arXiv:1812.01569v1 [cs.AI] 4 Dec 2018

Modeling Theory of Mind for Autonomous Agents with Probabilistic Programs

Iris R. Seaman
seaman.i@northeastern.edu

Jan-Willem van de Meent
j.vandemeent@northeastern.edu

David Wingate
wingated@cs.byu.edu

Abstract

As autonomous agents become more ubiquitous, they will eventually have to reason about the mental state of other agents, including those agents' beliefs, desires and goals – so-called *theory of mind* reasoning. We introduce a collection of increasingly complex theory of mind models of a "chaser" pursuing a "runner", known as the *Chaser-Runner* model. We show that our implementation is a relatively straightforward theory of mind model that can capture a variety of rich behaviors, which in turn, increase runner detection rates relative to basic (non-theory-of-mind) models. In addition, our paper demonstrates that (1) using a planning-as-inference formulation based on nested importance sampling results in agents simultaneously reasoning about other agents' plans and crafting counter-plans, (2) probabilistic programming is a natural way to describe models in which each uses complex primitives such as path planners to make decisions, and (3) allocating additional computation to perform nested reasoning about agents result in lower-variance estimates of expected utility.

1 Introduction

An autonomous agent that interacts with other agents needs to do more than simply perceive and respond to their environment. Eventually agents will need to reason about all of the complexities inherent in the real world, including the beliefs, intents and desires of other intentional agents. This is known as *theory of mind*, and is indispensable if we hope to one day create agents capable of empathy, "reading between the lines," and interacting with humans as peers.

In this paper, we explore how theory of mind can be implemented using nested simulations in the form of probabilistic programs. We develop a scenario involving two agents, a *chaser* and a *runner*. The chaser seeks to intercept the runner and the runner seeks to reach a goal location without detection. However, the runner's intended start location, goal location, and likely path to the goal are initially unknown to the chaser. We assume that the runner knows the current location of the chaser, but not where the chaser will move in the future. This results in a setting where both agents must reason about each other, and about how they reason about reasoning.

To simulate runner and chaser trajectories, we employ a variety of semi-realistic primitives, including path planners and visibility graphs. We formulate the model of the chaser and the runner as nested probabilistic programs, which are conditioned according to the desired behavior of the respective agents. The model of the chaser is conditioned to *maximize* the likelihood of detection, and the runner is conditioned to *minimize* likelihood of detection. The result is a probabilistic model over possible chaser trajectories. At each point of time, the chaser imagines possible future trajectories, along with possible runner trajectories, and selects a move that has a high relative expected utility. This planning-as-inference formulation [1] is a natural fit for probabilistic programming, which makes it straightforward to incorporate complex deterministic primitives into both models, and perform recursive Bayesian reasoning using the framework of nested importance sampling [2].

We evaluate our models in a variety of scenarios and demonstrate that nested Bayesian reasoning leads to rational behaviors which maximize utility respectively at each level. Our experiments show that our formulation leads to improved runner detection rates relative to basic models, and that allocating additional computation to perform nested reasoning about agents results in lower-variance estimates of expected utility.

Preliminary work, under review.

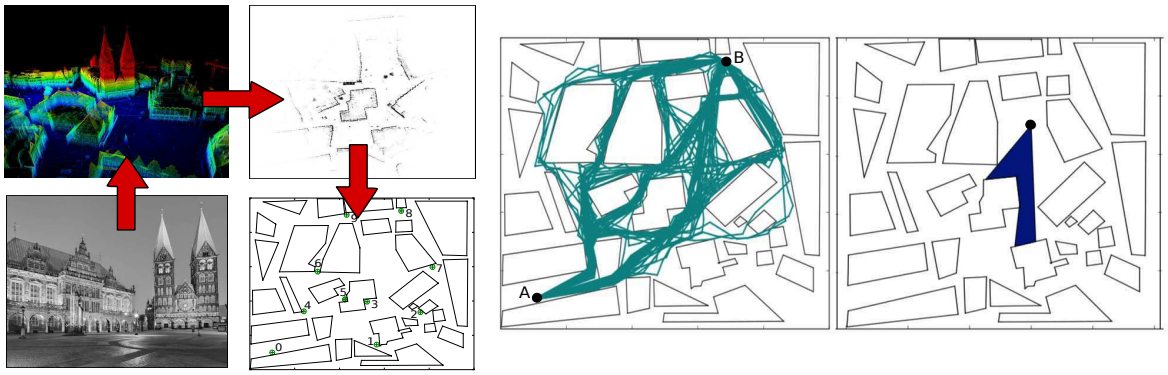

Figure 1: (a) We generate a coarse polygonal city map from point cloud data of the city of Bremen, Germany. (b) Visual distribution over paths a runner may take modeled with Random-Exploring Random Trees, RRTs, from points A to B. (c) a 45° isovist, or range of sight, of the chaser. The isovist is properly blocked by buildings.

2 Background

2.1 Theory of Mind

Human children develop theory of mind during their early years, generally between the ages of three and six [3, 4]. Bello and Cassimatis [5] explore this phenomenon with a computational model that suggests that the underlying cognitive shifts required for the development of theory of mind may be smaller than previously supposed. Goodman et al. [6] present a formal model that attempts to account for false belief in children, and later take the innovative approach of linking inference with causal reasoning [7]. Additionally, the same group explores language as a type of social cognition [8].

The development of theory of mind in machines leads naturally to interaction with their human counterparts. Awais and Henrich [9], Fern et al. [10], and Nguyen et al. [11] investigate collaboration between humans and robots in which the robot must determine the human's (unobservable) goal. In a complementary line of research, Sadigh et al. [12] explore the idea of active information, in which the agent's own behaviors become a tool for identifying a human's internal state.

Fully-developed theory of mind requires the possibility of nested beliefs. Koller et al. [13] present an inference algorithm for recursive stochastic programs. Frith and Frith [14] argue that theory of mind can be modeled using probabilistic programming, and demonstrate examples of nested conditioning with the probabilistic programming language, Church. Zettlemoyer et al. [15] address filtering in environments with many agents and infinitely nested beliefs.

