AFRL-RI-RS-TR-2019-124

SCALABLE PROBABILISTIC PROGRAMMING FOR LEARNING AND ACTING IN COMPLEX DOMAINS

BRIGHAM YOUNG UNIVERSITY

JUNE 2019

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nations. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2019-124 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

/ S /
STEVEN DRAGER
Work Unit Manager

/ S /
QING WU
Technical Advisor, Computing & Communications Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
JUNE 2019	FINAL TECHNICAL REPORT	JUN 2016 – DEC 2018

4. TITLE AND SUBTITLE

SCALABLE PROBABILISTIC PROGRAMMING FOR LEARNING AND ACTING IN COMPLEX DOMAINS

5a. CONTRACT NUMBER
FA8750-16-2-0209

5b. GRANT NUMBER
N/A

5c. PROGRAM ELEMENT NUMBER
61101E

6. AUTHOR(S)

David Wingate

5d. PROJECT NUMBER
PPMK

5e. TASK NUMBER
SB

5f. WORK UNIT NUMBER
YU

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Brigham Young University
3332 TMCB
Provo, UT 84602

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory/RITA
525 Brooks Road
Rome NY 13441-4505

DARPA
675 North Randolph St.
Arlington, VA 22203-2114

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RI

11. SPONSOR/MONITOR'S REPORT NUMBER
AFRL-RI-RS-TR-2019-124

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The research objective of this seedling effort was to create probabilistic programming languages (PPLs) that are scalable and expressive enough to serve as the foundation for complex agents (such as autonomous robots) that must blend low-level perception and high-level reasoning. The technical strategy was to improve the theory and implementation of PPLs through a mix of language refinement, novel inference algorithm development, and hardware acceleration. Key results from this project include, strong results on simulator development, development of the "Runner-Chaser" model, development of probabilistic Schelling games, and failed experiments on high-performance, low-level perception.

15. SUBJECT TERMS

Probabilistic programming, deep learning, theory of mind

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			**STEVEN DRAGER**
U	U	U	UU	70	19b. TELEPHONE NUMBER (*Include area code*) **N/A**

Contents

List of Figures

1 Summary

The research objective of this seedling was to create probabilistic programming languages (PPLs) that are scalable and expressive enough to serve as the foundation for complex agents (such as autonomous robots) that must blend low-level perception and high-level reasoning. Our technical strategy was to improve the theory and implementation of PPLs through a mix of language refinement, novel inference algorithm development, and hardware acceleration. An important dimension of the proposed work is to blend state-of-the-art techniques in PPLs with the superior signal processing capabilities of deep neural networks (DNNs), both for model construction and inference.

Key results from this project include:

- **Strong results on simulator development.** In October of 2018, we launched **Holodeck**, the simulator described in the original proposal. It was well received, with close to 400 stars, dozens of forks, and thousands of clones.
- **Development of the "Runner-Chaser" model.** A large portion of our work was done on using probabilistic programming to model theory-of-mind for a complex multi-agent interaction scenario. In this scenario, a chaser must outwit a runner by reasoning about his likely actions. Importantly, the chaser has a mental model of the runner, who has a mental model of the chaser. Both agents are situated in a physically realistic world with complex robotics primitives such as path planners, trajectory optimizers, and visibility graphs. The results are compelling, and illustrate the power of probabilistic programming to describe complex, nested reasoning.
- **Development of probabilistic Schelling games.** The Runner-Chaser models are adversarial, but the same theory-of-mind foundation can be used in cooperative scenarios as well. As part of this research, we developed coordination games, and ran a comprehensive suite of experiments to validate them.
- **Failed experiments on high-performance, low-level perception.** As an attempt to integrate low-level perception and high-level reasoning, we developed a Tensorflow (TF)-based framework for sparse, direct alignment (a Simultaneous Localization and Mapping (SLAM) technique from the robotics community). This project attempted to implement structure-from-motion as an optimization problem using the TF framework; I felt that this would be a worthwhile investigation as a complement to our theory-of-mind work. While moderately successful, it was ultimately abandoned because of the computational complexity of the method.

Key changes from the initial proposal include:

- **Emphasis on theory-of-mind**. The original proposal was framed in terms of using PPLs to model autonomous agents in complex domains, with an emphasis on performance. We quickly realized that it was necessary to craft complex models first, and work on performance second. This resulted in a strong emphasis on the Runner-Chaser and Schelling coordination models.
- **Emphasis on self-normalized importance sampling.** We also learned that performance was governed primarily by the quality of the inference algorithm, and less so by the primitives forming the foundation

of the language. As a result, we focused our efforts not on developing a new language, but on refining the core theory of nested importance sampling, as required by our models.

2 Introduction

Decision making for the average person occurs often and more naturally than realized, and is an essential component to learning, and most importantly, living. Many researchers are currently focusing their energy towards integrating artificial intelligence into society. However, implications arise as the problem of decision making for autonomous agents becomes more than perceiving and responding to an environment. It requires giving agents the ability to adapt to social behavior and perceiving the intentions, beliefs, and desires of other agents. Furthermore, agents require the ability to reason *constantly* to make new decisions based on new and past observations until they complete a goal.

In order for autonomous agents to reason and perform decision-making in real-world settings, their respective models need to be designed with the ability to simulate theory of mind. This requires modeling the mental states of other agents and responding after reasoning about their intentions, desires, and beliefs. This becomes important for agents when they need to predict the goals, beliefs, and intentions of other agents in order to make their own decisions. Theory of mind becomes especially useful for decision making when agents require prior knowledge of the world and of how other agents behave to make decisions when there is limited (useful) observations available.

As part of this project, we developed a planning-as-inference framework in which agents perform nested simulation to reason about the behavior of other agents in an online manner. As a concrete application of this framework, we use probabilistic programs to model a high-uncertainty variant of pursuit-evasion games in which an agent must make inferences about the other agents' plans to craft counter-plans. Our probabilistic programs incorporate a variety of complex primitives such as field-of-view calculations and path planners, which enable us to model quasi-realistic scenarios in a computationally tractable manner. We perform extensive experimental evaluations which establish a variety of rational behaviors and quantify how allocating computation across levels of nesting affects the variance of our estimators.

3 Methods, Assumptions, and Procedures

We use standard research methods in the field of machine learning. This includes building on others' work; integrating diverse ideas; attempting, whenever possible, to build on a theoretically sound foundation; carefully thought out metrics and benchmarks; and empirical validation of hypotheses.

It is the explicit goal of the Brigham Young University (BYU) Perception, Control and Cognition laboratory to create machine learning tools that are practically relevant across a wide variety of disciplines, and that have real-world impact. It is therefore our policy to release all code as open-source, to publish all results in reputable scientific venues, and to make all data publicly accessible via our website (http://pcc.cs.byu.edu).

Our key assumptions include:

- That probabilistic programming is (perhaps) the only viable way to model the thought processes of other agents;
- That probabilistic programming can ultimately be made fast enough for use in real-time agents;
- That deep learning can be integrated with probabilistic programming;
- That research into how to construct compelling autonomous agents will lead to breakthroughs for autonomous robots, drones, autonomous cars, and a diversity of application areas that can leverage probabilistic programming.

4 Results and Discussion

We have collected our key results into the following three sections. Section 4.1 reports on our work on theory of mind for the runner-chaser model (a submission for this work is currently under review at International Conference on Machine Learning (ICML) 2019); Section 4.2 discusses our work on Schelling coordination games; and Section 4.10 shows some detail about the Holodeck simulator.

4.1 Theory of Mind and interacting agents

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 section, we explore how theory of mind can be implemented on high-uncertainty pursuit-evasion games using nested simulations in the form of probabilistic programs. Contrary to classic pursuit-evasion problems, which are minimax games with fully-observable environments, we develop a domain in which a *chaser* agent must reason about possible locations of a *runner* agent, who seeks to avoid detection. The runner's intended start location, goal location, and likely path to the goal are initially unknown to the chaser. The runner knows the current location of the chaser, but not the chaser's future trajectory. This results in a setting where agents must reason about the reasoning of other agents under both state and outcome uncertainty.

We perform online planning in this domain using planning-as-inference approach [1]. We formulate the chaser and the runner models as nested probabilistic programs that are conditioned according to the desired behavior of the respective agent. The model of the chaser is conditioned to *maximize* the likelihood of detection, and the runner is conditioned to *minimize* likelihood of detection. At each point of time, the chaser imagines possible future trajectories, along with possible runner trajectories, and selects a move that has a high probability of detection.

Formulating our models as probabilistic programs also make it possible to incorporate semi-realistic deterministic primitives such as path planners and visibility graphs. Moreover, an advantage of our planning-as-inference approach is that we can employ nested importance sampling methods [2] for probabilistic program inference to perform recursive reasoning. This enables us to implement tractable inference in a partially-observable multi-agent domain with continuous actions. To our knowledge, this work is the first to adapt nested importance sampling methods to online planning in multi-agent systems.

We evaluate a range of scenarios to demonstrate that nested Bayesian reasoning leads to rational behaviors in which agents maximize the relative utility at each level in the model. We evaluate the effect of model complexity on runner detection rates relative to basic models. Finally we perform extensive experiments to quantify the effect of allocation of computational budget across levels in the model on the variance of the estimated expected utility.

4.1.1 Background

4.1.1.1 Pursuit-Evasion

Pursuit-evasion games are adversarial multi-agent problems that generally consist of two parties, the agents in pursuit (chasers) and those evading capture (runners). Although variants of this problem include altering the number of agents in each party and their respective traveling speeds, the objective is always to either minimize task-completion time, or to find the shortest distance the chaser(s) must travel in order to meet the runner at a single location (zero Euclidean distance apart) [3, 4]. In control theory, several assumptions are maintained in the off-line optimization process, (1) agents travel in straight lines in order to travel along shortest-distance trajectories, and (2) the environment (and other agents) are fully-observable. The optimization complexity increases with the additions of 2-dimensional polygonal obstacles and the reduction to partially-observable environments. Our formulation differs such that we develop partially observable variants of the pursuit-evasion problem which have a high obstacle count, limited field of view, and noisy trajectory planning. Since the chaser is unaware of the runner's location, the chaser's objective is to detect the runner before the runner reaches its goal, which is also unknown to the chaser.

4.1.1.2 Theory of Mind

Human children develop theory of mind during their early years, generally between the ages of three and six [5, 6]. Bello and Cassimatis [7] 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. [8] 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 [9]. Additionally, the same group explores language as a type of social cognition [10].

The development of theory of mind in machines leads naturally to interaction with their human counterparts. Awais and Henrich [11], Fern et al. [12], and Nguyen et al. [13] 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. [14] 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. [15] present an inference algorithm for recursive stochastic programs. Frith and Frith [16] 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. [17] 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. [18] 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 [19] 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 [19] considers problems for a single decision.

4.1.1.3 Probabilistic Program Inference

To represent our generative model cleanly and to perform inference in it, we employ the tools of probabilistic programming [20]. 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 is fast and only limited 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 [20]. 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 [21–25]. Methods based on importance sampling and sequential Monte Carlo (SMC) have become popular [26–30], due 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 (field-of-view calculations), or even distributions that involve optimization problems as a subcomponent of the distribution.

4.1.2 Simulation Primitives

Although probabilistic programming has previously been used to model theory of mind [19], past implementations have thus far considered relatively simple problems involving a small number of decisions. In this section, 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. 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 [31], shown in Figure 1 (a).

Path planning and trajectory optimization. We model paths using a RRT [32], 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 Figure 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 [33, 34]. Given a map, chaser location, and runner location, the isovist determines the likelihood that the runner was detected. Although an isovist usually uses a

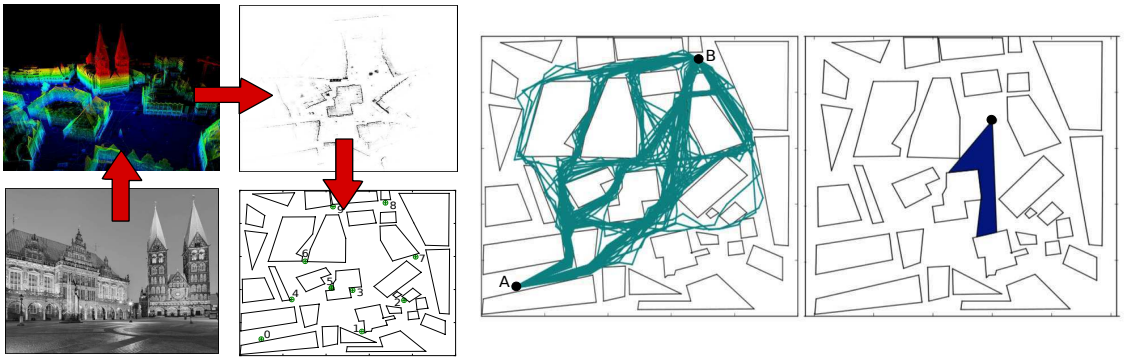

Figure 1: (a) We generate a coarse polygonal city map from point cloud data of the city of Bremen, Germany. (b) Visual distribution over paths a runner may take modeled with Random-Exploring Random Trees, RRTs, from points A to B. (c) a 45° isovist, or range of sight, of the chaser. The isovist is properly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

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 Figure 1 (c). The direction of the chaser's line of sight is determined by the imagined location of the runner.

4.1.3 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 imagined runner trajectories and assigns higher probability to trajectories in which detection is likely. 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 models work in tandem to create nuanced inferences about where the chaser believes the runner might be, and what counter-plan will maximize the 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 $x_1^{\mathrm{C}} = x_{\mathrm{START}}^{\mathrm{C}}$. For time points $t = 2 \ldots T$, the model samples a weighted partial trajectories $(x_{1:t}^{\mathrm{C}}, w_t)$ from the outermost CHASER model. After the final iteration, the model returns full trajectories $x_{1:T}^{\mathrm{C}}$.

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 $x^{\text{C}}_{\text{GOAL}}$ at random and uses the RRT planner to sample a possible future trajectory $x^{\text{C}}_{t:T}$. 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 $T^{\text{C}}_{\text{VISIBLE}}$ during which the runner is visible to the chaser. The chaser then conditions the sampled trajectories by defining a weight $w^{\text{C}} = \exp(\alpha\, T^{\text{C}}_{\text{VISIBLE}})$. As we will discuss below, this corresponds to assigning a utility proportional to $T^{\text{C}}_{\text{VISIBLE}}$ 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 $x^{\text{C}}_{1:t}$, together with a weight $w^{\text{C}} \cdot w^{\text{R}}$ 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 that of the chaser. We implement these assumptions in the probabilistic program as follows. The runner model first selects a start location $x^{\text{R}}_{\text{START}}$ and goal location $x^{\text{R}}_{\text{GOAL}}$ at random, and then samples a random trajectory $x^{\text{R}}_{1:T}$ using the RRT planner. The runner then imagines a future chaser trajectory by selecting a goal location $\tilde{x}^{\text{C}}_{\text{GOAL}}$ at random and sampling $\tilde{x}^{\text{C}}_{t:T}$ from the innermost model. We then condition this sample by computing the total time of visibility $T^{\text{R}}_{\text{VISIBLE}}$, based on both the known past trajectory $x^{\text{C}}_{1:t-1}$ and the imagined future trajectory $\tilde{x}^{\text{C}}_{t:T}$ of the chaser. Finally, we assign a weight $w^{\text{R}} = \exp(-\alpha T^{\text{R}}_{\text{VISIBLE}})$, which corresponds to a negative utility (i.e. a cost) proportional to $T^{\text{R}}_{\text{VISIBLE}}$ 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 $\tilde{x}^{\text{C}}_{\text{GOAL}}$ at random and then uses the RRT planner to sample a random future trajectory $\tilde{x}^{\text{C}}_{t:T}$. Since this model is not conditioned in any way, it returns weight 1.

4.1.4 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 sampling is *not* a form of nested Monte Carlo estimation as discussed in Rainforth et al. [35] and Rainforth [36]. We discuss the distinctions between the two methods below.

We implement conditioning using a planning-as-inference formulation [1, 37]. In planning-as-inference problems, a target density $\pi(x) = \gamma(x)/Z$ is defined in terms of a prior over trajectories $p(x)$ and the exponent of a utility or reward $R(x)$

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

Algorithm 1 Schematic representation of the Chaser-Runner model. The EPISODE model performs SMC in which moves are sampled 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 t, we propose K future trajectories for the chaser and $K \times L$ trajectories for the runner.

query EPISODE($x^{\mathrm{C}}_{\mathrm{START}}$) ▷ Episode model
  for k **in** $1 \dots K$ **do**
    $x^{\mathrm{C},k}_1 = x^{\mathrm{C}}_{\mathrm{START}}$
  for t **in** $2 \dots T$ **do**
    for k **in** $1 \dots K$ **do**
      $x^{\mathrm{C},k}_{1:t}, w^k_t \leftsquigarrow \mathrm{CHASER}(x^{\mathrm{C},k}_{1:t-1})$
    for k **in** $1 \dots K$ **do**
      $a \sim \mathrm{Categorical}\left(\frac{w^1_t}{\sum_k w^k_t}, \dots, \frac{w^K_t}{\sum_k w^k_t}\right)$
      $x^{\mathrm{C},k}_{1:t}, w^k_t = x^{\mathrm{C},a}_{1:t}, \frac{1}{K}\sum_k w^k_t$
  return $(x^{\mathrm{C},1}_{1:T}, w^1_T), \dots, (x^{\mathrm{C},K}_{1:T}, w^K_T)$
query CHASER($x^{\mathrm{C}}_{1:t-1}$) ▷ Outer Model
  $x^{\mathrm{C}}_{\mathrm{GOAL}} \sim \mathrm{Uniform}(\{x_{\mathrm{A}}, \dots, x_{\mathrm{J}}\})$
  $x^{\mathrm{C}}_{t:T} \sim \text{RRT-PLAN}(x^{\mathrm{C}}_{t-1}, x^{\mathrm{C}}_{\mathrm{GOAL}})$
  for l **in** $1 \dots L$ **do**
    $x^{\mathrm{R},l}_{t:T}, w^{\mathrm{R},l} \leftsquigarrow \mathrm{RUNNER}(x^{\mathrm{C}}_{1:t-1})$
    $T^{\mathrm{C},l}_{\mathrm{VISIBLE}} = \text{TIME-VISIBLE}(x^{\mathrm{R},l}_{t:T}, x^{\mathrm{C}}_{t:T})$
    $w^{\mathrm{R},l} = \exp(\alpha\, T^{\mathrm{C},l}_{\mathrm{VISIBLE}})$
  return $x^{\mathrm{C}}_{1:t}, w^{\mathrm{C}} \cdot \left(\frac{1}{L}\sum_l w^{\mathrm{R},l}\right)$
query RUNNER($x^{\mathrm{C}}_{1:t-1}$) ▷ Middle Model
  $x^{\mathrm{R}}_{\mathrm{START}} \sim \mathrm{Uniform}(\{x_{\mathrm{A}}, \dots, x_{\mathrm{J}}\})$
  $x^{\mathrm{R}}_{\mathrm{GOAL}} \sim \mathrm{Uniform}(\{x_{\mathrm{A}}, \dots, x_{\mathrm{J}}\})$
  $x^{\mathrm{R}}_{1:T} \sim \text{RRT-PLAN}(x^{\mathrm{R}}_{\mathrm{START}}, x^{\mathrm{R}}_{\mathrm{GOAL}})$
  $\tilde{x}^{\mathrm{C}}_{t:T}, \tilde{w}^{\mathrm{C}} \leftsquigarrow \text{NAIVE-CHASER}(x^{\mathrm{C}}_{t-1})$
  $T^{\mathrm{R}}_{\mathrm{VISIBLE}} = \text{TIME-VISIBLE}(x^{\mathrm{R}}_{1:T}, \{x^{\mathrm{C}}_{1:t-1}, \tilde{x}^{\mathrm{C}}_{t:T}\})$
  $w^{\mathrm{R}} = \exp(-\alpha\, T^{\mathrm{R}}_{\mathrm{VISIBLE}})$
  return $x^{\mathrm{R}}_{t:T}, w^{\mathrm{R}} \cdot \tilde{w}^{\mathrm{C}}$
query NAIVE-CHASER(x^{C}_{t-1}) ▷ Inner Model
  $\tilde{x}^{\mathrm{C}}_{\mathrm{GOAL}} \sim \mathrm{Uniform}(\{x_{\mathrm{A}}, \dots, x_{\mathrm{J}}\})$
  $\tilde{x}^{\mathrm{C}}_{t:T} \sim \text{RRT-PLAN}(x^{\mathrm{C}}_{t-1}, \tilde{x}^{\mathrm{C}}_{\mathrm{GOAL}})$
  return $\tilde{x}^{\mathrm{C}}_{t:T}, 1$

The normalizing constant $Z = \mathbb{E}[\exp(R(x))]$ is sometimes referred to as the desirability [38].

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

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

In this density, the reward $\alpha(T^{\text{C}}_{\text{VISIBLE}} - T^{\text{R}}_{\text{VISIBLE}})$ 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 on 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.

4.1.5 Nested Importance Sampling

We can perform inference in the chaser-runner model using Monte Carlo methods for probabilistic programs. Algorithm 1 defines an importance sampling scheme. At each time t, we sample $x^{\text{C}}_t \sim \pi_t(x^{\text{C}}_t \mid x^{\text{C}}_{t-1})$ from the marginal of the target density above. To do so, we sample K particles from the CHASER model (line 6). For each sample, in the CHASER, we draw L samples from the RUNNER model (line 15). We then perform resampling to select K of the resulting $K \cdot L$ particles, which corresponds to performing SMC sampling within the EPISODE model (lines 8-9).

