significantly better performance than OR with fewer observations, converging around 2 orders of magnitude faster. In addition, TBA-EES has slightly better performance than TBA-MAP with few observations, but the two variants are indistinguishable as the number of observations grows. This shows that TBA is a better method than OR at the task of estimating the execution skill of a rational agent in 1D-Darts.

6 Experimental Domain: Billiards

The computational billiards game used for the experiments is the game of eight ball. This game consists of two players playing on a table initially filled with 15 target balls of three types: stripes, solids, and the eight ball. The players act by using a cue stick to send a cue ball towards the other balls on

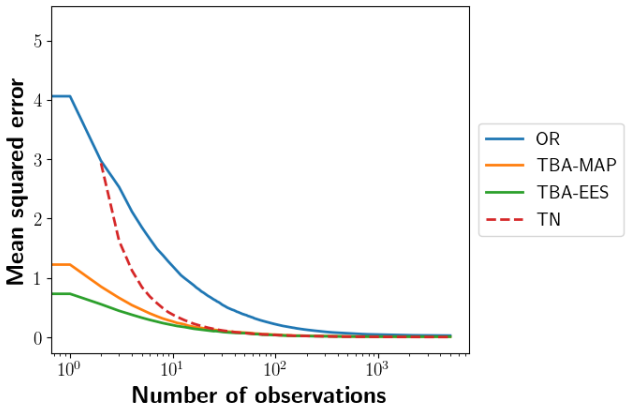

Figure 3: 1D-Darts Method Comparison

the table, with the aim of knocking the target balls into the pockets of the table. The goal of the players is to be the first to pocket all of either the striped or solid balls, followed by the eight ball. If a player's shot successfully pockets one of their balls, then that player gets to act again.

In computational billiards, the states of a billiards table are represented by the locations of all of the balls on the table. An action is specified by five real numbers that completely specify the position, orientation, and velocity of the cue stick. These parameters are v, ϕ, θ, a, and b. v represents the cue stick velocity upon striking the cue ball, ϕ represents the angle of the cue stick when viewed from overhead, θ represents the angle of the cue stick above the table, and a and b designate the position on the cue ball where the cue stick strikes, which plays a big role in imparting spin, or "english", to the cue ball. ϕ and θ are measured in degrees, v in m/s, and a and b are measured in millimeters. Each component of the selected action is perturbed by noise drawn from an independent zero-mean Gaussian distribution with a different standard deviation. The perturbed shot is then executed on the deterministic FastFiz physics simulator (Archibald, Altman, and Shoham 2009),

The action parameter that has the most impact on the success of an individual shot is the ϕ value, as this specifies the direction that the cue ball will initially travel. The other parameters main effect is on the location of the cue ball at the end of the shot. For this reason we first focus on estimating σ_ϕ, the standard deviation of the noise applied to the ϕ parameter for the observed agent.

Since this domain explicitly features imperfect execution skill, it is a good testbed for our problem of estimating an agent's execution skill.

Billiards Agent

Decision-making in the billiards domain is complex and has been the focus of significant research efforts (see section 2 for examples). Suffice it to say that exhaustive calculation of the optimal action as was done in the domain of 1D-darts is not feasible. To have an agent with as much strategic skill as possible, we utilized CueCard (CC) (Archibald, Altman, and Shoham 2009), a previous championship computational billiards agent for use in the experiments. CueCard selects actions taking into account its execution skill level.

The OR Method in Billiards

The OR method relies entirely on observed rewards. One difficulty with billiards is that the only true reward comes at the end of the game. Because of this, in the experiments, the success of an individual shot was used as a proxy for the true reward. If a shot is successful, that is, if it pockets a valid target ball, then this will be considered as a reward of 1. If the shot fails to successfully pocket a ball, then the reward will be 0. Not all shots in billiards are intended to pocket a ball. These other types of shots are generally called *safety* shots, and their goal is to put the opponent in a particularly tricky situation. These types of shots are, however, relatively rare in practice, and so we hypothesize that this proxy reward function will suffice for present purposes.

Since it is impossible to compute the expected shot success probability for a given state and execution skill hypothesis in a timely manner, we instead adapted the OR method by assuming the existence of historical mean shot success rate statistics for agents of different rough execution skill levels. This is a non-trivial assumption, and it is unlikely to be available in many settings, but using this will provide a way to compare the performance of OR in billiards to TBA.