To our knowledge, our work is the first to model nested reasoning about agents in a time-dependent manner. Prior work by Baker et al. [16] develops a Bayesian framework for reasoning about preferences of individual agents based on observed time-dependent trajectories. Our work differs in that our environment is not discretized into a grid world, and as such represents a continuous action space. Work by Stuhlmüller and Goodman [17] employed probabilistic programs to model nested reasoning about other agents. Relative to this work, our work differs in that agents update and *act upon* their beliefs of other agents in a time-dependent manner, whereas the work by Stuhlmüller and Goodman [17] considers problems in which there is a single decision.

2.2 Probabilistic Program Inference

To represent our generative model cleanly and to perform inference in it, we employ the tools of probabilistic programming [18]. This allows us to define probabilistic models that incorporate control flow, libraries of deterministic primitives, and data structures. A probabilistic program is a procedural model that, when run unconditionally, yields a sample from a prior distribution. Running probabilistic programs forward can be quite fast, and is limited only by the native speed of the interpreter for the language.

Inference in probabilistic programming involves reasoning about a target distribution that is conditioned by a likelihood, or more generally a notion of utility [18]. Inference for probabilistic programs is difficult because of the flexibility that probabilistic programming languages provide: an inference algorithm must behave reasonably for any program a user wishes to write. Many probabilistic programming systems rely on Monte Carlo methods due to their generality [19–23]. Methods based on importance sampling and SMC have become particularly popular [24–28], owing to their simplicity and compositionality [2].

For our purposes, the most important feature of probabilistic programming languages is that they allow us to freely mix deterministic and stochastic elements, resulting in tremendous modeling flexibility. This makes it relatively easy to (for example) describe distributions over Rapidly-Exploring Random Tree (RRTs), isovists, or even distributions that involve optimization problems as a subcomponent of the distribution.

3 Simulation Primitives

Although probabilistic programming has previously been used to model theory of mind [17], past implementations have thus far considered relatively simplistic problems involving a small number of decisions. In this paper, we not only model a setting in which agents must reason about future events, but also do so in a manner that involves reasoning about properties of the physical world around them. To enable this type of reasoning, we will employ a number of semi-realistic simulation primitives.

The environment. To search for and intercept the runner, the chaser requires a representation of the world that allows reasoning about starting locations, goals, plans, movement and visibility. We use a polygonal model designed around a known, fixed map of the city of Bremen, Germany [29], shown in Fig. 1 (a).

Path planning and trajectory optimization. We model paths using a RRT [30], a randomized path planning algorithm designed to handle nonholonomic constraints and high degrees of freedom. We leverage the random nature of the RRT to describe an entire distribution over possible paths: each generated RRT path can be viewed as a sample from the distribution of possible paths taken by a runner (see Fig. 1 (b)). RRTs naturally consider short paths as well as long paths to the goal location. To foreshadow a bit, note that because we will be performing inference over RRTs conditioned on not being detected, the runner will naturally tend to use paths that minimize the chance of detection, which are often, but not always, the shortest and most direct. Our RRTs are refined using a trajectory optimizer to eliminate bumps and wiggles.

Visibility and detection. Detection of the runner by the chaser is modeled using an isovist, a polygon representation of the chaser's current range of sight [31, 32]. Given a map, chaser location, and runner location, the isovist determines the likelihood that the runner was detected. Although an isovist usually uses a 360 degree view to describe all possible points of sight to the chaser, we limit the range of sight to 45 degrees, and add direction to the chaser's sight as seen in Fig. 1 (c). The direction of the chaser's line of sight is determined by the imagined location of the runner.

4 The Chaser-Runner Model

To model theory of mind, we will develop a nested probabilistic program in which a Chaser plans a trajectory by maximizing the probability of interception relative to imagined runner trajectories. The model for runner trajectories, in turn, assumes that the runner imagines chaser trajectories and avoids paths with a high probability of interception.

Our model has four levels: the **episode model** samples a sequence of moves by the chaser. Each move is sampled from the **outermost model**, which describes the beliefs of the chaser about the expected utility of moves. This model compares future chaser trajectories to possible runner trajectories and assigns higher probability to trajectories in which the runner is likely to be detected. The runner trajectories are in turn sampled from the **middlemost model**, which minimizes detection probability based on imagined chaser trajectories that are sampled from the **innermost model**. These three models work in tandem to create nuanced inferences about where the chaser believes the runner might be, and how it ought to counter-plan to maximize probability of detection.

Algorithm 1 shows pseudo-code for the Chaser-Runner model, formulated as nested probabilistic programs, which we refer to as queries. Together, these programs define a planning-as-inference problem [1] in which queries generate weighted samples, resulting in a nested importance sampling [2] scheme that we describe in more detail below.

The episode model initializes the location of the chaser to a specified start location $x_1^{\text{C}} = x_{\text{START}}^{\text{C}}$. For subsequent time points $t = 2 \dots T$, the model samples a weighted partial trajectory $(x_{1:t}^{\text{C}}, w_t)$ from the outermost CHASER model. After the final iteration, the model returns the completed trajectory $x_{1:T}^{\text{C}}$ and the product over incremental weights $w_{1:T}$.

The outermost model describes the chaser's plan for trajectories, given the chaser's belief about possible runner trajectories. The chaser selects a goal location $x_{\text{GOAL}}^{\text{C}}$ at random and uses the RRT planner to sample a possible future trajectory $x_{t:T}^{\text{C}}$. Note that this trajectory is random, owing to the stochastic nature of the RRT algorithm. In order to evaluate the utility of this trajectory, the chaser imagines a possible runner trajectory by sampling from the middlemost RUNNER model. The chaser then evaluates the utility of the trajectory by using an isovist representation to determine the number of time points $T_{\text{VISIBLE}}^{\text{C}}$ during which the runner is visible to the chaser. The chaser then conditions the sampled trajectories by defining a weight $w^{\text{C}} = \exp(\alpha\, T_{\text{VISIBLE}}^{\text{C}})$. As we will discuss below,

Algorithm 1 Probabilistic program implementation of the Chaser-Runner model. The EPISODE model samples moves from a nested CHASER model, which in turn simulates runner trajectories from a second nested RUNNER model. The CHASER model is conditioned to *maximize* the probability of future detections, whereas the RUNNER model is conditioned *minimize* both past and future detections. At each time step t, we propose K future trajectories for the chaser and $K \times L$ trajectories for the runner. These trajectories are then resampled to K partial trajectories $x^{\mathrm{C}}_{1:t}$, resulting in a SMC sampler for the CHASER model.