When $L = 1$, this sampling scheme reduces to standard SMC inference for probabilistic programs [28]. When $L > 1$ it can be understood as a form of nested importance sampling [2]. Note that in this sampling scheme, each of the L samples corresponds to a *different* runner trajectory $x^{\text{R},k,l}_{1:T}$, but that the reward for this trajectory is evaluated relative to the *same* past $x^{\text{C},k}_{1:t-1}$ and imagined future $x^{\text{C},k}_{t:T}$ 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 problem if we defined the weight

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

by averaging the reward over chaser trajectories

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

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

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

This is sometimes referred to as nested conditioning, in the context of probabilistic programming systems [36]. 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.

The approach in Algorithm 1 differs subtly from standard nested importance sampling approaches. Nested importance sampling was introduced as a means of reasoning about proposals in importance sampling that are themselves generated by means of another importance sampling algorithm. Concretely, suppose that we

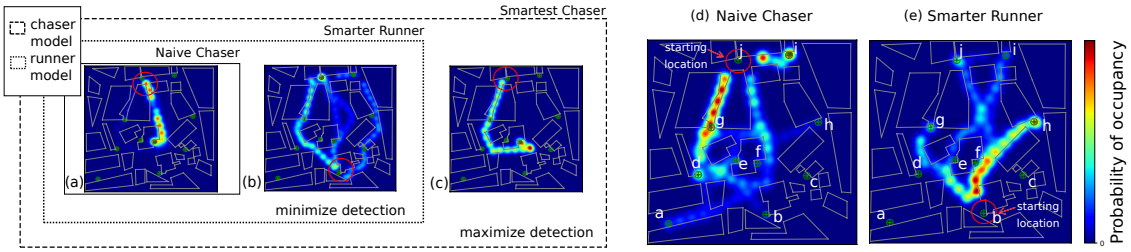

Figure 2: **Chaser and runner trajectories** in the innermost, middlemost, and outermost models where locations circled in red are the starting locations for each agent. We show posterior distributions of L *runner* and *naive chaser* paths, when $(K, L) = 128, 16$ for a single resampled sample k. (a)-(c) 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 .

have an importance sampling mechanism that targets an unnormalized proposal density $x, w^{\mathrm{P}} \leftsquigarrow \gamma^{\mathrm{P}}(x)$ and wish to define an importance sampler that for an unnormalized target density $x, w^{\mathrm{T}} \leftsquigarrow \gamma^{\mathrm{T}}(x)$, then we can do so by defining the importance weight

$$w^{\mathrm{T}} = \frac{\gamma^{\mathrm{T}}(x) w^{\mathrm{P}}}{\gamma^{\mathrm{P}}(x)}, \qquad\qquad x, w^{\mathrm{P}} \leftsquigarrow \gamma^{\mathrm{P}}(x).$$

A corollary of this identity is that we may compose importance samplers to sample different subsets of variables in any generative model, e.g. we could propose using two importance samplers

$$x_2, w_2^{\mathrm{P}} \leftsquigarrow \gamma^{\mathrm{P}}(x_2 \mid x_1) \qquad\qquad x_1, w_1^{\mathrm{P}} \leftsquigarrow \gamma^{\mathrm{P}}(x_1).$$

and define the importance weight

$$w^{\mathrm{T}} = \frac{\gamma^{\mathrm{T}}(x_1, x_2)\, w_1^{\mathrm{P}}\, w_2^{\mathrm{P}}}{\gamma^{\mathrm{P}}(x_2 \mid x_1)\gamma^{\mathrm{P}}(x_1)}.$$

In the sampling scheme Algorithm 1, we sequentially sample moves $x_t^{\mathrm{C}}, w_t^{\mathrm{C}} \leftsquigarrow \gamma_t(x_t^{\mathrm{C}} \mid x_{t-1}^{\mathrm{C}})$ from an importance sampler which has the same unnormalized density as the target, which can be understood as a special case of nested importance sampling in which $\gamma^{\mathrm{T}}(x) = \gamma^{\mathrm{P}}(x)$.

4.1.6 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*, which serve to evaluate the trade-offs in allocating our sample budget across different levels of nesting in the model.

4.1.6.1 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

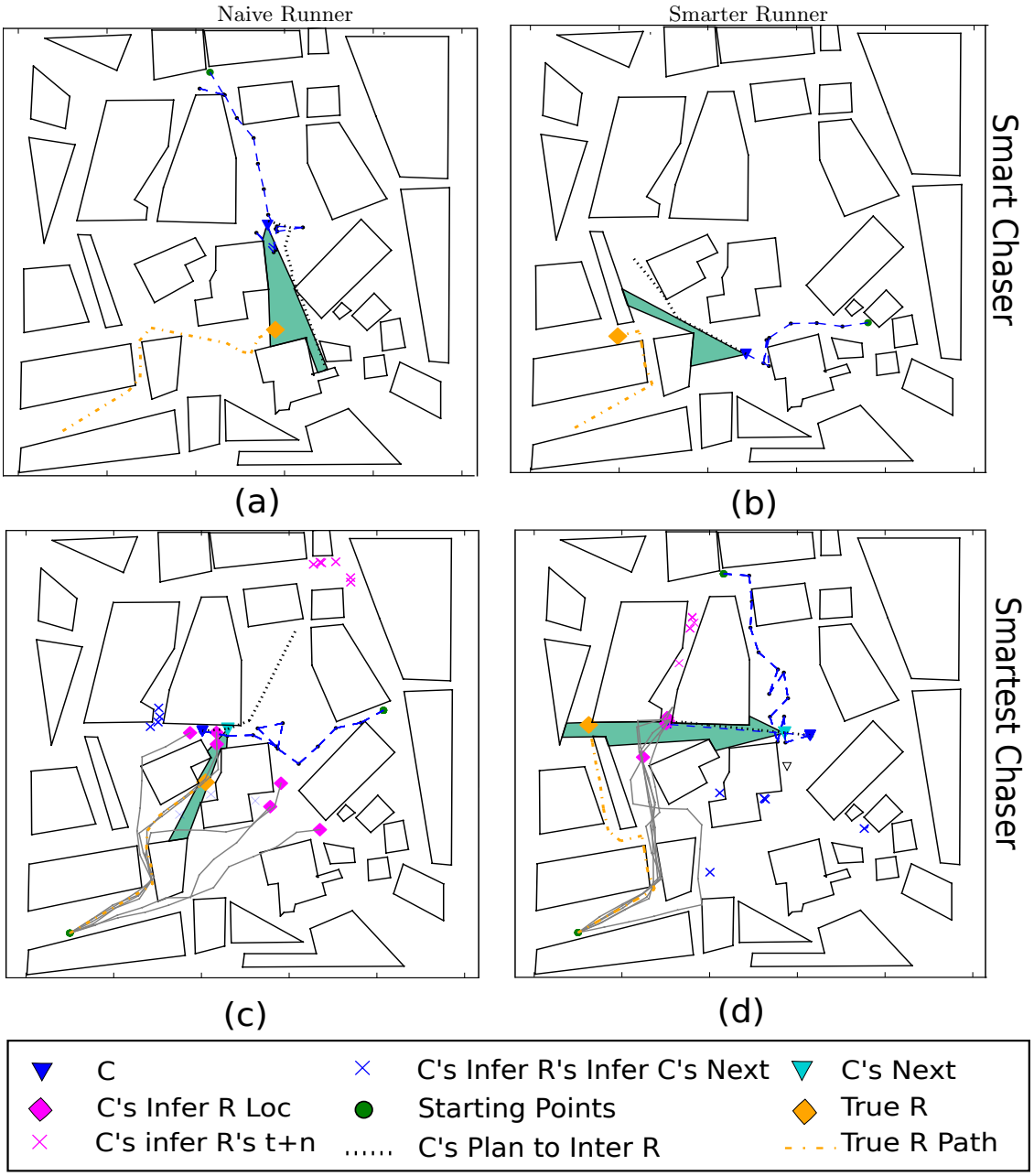

Figure 3: (a) Smart chaser playing against a naive runner, where the chaser anticipates the the intersection point and heads in the correct direction to detect the runner. (b) Smart chaser playing against a smarter runner. (c) Smartest chaser against the naive runner. (d) Smartest Chaser infers the runner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

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.

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.

4.1.6.2 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.

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.

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.

Table 1: Detection Rates for Agent Model Variants

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

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.

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.

4.1.6.3 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 $K\,L = 2048$ 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}, which corresponds to just over 4 million calls to the RRT planner.

In Figure 4 (top row), we show the log mean log weights for the chaser (left) and runner (right) at each time

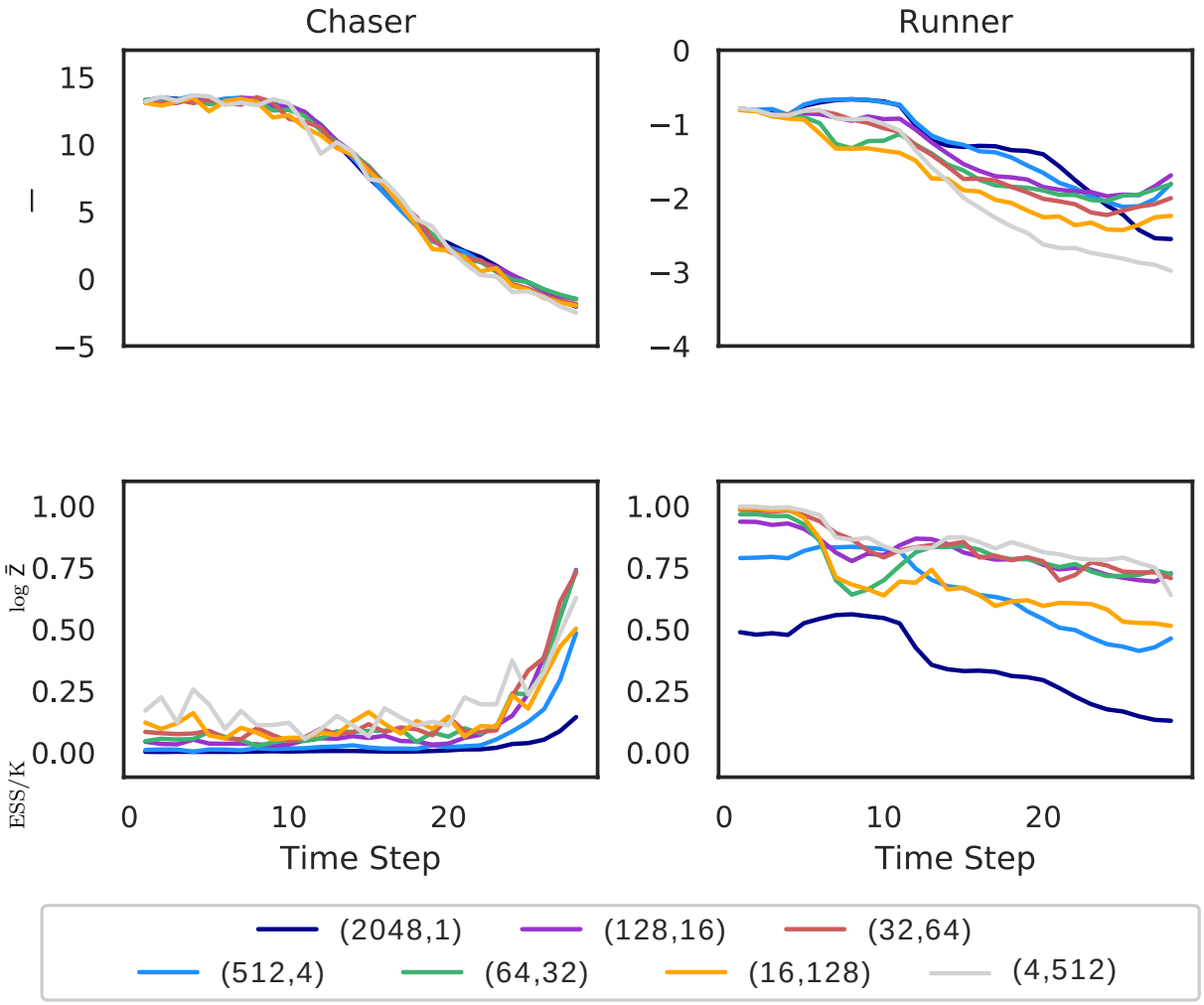

Figure 4: Log mean log weights, $\log \bar{Z}$, and Fractional ESS (normalized by K) as a Function of Time for each sample budget. **Top Row:** $\log \bar{Z}^{\mathrm{R}}$ for the middlemost model (left), and $\log \bar{Z}^{\mathrm{C}}$ the outermost model, (right). **Bottom Row:** The fractional ESS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

point t

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

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

To evaluate the weight variance at each time step, we compute the effective sample size (ESS), which for a

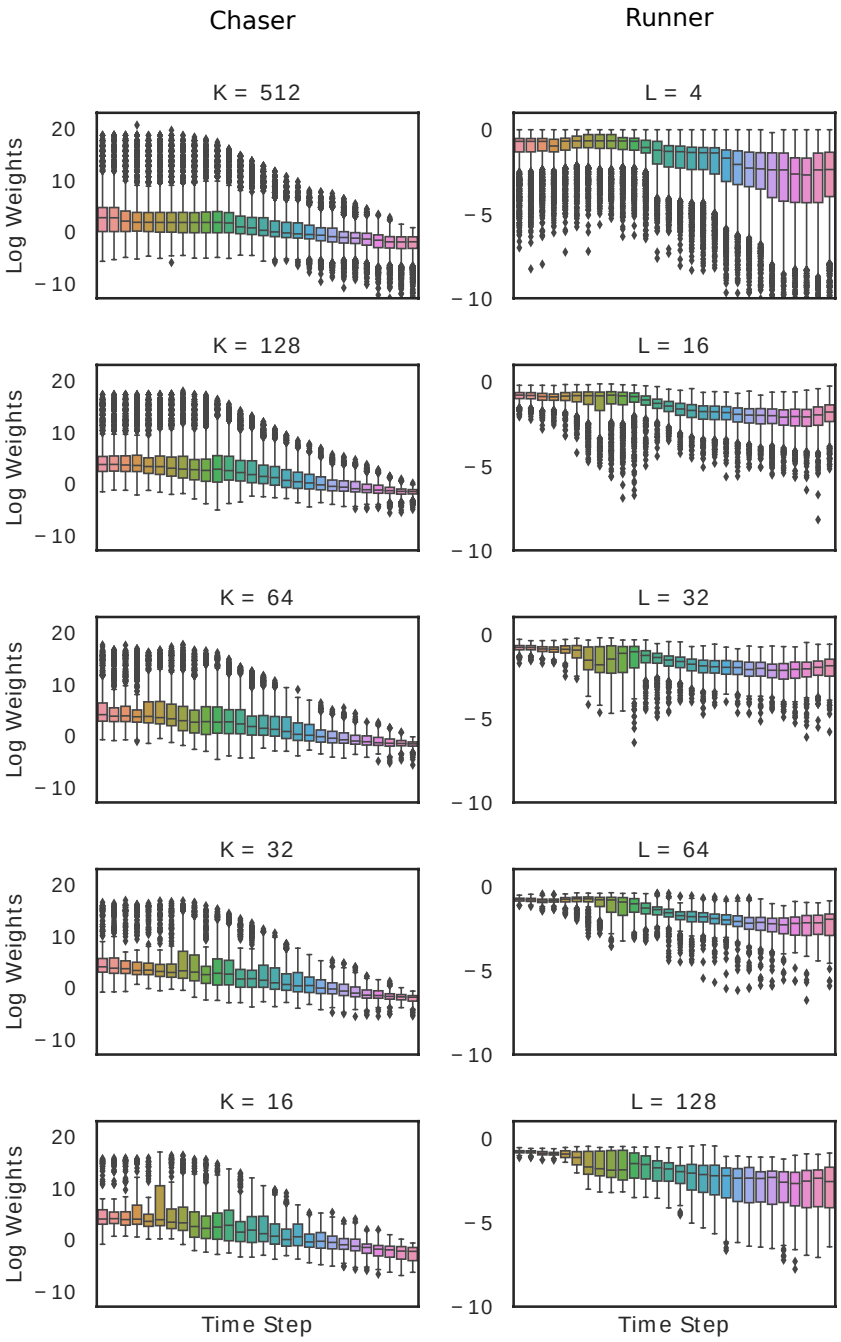

Figure 5: Box plots showing quantiles of log weights for the runner (left) and chaser (right) at each time step in the simulation for varying K and L.

set of $K \cdot L$ weights $\{w^{k,l}\}$ is defined as

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

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 less 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.

4.1.7 Conclusion and Future Work

In this section, we have introduced a high-uncertainty variant of pursuit-evasion games where agents are required to reason about other agents' reasoning in order to accomplish their respective goals of pursing or evading other agents. We developed a planning-as-inference framework that enables us to perform online planning by way of nested simulations. By formulating our models as probabilistic programs, we can incorporate semi-realistic deterministic primitives such as path planners and field-of-view calculations.

To our knowledge, nesting importance sampling methods have not previously been applied to online planning in multi-agent systems. Relative to existing approaches that model theory of mind with nested probabilistic programs, our work is the first to reason about agents in a time-dependent manner by repeatedly making inferences at each step of a simulation. We empirically demonstrate that nested Bayesian reasoning leads to rational behaviors and that increasing model complexity to incorporate reasoning about reasoning outperforms non-nested models. Finally, our empirical evaluations indicate that nested reasoning results in lower-variance estimates of expected utility.

An advantage of probabilistic programming approaches is their compositionality. While here we assume knowledge of a high-level map, our framework could be applied to a joint model that blends high-level reasoning with low-level perception. In such models, inferences in theory of mind models could go beyond goals and paths, and could serve to infer (for example) the existence of objects or other agents seen by the runner, but not by the chaser. A future line of research is how to enable such integrated models via inference meta-programming architectures.

4.2 Schelling Coordination games

This section works on a special application of applying theory of mind to autonomous agents. As well, we do so with the tools of probabilistic programming. Specifically we work with two simulated agents, whom we refer to as Alice and Bob. These experiments especially focus on analyzing the behavior of agents when attempting to model their complexities. As well, we point out attributes in our models that may explain the manifested behavior of these agents interacting. This section focuses on having Alice and Bob play a series

of games where their objectives are either to avoid one another, end up at the same goal location, or find one another on a complex map filled with obstacles. Each of these games allow different kinds of observations for the agents when trying to reach their goals. For example, in games where Alice and Bob must end at the same location, they will have a fully-observable environment of the map and of each other's locations, but when Alice and Bob are placed in a more convoluted map and must find one another, their environment becomes partially observable. This means that Alice and Bob are both unaware of each other's movements until they end up in place where they have line of sight with one another. Then they can use that information to end up n the same location. In each of these games, we experiment with how we can apply theory of mind, and test if it plays an important role in their outcomes, especially when there is limited available observations for the agents. If one agent has the prior knowledge that the other agent is also looking to find its location, then the goal is to have it simulate the actions of the other and infer what the other agent would plan based on what the agent's own objective is. However, these models increase in complexity as we attempt to simulate theory of mind and as well, computation time increases. Therefore, our experiments will demonstrate if applying such methods are valuable and enhancing to our agents' behavior.

This section's objective is to first give a short discussion of a previous example of a cognitive game, which we base our experiments on. Then we discuss previous approaches to simulating decision making for agents, and its progression towards probabilistic programming, more specifically, planning as inference. The rest of the section focuses on a series of experiments that demonstrate how to write probabilistic programs that can perform goal inference, and most importantly, simulate theory of mind. Our experiments begin by demonstrating how goal inference can be used with path-planning of other agents. We discuss how setting up probabilistic programs can help us leverage observations and noisy data. As well, we discuss inference algorithms and how the number of samples influence posterior samples. We show different examples where agents require inferring the goals of other agents in order to accomplish a goal. To compare, we discuss how we can add theory of mind into models in order to simulate "thinking" about agents that "think" about us as well. We compare how these two models influence the behavior of agents while attempting to accomplish their respective goals. We move towards explaining more complex situations where agents require more complex models in order to act rationally and meet their goals. We discuss the complexity of the problem and how the models must adjust. Then we show how adjusting the models influence agent behavior.

4.3 Related Works

An example of a coordination game is discussed in [19]'s work centered on *reasoning about reasoning* through nested conditional inference. Their example introduces a scenario of two agents, Alice and Bob, who desire to meet at some location. These agents share common knowledge that there are two possible meeting locations. However, one location is slightly more popular than the other. [19] formalize this game as a conditional distribution where one agent conditions its location on its partner choosing the same location. Since each of their agents begin with a bias of the other agent choosing the more popular place with higher probability, each are more likely to go to the more popular place. This is specifically demonstrated as an instance of *planning as inference*, where they transform the problem from maximizing the expected utility to maximizing the likelihood. In their experiments, the proposed approach demonstrates convergence of the most popular meeting location.

This particular experiment is designed for the agent to make a single final decision of where it must go to

increase its chances of meeting the other agent at the same location. However, this example lacks experiments needed to demonstrate how nested conditioning can apply to more convoluted and realistic models. An example of a more complicated model would be one where agents constantly need to make new decisions conditioned on commonly known prior knowledge and a collection of past and new observations.

Baker [39] illustrates a scenario where they successfully model the beliefs, intentions, and desires of an agent while traveling on a campus map looking for a particular food truck. This is attempted with an implementation of a theory-based Bayesian (TBB) framework, which models the structured knowledge of theory of mind. Although they are able to model theory of mind through time with new and past observations, this implementation does not demonstrate coordination with other agents or autonomous decision making.