To be specific, we utilized a database of 6 CueCard agents, each with known execution skill level and between 30,000 and 60,000 observed shots. From these shots we calculated the average reward/shot success probability for each agent. Then, we classified the agents into 3 different categories (expert, average, novice) and computed the mean of the success probabilities for all the agents inside each of the buckets. These mean average rewards and their accompanying execution skill levels were then used in the OR method described in section 3.

The Bayesian Approach in Billiards

The full Bayesian approach (TBA) described in section 4 requires a set of focal actions derived from the state. These are actions that, when unperturbed, should lead to high reward. In the case of billiards, we again use a successful shot as a proxy for reward and seek actions that should pocket a ball. This set of actions can be geometrically generated for a given state. In 8-ball the acting agent has to indicate before their shot the object ball that they intend to pocket, as well as the target pocket where the object ball is hoped to end up. These focal actions consist of ϕ angles that can be generated for each of the following types of shots:

- **Straight-in Shot** - the ϕ angle that will direct the cue ball directly into the object ball and send it directly into the middle of the target pocket
- **Bank Shot** - direct the cue ball into the object ball, sending it first bouncing off of a rail and then into the middle of the target pocket.
- **Kick Shot** - direct the cue ball first into a rail and then into the object ball, sending it straight into the middle of the target pocket.

- **Combo Shot** - direct the cue ball first into one legal object ball, sending it into another legal object, with the second object ball sent directly into the center of the pocket.

A set of these actions is generated for the object ball and target pocket, together with all feasible rail and intermediate object ball combinations respectively. Another requirement of TBA is the parameter β. This was determined by parameter sweep and validation on a subset of the data. β was set for the experiments with CueCard to a value of 0.55. TBA was run with 100 hypothesis execution skill levels evenly space in the interval [0.01, 0.9].

Experimental Setup

To generate the data for the experiments, CueCard played in matches against itself with a number of different execution skill levels. The execution skill levels used were 0.025, 0.125, 0.25, 0.375, 0.5 and 0.75. These levels indicate σ_ϕ, the standard deviation of the zero-mean Gaussian noise applied to the ϕ action component.

For each experiment, a random game was repeatedly selected for an agent and all shots from that game were observed in sequence by the methods. This process was repeated until the 1,000 shots had been observed. The OR and TBA methods then generated their predictions in an online manner. After obtaining each new observation, each method produced a new estimate.

A total of around 1,900 experiments were performed for each one of the different agents. The results shown are averaged across all experiments.

Billiards Experimental Results

Figure 4 shows the mean squared estimation error (MSE) for each method, averaged across all of the different execution skill level agents. This gives a sense of the convergence of the different methods and shows how they compare at a high level. The true noise method is included for comparison. Both variants of TBA noticeably outperform OR, and seems to stay close to TN over the whole timeline. TBA with the EES prediction appears to have slightly better performance overall than TBA with the MAP prediction. In comparing TBA and OR we note again that without historical shot success data, OR couldn't even be run in this setting, while TBA simply needs to do some geometric calculations on the state to produce its estimate.

Significance

Just looking at the average mean squared error across all of the agents it is hard to tell how significant this result is, since we are just looking at numbers, although it is good that the TBA estimates look very comparable to, and sometimes better than, TN. To get a better qualitative idea of how successful the estimation methods are, the following figures are provided. Figures 5 and 6 show the mean estimates for a single agent after each number of observations. Each figure shows these mean estimate plots for CueCard agents with six different execution skill levels on the same plot. Each execution skill level CueCard agent is reflected by a horizontal