$Q^{\text{CHASER}} \leftarrow$ **resample**(**importance**(CHASER, K), K)
$Q^{\text{RUNNER}} \leftarrow$ **importance**(RUNNER, L)
$Q^{\text{NAIVE-CHASER}} \leftarrow$ **importance**(NAIVE-CHASER, 1)
query EPISODE($x^{\mathrm{C}}_{\text{START}}$) ▷ Episode model
  $x^{\mathrm{C}}_1 \leftarrow x^{\mathrm{C}}_{\text{START}}$
  for t **in** $2 \dots T$ **do**
    $x^{\mathrm{C}}_{1:t}, w_t \leftsquigarrow Q^{\text{CHASER}}(x^{\mathrm{C}}_{1:t-1})$
  return $x^{\mathrm{C}}_{1:T}, \prod_t w_t$
query CHASER($x^{\mathrm{C}}_{1:t-1}$) ▷ Outer Model
  $x^{\mathrm{C}}_{\text{GOAL}} \sim \text{Uniform}(\{x_{\text{A}}, \dots, x_{\text{J}}\})$
  $x^{\mathrm{C}}_{t:T} \sim \text{RRT-PLAN}(x^{\mathrm{C}}_{t-1}, x^{\mathrm{C}}_{\text{GOAL}})$
  $x^{\mathrm{R}}_{t:T}, w^{\mathrm{R}} \leftsquigarrow Q^{\text{RUNNER}}(x^{\mathrm{C}}_{1:t-1})$
  $T^{\mathrm{C}}_{\text{VISIBLE}} \leftarrow \text{TIME-VISIBLE}(x^{\mathrm{R}}_{t:T}, x^{\mathrm{C}}_{t:T})$
  $w^{\mathrm{C}} \leftarrow \exp(\alpha\, T^{\mathrm{C}}_{\text{VISIBLE}})$
  return $x^{\mathrm{C}}_{1:t}, w^{\mathrm{C}} \cdot w^{\mathrm{R}}$
query RUNNER($x^{\mathrm{C}}_{1:t-1}$) ▷ Middle Model
  $x^{\mathrm{R}}_{\text{START}} \sim \text{Uniform}(\{x_{\text{A}}, \dots, x_{\text{J}}\})$
  $x^{\mathrm{R}}_{\text{GOAL}} \sim \text{Uniform}(\{x_{\text{A}}, \dots, x_{\text{J}}\})$
  $x^{\mathrm{R}}_{1:T} \sim \text{RRT-PLAN}(x^{\mathrm{R}}_{\text{START}}, x^{\mathrm{R}}_{\text{GOAL}})$
  $\tilde{x}^{\mathrm{C}}_{t:T}, \tilde{w}^{\mathrm{C}} \leftsquigarrow Q^{\text{NAIVE-CHASER}}(x^{\mathrm{C}}_{t-1})$
  $T^{\mathrm{R}}_{\text{VISIBLE}} \leftarrow \text{TIME-VISIBLE}(x^{\mathrm{R}}_{1:T}, \{x^{\mathrm{C}}_{1:t-1}, \tilde{x}^{\mathrm{C}}_{t:T}\})$
  $w^{\mathrm{R}} \leftarrow \exp(-\alpha\, T^{\mathrm{R}}_{\text{VISIBLE}})$
  return $x^{\mathrm{R}}_{t:T}, w^{\mathrm{R}} \cdot \tilde{w}^{\mathrm{C}}$
query NAIVE-CHASER(x^{C}_{t-1}) ▷ Inner Model
  $\tilde{x}^{\mathrm{C}}_{\text{GOAL}} \sim \text{Uniform}(\{x_{\text{A}}, \dots, x_{\text{J}}\})$
  $\tilde{x}^{\mathrm{C}}_{t:T} \sim \text{RRT-PLAN}(x^{\mathrm{C}}_{t-1}, \tilde{x}^{\mathrm{C}}_{\text{GOAL}})$
  return $\tilde{x}^{\mathrm{C}}_{t:T}, 1$

this corresponds to assigning a utility proportional to $T^{\mathrm{C}}_{\text{VISIBLE}}$ in a planning-as-inference formulation. The model discards most of the imagined future trajectory, keeping only the next time point x^{C}_t, and returns the partial trajectory $x^{\mathrm{C}}_{1:t}$, together with a weight $w^{\mathrm{C}} \cdot w^{\mathrm{R}}$ that reflects the utility of the chaser and the runner.

The middlemost model describes the chaser's reasoning about possible runner trajectories. We assume that the chaser models a worst-case scenario where the runner is aware of the chaser's location. This could be, for example, because the runner uses a police scanner to listen in on the chaser's reported location. Moreover, we assume that at any point in time, the episode only continues when the chaser has not yet detected the runner. Finally, we assume that the runner will seek to avoid detection by imagining a chaser trajectory, and then selecting a trajectory that will not intersect the that of the chaser. We implement these assumptions in the probabilistic program as follows. The runner model first selects a start location $x^{\mathrm{R}}_{\text{START}}$ and goal location $x^{\mathrm{R}}_{\text{GOAL}}$ at random, and then samples a random trajectory $x^{\mathrm{R}}_{1:T}$ using the RRT planner. The runner then imagines a future chaser trajectory by selecting a goal location $\tilde{x}^{\mathrm{C}}_{\text{GOAL}}$ at random and sampling $\tilde{x}^{\mathrm{C}}_{t:T}$ from the innermost model. We then condition this sample by computing the total time of visibility $T^{\mathrm{R}}_{\text{VISIBLE}}$, based on both the known past trajectory $x^{\mathrm{C}}_{1:t-1}$ and the imagined future trajectory $\tilde{x}^{\mathrm{C}}_{t:T}$ of the chaser. Finally, we assign a weight $w^{\mathrm{R}} = \exp(-\alpha T^{\mathrm{R}}_{\text{VISIBLE}})$, which corresponds to a negative utility (i.e. a cost) proportional to $T^{\mathrm{R}}_{\text{VISIBLE}}$ in the planning-as-inference formulation.