4.4 Agents as Probabilistic Programs

The flexibility of modeling agents with probabilistic programs is infinite, which is a benefit to the designer of such models. As with design, approaches vary. One recognized approach of modeling agents with probabilistic programs is by the implementation of agents that compute rational policies. This allows for the agent to be modeled in a very structured design that use environments, sets of actions, transition functions, and utility functions. The decision rule is take an action that maximizes utility. Usually, these agents are designed to take a state as input and return an action. This approach can be expanded to more complicated states, actions, and so forth. However, there is an alternative to computing the optimal action for any problem, and we do this by treating the act of choosing an action as an inference problem.

One simple way of switching from a policy to an inference approach is to instead sample random actions from a uniform distribution and condition on the preferred outcome. We can then infer from our observed consequences what action caused our preferred outcome. This is known as *planning as inference* [40]. Our agents solve the same problem as the maximizing utility agent, but uses planning as inference. This is the approach that we will use for our experiments. We create probabilistic programs and use inference algorithms to sample from the posterior distribution, and then use those outcomes to help the agents make rational decisions to complete their goal.

4.5 The Big Picture

Our efforts are toward simulating intelligent agent interaction with the application of theory of mind on other agents. These agents may have different objectives, therefore it's important for these agents to have the ability to model the mental states of *several* other agents. For example, consider the game where agents Alice and Bob share a common goal of meeting at some location on the map. However, Alice and Bob are unaware of each other's location and must take under consideration the challenge of *remaining* undetected by a third agent. We call this third agent Chuck, who is actively searching for at least one of the other two agents. This game is a mix of collaboration between Alice and Bob, and opposition, between the team of Alice and Bob, and Chuck. This problem introduces several new complexities: (1) Alice and Bob must independently use theory of mind to converge on a mutual meeting location, and (2) simultaneously implement theory of mind to do so while conditioning on not being detected by Chuck. In summary, Alice and Bob must consider more than just the single goal of inferring where the best location is to meet, but must also consider doing

Figure 6: Agents Alice and Bob, **A** and **B** on the map respectively, are collaborative agents that share the common goal of meeting at some unknown location on the map without getting detected by the opposing agent Chuck, **C**. (shown on the left). On the right, we show an example of a possible scenario of the problem where agent A and B meet on the top right of the map without detection from agent C.

so without being detected by Chuck. Figure 6 shows an example of the three agents on a map of Bremen Germany, [31], and where the three agents begin a simulation of the proposed problem. Alice and Bob are represented as **A** and **B** in the figure respectively. On the left-most map, Alice and Bob begin at opposite ends of the map, while Chuck, **C**, begins at the center. On the right of Figure 6, we show an ideal example of how Alice and Bob can plan to accomplish both goals of meeting with one another while remaining undetected. Figure 7 illustrates a Bayesian model for the described problem, where Alice's plan to meet Bob depends on her (i) true location, (ii) belief of Bob's location, (iii) belief of Bob's belief of Alice's plan, (iv) belief of Chuck's location, (v) any observations of Chuck's location, and, of course, (vi) the map. Bob's plan to meet Alice requires similar relationship dependencies, and lastly, detection of Alice or Bob depends on Chuck's true location.

In efforts towards our *big picture* goal, we begin with the intention of simulating agents that collaborate to accomplish some goal. Since we are working towards agents planning on a convoluted map, it's important to start out by creating probabilistic programs that can be used to infer the future locations and goals of other agents. For example, if Alice and Bob can predict in what area of the map they should go to find one another, it would be an advantage towards meeting their goal. As well, if they could predict locations where Chuck will most likely be, Alice and Bob could avoid those areas. Therefore, the next section will discuss how goal inference works exactly and how agents can use it to predict future steps and goals of other agents.

4.6 Goal Inference

An important element to implementing theory of mind or any collaboration, is to have the ability to perform goal inference for agents. By embedding the ability to perform goal inference into agents, they will be able to

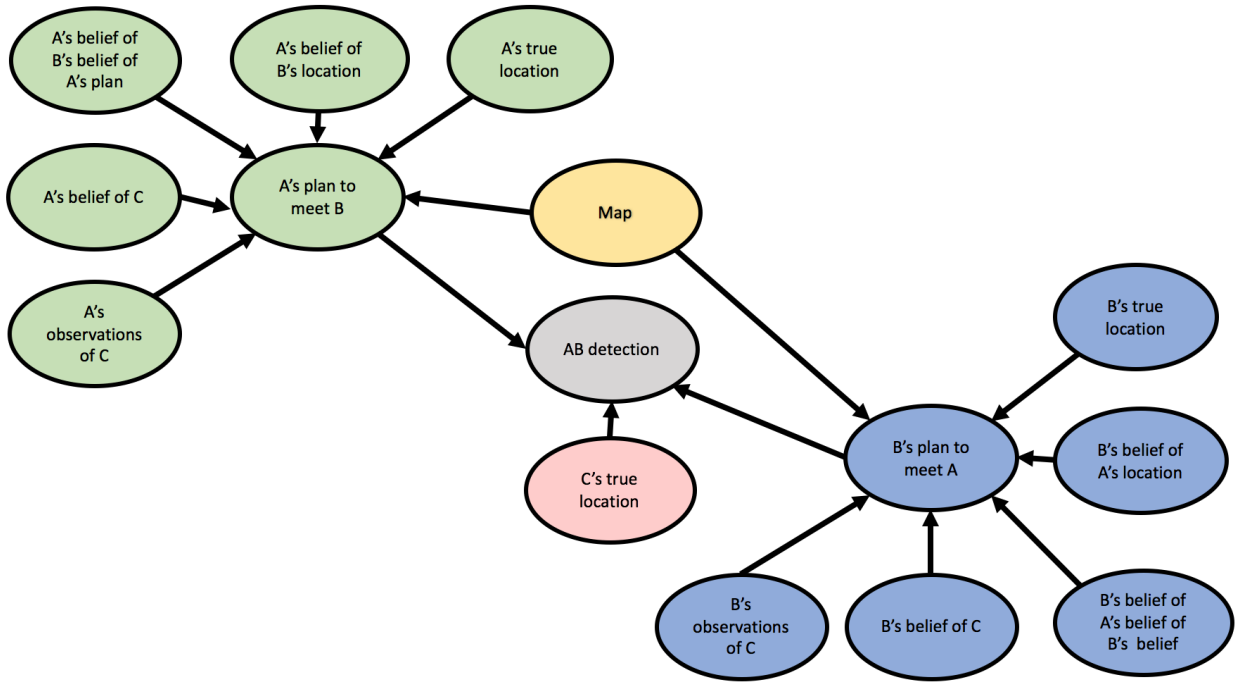

Figure 7: Bayesian model of the Alice, Bob, and Chuck mixed collaborative and competitive scenario. See text for details.

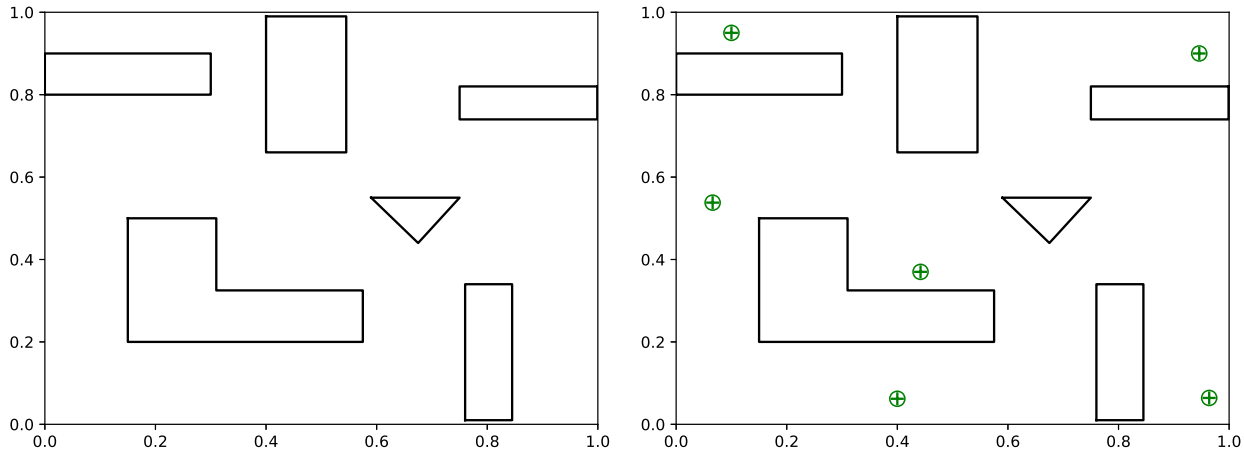

Figure 8: Left: Basic map layout with 6 basic obstacles of various shapes and sizes. Right: Marked in green circle-crosses, we show possible start and goal locations for this basic map.

predict future locations of other agents, thus giving them the ability to make informed decisions contingent on their own respective goals. For example, if one agent is actively trying to avoid another agent, that agent requires the ability to infer future goals of the other agent to actively avoid intercepting its path to its own goal location. In future sections, we will discuss and demonstrate how goal inference is essential to model other agents and make decisions based on its own mental models of other agents performing goal inference.

We will begin this section by first briefly describing how we score traces of probabilistic programs. Then we discuss details about path planning that will lead to the discussion of how we can create a simple model for goal inference. With that simple model, we will simulate agents applying goal inference on other agents, and then on one another. In summary, we will demonstrate agents playing a series of, what we call, "Schelling Coordination Games", which were inspired from the original experiments using nested inference.

We mainly use two inference algorithms in our experiments, Importance Sampling and Metropolis Hastings. Each of these inference algorithms require traces to be scored, or to be given a weight. For this reason we apply the scoring technique described in [41]. Basically, this method computes the log-density of a trace even if 'auxiliary' random variables are involved and cannot be exactly marginalized out. This especially becomes handy for goal inference, since it involves path planning. After some math manipulation, it turns out that this method scores only when traces are conditioned. If a trace is not conditioned, then the score does not exist for that trace. This is because the score ultimately becomes the the sum of the log scores of the elementary random primitives in the probabilistic program. Therefore, all conditioned traces seen in our experiments will be scored as such.

4.6.1 Test Environment

We begin experiments on the following map shown in Figure 8, with 6 main obstacles of various shapes and sizes. In the figure, we also show possible start and goal locations for agents in crossed green circle markers.

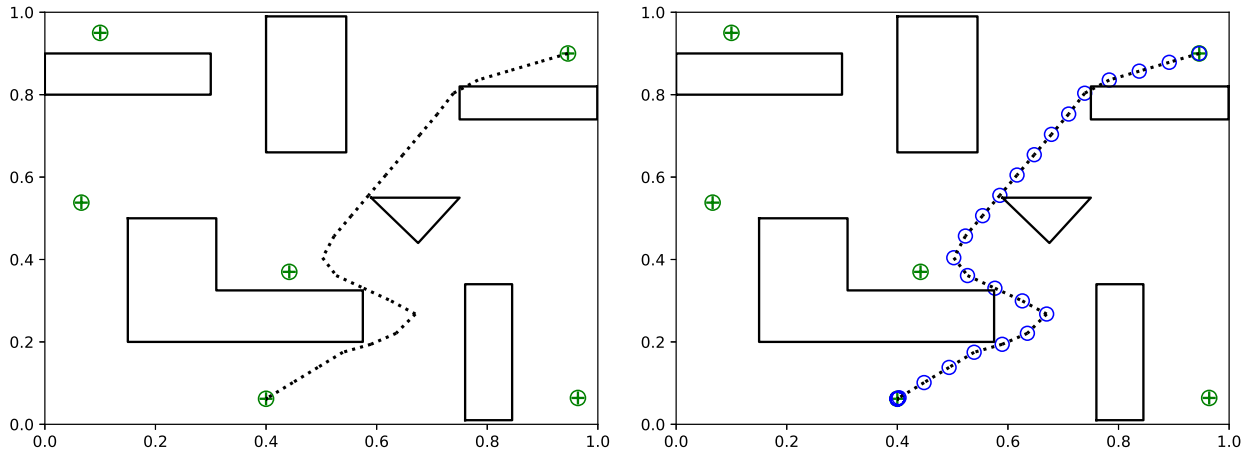

Figure 9: Left: We show an example of an agent's plan. Right: We show the time steps of how an agent traverses its plan in blue circles.

These locations serve as a bases of where agents will generally begin their planning and where they will end, i.e. reach their goal. These locations were chosen to cover different surrounding areas of obstacles on the map. In this map agents are allowed to travel from one of the set locations (seen in Figure 8) to another. However, the path from one location to another may vary. In Figure 9, we demonstrate what a path for an agent looks like. On the left side of the figure we see the overall path of an agent represented in a dotted black path. On the right we show particular time steps, represented in blue circles. Each of these time steps are observations from the path. This means that each of those times, we observe at which location along the path the agent will be. Each path has a set number of observations. If the agent reaches its discrete goal location before the end of the set number of time steps, the agent "hovers" on the goal location in each of the remaining time observations. Our paths are created using RRTs. This gives us the flexibility to have different paths from a particular start and goal location. As shown in previous work, the RRT tends to manifest itself in a very wavy, or noisy manner. Therefore, we optimize the path in such a way that we remove most of the noise. We keep some noise to create diversity between paths from the same respective start and goal locations. We demonstrate this in the next section. As well, we demonstrate in later examples how we can change the speed in which the agent travels along a path, but for now, the intervals in which are seen in Figure 9 are the intervals we will use for first basic map shown.

4.6.2 A Simple Planning Generative Model

Now that we have a path planner, we can move on to building a generative model which we can use to perform goal inference. The objective we want to accomplish when performing goal inference is to be able to infer which (goal) location an agent is traveling to when we observe its past movements. To be able to make those kinds of inferences we need to begin by building a model that simulates an agent traveling from one location to another. To build such a model, we will need to make decisions on which variables in the model should be random variables. Since we will need to condition on both the start and goal locations of an

Algorithm 2 Basic Planning Model

1: Given: map
2: Variables: β = last time step
3: t $\sim$ uniform$(1, \beta)$
4: start $\sim$ Categorical(possible start locations)
5: goal $\sim$ Categorical(possible goal locations)
6: plan = optim_RRT(start, goal) + noise

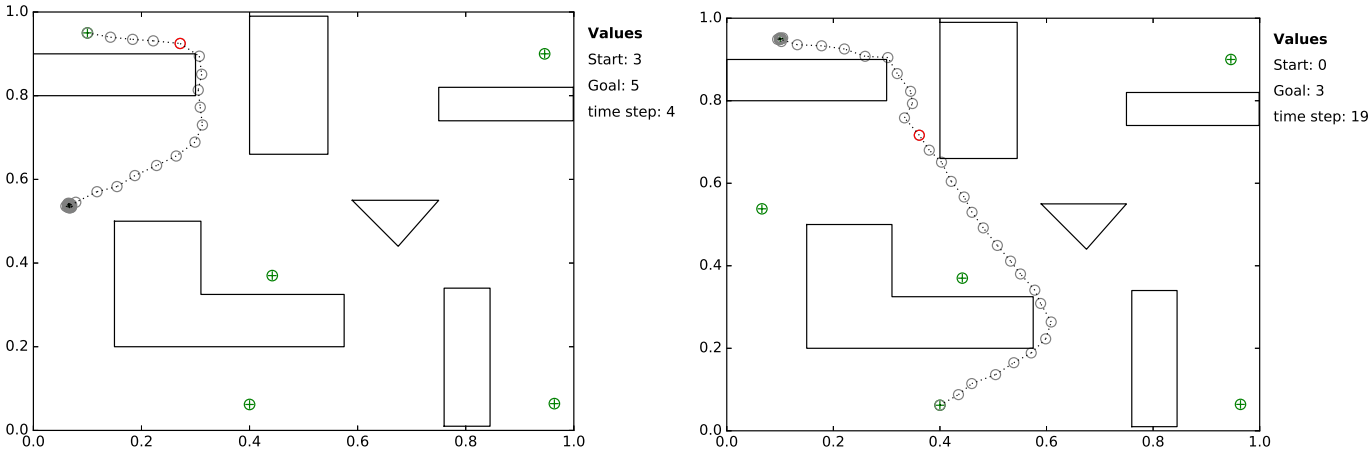

Figure 10: We show unconditioned forward traces of the basic planning model. The red circle on the path represents the time step i.e. the agent's location at that time step. On the left, we show a trace where the agents has planned path from locations 3 and 5, and its time step is at 4. The agent is traveling downward. On the right, we show another sampled trace from the prior where the agent is at time step 19 on a plan from location 0 to 3. The agent is traveling upward.

agent's plan, both the start and goal location need to be described with random variables. In addition, we will need to condition on past observations of the agent, therefore each time step will need to be a random variable. Algorithm 2 shows an example of how we can set up a probabilistic program for this scenario. The joint probability distribution of this model is $\mathrm{P}(\mathrm{start}, \mathrm{goal}, \mathrm{step}_1, \mathrm{step}_2, ...\mathrm{step}_\beta)$ where *start* represents the discrete starting location of the agent and the *goal* is the discrete goal location. Once the program has sampled the start and goal locations, we can use our path planner to get a coherent path with observed time steps. However to transform the latent variables of each time step into random variables, we must add noise to the original realizations. We do this using a normal distribution centered at the original realization at time *t*.

To test if this model behaves as desired, we naturally run unconditioned forward samples using the probabilistic program. These are traces from the prior distribution. Figure 10 shows two samples which the basic planning model generated. On the left, we see that the time step is at 4, and the agent is traveling downward the map. It starts at location 3 and plans to end at 5. On the right, the time step is at 19, and the agent is traveling upward the map. It starts at location 0 and ends at 3. We expect the samples to vary in realizations for all random variables. This means that samples from the prior should produce a variety of possible instances for the scenario. We can have paths from and to any of the possible goal locations and have those paths vary as

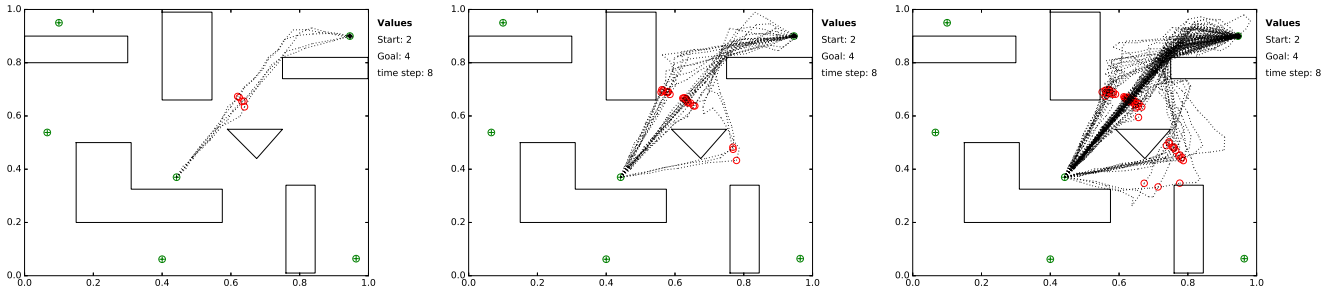

Figure 11: We show conditioned forward traces of the basic planning model. We condition the start, goal, and time random variables. The first figure shows 5 condition samples. The second shows 25, and the third shows 100 samples. We see how we get a distribution of paths when we do not optimize the path entirely.

well. In addition, we can generate instances at varying time steps.

We can go further to see what kind of results we get when we condition the start, goal, and time steps. The first plot in Figure 11 shows 5 samples where we condition the start variable to be 2, the goal to be 4, and the time step to be 8. The second shows 25 samples and the third shows 100 samples with the same conditions. We see from these figures how we are able to get a nice distribution of paths from one location to another. This justifies why we do not optimize our paths fully and why we will now refer to them as *semi-optimized*.

Now that we have seen that the generative model produces scenarios as desired, we can run inference on the model to determine goal locations based on past observations of agents. Since our model is not very complex, and we simply desire to infer the goal location, we can use simple Markov Chain Monte-Carlo (MCMC) inference algorithms. For comparison, we implement two inference algorithms, Importance Sampling (IS) and Metropolis Hastings (MH). Although both of these algorithms behave differently, we see that both produce similar posterior samples. Before allowing the agents to begin playing games, i.e. running simulations of agents moving and simultaneously performing goal inference, we set up a scene where we can test our inference algorithms. We begin these experiments by randomly choosing a consistent starting and goal location, 4 and 0 respectively. Then we created a random semi-optimized path and used that path to condition the the first 10 time steps. We run inference algorithms to estimate the $\mathrm{P}(\mathrm{goal}|\mathrm{start}, \mathrm{step}_1, \mathrm{step}_2, ..., \mathrm{step}_{10})$.