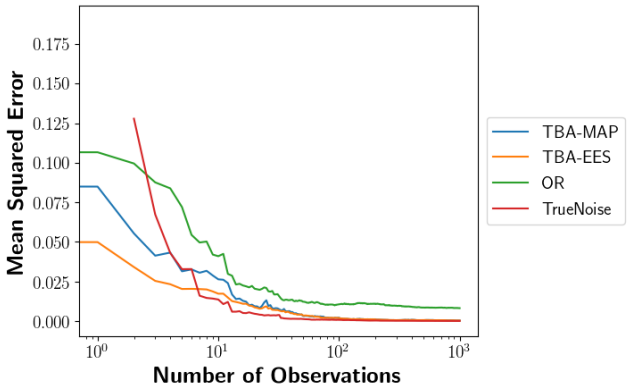

Figure 4: Billiards MSE Comparison on CC

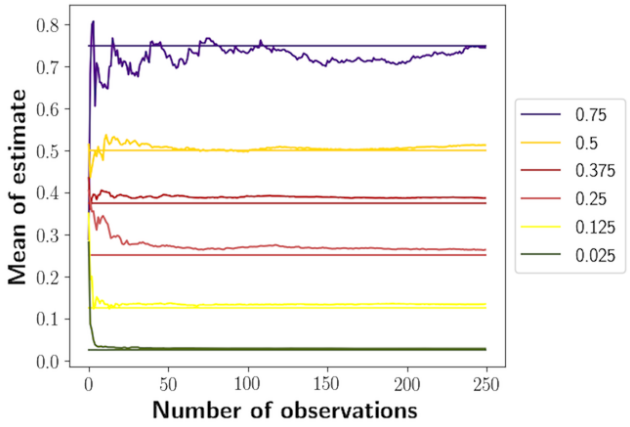

Figure 5: Estimate Comparison for TBA-MAP on CC

line at their true execution skill level. The estimate for that agent is shown in the same color.

These figures show how quickly the estimates converge, and how quickly and decisively they differentiate between the agents of different execution skill levels. As an example of how much these differences matter to the success of the agents, the CueCard agent with an execution skill level of 0.1875 will win against an 0.25 agent 56.24% of the time, but win against an 0.125 agent only 41.93% of the time.

Figure 5 shows these results for TBA while Figure 6 shows them for OR. This shows how much more quickly TBA is able to differentiate between the different agents, often after observing fewer than 50 actions. On the other hand OR struggles to have accurate predictions outside of a few agents with medium execution skill, and the estimate of the 0.75 execution skill agent converges to an estimate closer to 0.5 than 0.75. A typical billiards game includes around 9-15 actions from a player, so this means that in 4 or 5 games, TBA could have an estimate that could help determine the skill level of a player, which could be important in interacting with or predicting the later performance of that player. For example, after watching a few games, a player might have to decide whether or not to face the observed player in

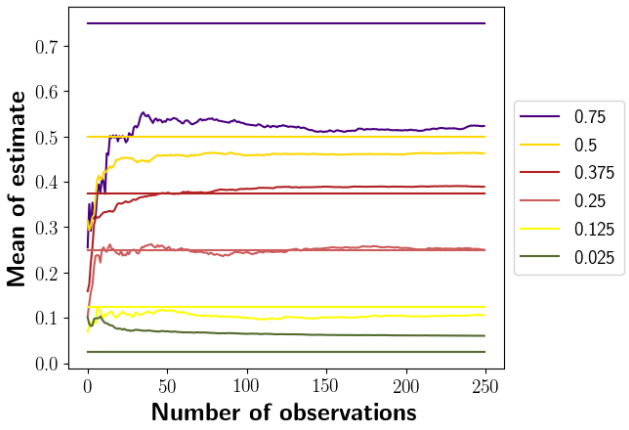

Figure 6: Estimate Comparison for OR on CC

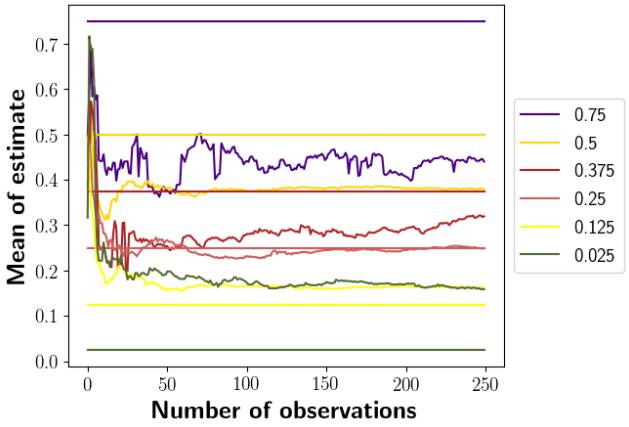

Figure 7: Estimate Comparison for TBA-MAP on RA

a match. Having an idea of their skill level and an ability to predict their future success would be crucial to help make this decision.