The innermost model describes future chaser trajectories imagined by the runner. This model is the simplest of all the models in our nested formulation. Given the previous location x^{C}_{t-1} of the chaser, the runner imagines a goal location $\tilde{x}^{\mathrm{C}}_{\text{GOAL}}$ at random and then uses the RRT planner to a sample a random future trajectory $\tilde{x}^{\mathrm{C}}_{t:T}$. Since this model is not conditioned in any way, it returns weight 1.

5 Planning as Inference Formulation

The Chaser-Runner model performs two levels of nested inference. At the episode level, we infer the next time point x^{C}_t, conditioning on expected future detections. In order to evaluate this likelihood, we simulate runner trajectories that are conditioned to avoid future detections. We will perform inference using a nested importance sampling scheme [2], which is a generalization of importance sampling in which weighted samples at one level in the model are used as proposals at other levels in the model. Note that nested importance samplign is *not* a form of nested Monte Carlo estimation as discussed in Rainforth et al. [33] and Rainforth [34]. We discuss the distinctions between the two methods below.

We implement conditioning using a planning-as-inference formulation [1, 35]. In planning-as-inference problems, the target density $\pi(x) = \gamma(x)/Z$ is defined in terms of an unnormalized density

$$\gamma(x) = \exp(R(x))p(x), \tag{1}$$

which in turn is defined in terms of prior $p(x)$ and a utility or reward $R(x)$. The normalizing constant $Z = \mathbb{E}[\exp(R(x))]$ is sometimes referred to as the desirability [36].

The Chaser-Runner model in Algorithm 1 defines a sequence of unnormalized densities

$$\gamma_t(x^{\mathrm{C}}_{t:T}, \tilde{x}^{\mathrm{C}}_{t:T}, x^{\mathrm{R}}_{1:T}) = \exp[\alpha(T^{\mathrm{C}}_{\mathrm{VISIBLE}} - T^{\mathrm{R}}_{\mathrm{VISIBLE}})] \\ p(x^{\mathrm{C}}_{t:T}|x^{\mathrm{C}}_{t-1})p(\tilde{x}^{\mathrm{C}}_{t:T}|x^{\mathrm{C}}_{t-1})p(x^{\mathrm{R}}_{1:T}).$$

In this density, the reward $\alpha(T^{\mathrm{C}}_{\mathrm{VISIBLE}} - T^{\mathrm{R}}_{\mathrm{VISIBLE}})$ depends on the *difference* between the number of time points during which the chaser expects that the runner will be visible, and the number of time points during which the runner expects to be visible based the imagined chaser trajectory (which reflects a more naive model of the chaser). In other words, the the chaser aims to identify trajectories that will result in likely detections of the runner, under the assumption that the runner will avoid trajectories where detection is likely given a naive chaser model.

6 Nested Importance Sampling

We can perform inference in the chaser-runner model using Monte Carlo method for probabilistic programs. Algorithm 1 defines an importance sampling scheme. At each time t, we sample $x^{\mathrm{C}}_t \sim \pi_t(x^{\mathrm{C}}_t)$ from the marginal of the target density above. To do so, we sample K particles from the CHASER model. For each sample, we draw L samples from the RUNNER model. We then perform resampling to select K of the resulting $K \cdot L$ particles with, which corresponds to performing SMC sampling within the EPISODE model.

To denote this sampling scheme, we define query distributions in lines 1-3 of Algorithm 1. We assume an operator **importance** that accepts a query and a number of samples L and returns a transformed query Q that accepts K samples, and returns $K \times L$ weighted samples. We additionally assume a **resample** operation that accepts a query and a sample count K and returns a new query that resamples K samples from a query, down-sampling or up-sampling if necessary.

When $L = 1$, this sampling scheme reduces to standard SMC inference for probabilistic programs [26]. When $L > 1$ it can be understood as a a form of nested importance sampling [2]. Note that in this sampling scheme, each of the L samples corresponds a *different* runner trajectory $x^{\mathrm{R},k,l}_{1:T}$, but that the reward for this trajectory is evaluated relative to the *same* past $x^{\mathrm{C},k}_{1:t-1}$ and imagined future $x^{\mathrm{C},k}_{t:T}$ trajectory for the chaser.

As noted above, nested importance sampling is not the same as nested Monte Carlo estimation. In nested Monte Carlo problems, we compute an expectation of the form $\mathbb{E}[f(y, \mathbb{E}[g(y, z)])]$, which is to say that we compute an expectation in which, for each sample y, we need to compute an expected value by marginalizing over samples z. In the chaser-runner problem, we would obtain a nested Monte Carlo probelm if we defined the weight

$$w^k_t = \exp\left[\hat{R}(x^{\mathrm{C},k}_{1:T})\right],$$

by averaging the reward over chaser trajectories

$$\hat{R}(x^{\mathrm{C},k}_{1:T}) = \frac{1}{L}\sum_{l=1}^{L} R(x^{\mathrm{C},k}_{1:T}, x^{\mathrm{R},k,l}_{1:T}).$$

In nested importance sampling, we select a particle $x^{\mathrm{C},k}_{1:T}$ according to the average weight

$$w^k_t = \frac{1}{L}\sum_{l=1}^{L} \exp\left[R(x^{\mathrm{C},k}_{1:T}, x^{\mathrm{R},k,l}_{1:T})\right].$$

This is sometimes referred to as nested conditioning, in the context of probabilistic programming systems [34]. For any choice of L, this is a valid importance sampling scheme in which the importance weight provides an unbiased estimate of the normalizing constant.