The first algorithm we tested with was Metropolis Hastings. We experimented by sampling 8 posterior samples using 32 particles. In the top-left plot in Figure 12, we see the results from running MH. The eight posterior samples are shown in red dashed lines, where again the blue circles indicated past observed locations. This shows that three of the samples show the goal location to be at location 2, four of the samples show the goal as location 0, and one sample as location 1. To further evaluate what kinds of samples we get from the posterior, we increase the number of samples and particles. On the right of that top-left plot, we show the results of running MH to sample 32 traces from the posterior using 64 particles. We conditioned on the same start and goal locations, however we simulated another random path to condition the first 10 time steps from. Based on the past observations, we see that we get similar outcomes for the probabilities of the goal locations. We also see that the choices of goal locations are rational according to past observations. We ran the same experiments using Importance Sampling. We conditioned the same random values, and generated new random paths form which to condition on. Again we see that the outcomes produce rational

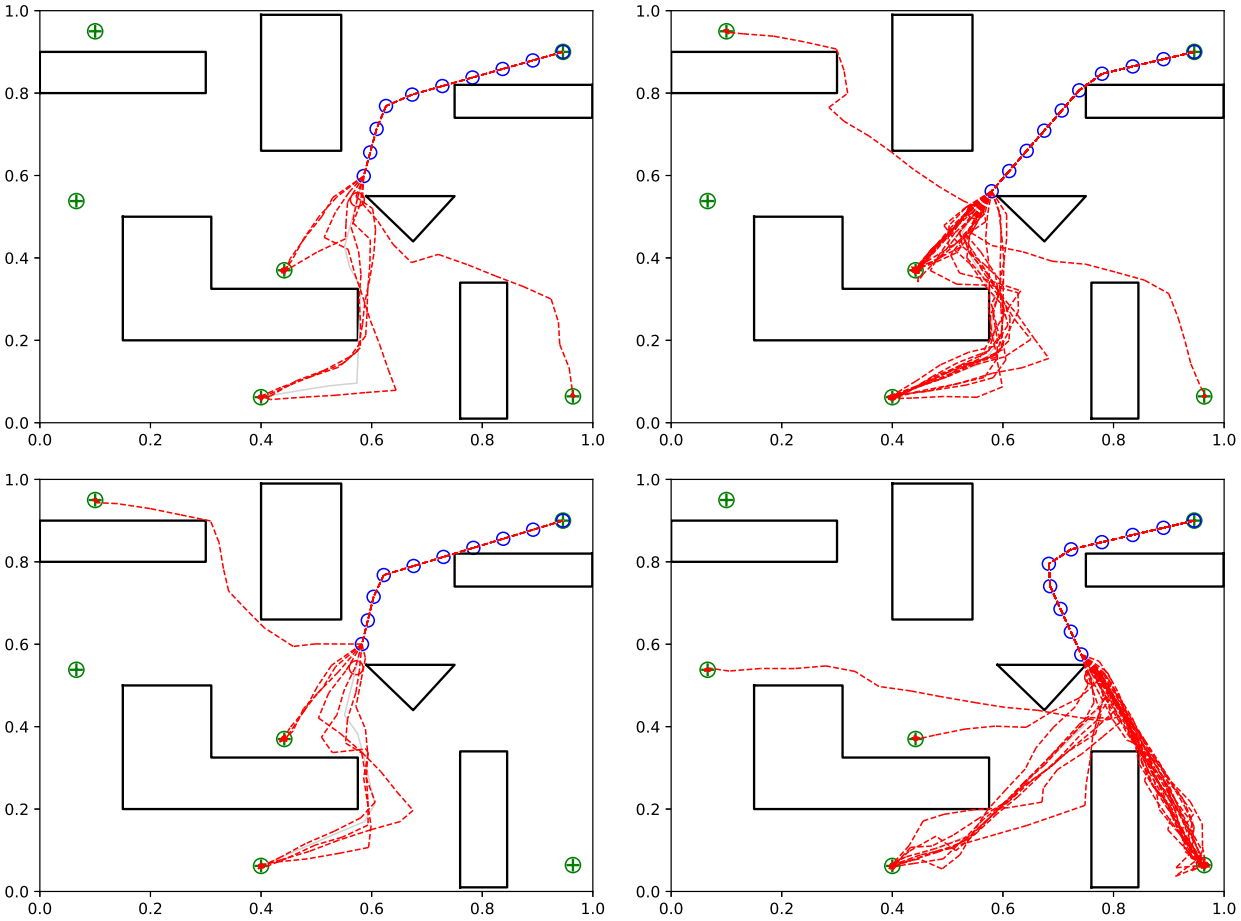

Figure 12: In each of these experiments we conditioned the start and goal location. Then we randomly generated a path between the two locations. Once we have the new random path, we conditioned the first 10 time steps. Top Left: Goal inference using Metropolis Hastings. 8 samples from the posterior using 32 particles with Metropolis Hastings. To its right, we increase the posterior samples to 32 and the number of particles to 64. Bottom Right: Goal inference using Importance Sampling. 8 samples from the posterior using 32 particles with Importance Sampling. Bottom Right: Goal inference using Importance Sampling. 32 samples from the posterior using 64 particles.

Algorithm 3 Basic Goal Inference Simulation

```
Given: map, start, planned-path
Variables: β = last_time_step
for t = 1 : β do
    for i : t = 10 do
        condition(planned_path[i])
    inferred_goals = run_inference()
```

goal locations based on the past time steps of agent movements.

We saw from setting up scenarios that our basic planning model can be used to perform goal inference when we observe the time steps of an agent. The next experiment we set up is to see how the inferred goals of an agent change as the agent moves. This means that we must perform goal inference at each new observations we gain from the agent. In this experiment we precomputed a random path from locations 4 to 1. Then we simulated the agent moving at each time step. At each time step we conditioned the model to include new observations. We expected that as we gained new observations the inferred goal would converge to a single location. We set up the described simulation in Algorithm 3. Figure 13 shows the posterior samples for the rest of the path and the goal for the agent at different time steps. The plot on the top-left shows the pre-planned path in light grey. The start location is where the blue circle is located, i.e. the agent's past observations. The goal location is at location 1, highlighted in red. To its left on the same figure, we see inferences for the goal location on times step 3. We see that every goal is a possible candidate. On the bottom-left, at time step 10, we see that number of goal locations decreased to two, and at time step 16, the goal is very obvious and our inference algorithm captures that. To show how the probabilities of each goal changes with more observations, we show Figure 14. The left plot shows the probabilities for a new simulation where the start location remains at 4 and the goal is at location 1. At the beginning of the plot we see how the probabilities for each goal remains low and quickly changes as time progresses. By time step 10, the goal with the highest probability is at location 1, with location 0 coming in second highest. However, as more observations are gathered, the probability of goal 1 continues to be higher. Next to that plot, we show another simulation where we change the goal location to now be at location 3. In this simulation, the goal does not become obvious until time step 14. We see from these experiments that agents are able to successfully perform goal inference on other agents. It is obvious as well that the agent infers more correct goals with more observations.

4.7 Fully Observable Collaboration Games

Now that we have demonstrated how we perform goal inference, we can add an element of collaboration to the generative models. By adding a second agent to the experiment, we can test how well an agent performs decision making when applying the goal inference on another agent. This section focuses on running simple collaborative games with two agents. The environment for these experiments will be performed on the same map as the previous goal inference experiments. Agents will remain in a fully-observable environment where they are aware of other agent locations even if they do not have a line of sight. In each of these experiments, we test to see how inferring the goals of the other agent influences their own movements. In all, we hope that the agents can predict the goal of the other agent so that they end up at the same goal before time runs out.

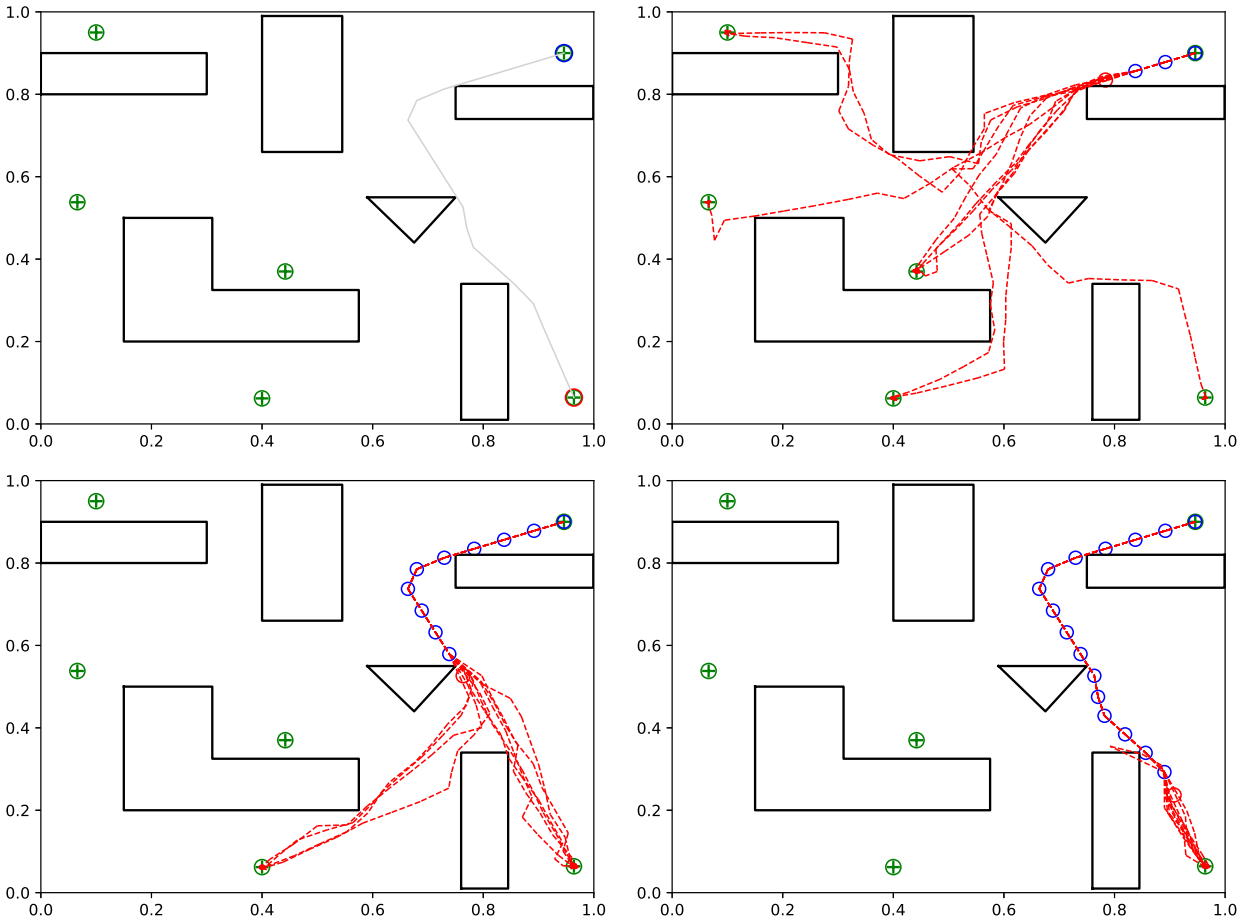

Figure 13: Top-Left: We show the pre-planned path of an agent from location 4 to 1. On the top-right, we show how all goal locations are possibilities based on current observations. On the bottom-left, we show how we converge to 2 possible goals, and on the bottom-right, we show how we converge to location 1 as the agent's final goal.

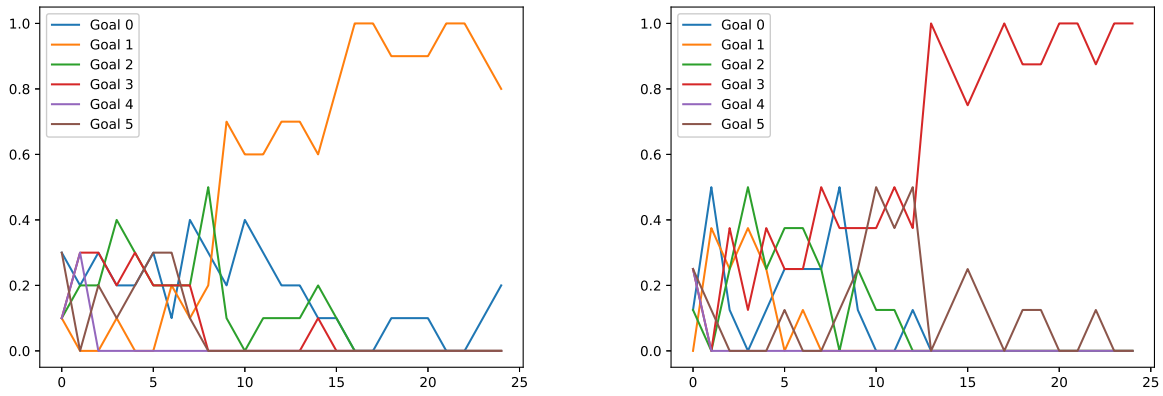

Figure 14: Left: We show the probabilities of each goal at each time step where the starting location was at 4 and the true goal is at location 1. On the right, we show the probabilities of each goal at each time step in a different goal location of 3. Each case reveals how the apparent goal location becomes more apparent with different amount of observations.

Since we are dealing with a fairly simple map, we expect the agents to successfully end at the same goal within the given time interval of 25 time steps. We test with two main games. The first we call *Follow the Leader* which we will discuss first, and the second we refer to as *Agent See, Agent Do*. The rest of this section will discuss the details of each game and their results.

4.7.1 Collaboration Experiment: Follow the Leader

In this experiment, we have two agents on the map. One agent, which we will refer to as *Alice*, preplans a path from a known location to another *unknown* goal location. The second agent, Bob, has a fully observable view of the map and of where Alice is, however Bob has no other information. Only of Alice's past movements as time goes on. Bob does not know what Alice's goal location is, nor the path in which she plans to take to get to her destination. Bob's objective in this experiment is to simply observe her movements, perform goal inference, and try to end at the same goal location as Alice. For obvious reasons, we refer to this experiment, or game, as *follow the leader*. We set up the simulation similar to Algorithm 3 and add the additional action of Bob moving towards the inferred goal location of Alice. Bob conditions the model in the same manner as before, on past observations. We determine whether the agents have successfully completed the game when Bob reaches the same goal location as Alice in 25 or less time steps. We begin experiments where agents begin at different locations. Figure 15 shows a typical example of how Bob behaves. The yellow circle in the figure represents Alice, who has a pre-planned path to an unknown goal. The blue circle represents Bob. Figure 15 shows the progression of both agents' movement through time. Specifically, we show the agents at time steps 1, 7, 9, 14, and 18. We chose these time steps to demonstrate the obvious decisions made by both agents. By time step 7, we see that Bob is confident that Alice's goal is toward the top of the map, which means he believes the goal is location 3. By time step 14, we see that Alice's goal is obviously at location 3,

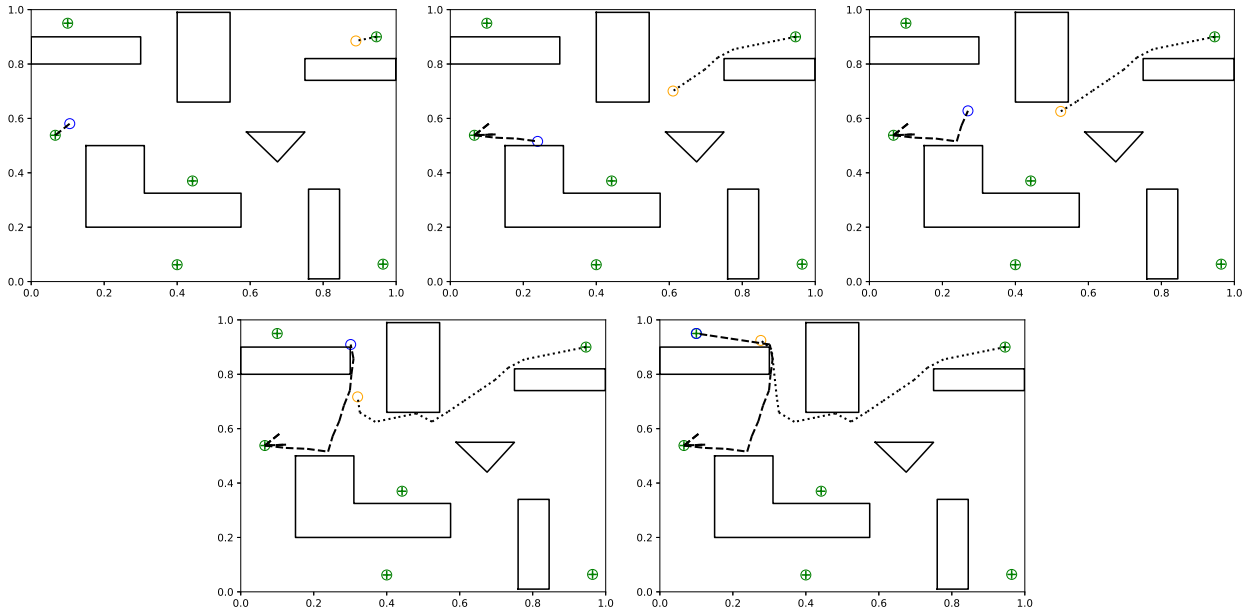

Figure 15: ***Follow the Leader* - One Agent Performs Goal Inference.** We show two agents playing *follow the leader* where one agent, Alice (the yellow circle), preplans a path to unknown goal. As time goes on, Alice travels this path to her goal. Bob, the second agent and the blue circle, observes her path at each time step and performs goal inference. He then takes a step toward the most probable goal Alice is heading to. The paths for each agents are shown for both agents as they travel at time steps 1, 7, 9, 14, and 18. At time step 9, Bob is convinced that Alice's goal is at location 3. The following time steps reveal that Bob's inferences were accurate and Alice then joins Bob at the same goal location.

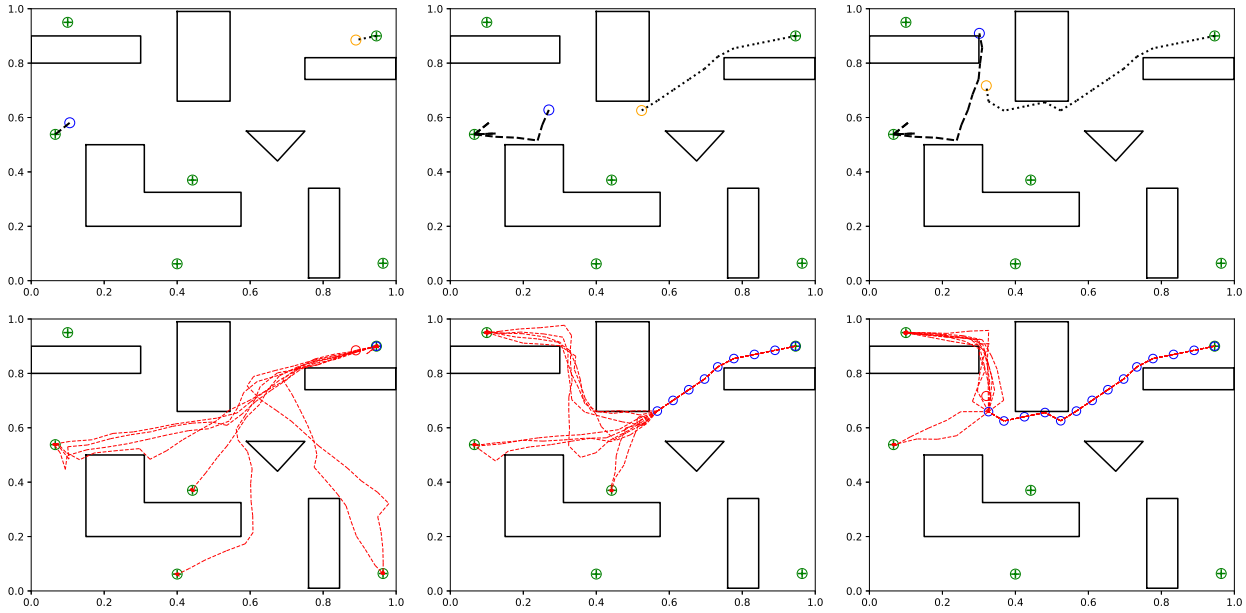

Figure 16: **Under the Hood during *Follow the Leader.*** Time steps 1, 9, and 14 of the same follow the leader game between the two agents, Alice and Bob. Below each time step, we show the goal inference "Bob" performs in the form of sample paths from the posterior to possible goals for Alice. We show that the number of possible goal locations decreases as Bob gathers more observations.

and at time step 18, Bob has reached the goal and waits for Alice to reach the goal. Although we show an example where Bob reaches the correct goal before Alice, not all simulations demonstrate that same effect. Sometimes Bob arrives either right before or after Alice does. We still consider this behavior successful since Bob had to infer Alice's goal successfully to even be within the proximity of the correct goal location before time ran out.

Although we have shown you the results of the typical *follow the leader* game, we now look "under the hood" of the inference algorithms just as we have shown before with the simple goal inference experiments. Figure 16 shows another example of the *follow the leader* game where both agents start in their same two locations, however Alice's pre-planned path is a bit different from before. We show the time steps of 1, 9, and 14 (from left to right) of the simulation. The top row of the figure show the agent movements just as before and right below each time step we show 8 posterior samples using 32 particles. We see once again that as Alice travels, her inferred goal locations narrow down to a single goal location. Bob chooses the goal location of Alice as the most probable path of all the posterior samples. We can see in time step 14, that location 3 is significantly more probable than any of the other goal locations. We note that these experiments were conducted with Importance Sampling. We could have used MH since they produce similar posterior samples, but we simply chose one.