7 Testing the Limits

After observing the success of TBA at estimating the execution skill level of a billiards agent with high strategic skill, we ran another set of experiments to see how well it could do at estimating the execution skill level of a billiards agent with low strategic skill. Random agent (RA) is an agent that randomly samples ϕ directions and simulates an action with each ϕ value a number of times to estimate the success probability of a shot with that ϕ value. Whichever ϕ direction that successfully pockets a ball most frequently is used in the final selected shot. Could TBA still produce useful estimates in this case? Could OR?

We did a parameter sweep for the random agent to set β for TBA. The results of the TBA estimator with this $\beta = 0.02$ is shown in Figure 7. The estimates are clumped in the center of the region, and significantly over and underestimate the extreme execution skill levels respectively. The

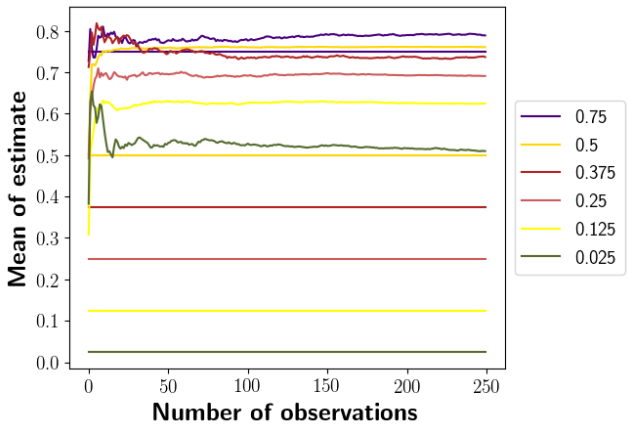

Figure 8: Estimate Comparison for OR on RA

estimates do properly reflect the ordering among the agents, in terms of their execution skill level. This could still be useful for ranking agents, but not at extracting their precise execution skill levels.

Figure 8 puts the TBA results in perspective. These results show the performance of the OR method on the random agent. The method gets the correct ordering over agents, but the skill levels are more closely bunched together and much farther away from the true execution skill levels. This result indicates that TBA is much more useful than OR when dealing with less rational agents. To get a sense of how inferior the random agent is to CueCard, the random agent with an execution noise level of 0.125 wins only around 2.59% of their games against a CueCard agent with the same noise level. The fact that TBA was able to pull any useful information out of such an unintelligent agent can be viewed positively. It also raises a lot of questions for future work.

8 Conclusions

The execution skill level of agents is an important characteristic of agents in many continuous domains. We introduce a robust Bayesian approach that uses the actions an agent selects and information about the state to estimate that agent's execution skill level. This approach was shown to outperform the only previous existing algorithm for this problem. It was also shown to be able to accurately estimate execution skill levels and meaningfully differentiate between agents of different execution skill levels in computational billiards.

In the future, we plan to investigate extensions and modifications of TBA to discover a method that performs well even for agents with less planning skill, like the random agent. We also plan to investigate methods for extracting not only an agent's execution skill level, but also their planning skill level. Furthermore, we also plan to apply these ideas to other continuous domains.

References

Ahmad, Z. F.; Holte, R. C.; and Bowling, M. 2016. Action selection for hammer shots in curling. In *IJCAI*, 561–567.

Archibald, C.; Altman, A.; and Shoham, Y. 2009. Analysis of a winning computational billiards player. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 1377–1382.

Archibald, C.; Altman, A.; and Shoham, Y. 2010. Success, strategy and skill: an experimental study. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, 1089–1096. International Foundation for Autonomous Agents and Multiagent Systems.

Archibald, C.; Altman, A.; and Shoham, Y. 2016. A distributed agent for computational pool. *IEEE Transactions on Computational Intelligence and AI in Games* 8(2):190–202.

Archibald, C., and Nieves-Rivera, D. 2018. Execution skill estimation. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '18, 1859–1861. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems.

Archibald, C., and Shoham, Y. 2011. Hustling in repeated zero-sum games with imperfect execution. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, 31–36.