7 Experiments

We carry out three categories of experiments: 1) *trajectory visualization experiments*, in which we qualitatively evaluate what forms of rational behavior arise in our model depending on conditioning, 2) *detection rate experiments*, which test to what extent a more accurate model of a runner enables the chaser to detect the runner most often, and 3) *sample budget experiments*, where we quantify the trade-offs in allocating our sample budget at different levels of the model.

7.1 Visualization of Trajectories

Before carrying out a more quantitative evaluation of the chaser-runner model, we visualize sampled trajectories to show how nested inference converges empirically to rational behavior at each level of the model. We begin by considering a simplified scenario in which we assume fixed start and goal locations. These locations are known to both the chaser and the runner, which means that the chaser and runner do not have to perform inference over possible goal locations. Figure 2 (a) shows a heat map of naive chaser paths in the innermost model, which are conditioned on the start and goal locations. In Figure 2 (b), we show a heat map of runner paths, in which the runner travels in the opposite direction along the same two locations. We observe that the runner avoids direct routes so as to minimize chance of detection. In Figure 2 (c) we show a heat map of chaser trajectories in the outermost model, which shows that the chaser selects paths that are likely to lead to interception of the runner. Together, Figures 2 (a)-(c) demonstrate how our Chaser-Runner model can perform planning conditioned on start and end locations.

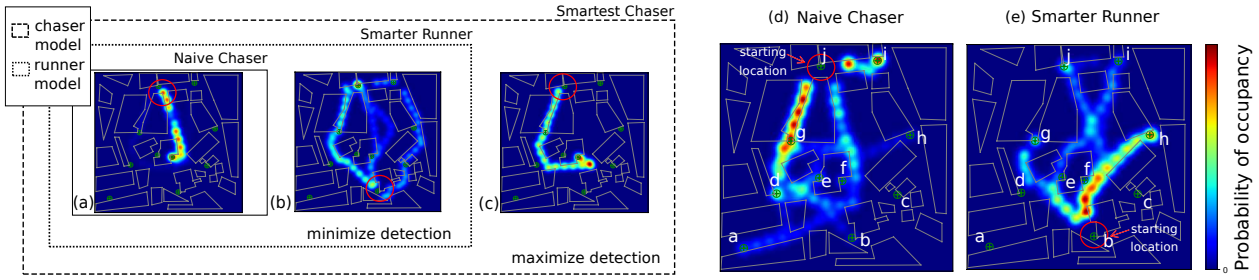

Figure 2: **Chaser and runner trajectories** in the innermost, middlemost, and outermost models where locations circled in red are the starting locations for each agent. We show posterior distributions of L *runner* and *naive chaser* paths, when $(K, L) = 128, 16$ for a single resampled sample k. (a)-(c) show posterior distributions over paths after running importance sampling where we condition the start and goal locations for each agent. Figures (d)-(e) show posterior paths after we only condition the start locations for the agents .

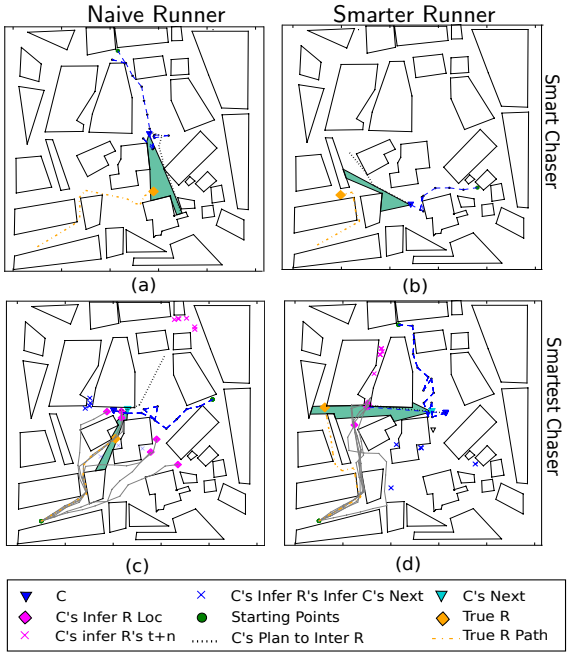

Figure 3: (a) Smart chaser playing against a naive runner, where the chaser anticipates the the intersection point and heads in the correct direction to detect the runner. (b) Smart chaser playing against a smarter runner. (c) Smartest chaser against the naive runner. (d) Smartest Chaser infers the runner locations to be more hidden, avoiding the center of the map. The chaser successfully detects the smarter runner.

In Figure 2 (d)-(e), we visualize L *naive* chaser and runner paths from a single k sample (sampled proportionally to importance weights) at times step $t \geq 3$. The runner paths in Figure 2 (e) once again avoid detection relative to the naive chaser paths in Figure 2 (d). Although the naive chaser travels directly toward goal locations from the upper end of the map, in this particular k sample, the naive chaser most often remains on the left side of the map. This results in the runner traveling through the center of city to minimize probability of detection, but more often planning toward location **h**. This is a case where the RRT planner provides the runner with a shorter and direct plan to minimize detection from the chaser.

7.2 Detection Experiments

To evaluate the influence of nested modeling on resulting plans, we compare detection rates in the full chaser-runner model to detection rates in three simplified models. We run simulations using two types of runners. We refer to the runner from the full model as the *smarter runner*, and also consider a *naive runner* which samples from the RRT planner in the same manner as the naive chaser. We similarly consider two chaser models. We refer to the chaser from the full model as the *smartest chaser*. We additionally consider a simplified model in which the chaser assumes a naive runner, which we refer to as a *smart chaser*.

These two runner and chaser models together yield 4 modeling scenarios. Table 1 shows the average detection rate over 50 restarts for each scenario. Figure 3 shows illustrative trajectories. In this figure, 'C' stands for Chaser; 'R' stands for runner. The blue triangle represents the chaser's true, current location. Blue dashed lines represent the past chaser trajectory whereas crosses mark future locations imagined by the runner. Magenta diamonds represent samples of inferred runner locations; magenta crosses represent inferred future runner trajectories.