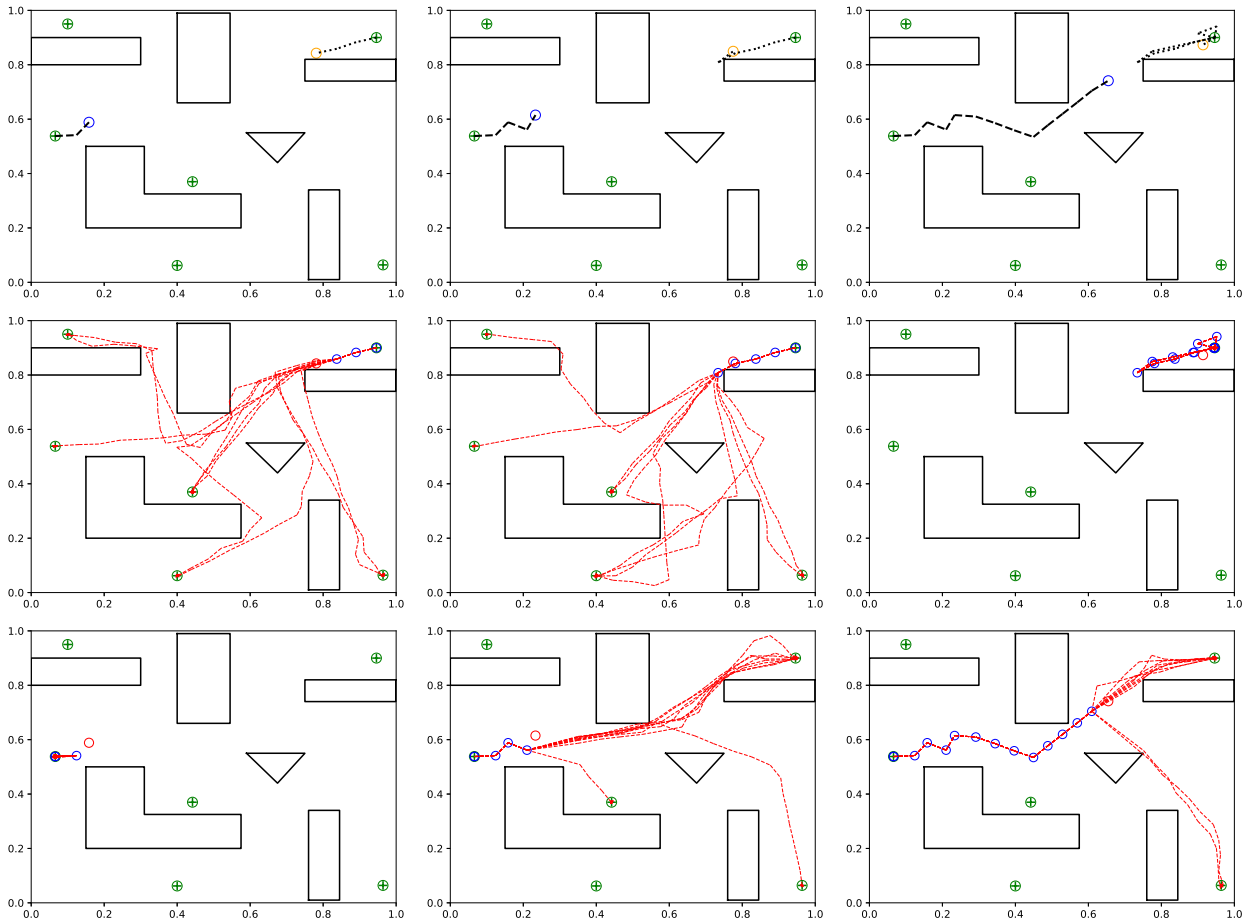

Figure 17: **Double Goal Inference while Traveling.** This is a typical example of how two agents behave when both are simultaneously performing goal inference one one another at each time step. We show time steps 3, 5, and 14 from left to right. The first row show the movements of both agents: Alice is the yellow circle and Bob is the blue circle. The second, or middle row represents samples from the posterior as Bob performs goal inference on Alice for time steps 3, 5, and 14. The last row shows the samples from the posterior that Alice gets when she performs goal inference on time steps 3, 5, and 14. By time step 3, we see how Alice returns to her original starting point. We show this example to demonstrate how simple goal inference often makes one of the agents return to their original starting point. Eventually both agents converge on the same goal location, however it is commonly one of the agent's starting location.

4.7.2 Collaboration Experiment: Agent See, Agent Do

This next experiment takes away any pre-planning for agents. This means that Alice now has to do the same as Bob, conduct goal inference to end at the same goal location as Bob. In this simulation, both agents conduct goal inference simultaneously on one another. Once again, the same basic planning model is used and the same random variables are conditioned. Movements of each agent are done with the intention to end up at the same place as the other agent. Using the same map as before, we place the agents in random locations on the map. However, for consistency, we show examples where the agents begin in the same start locations as the previous experiment of *follow the leader*. Figure 17 shows a typical simulation of the results of Alice and Bob playing this game. Each column represents a time step in the simulation, we show time steps 3, 5, and 14 from left to right. The top row shows the agents' movements in each time step. The second row shows the 8 posterior samples using 32 particles that Bob performs on Alice. The third row shows Alice running goal inference on Bob on each of those time steps. The behavior in these experiments are particularly interesting due to how often Alice and Bob manifested ending at either one of their own starting location. As the agents did inference on one another and moved toward each other, the past observations indicated to the other that the agent was already heading to that agent's location. Early on, the other agent is still near their own starting location. Here we see by time step 14, Alice has decided to turn around and go back to her own starting location so that she can end at the same goal location as Bob. In the end, the agents did well in the game and accomplished their objectives, but they were done in a rather unexpected manner. It was expected that the agents would end in other locations since observations should have indicated heading away from those starting locations. However, in our case, simple goal inference manifested itself differently when two agents were applying it to one another.

4.7.3 Agent See, Agent Do, with Theory of Mind

Now that we have two agents interacting with one another, we can further experiment by testing if and how implementations of theory of mind into the model can affect the basic behavior of the agents. In the previous section we have shown Alice and Bob playing *agent see, agent do*. We saw how their next movements and overall behavior were entirely dependent on the movements of the other agent. By implementing theory of mind, we hope that agents will manifest a slightly different behavior while attempting to end at the same goal location.

4.7.4 Describing Theory of Mind and Its Implementation

When we say that we desire to embed or have an autonomous agent simulate theory of mind, we mean that we want to give an agent the ability to create its own mental model of what the other agent might be modeling itself when trying to accomplish its own objective. This requires agents modeling other agents that model as well. We can imagine that our model, ie. our probabilistic program, will now become quite intricate. However this does not mean that we can not take advantage of previously designed basic models. In fact, since now our models must consider simulating other agent's models, we require layering of models, each layer representing an agent's perspective. In our case, we would first model the principle agent, Alice, who is then trying to model Bob modeling about her. So then, Alice's model, the outer model, would include a

nested model of Bob. Bob's model would model Alice in return. These basic models that we have already designed can serve as the inner, or nested model. A nested model will help simulate theory of mind and help the agent, Alice, understand the beliefs and intentions of the other agent, Bob, to reason about them. By designing a mental model of the mental model of the other agent, we hope that agents will have the ability to form their own beliefs of the other agent and make better decisions compared to non-theory-of-mind models. If the agent can "think" in the perspective of the other agent and take advantage of the basic knowledge of the other agent's desires, it would then be possible to find the probabilities of beliefs of the other agents. For example, if we continue to play our *agent see, agent do* game, and focus on a single agent performing inference, then we can describe what we desire to happen in a model in order to simulate theory of mind.

Overall, the goal is to find Bob's most probable goal *given* the map, Bob's start location, Alice's start location, Alice's previous time steps, and the observations of Bob's location. However, to get improved probabilities of goal locations for Bob, Alice must first use the information she already has to model what Bob would most likely do. Therefore, Alice uses the basic planning model to condition in the perspective of Bob. Alice's version of Bob desires to also locate Alice and end at the same location as she. Therefore, Alice conditions the basic planning model as if she were Bob. She conditioned the previous time steps with her own movements. Then she finds what her own most probable goal would be since Bob would do that to infer where she is most likely to go (this is assuming that Bob had previous observed all of Alice's movements, which he did not). Then using that information, Alice conditions her own mental model of Bob's goal location to be the same as hers.

In the end, we require two inferences to take place. In fact, we must run inference in a model that runs inference. Equation 2 shows how we condition the theory of mind model when running inference. In order to find the most probable goal that Bob will go to, we require an additional value, $goal_A$, which we find using nested inference. Equation 3 shows how we condition the inner model to find $goal_{inferred\_A}$, which we use as $goal_A$ in Equation 2. We show the *Theory of Mind Planning Model* in Algorithm 4 as a probabilistic program.

$$P_{max}(\mathrm{goal_B}|\mathrm{map, t, start_B, step_{B_{t-1}}, ..., step_{B_1}, goal_A = goal_{inferred\_A}, same\_goal = True}) \quad (2)$$

$$P_{max}(\mathrm{goal_{inferred\_A}}|\mathrm{map, t, start_A, step_{A_t}, step_{A_{t-1}}, ..., step_{A_1}}) \quad (3)$$

4.7.5 Results

Figure 18 shows three typical examples of the kind of behavior that is manifested by our theory of mind model. Each row represents an individual simulation. Each column represents time steps (from left to right) of the simulation's progression. In each of these simulations we conditioned the starting locations for the agents to be location 4 and 5, then we give each agent the ability to perform nested goal inference on one another with the objective to end at the same goal location. In the first column of Figure 18, we see time steps where either agents are within proximity of one another and have started to follow one another. The second column shows time steps where the agents have either met at the same location, but not at a goal location, or

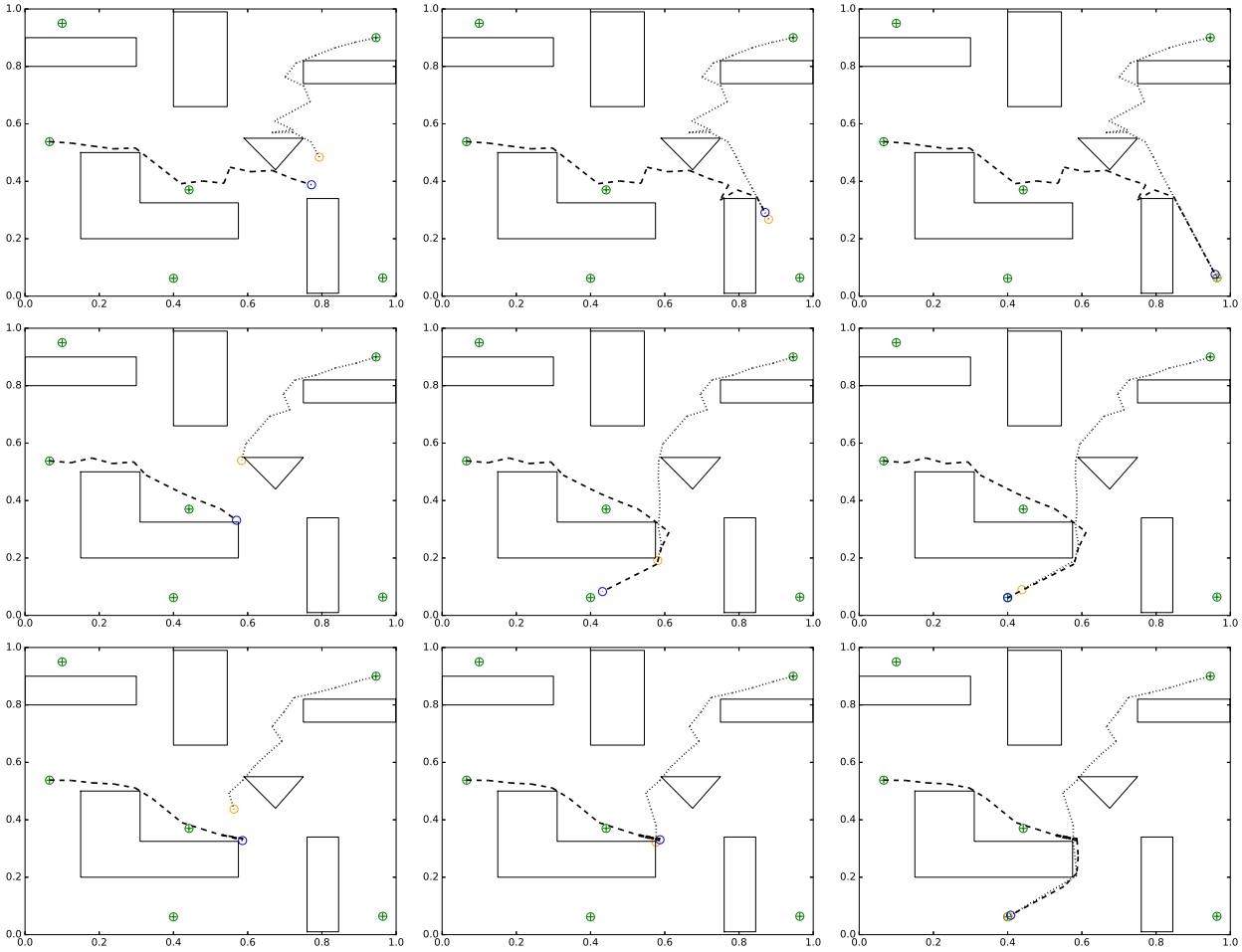

Figure 18: **Nested Goal Inference to Simulate Theory of Mind.** In each row, we show three time steps of a simulation of both agents, Alice and Bob, using nested goal inference on one another. These are typical results of how nested inference influences collaborative agents for both agents. The first column shows time steps where the agents starting getting close to one another, but still undecided of which goal each will end up at. The second column shows the time step where both agents meet up or start following one another to a specific and obvious goal. The last column shows the agents at the same goal location after traveling together for some time steps.

Algorithm 4 Theory of Mind Planning Model

Given: map, other_observations, my_past_observations
other_start $\sim$ Categorical(possible start locations)
other_goal $\sim$ Categorical(possible goal locations)
other_plan = optim_RRT(other_start, other_goal) + noise
my_most_probable_goal = run_inference(basic_planning_model | my_past_observations)
if other_goal == my_most_probable_goal **then**
    p = 0.999
else
    p = 0.001
same_goal $\sim$ Bernoulli(p)

have now officially begun following one another. The last column shows how the agents have started to move together towards the same goal.

This behavior is slightly different compared to when the agents were simply running goal inference on one another. We can even argue that these agents behave in a more collaborative manner since they first met at the same location and then, together, go to the same goal location in sync. However, we cannot disregard the value which goal inference provides since it becomes the nested model and inference method in the theory of mind planning model. Figure 19 shows further results from experiments using this model. We show two simulations (one in each row) of the agents running nested goal inference on one another. The first column shows the overall behavior of the simulation, however on the second column we see the goal inferences Alice made on Bob, and on the last column, we show the inference Bob made on Alice. Both the second and third columns show plots of probability for each goal location as time progresses. We see in both of these experiments that Alice figured out which goal Bob was going towards in an earlier time step than when Bob was performing inference on Alice. However, we see that both agents converge on the same goal locations in the end. The top row's simulation converges on location 2, and on the bottom row's they converge to 0.

4.8 Partially Observable Collaboration Games

Although goal inference is ideal to find the location goals of other agents, this method becomes rather less useful when trying to infer the location of another agent in a partially observable environment. As seen before, goal inference is most useful when you have more observations. However if the agents are placed on opposite ends of the map and do not have a line of sight with one another, then, in this setting, they are unaware of each other's current location. Therefore, basic goal inference is difficult to apply.

4.8.1 Collaboration Experiment: Run and Seek

4.8.2 Setup

The next scenario we move on to is a setting where agents, Alice and Bob, are placed in a more complex map and can only see, or observe, in a 45-degree field of view. In addition, we limit the agent's vision in

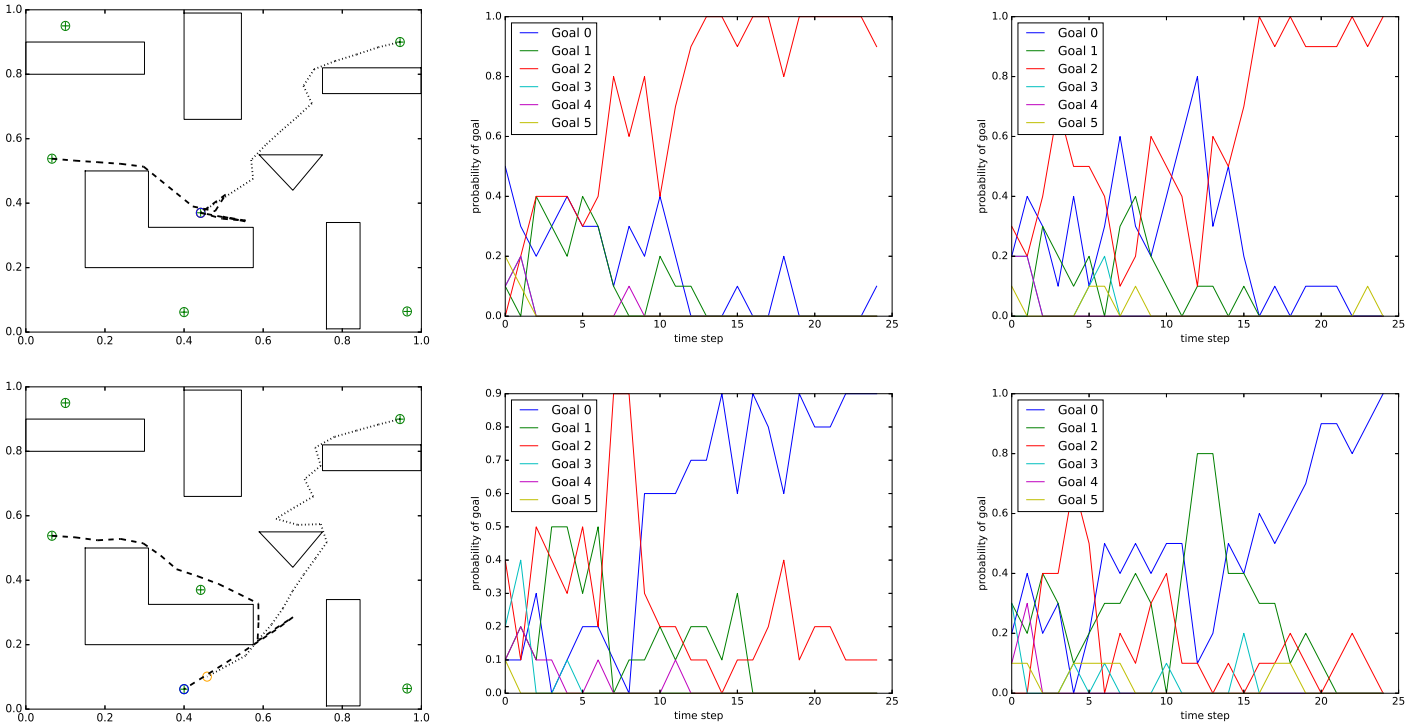

Figure 19: We show two simulations of nested goal inference (theory of mind): simulation 1 is on the top row, and simulation 2 is on the bottom row. On the left-most column we see the results of each simulation. The second column shows the goal inference Alice made on Bob, and the last column shows the goal inference Bob made on Alice. We can clearly see how both agents converged to the same goal location, although each simulation shows different goal locations.

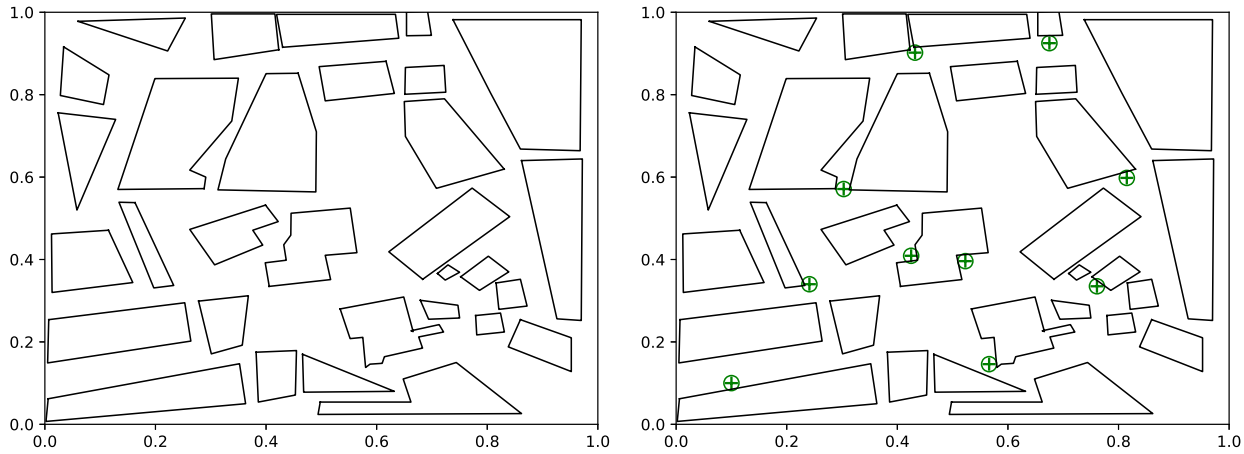

Figure 20: Left: the Bremen-Map layout with 31 obstacles of various shapes and sizes. Right: Marked in green circle-crosses, we show possible start and goal locations for the Bremen-Map.

the sense that now they cannot see through walls or too far ahead from their own respective locations on the map. Since each of the agents do not have any observations of the other agent, goal inference is as good as sampling from the prior. This means that as time continues, our inferences of possible goals for other agents do not converge to any small number of goals. Since we limit agent observations, yet still desire the agents to have the objective of locating one another, we must adjust our models. An agent cannot simply infer the goal of the other agent and go there, now it must consider planning paths that are most likely to increase the probability of detecting the other agent. As well, the agent can plan in such ways that increase the likelihood of the other agent detecting them first. Once the agents find each other, they can use those observations to try to end at the same goal location, or at least that is the behavior that we desire to manifest.