Bard, N.; Johanson, M.; Burch, N.; and Bowling, M. 2013. Online implicit agent modelling. In *Proceedings of the Twelfth International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 255–262.

Bard, N.; Nicholas, D.; Szepesvari, C.; and Bowling, M. 2015. Decision-theoretic clustering of strategies. In *Proceedings of the Fourteenth International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*. To Appear.

Billings, D.; Papp, D.; Schaeffer, J.; and Szafron, D. 1998. Opponent modeling in poker. In *AAAI/IAAI*, 493–499.

Borm, P., and Genugten, B. 2001. On a relative measure of skill for games with chance elements. *TOP: An Official Journal of the Spanish Society of Statistics and Operations Research* 9(1):91–114.

Bowling, M., and Veloso, M. 2004. Existence of multiagent equilibria with limited agents. *Journal of Artificial Intelligence Research* 22:353–384. A previous version appeared as a CMU Technical Report, CMU-CS-02-104.

Carmel, D., and Markovitch, S. 1995. Opponent modeling in multi-agent systems. In *International Joint Conference on Artificial Intelligence*, 40–52. Springer.

Davis, T.; Burch, N.; and Bowling, M. 2014. Using response functions to measure strategy strength. In *Proceedings of the Twenty-Eighth Conference on Artificial Intelligence (AAAI)*, 630–636.

Dreef, M.; Borm, P.; and Genugten, B. v. d. 2002. On strategy and relative skill in poker. Discussion Paper 59, Tilburg University, Center for Economic Research.

Dreef, M.; Borm, P.; and van der Genugten, B. 2004. A new relative skill measure for games with chance elements. *Managerial and Decision Economics* 25(5):255–264.

Greenspan, M.; Lam, J.; Leckie, W.; Godard, M.; Zaidi, I.; Anderson, K.; Dupuis, D.; and Jordan, S. 2008. Toward a competitive pool playing robot. *IEEE Computer Magazine* 41(1):46–53.

Jiang, A. X.; Yin, Z.; Zhang, C.; Tambe, M.; and Kraus, S. 2013. Game-theoretic randomization for security patrolling with dynamic execution uncertainty. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, 207–214. International Foundation for Autonomous Agents and Multiagent Systems.

Kaminka, G. A.; Vered, M.; and Agmon, N. 2018. Plan recognition in continuous domains. In *Proceedings of 32nd AAAI Conference*.

Landry, J. F.; Dussault, J. P.; and Beaudry, E. 2015. A straight approach to planning for 14.1 billiards. *IEEE Transactions on Computational Intelligence and AI in Games* PP(99):1–1.

Larkey, P.; Kadane, J. B.; Austin, R.; and Zamirm, S. 1997. Skill in games. *Management Science* 43(5):596–609.

Russell, S., and Norvig, P. 2009. *Artificial Intelligence: A Modern Approach*. Upper Saddle River, NJ, USA: Prentice Hall Press, 3rd edition.

Schadd, F.; Bakkes, S.; and Spronck, P. 2007. Opponent modeling in real-time strategy games. In *GAMEON*, 61–70.

Smith, M. 2007. PickPocket: A computer billiards shark. *Artificial Intelligence* 171:1069–1091.

Sturtevant, N.; Zinkevich, M.; and Bowling, M. 2006. ProbMaxn: Opponent modeling in n-player games. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI)*, 1057–1063.

Thrun, S.; Burgard, W.; and Fox, D. 2005. *Probabilistic Robotics*. Cambridge, MA: MIT Press.

Van Valkenhoef, G.; Ramchurn, S. D.; Vytelingum, P.; Jennings, N. R.; and Verbrugge, R. 2010. Continuous double auctions with execution uncertainty. In *Agent-Mediated Electronic Commerce. Designing Trading Strategies and Mechanisms for Electronic Markets*. Springer. 226–241.

Yee, T.; Lisỳ, V.; and Bowling, M. H. 2016. Monte carlo tree search in continuous action spaces with execution uncertainty. In *IJCAI*, 690–697.

Yin, Z.; Jain, M.; Tambe, M.; and Ordónez, F. 2011. Risk-averse strategies for security games with execution and observational uncertainty. In *AAAI*.