1. Naive Runner, Smart Chaser. A smart chaser can reliably intercept a naive runner. Figure 3 (a) illustrates a successful detection. We observe that the chaser typically navigates to the center of the map. Since the shortest path between most points crosses the center of the map, this allows the chaser to intercept the runner with high probability.

	Naive Runner	Smarter Runner
Smart Chaser	$(49/50) = 0.98$	$(18/50) = 0.36$
Smartest Chaser	$(49/50) = 0.98$	$(28/50) = 0.56$

Table 1: Detection Rates for Agent Model Variants

2. Smarter Runner, Smart Chaser. When we increase the model complexity of the runner, the detection probability decreases. Figure 3 (b) illustrates a prototypical result. The *smarter runner* expects the chaser to remain in the center of the map, as it is trying to head off a naive agent, and successfully avoids the center of the map. In Figure 3 (b), the runner is seen swerving sharply left taking a longer path around the perimeter of the city to reach its goal. As a result, the chaser is unable to find the runner for the rest of the simulation. The average detection rate is 0.36, which means that a smarter runner is able to avoid a misinformed chaser in most episodes.

3. Naive Runner, Smartest Chaser. In this experiment, the chaser assumes a smarter runner, even though the runner's behavior is in fact naive. Figure 3 (c) illustrates a prototypical result. Here, the multimodality of the model's inferences is apparent: the chaser predicts two possible modes where the runner could be (clusters of magenta triangles), but assigns more probability mass to the upper (correct) cluster; the result is that the chaser plans a path to that location, which results in a detection. As it turns out, this model variant results yields a detection rate of 0.98, which is the same as that of in scenario 1, where the chaser has an accurate model of the naive runner.

4. Smarter Runner, Smartest Chaser. Figure 3 (d) shows a prototypical result from the full chaser-runner model, which results in a successful detection. The chaser anticipates that the runner will avoid highly visible areas of the map and travel through alley ways and around the city.

This experiment yielded a detection rate of 0.56, which is significantly higher than the detection rate of 0.36 in experiment 2.

Discussion. These 4 scenarios illustrate that when the runner reasons more deeply, he evades more effectively; Conversely when the chaser reasons more deeply, he intercepts more effectively. Furthermore, we show that a single, unified inference algorithm can uncover a wide variety of intuitive, rational behaviors for both the runner and the chaser.

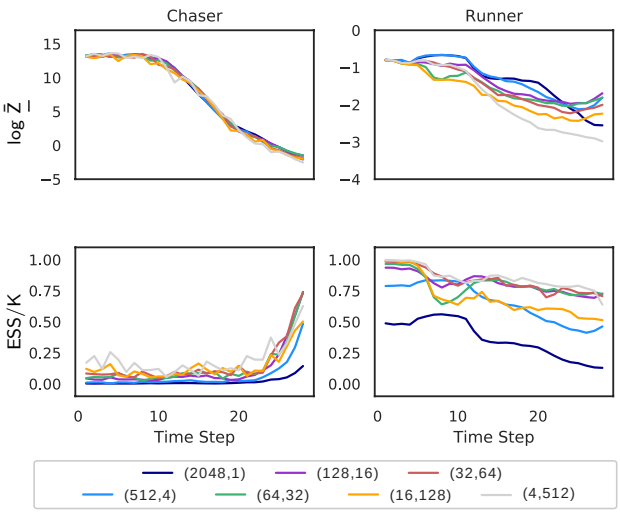

Figure 4: Log mean log weights, $\log \bar{Z}$, and Fractional ESS (normalized by K) as a Function of Time for each sample budget. **Top Row:** $\log \bar{Z}^{\mathrm{R}}$ for the middlemost model (left), and $\log \bar{Z}^{\mathrm{C}}$ the outermost model, (right). **Bottom Row:** The fractional ESS for each varying K and L.

7.3 Sample Budget Experiments

To evaluate how the allocation of computational resources to different levels of the model affects the variance of our importance sampling estimator, we carry out experiments in which we set K and L to

$$(K, L) = (2048, 1), (512, 4), (128, 16), \ldots (4, 512)$$

This fixes the total computation budget to $KL = 2048$ samples, which allows us to assess how many samples from the runner are needed to effectively evaluate utilities in the chaser model.

In this experiment, we perform $R = 10$ independent episode restarts for 7 combinations of (K, L) values. For each episode we compute K chaser trajectories and $K \cdot L$ runner trajectories for $T = 28$ time steps. In other words, we compute $7 \cdot R \cdot T \cdot K \cdot L$ runner trajectories w^{R}, just over 4 million in total.

In Figure 4 (top row), we show the log mean log weights for the chaser (left) and runner (right) at each time point t. These are computed as follows

$$\log \bar{Z}_t^{\mathrm{C}} = \frac{1}{R} \sum_{r=1}^{R} \log \left(\frac{1}{K} \sum_{k=1}^{K} \sum_{l=1}^{L} w_t^{\mathrm{C},k,l} w_t^{\mathrm{R},k,l} \right)$$

$$\log \bar{Z}_t^{\mathrm{R}} = \frac{1}{R} \sum_{r=1}^{R} \log \left(\frac{1}{KL} \sum_{k=1}^{K} \sum_{l=1}^{L} w_t^{\mathrm{R},k,l} \right)$$

For each sample budget, $\bar{Z}_t^{\mathrm{C}}$ decreases (left) as a function of time while $\bar{Z}_t^{\mathrm{R}}$ remain relatively stable independent of time (right). The decrease in $\bar{Z}_t^{\mathrm{C}}$ is to