Since we are working with a scenario where the agents cannot initially see each other, we introduce a more complex map. Figure 20 shows the new map for the agents. It is composed of 31 obstacles of various shapes and sizes. On the right of the figure we show 10 possible start and goal locations for this map. These locations were chosen so that we can have the agents begin and end in all areas of the map, but mostly they were chosen with the interest of not making it easy for agents to see one another until they move out of those locations. We do note that agents are placed randomly on starting locations with the condition that they begin at least 0.4 (in a 0 to 1 scale) distance apart.

Since we are transitioning to another map, we also address that we update the speed in which the agent travels on a planned path. We set up paths in such a way that each time step represents a specific observed time, as discussed previously. When we slow the speed of an agent, we expect to see the locations of each time step along the path to be shorter in distance with one another. However as the speed increases, we expect the locations of the agent to become farther apart at each observed time step. Figure 21 demonstrates how the locations differ at each observation and as we adjust the speed. On the bottom row of the figure, we show the original path of an agent, each column showing a randomly generated path from the same start and goal locations. On the top row, we show each time step along the path, indicated in blue circles. The left-most plot shows the agent traveling the slowest along the path. On the right-most plot, we show the agent traveling the

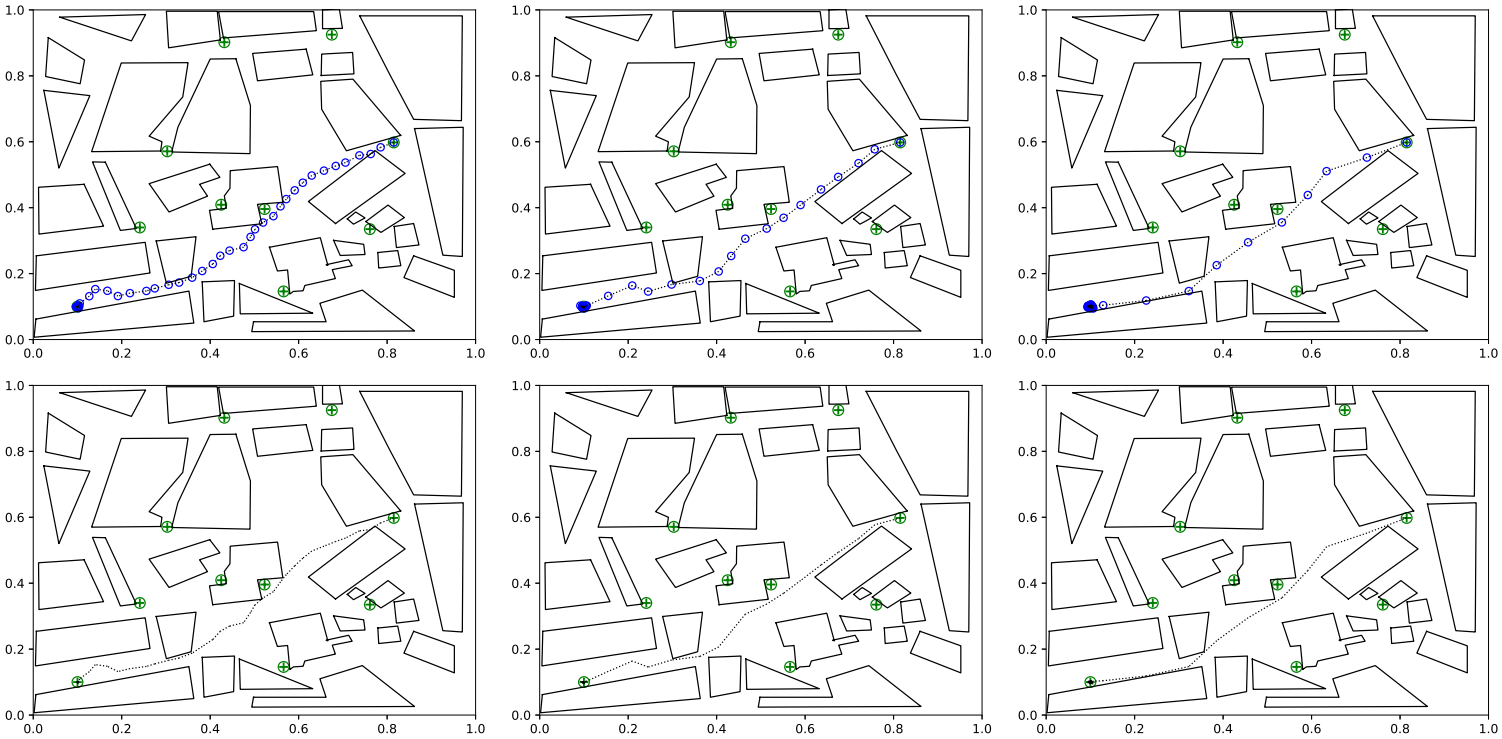

Figure 21: Here we show examples of how the speed can be adjusted once the path has been optimized. The first column shows an example of a very slow speed. The second column shows a faster speed, and the last column shows a very fast speed.

fastest along the path. The center plot shows a medium speed. For experiments, we choose a speed most similar to the center plot, as we figure it will decrease computation time and adequately demonstrate our models' behaviors. We also add that since agents require the means to "see", or detect another agent, we use Isovist calculations. Isovist is simply the technical terms for calculating the polygon representation on a map of an agent's field of view. If an agent is within the field of view of another agent, then that agent is considered to be located inside the Isovist. The field of view polygon will be seen in future plots where we show forward runs of our models computing Isovists.

4.8.3 Designing the Run and Seek Probabilistic Program

As mentioned before, now that we do not have all observations of the other agent, our task, which is to plan to the same location as the other agent, becomes vastly more complicated and difficult. We begin by discussing what variables become important in a problem such as this. The objective for the agents remains to end at the same location, therefore we must keep that objective as part of our newer model. In addition, this means we also need planning, therefore we keep our current path planner. The new element that our model must capture is detecting the other agent. As this is a partially observable environment, we need the model to keep track of when the other agent was actually detected. We can describe this in the following way:

$$\begin{aligned} &P_{max}(\mathrm{goal_A}|\mathrm{map, t, start_B, start_A, step_{A_1}, ..., step_{A_{t-1}}},\\ &\mathrm{detection_{t-1} = False, detection_{t-2} = False, ..., detection_{t=0} = False},\\ &\mathrm{detection_t = True, ..., detection_{t=\beta} = True, same\_goal = True}) \end{aligned} \tag{4}$$

Equation 4 shows that to find the goal with the maximum probability we need to condition given the map, a time step t, the previous locations of the agent itself, $start_A$, $step_{A_1}$, ..., $step_{A_{t-1}}$, and the starting location of the other agent, $start_B$. Now that we must consider finding the agent, paths must also explain the fact that the other agent was not detected previously, since it has not been found. Therefore we condition past detections to be false, $detection_{t-1} = False, detection_{t-2} = False, ..., detection_{t=0} = False$. The same logic applies with desiring future detections. Therefore we additionally condition future detections to be true, $detection_t = True, ..., detection_{t=\beta} = True$. Once the agents locate one another, they should use those observations to keep future detections true, and they must simultaneously travel to the same goal, $same\_goal = True$.

With this in mind, we can design our probabilistic program which we can use to run inference and approximate $P_{max}(goal_B)$. Algorithm 5 shows how we designed the *Run and Seek* model. We first begin by modeling the path of the other agent, lines 4 - 6. Then we model the agent's own plan, lines 7 - 9. Next, we check detections for each time and add some noise, lines 10 - 17. Lastly, we see if the plans had the same goal location.

Now that we have a model formed, we can see how a forward run of the *Run and Seek* model looks like. Figure 22 shows examples of unconditioned forward runs from this model. As expected, we should get a variety of scenarios from the model. Once again, Alice, or Agent A, is depicted as the yellow circle, and Bob, Agent B, is the blue circle. The red circles on the paths indicate time steps where Bob was detected, and the solid teal polygons describe the field of view (Isovist) which Alice calculated when she detected Bob.

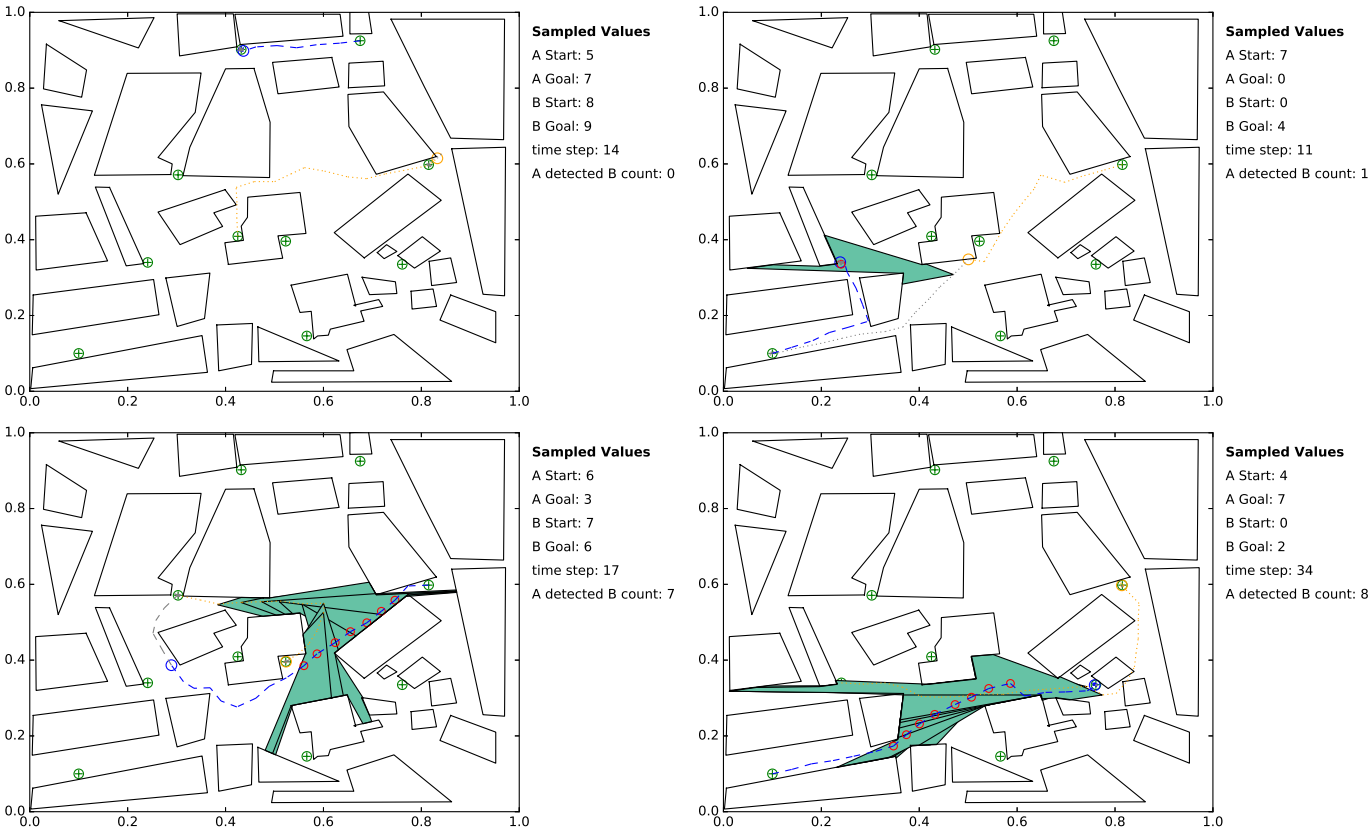

Figure 22: Samples from the unconditioned model *run and seek* Top Left: A forward run where agents planed and did not detect one another at any time from their start to goal location. Top Right: forward run where the Agent A's plan will detect Agent B once. Bottom Left: Agent A makes a plan that detected Agent B in his plan 7 times. Bottom Right: Agent A detects agent B 8 times. In all of these forward runs, we show the field of view that Agent A had at each time Agent B was detected. See text for more details.

Algorithm 5 Run and Seek Model

Given: map, detection observations, my past movements
Variables: $\beta = \text{last time step}$
t $\sim \text{uniform}(1, \beta)$
other_start $\sim \text{Categorical}(\text{possible start locations})$
other_goal $\sim \text{Categorical}(\text{possible goal locations})$
other_plan = $\text{optim\_RRT}(\text{other\_start}, \text{other\_goal}) + \text{noise}$
start $\sim \text{Categorical}(\text{possible start locations})$
goal $\sim \text{Categorical}(\text{possible goal locations})$
plan = $\text{optim\_RRT}(\text{start}, \text{goal}) + \text{noise}$
for $i = 1 : \beta$ **do**
    if isovist(plan[i], other_plan[i]) == True **then**
        p = 0.999
    else
        p = 0.001
    $\text{detected}_{t_i} \sim \text{Bernoulli}(p)$
if other_goal == goal **then**
    p = 0.999
else
    p = 0.001
same_goal $\sim \text{Bernoulli}(p)$

On the top left, we see that the agents planned, but Alice did not detect Bob at any of the time steps. To its right, we see a scenario where Alice makes a plan that detects Bob in a future time step. On the bottom-left, we see a scenario where Alice happened to make a plan where she detected Bob as they both, Alice and Bob, traveled along their paths. Lastly, on the bottom-right, we see another example where Alice detected Bob along his path (as she travels as well). All of these samples are plausible scenarios the agents could encounter. However, we must still see what kind of behavior is manifested by the generative model when we start conditioning on the random variables.

The following experiments will describe outcomes from the probabilistic program when we choose to condition several random variables. We note that these outcomes are scored as discussed briefly in the goal inference section. Figure 23 show four outcomes of the conditioned generative model. In each of these outcomes, a log-score is shown right below each plot. These scores are relative to one another, therefore, the better the score, the better the outcome, or trace, matched the conditioned values. In each of these instances, we conditioned the starting location for Alice and Bob to be at locations 2 and 9, respectively. As well, we conditioned $t = 5$, all past detections to be false, and all future detections to be true. We conditioned the $same\ goal$ random variable to be true, and the first 4 time steps for Bob. In each trace shown in Figure 23, we show how traces can be given higher scores than others. For example, if we compare the the top-left plot with the top-right, the plot of the right scores slightly better. This is because the future path of Bob describes his past time steps better than that of the left. Logically, a person would say that Bob is most likely to continue his movement towards the right of the map, than left. If we compare both of the top plots with that of the bottom-left, we see that they are much more reasonable traces than expressed in the bottom-left plot. It seems in that particular trace, that Bob jumps over a building and continues his path. On the bottom-right plot, we

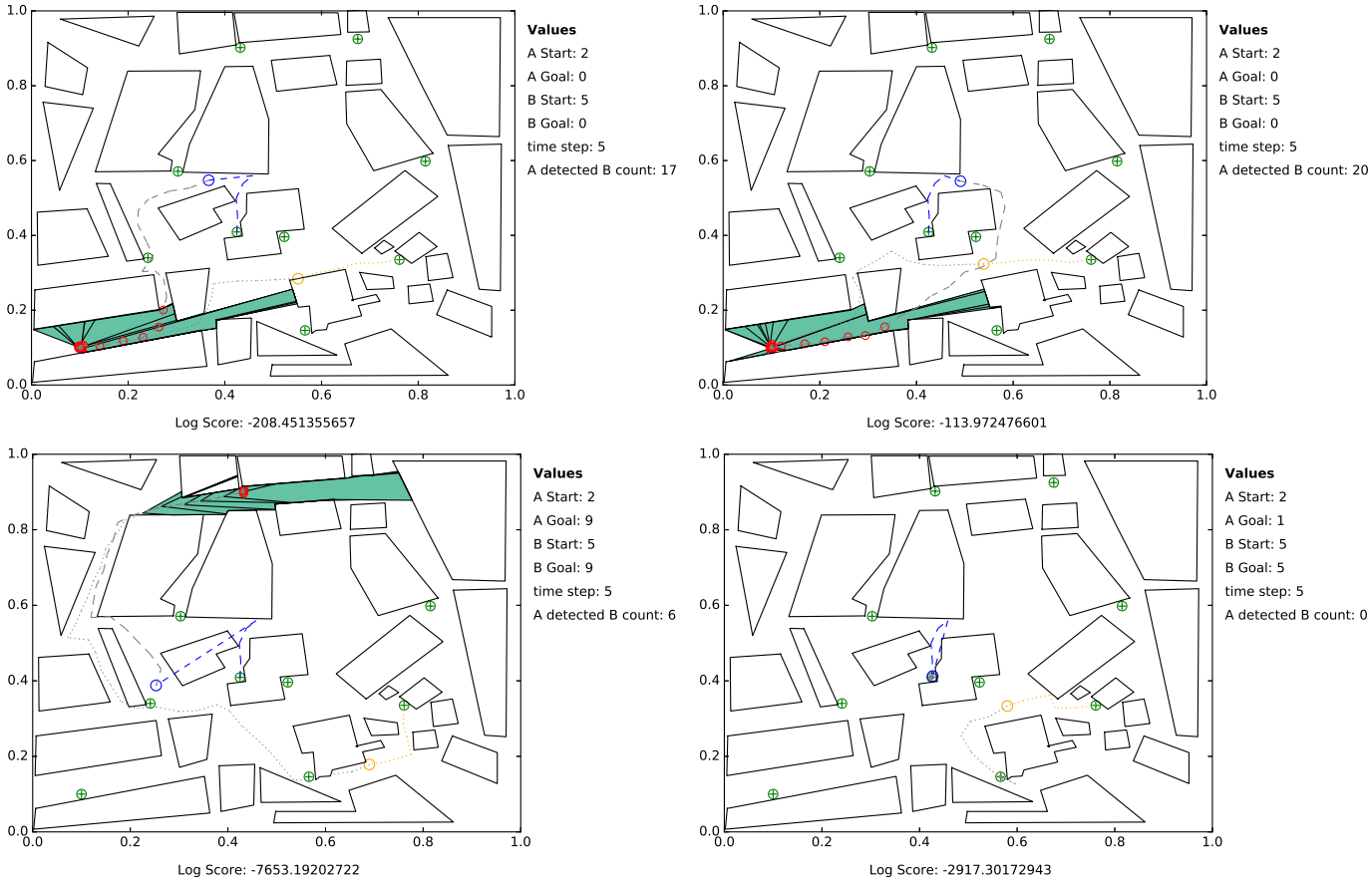

Figure 23: Samples from conditioning the *run and seek* model. In each of the forward runs shown, we show the log score computed given the conditioned values. It's important to note that we condition the first 5 time steps of Agent B, the starting location of Agent A, past detections to be False, and future detections to be True. These scores are relative: the higher the score, the better the forward run matched the conditioned variables. See text for more details.

see how the trace log-score is in-between the first three examples. This is a scenario that could occur, but it is unlikely that an agent would return to its goal after four steps. If its intended goal was to its starting location to begin with, it would have most likely stayed in that starting location. It also scores lower due to the lack of future detections of Bob.

In Figure 24 we show how the agents behave differently when we change the conditioning of future detections to be false instead of true. Notice how the trace with the best score is where the agents plan to different locations, and do so without Alice detecting Bob. As well, notice how Bob's future plan is coherent with his past time steps. On the top-right, we encounter another trace that scores well. In this case, Alice's start location is the same as her goal, which is perfectly fine if she plans to not have any future detections of Bob. On the bottom-right, Bob's future path is again a bit unusual, which is why its score becomes significantly lower. Lastly, on the bottom-right we show a trace that does not follow the conditioning well. There are several future detections of Bob, and Bob's future plan is again not coherent to his past time steps.

4.8.4 Results

After seeing what kinds of results we get after conditioning the model, we can now perform inference to sample from the posterior distribution. As mentioned before, we test our experiments with two inference algorithms, Importance Sampling and Metropolis Hastings. In each of these inference algorithms, we require scoring, or weighting a trace from the model. As well, it requires a known constant of particles to score. As a means to choose the number of particles needed to run inference and get higher log-scored traces, we show the typical trace outcomes with 16, 32, 64, and 124 particles. Figure 25 shows these results each with 5 samples from the posterior. In each of these examples, we conditioned the same first four time steps of Bob's path, the starting location for Alice, past detection to be false, and future detections to be true. On the top-left, we show 5 posterior samples after using 16 particles, on the top-right we show another 5 posterior samples with 32 particles. On the bottom row, we show samples with 64 and 124 particles, left to right, respectively. If we compare the future paths for Bob, we see that the most logical paths become most apparent when using 32 or more particles. We also show that we see more future detections of Bob after 32 particles, and conclude that when running more particles than that, results are slightly better. However, results are similar enough to justify our selection for the number of particles. We note that we used Importance Sampling when running inference in this experiment.