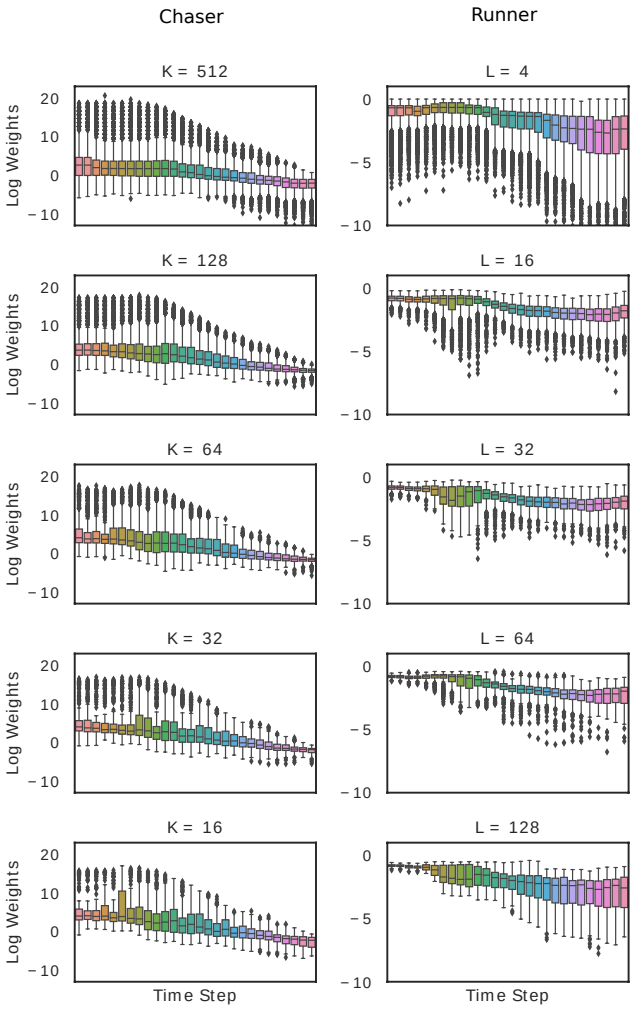

Figure 5: Box plots showing quantiles of log weights for the runner (left) and chaser (right) at each time step in the simulation for varying K and L.

be expected, given that the probability of intercepting the runner decreases as we approach the end of the episode.

To get a evaluate the weight variance at each time step, we compute the effective sample size (ESS), which for a set of $K \cdot L$ weights $\{w^{k,l}\}$ is defined as

$$\text{ESS} = \Big(\sum_{k=1}^{K}\sum_{l=1}^{L} w^{k,l}\Big)^2 \Big/ \Big(\sum_{k=1}^{K}\sum_{l=1}^{L}(w^{k,l})^2\Big).$$

Figure 4 (bottom row), shows the fractional ESS (normalized by $K \cdot R$) as a function of time for each sample budget. The effective sample size for the chaser weights increases over the course of the episode, reflecting that inference becomes easier owing to the previously mentioned conclusion of progressively decreasing runner detection probabilities as we reach the end of the episode.

Figure 5 shows quantiles with respect to restarts for log mean weights, which further confirms the trend in Figure 4. We show higher median log weights and lower number of outliers as K decreases and L increases, $(K, L) = (16, 128)$ results show that computed log weights are less robust when we draw a smaller number of samples from the outermost model.

8 Conclusion and Future Work

In the beginning of this paper, we considered the question, "How do we give autonomous agents the ability to infer the mental state of other agents?", and more importantly, "How do we reason about that mental state for decision making and planning?" We have taken a step towards this goal by contributing a model with several novel elements, including complex path planning, visibility and nested planning-as-inference. We have shown how relatively straightforward models of theory of mind can capture a variety of rich behavior, and that probabilistic programming is a natural way to describe those models. We experimentally demonstrated that runner detections increase as we increase the complexity of the chaser model, therefore showing that more complex models produce improved behavior, and thus improved detection rates. Additionally we show that nested reasoning results in lower-variance estimates of expected utility.

One of the virtues of a Bayesian approach is compositionality. While we assumed access to a high-level map, the same framework could be applied to a joint model that blends high-level reasoning with low-level perception. In such models, inferences driven by theory of mind models could go beyond goals and paths, and could additionally infer (for example) the existence of objects or other agents seen by the runner, but not by the chaser. Such integrated models may require inference metaprogramming; but how best to make such models computationally tractable is an open question.

9 Acknowledgements

We would like to acknowledge helpful feedback from a number of reviewers. IRS and DW gratefully acknowledge the support of DARPA under grant FA8750-16-2-0209, and additional support from the NSF CUAS. IRS and JWM additionally acknowledge support from startup funds provided by Northeastern.

References

[1] Marc Toussaint, Stefan Harmeling, and Amos Storkey. *Probabilistic inference for solving (PO)MDPs*. Tech. rep. EDI-INF-RR-0934. University of Edinburgh, 2006.

[2] Christian Naesseth, Fredrik Lindsten, and Thomas Schon. "Nested Sequential Monte Carlo Methods". In: *International Conference on Machine Learning*. 2015, pp. 1292–1301.

[3] Henry M Wellman. "The child's theory of mind". In: (1990).

[4] Nick Chater, Joshua B Tenenbaum, and Alan Yuille. "Probabilistic models of cognition: Conceptual foundations". In: *Trends in cognitive sciences* 10.7 (2006), pp. 287–291.

[5] Paul Bello and Nicholas Cassimatis. "Developmental accounts of theory-of-mind acquisition: Achieving clarity via computational cognitive modeling". In: *Proceedings of the Twenty-Eighth Annual Conference of the Cognitive Science Society*. 2006, pp. 1014–1019.

[6] Noah D Goodman, Chris L Baker, Elizabeth Baraff Bonawitz, Vikash K Mansinghka, Alison Gopnik, Henry Wellman, Laura Schulz, and Joshua B Tenenbaum. "Intuitive theories of mind: A rational approach to false belief". In: *Proceedings of the twenty-eighth annual conference of the cognitive science society*. 2006, pp. 1382–1387.

[7] Noah D Goodman, Chris L Baker, and Joshua B Tenenbaum. "Cause and intent: Social reasoning in causal learning". In: *Proceedings of the 31st annual conference of the cognitive science society*. Citeseer. 2009, pp. 2759–2764.

[8] Noah D Goodman and Andreas Stuhlmüller. "Knowledge and implicature: Modeling language understanding as social cognition". In: *Topics in cognitive science* 5.1 (2013), pp. 173–184.

[9] Muhammad Awais and Dominik Henrich. "Human-robot collaboration by intention recognition using probabilistic state machines". In: *Robotics in Alpe-Adria-Danube Region (RAAD), 2010 IEEE 19th International Workshop on*. IEEE. 2010, pp. 75–80.