Using 32 particles, we now run inference where we condition the same variables as before. Except we show two difference scenarios: samples from the posterior when we condition future detections to be **false** and samples when we condition future detections to be **true**. Figure 26 shows posterior samples in each plot after running inference. Each shows 5 samples from the posterior when using 32 particles in Importance Sampling. Similar results were generated when using Metropolis Hastings. For all scenarios, we conditioned the starting locations for Agent A and B consistently. In the figure, the top row shows posterior samples where we conditioned future detections to be false. In this particular scenario, we expected Agent A to avoid detecting Agent B. We call this scenario *Run and Avoid*. The bottom row in this figure is the opposite: we conditioned future detections to be true. We refer to this scenario as *Run and Find*. Each red circle along a path indicates a *true* detection at that particular time step. We can easily see how the behavior changes when we condition differently. The top row shows Agent A heading West generally, and the bottom row shows Agent A heading North. When the agent plans to move towards the center of the map, we easily see that it's

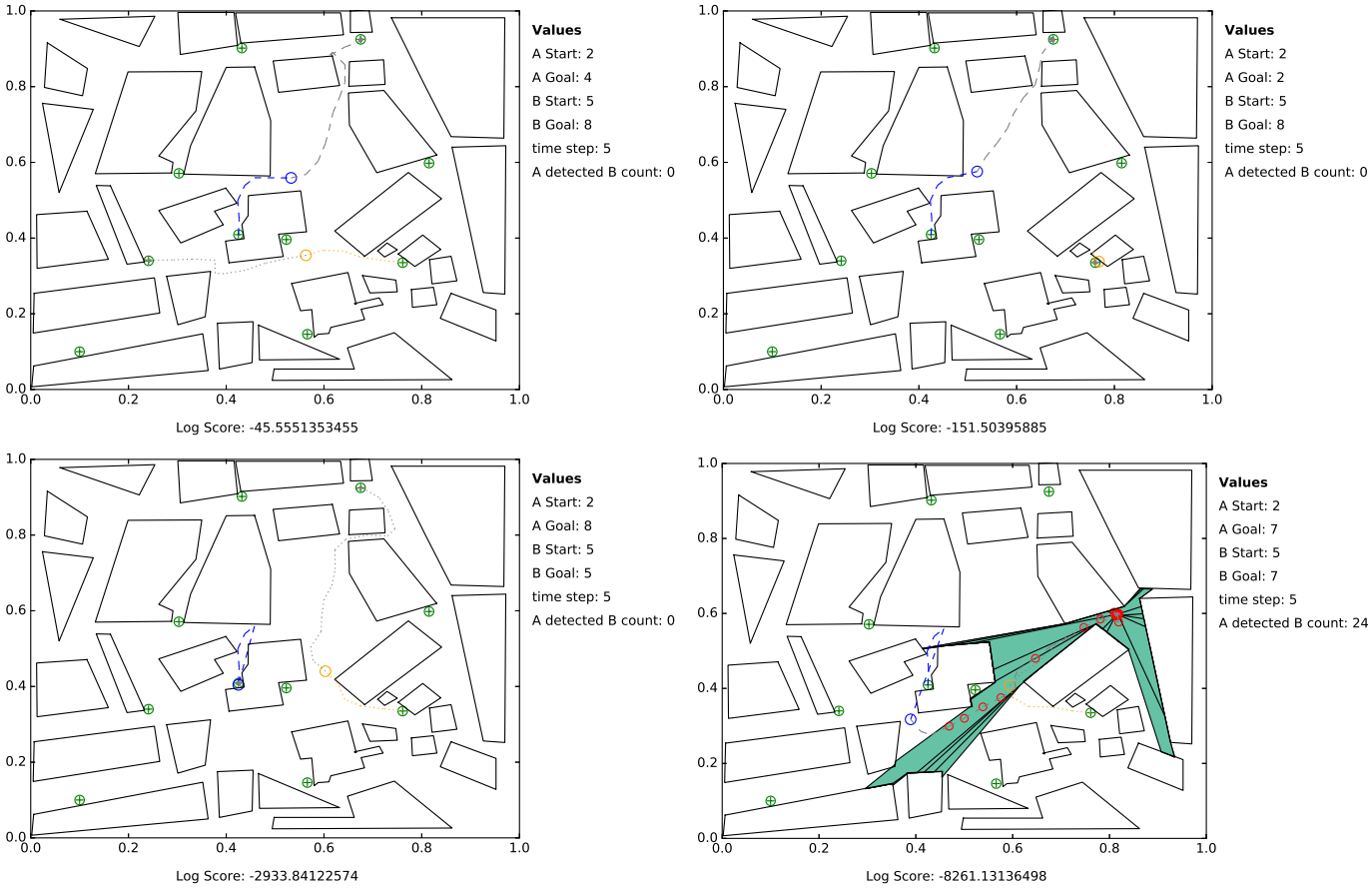

Figure 24: Samples from conditioning the *run and seek* model. In each of the forward runs shown, we show the log score computed given the conditioned values. We note that we keep similar conditioned values in Figure 23, but change future detections to be False. We demonstrate that changing what variables we condition, changes the behavior of the agent. We expect simulations to score higher where Agent B plans a rational future path and Agent A avoids detecting Agent B. See text for more details.

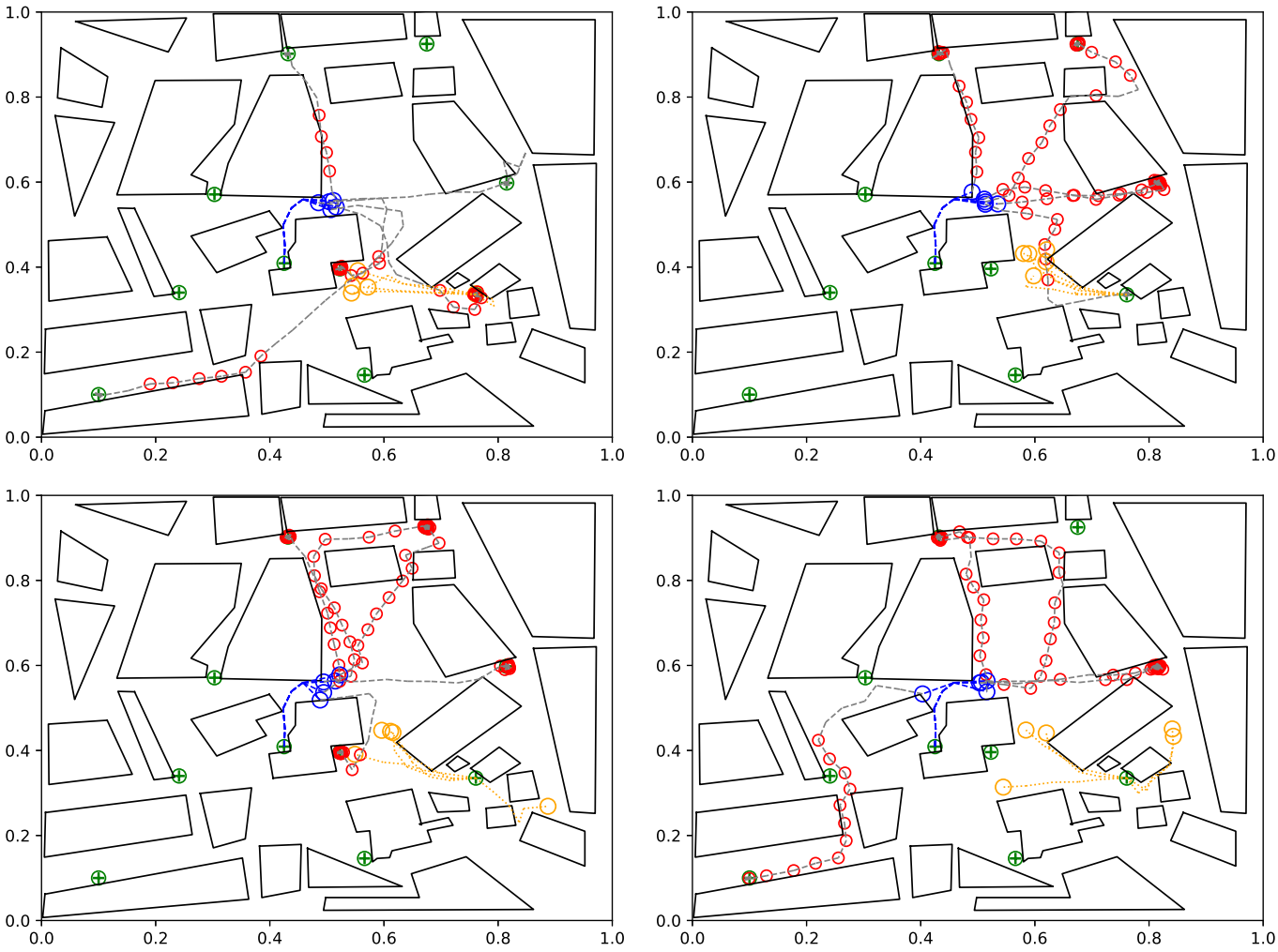

Figure 25: We show a comparison of running importance sampling with different number of particles. Each figure show 5 samples from the posterior. The top left figure show the posterior samples using 16 particles, the top right uses 32 particles, and the bottom left and right figures use 64 and 124 particles respectively. We see that we start getting better samples from the posterior when we use 32 or more particles in our inference algorithms.

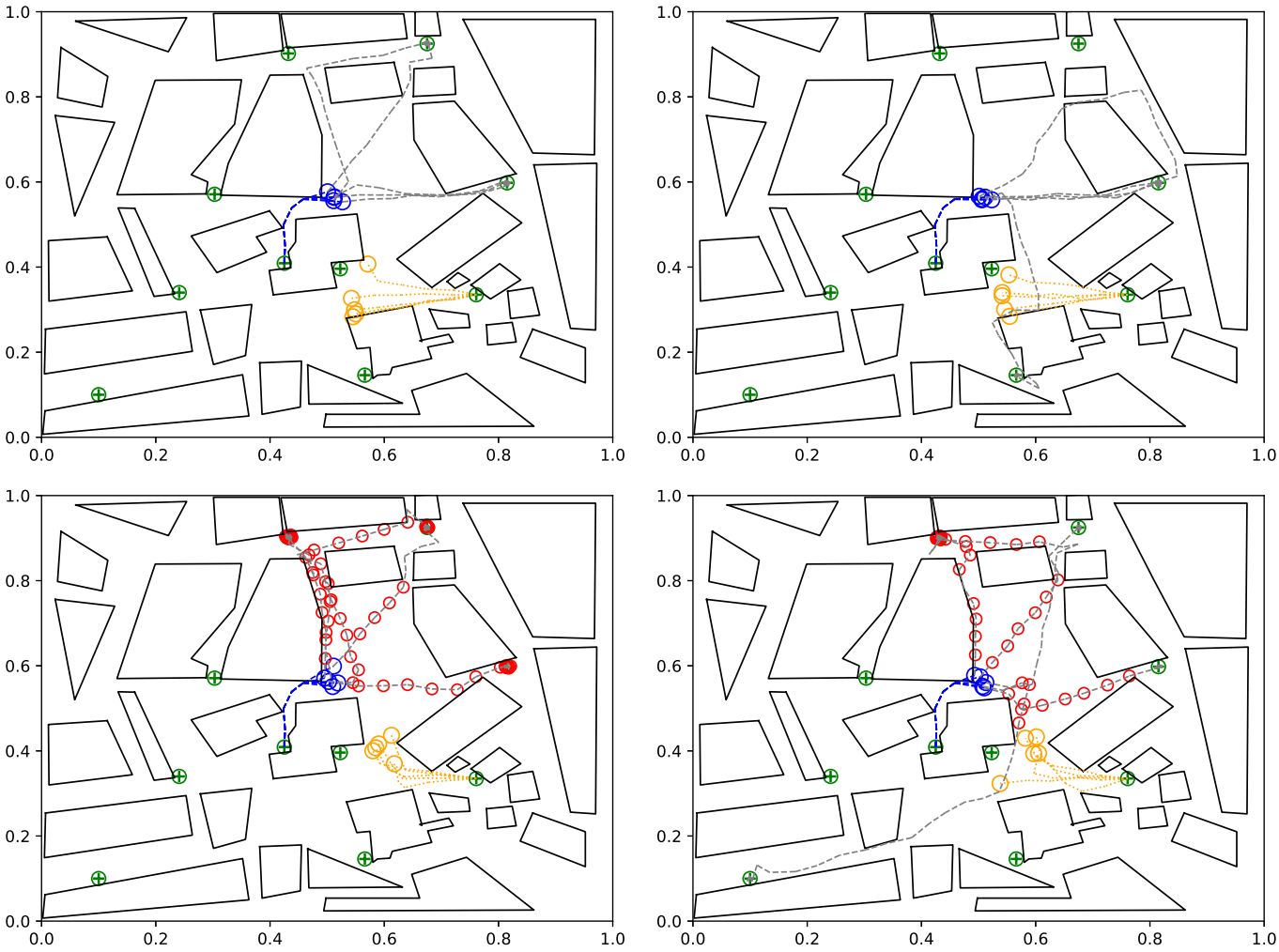

Figure 26: We show posterior samples in each figure after running inference. Each shows 5 samples from the posterior with 32 particles in Importance Sampling. Similar results were generated when using Metropolis Hastings instead. For all scenarios, we conditioned the starting locations for Agent A and B consistently. However, the top row shows posterior samples where we conditioned future detections to be False. We expected Agent A to avoid detecting Agent B. We call this scenario *Run and Avoid*. The bottom is the opposite. We conditioned future detections to be True. We refer to this scenario as *Run and Find*. Each read circles indicated a True detection at that particular time step. We can easily see how the behavior changes when we condition differently. The top row shows Agent A heading West generally. The bottom row shows Agent A heading North, towards the center of the Map to increase the probability of detecting Agent B. Note: The field of view polygon was purposely removed from the the figures to simplify the figure.

to increase the probability of detecting Agent B. (The field of view polygon was purposely removed from the the figures to simplify the figure.)

Since we already have run experiments with the *run and seek* models, we now allow agents A and B to use these models on one another to find each other. In these experiments we place both agents on random, but different start locations on the map. Both agents are aware of the other's starting location. However, that is the only location they have to find one another until they have a line of sight. At each time step they perform goal inference using the run and seek model and condition on where they are currently located on the map, their past observations (lack of detecting the other agent), and move towards the goal location that it currently believes the other agent is heading towards. As time progresses, we condition the past moves of the agent in the model, as well as their respective observations in those time steps. We ran simulations of the two agents running inference against one another to see if they could find one another with little information. At each time step the agent collects 5 samples from the posterior using 32 particles.

We learned during experimentation that when we conditioned the agent's own past observations after some time, the posterior samples were not helpful when trying to infer the agent's goal location. This was due to the agent's own past movements being noisy and not obviously moving in particular direction. The posterior samples were poor due to the path planner's inability to generate paths with such noise. Common past movements that created complications for the model involved paths circling in a particular area for some time, and paths that headed towards a direction and then changed spontaneously to another, creating a way-point along the path. Any time the agent created obvious way-points with its movements, the true path became one that could not be easily described by our generative model. For this reason, instead of conditioning on all past movements for the agent, we only condition the starting location and the previous time step. In a way, this disregarded unhelpful past movements, and simply left past observations to be sampled from the prior.

Results for these experiments depended on where the agents were randomly placed on the map for their starting locations. It was fairly common for agents to converge on goal location within 25 time steps if they were placed on nearer starting locations. It was also common for agents to find one another when placed on farther starting locations, but in cases such as these, they failed to reach a common goal location in time. Figure 27 shows four examples of when we placed the agent on nearer starting locations. On the top left, we show an example where both Alice and Bob ended at the same goal location (one that was center to both of their starting locations). On the top right, we show an example where Alice and Bob found one another, but were still trying to end at the same goal location when time ran out. The two examples on the bottom row are cases where the agents converged successfully on other same goal locations. Figure 28 shows four examples of when we placed Alice and Bob on farther starting locations. Three out of four of these plots are examples of when both Alice and Bob would find one another, but had still not converged to a goal location before 25 time steps. On the bottom-right example, we show Alice and Bob still looking for one another when time ran out. Both of these are typical examples of the kind of results our models would produce in the *run and seek* game. It was rare for agents to end on the same goal location when they were placed farther apart.

4.8.5 Run and Seek with Theory of Mind

Although the agents were able to succeed in some of the *run and seek* games, we decided to see how well a model with the same objective would behave when we embedded theory of mind. This, of course, would also mean that we would increase the complexity of the model thus increasing computation time during inference.

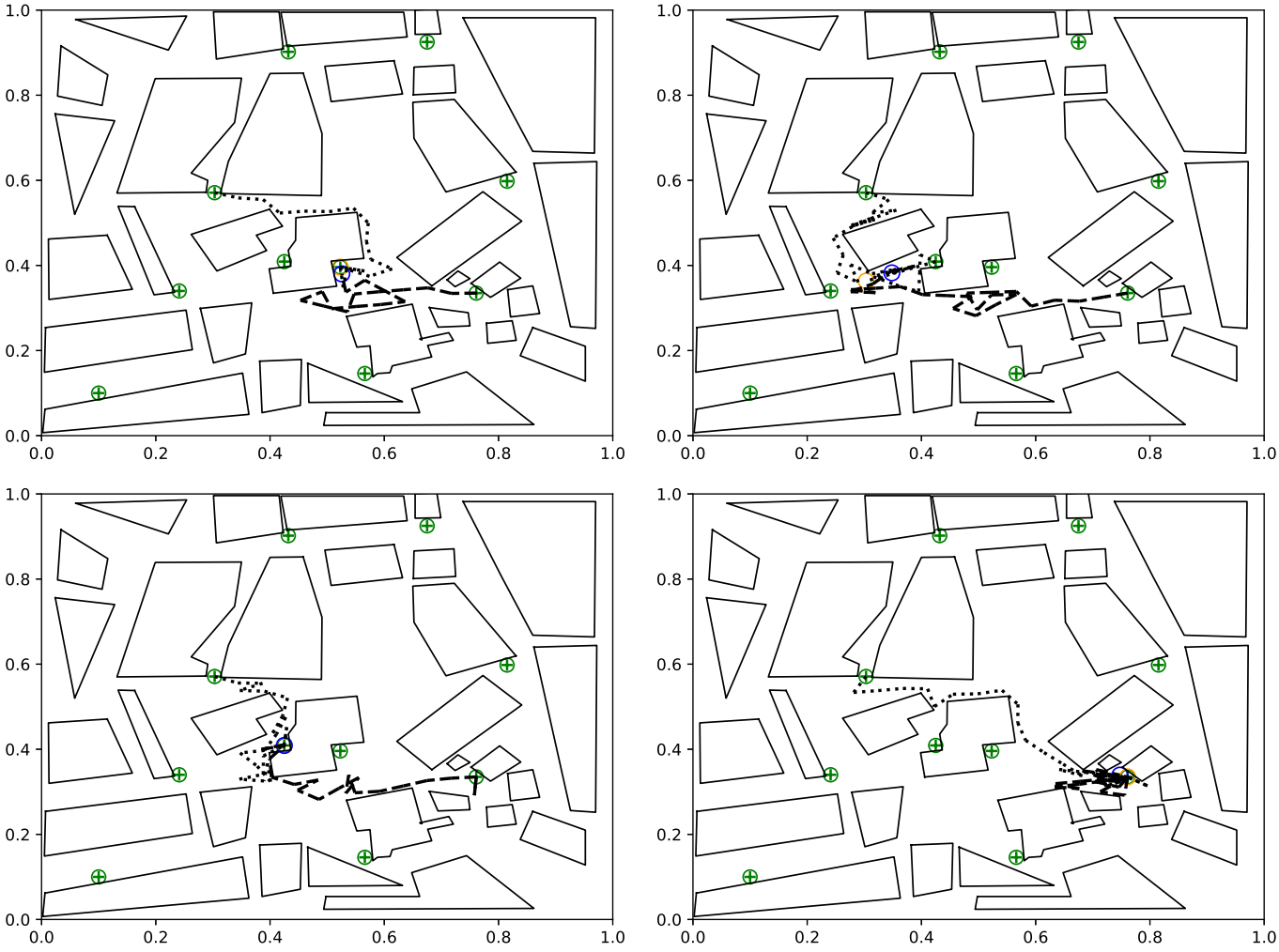

Figure 27: Four examples of Bob and Alice in a partially observable environment with objectives to find one another. In these examples we show Alice and Bob starting in nearer starting location. They apply the *run and seek* model and perform inference to search for one another. Three of the four cases show the agents converging on the same goal location. The plot on the top-right shows an example of when Alice and Bob found one another, but were still attempting to end on the same goal location when time ran out. See text for more details.