[10] Alan Fern, Sriraam Natarajan, Kshitij Judah, and Prasad Tadepalli. "A Decision-Theoretic Model of Assistance." In: *IJCAI*. 2007, pp. 1879–1884.

[11] Truong-Huy Dinh Nguyen, David Hsu, Wee-Sun Lee, Tze-Yun Leong, Leslie Pack Kaelbling, Tomas Lozano-Perez, and Andrew Haydn Grant. "Capir: Collaborative action planning with intention recognition". In: *arXiv preprint arXiv:1206.5928* (2012).

[12] Dorsa Sadigh, S Shankar Sastry, Sanjit A Seshia, and Anca Dragan. "Information gathering actions over human internal state". In: *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE. 2016, pp. 66–73.

[13] Daphne Koller, David McAllester, and Avi Pfeffer. "Effective Bayesian inference for stochastic programs". In: *AAAI/IAAI*. 1997, pp. 740–747.

[14] Chris Frith and Uta Frith. "Theory of mind". In: *Current Biology* 15.17 (2005), R644–R645.

[15] Luke Zettlemoyer, Brian Milch, and Leslie P Kaelbling. "Multi-agent filtering with infinitely nested beliefs". In: *Advances in neural information processing systems*. 2009, pp. 1905–1912.

[16] Chris L Baker, Rebecca Saxe, and Joshua B Tenenbaum. "Action understanding as inverse planning". In: *Cognition* 113.3 (2009), pp. 329–349.

[17] Andreas Stuhlmüller and Noah D Goodman. "Reasoning about reasoning by nested conditioning: Modeling theory of mind with probabilistic programs". In: *Cognitive Systems Research* 28 (2014), pp. 80–99.

[18] Jan-Willem van de Meent, Brooks Paige, Hongseok Yang, and Frank Wood. "An Introduction to Probabilistic Programming". In: *arXiv:1809.10756 [cs, stat]* (Sept. 2018). arXiv: `1809.10756 [cs, stat]`.

[19] Noah Goodman, Vikash Mansinghka, Daniel Roy, Keith Bonawitz, and Joshua Tenenbaum. "Church: a language for generative models". In: *Uncertainty in Artificial Intelligence (UAI)*. 2008.

[20] Brian Milch, Bhaskara Marthi, Stuart Russell, David Sontag, Daniel L. Ong, and Andrey Kolobov. "BLOG: Probabilistic Models with Unknown Objects". In: *International Joint Conference on Artificial Intelligence (IJCAI)*. 2005, pp. 1352–1359.

[21] Avi Pfeffer. "IBAL: A probabilistic rational programming language". In: *International Joint Conference on Artificial Intelligence (IJCAI)*. 2001, pp. 733–740.

[22] Stan Development Team. *Stan: A C++ Library for Probability and Sampling, Version 2.4*. 2014.

[23] Vikash Mansinghka, Daniel Selsam, and Yura Perov. "Venture: a higher-order probabilistic programming platform with programmable inference". In: *arXiv preprint arXiv:1404.0099* (2014).

[24] L. M. Murray. "Bayesian state-space modelling on high-performance hardware using LibBi". In: *arXiv preprint arXiv:1306.3277* (2013).

[25] Adrien Todeschini, François Caron, Marc Fuentes, Pierrick Legrand, and Pierre Del Moral. "Biips: Software for Bayesian Inference with In-

teracting Particle Systems". In: *arXiv preprint arXiv:1412.3779* (2014).

[26] Frank Wood, Jan Willem van de Meent, and Vikash Mansinghka. "A New Approach to Probabilistic Programming Inference". In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2014, pp. 1024–1032.

[27] Noah D Goodman and Andreas Stuhlmüller. *The Design and Implementation of Probabilistic Programming Languages*. http://dippl.org. Accessed: 2017-8-22. 2014.

[28] Hong Ge, Kai Xu, and Zoubin Ghahramani. "Turing: A Language for Flexible Probabilistic Inference". In: *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*. Ed. by Amos Storkey and Fernando Perez-Cruz. Vol. 84. Proceedings of Machine Learning Research. Playa Blanca, Lanzarote, Canary Islands: PMLR, 2018, pp. 1682–1690.

[29] Dorit Borrman and Andreas Nuchter. *Dataset generated by Dorit Borrman and Andreas Nuchter of Jacobs University Bremen*. http://kos.informatik.uni-osnabrueck.de/3Dscans/. Accessed: 2017. 2017.

[30] Steven M LaValle. "Rapidly-exploring random trees: A new tool for path planning". In: (1998).

[31] M L Benedikt. "To Take Hold of Space: Isovists and Isovist Fields". In: *Environment and Planning B: Planning and Design* 6.1 (1979), pp. 47–65. DOI: 10.1068/b060047. eprint: http://dx.doi.org/10.1068/b060047. URL: http://dx.doi.org/10.1068/b060047.

[32] Vlad I Morariu, V Shiv Naga Prasad, and Larry S Davis. "Human activity understanding using visibility context". In: *IEEE/RSJ IROS Workshop: From sensors to human spatial concepts (FS2HSC)*. 2007.

[33] Tom Rainforth, Rob Cornish, Hongseok Yang, Andrew Warrington, and Frank Wood. "On Nesting Monte Carlo Estimators". en. In: *International Conference on Machine Learning*. July 2018, pp. 4267–4276.

[34] Tom Rainforth. "Nesting Probabilistic Programs". In: *Uncertainty in Artificial Intelligence* (Mar. 2018). arXiv: 1803.06328.

[35] Jan-Willem van de Meent, Brooks Paige, David Tolpin, and Frank Wood. "Black-Box Policy Search with Probabilistic Programs". In: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics* (2016), pp. 1195–1204.

[36] Emanuel Todorov. "Efficient Computation of Optimal Actions." In: *Proceedings of the National Academy of Sciences of the United States of America* 106.28 (2009), pp. 11478–11483. ISSN: 0710743106. DOI: 10.1073/pnas.0710743106.