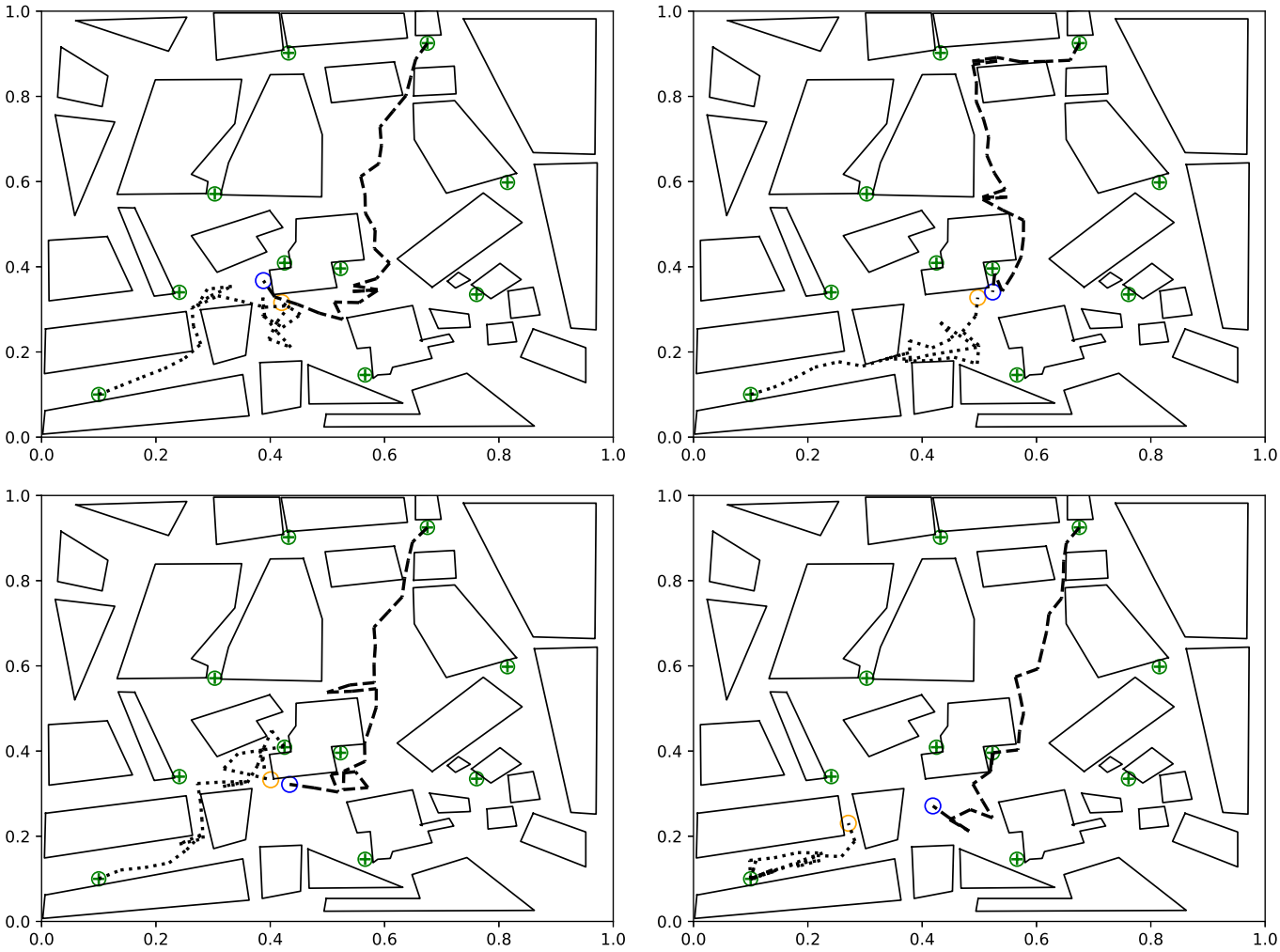

Figure 28: Typical examples of agents in a partially observable environment playing the *run and seek* game when the agents were placed in far starting locations. Three out of four examples show Alice and Bob near one another, but still not at the same goal location. On the last example, the bottom-right, we show an example where Alice and Bob have still not found one another. See text for more details.

Even though embedding theory of mind into our already complex model would mean increase of computation time, we would allow the agents to form their own beliefs of what the other agent is planning to do. Before designing a model that can simulate theory of mind, we must consider the objective of the other agent and how it's trying to accomplish it. By figuring out these details, we can best design a model for Alice to have of Bob's mental model. Since Alice knows that Bob also desires to end at the same location as Alice, she must use a model that can manifest that kind of behavior. Since we designed such a model in the previous section, the basic *run and seek* model will represent what Alice thinks Bob is thinking, which will then allow her to form her own beliefs of Bob's objectives and future behavior.

Since the objective is to find and end at the same goal, we still want the agents to basically calculate which goal the other agent is most likely to go so that agent can end there as well. We can do this in a nested manner. Just like in our *run and seek* game, we need to simulate both agents. As well, we will need to describe the nested model as the other agent's mental model of what's happening in the world. Therefore, to find the goal that the other agent, Bob, is most likely to go to, Alice must condition on the map, t, the start locations of both agents, past detections as false, future detections as true, and the inferred goal of the nested agent, Bob, which he *believes* Alice will end at.

$$\begin{aligned} &P_{max}(\mathrm{goal_A}|\mathrm{map, t, start_B, start_A},\\ &\mathrm{detection_{t-1} = False, detection_{t-2} = False, ..., detection_{t=0} = False},\\ &\mathrm{detection_t = True, ..., detection_{t=\beta} = True, same\_goal = True}) \end{aligned} \tag{5}$$

$$\begin{aligned} &P_{max}(\mathrm{goal_{inferred\_A}}|\mathrm{map, t, start_A, step_{A_t}, step_{A_{t-1}}, ..., step_{A_1}, start_B},\\ &\mathrm{detection_{t-1} = False, detection_{t-2} = False, ..., detection_{t=0} = False},\\ &\mathrm{detection_t = True, ..., detection_{t=\beta} = True, same\_goal = True}) \end{aligned} \tag{6}$$

Algorithm 5 shows how we find the goal that Alice should travel to; the goal with the max probability conditioned on future detections and the same goal as Bob. The $same\_goal$ variable is dependent on the latent variable of Bob's inferred goal of Alice, which is found in Equation 6, which is conditioned on values generated, or additionally conditioned on the outer model. Algorithm 6 shows how we design the model based on the these equations. Notice how the basic *run and seek* model and the *theory of mind (TOM) run and seek* model are very similar. Instead of modeling the other agents, the model in Algorithm 6 replaces the generation of the other agent's plan by first running inference on the nested model, which is in fact the *run and seek* model in Algorithm 5.

Just as before, we test the model by running a conditioned scenario. We set up a scene where we condition with starting location of Alice and Bob as 1 and 8, respectively. The time step is conditioned to $t = 8$, and we condition a single location for both agents. This location is set up so it's considered as the first observed location. Therefore, we condition the previous time step with detection equal to true, along with all previous detections as false, and all future detections as true. Figure 29 shows two examples of a single posterior sample of the *TOM run and seek* model, with 3 nested posterior samples using the basic nested *run and seek* model. This sample from the posterior is from using Importance Sampling on both layers with 32 particles. These examples are typical of a posterior sample. Again, the yellow circle represents Alice, and the blue

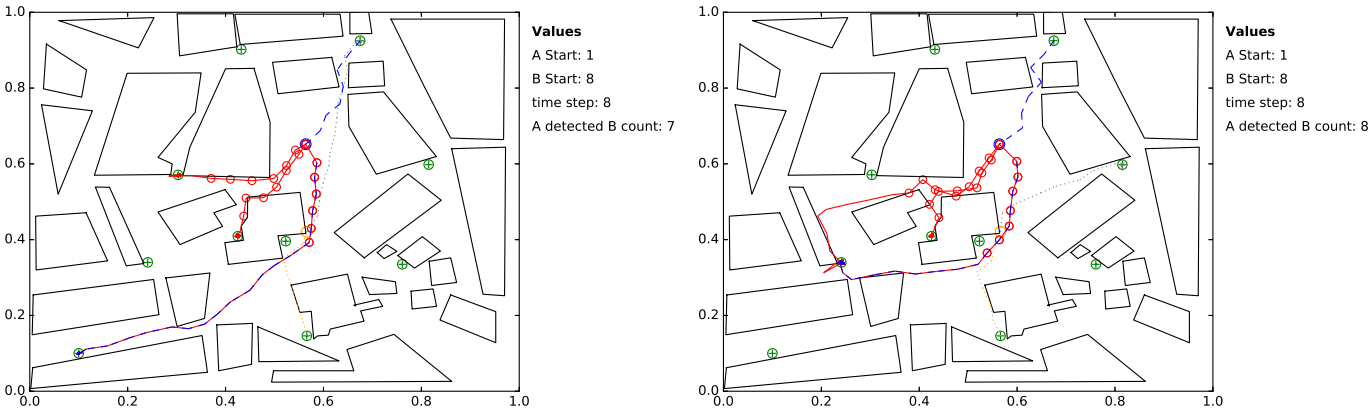

Figure 29: Two examples of a single posterior sample of the *TOM run and seek* model, with 3 nested posterior samples using the basic *run and seek* model. This sample from the posterior is from using Importance Sampling on both layers with 32 particles. These examples are typical of a posterior sample. Again, the yellow circle represent Alice, and the blue represents Bob. The red circles represent detections at those particular time steps. The three red paths represent posterior samples of Bob's future plans. Out of those 3 samples, 1 was chosen using the most likely goal location from the posterior samples. This path is highlighted with a combination of blue and red. In each of these examples, Bob has coherent future paths, and so does Alice. Although neither examples shows the agents traveling to the same goal location, they do represent future plans to remain (with the highest detection rate) in sight of the other agent.

Algorithm 6 Theory of Mind (TOM) Run and Seek Model

Given: map, detection observations, my past movements
Variables: $\beta = \text{last time step}$
t $\sim \text{uniform}(1, \beta)$
start $\sim \text{Categorical}(\text{possible start locations})$
goal $\sim \text{Categorical}(\text{possible goal locations})$
plan = $\text{optim\_RRT}(\text{start}, \text{goal}) + \text{noise}$
$\text{other\_plan}, \text{other\_goal} = \text{run\_inference}(\text{basic\_run\_and\_seek\_model}|\text{plan})$
for $i = 1 : \beta$ **do**
    if isovist(plan[i], other_plan[i]) == True **then**
        p = 0.999
    else
        p = 0.001
    $\text{detected}_{t_i} \sim \text{Bernoulli}(p)$
if other_goal == goal **then**
    p = 0.999
else
    p = 0.001
same_goal $\sim \text{Bernoulli}(p)$

represents Bob. The red circles represent detections of Bob at those particular time steps. The three red paths represent the *nested* posterior samples of Bob's future plans. Out of those 3 samples, 1 was chosen using the most likely goal location from the posterior samples. This path is highlighted with a combination of blue and red. In each of these examples, Bob has coherent future paths, and so does Alice. Although neither examples shows the agents traveling to the same goal location, they do represent future plans that allow the agents to remain (with the highest detection rate) in sight of the other agent.

4.8.6 Theory of Mind Nested Inference

In these experiments, we decided to use the same number of posterior samples and particles in both layers of inference as in the previous theory of mind implementation. This means in each layer we sample 5 samples from the posterior using 32 particles. Since we are running a total of 25,600 forward runs of outer and nested models, and each agent in our simulation is running nested inference, about two time steps progress per day. Unfortunately this means that experiments are still running for this section.

4.9 Conclusion and Future Work

In reference to our *big picture*, we are working towards creating generalized generative models that can simulate the behavior necessary to make informed decisions with little information. Goal inference, as demonstrated, is essential when inferring the movements of other agents. We also see how embedding theory of mind into models affect the behavior of agents, even ever so slightly. We saw how keeping the environment

fully observable allowed the agents to make informed and progressive decisions that lead them to doing well in the collaborative games. Once we transitioned the setting into a partially observable environment, the tasks became much more difficult and the agents behaved differently.

There are several factors to consider now that we have experimented with the models we designed in our partially observable games. When we are working with agents that are searching for other agents, our path planners need to be able to explain the behavior of searching. This means that we need to have path planners that explain way points, and idling. However, if agents are not searching for one another, these models are appropriate, and they are better at explaining movements that travel towards a direct goal in mind.

We also know that computation time is an issue when running nested inference on complex models. As we make these models more realistic, we need to take into account that we would like these models to be used in real time. Therefore future work consists on decreasing the computation time.

Lastly, once we can have these agents work collaboratively, we will introduce a third opposing agent, Chuck. Chuck's objective will be simple, it will be to detect one of the two agents, Alice and Bob, before they reach their goal. This particular situation increases the complexity of Alice and Bob's models. Not only will they need to do inference to end at the same goal, but must do so without getting detected by this third agent. Therefore, we will be introduced to designing models that consists of more than one theory of mind implementation.

In retrospect, we see that we are able to design agents that can simulate collaborative behavior. After seeing what kind of results our nested inference experiments for the partially observable environment produces, we will be able to analyze if nested inference really makes a difference in behavior.

4.10 Holodeck

The Holodeck simulator is a high-quality, visually rich simulator designed to be used in a variety of machine learning contexts. See Figure 30 for some examples of the kinds of domains it is capable of modeling. Its features include:

- A programmatic interface to python.
- A variety of agents (spheres, cars, boats, and humanoid robots)
- A variety of environments (deserts, oceans, cities, farms)
- Programmable control of the time-of-day, weather, and lighting
- A variety of tasks
- Support for multi-agent tasks
- Support for containerized, headless operation in GPU-accelerated environments.

In addition to spurring research in our own lab, Holodeck has also been used by other labs here on BYU campus. For example, the BYU Multiple AGent Intelligent Coordination and Control (MAGICC) Lab is

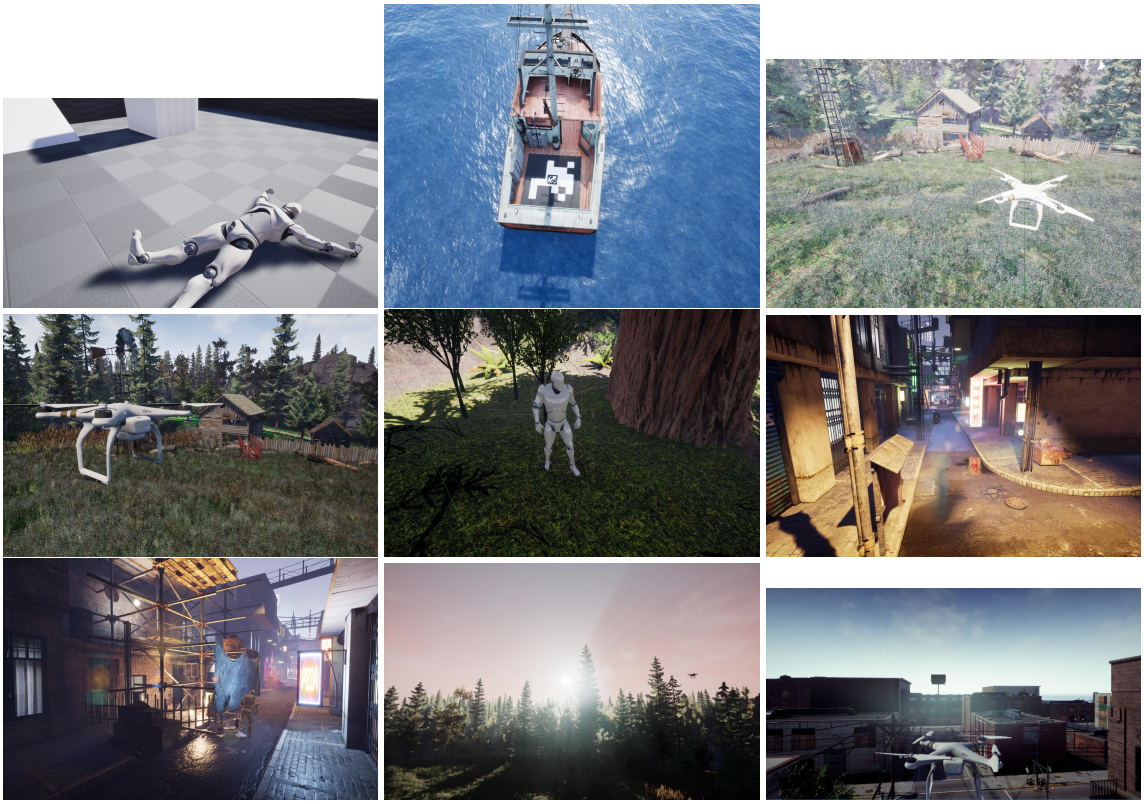

Figure 30: Screenshots from the holodeck, illustrating a variety of environments and agents.

currently using Holodeck for several projects to advance their research in unmanned aerial vehicle (UAV) perceptron and control. One of these projects is an ocean world where a UAV attempts to land on the back of a boat in choppy waters which the Holodeck team custom built. This world allows the MAGICC lab team to simulate and evaluate methods for perceiving landing sites and safely landing UAVs in complex environments. They also have used Holodeck as a simulator for research into plane orientation prediction where a gimbal camera attempts to track a plane and output the orientation based only on monocular images.

Holodeck has also become a regular tool for robotics vision classes at BYU. These classes use Holodeck for computer vision and controls problems such as detecting and navigating around obstacles and mapping city environments.

Holodeck has become a regular tool in the Perception Control and Cognition lab in performing reinforcement learning (RL) experiments researching ways to control complex humanoid robots, testing new RL algorithms and exploring multi agent RL interaction.

Holodeck has proved to be an invaluable tool in quickly iterating and running simulated experiments. Its visual fidelity, customizability and ease of use allows it to be extremely useful in a number of fields and use cases.

Holodeck had its first release on October 14, 2018. It was well received within the machine learning community and has benefited from bug reporting, feature suggestion and other community contributions.

5 Conclusions

We are deeply grateful to the Defense Advanced Research Projects Agency (DARPA) and the Probabilistic Programming for Advancing Machine Learning (PPAML) program for their generous support of this seedling. We have made significant progress towards cognitively plausible, practically relevant models that could potentially transform the way we think of and implement robotic systems. Perhaps more than anything, this seedling has laid the groundwork for continuing research into the intersection of probabilistic programming, deep learning, and cognitive robotics. While there are still many open questions, it is an exciting time to be developing, and harnessing, new and powerful tools to help solve some of mankind's most aspirational challenges.

6 Bibliography

[1] Marc Toussaint, Stefan Harmeling, and Amos Storkey. Probabilistic inference for solving (PO)MDPs. Technical Report 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*, pages 1292–1301, 2015.

[3] Fuhan Yan, Jiuchuan Jiang, Kai Di, Yichuan Jiang, and Zhifeng Hao. Multiagent pursuit-evasion problem with the pursuers moving at uncertain speeds. *Journal of Intelligent & Robotic Systems*, pages 1–17, 2018.

[4] Venkata Ramana Makkapati, Wei Sun, and Panagiotis Tsiotras. Optimal evading strategies for two-pursuer/one-evader problems. *Journal of Guidance, Control, and Dynamics*, 41(4):851–862, 2018.

[5] Henry M Wellman. The child's theory of mind. 1990.

[6] Nick Chater, Joshua B Tenenbaum, and Alan Yuille. Probabilistic models of cognition: Conceptual foundations. *Trends in cognitive sciences*, 10(7):287–291, 2006.

[7] 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*, pages 1014–1019, 2006.

[8] 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*, pages 1382–1387, 2006.

[9] 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*, pages 2759–2764. Citeseer, 2009.

[10] Noah D Goodman and Andreas Stuhlmüller. Knowledge and implicature: Modeling language understanding as social cognition. *Topics in cognitive science*, 5(1):173–184, 2013.

[11] 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*, pages 75–80. IEEE, 2010.

[12] Alan Fern, Sriraam Natarajan, Kshitij Judah, and Prasad Tadepalli. A decision-theoretic model of assistance. In *IJCAI*, pages 1879–1884, 2007.

[13] 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. *arXiv preprint arXiv:1206.5928*, 2012.

[14] 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*, pages 66–73. IEEE, 2016.

[15] Daphne Koller, David McAllester, and Avi Pfeffer. Effective bayesian inference for stochastic programs. In *AAAI/IAAI*, pages 740–747, 1997.

[16] Chris Frith and Uta Frith. Theory of mind. *Current Biology*, 15(17):R644–R645, 2005.

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

[18] Chris L Baker, Rebecca Saxe, and Joshua B Tenenbaum. Action understanding as inverse planning. *Cognition*, 113(3):329–349, 2009.

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

[20] Jan-Willem van de Meent, Brooks Paige, Hongseok Yang, and Frank Wood. An Introduction to Probabilistic Programming. *arXiv:1809.10756 [cs, stat]*, September 2018.

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

[22] 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)*, pages 1352–1359, 2005.

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

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

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

[26] L. M. Murray. Bayesian state-space modelling on high-performance hardware using LibBi. *arXiv preprint arXiv:1306.3277*, 2013.

[27] Adrien Todeschini, François Caron, Marc Fuentes, Pierrick Legrand, and Pierre Del Moral. Biips: Software for Bayesian inference with interacting particle systems. *arXiv preprint arXiv:1412.3779*, 2014.

[28] 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)*, pages 1024–1032, 2014.

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

[30] Hong Ge, Kai Xu, and Zoubin Ghahramani. Turing: A language for flexible probabilistic inference. In Amos Storkey and Fernando Perez-Cruz, editors, *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pages 1682–1690, Playa Blanca, Lanzarote, Canary Islands, 09–11 Apr 2018. PMLR.

[31] Dorit Borrman and Andreas Nuchter. Dataset generated by dorit borrman and andreas nuchter of jacobs university bremen. http://kos.informatik.uni-osnabrueck.de/3Dscans/, 2017. Accessed: 2017.

[32] Steven M LaValle. Rapidly-exploring random trees: A new tool for path planning. 1998.

[33] M L Benedikt. To take hold of space: Isovists and isovist fields. *Environment and Planning B: Planning and Design*, 6(1):47–65, 1979. doi: 10.1068/b060047. URL http://dx.doi.org/10.1068/b060047.

[34] 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.

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

[36] Tom Rainforth. Nesting Probabilistic Programs. *Uncertainty in Artificial Intelligence*, March 2018.

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

[38] Emanuel Todorov. Efficient computation of optimal actions. *Proceedings of the National Academy of Sciences of the United States of America*, 106(28):11478–11483, 2009. ISSN 0710743106. doi: 10.1073/pnas.0710743106.

[39] Chris L Baker and Joshua B Tenenbaum. Modeling human plan recognition using bayesian theory of mind. *Plan, activity, and intent recognition: Theory and practice*, pages 177–204, 2014.

[40] Solway A. and Botvinick MM. Goal-directed decision making as probabilistic inference: a computational framework and potential neural correlates. *Psychol Rev.*, (119(1)), 2012.

[41] Marco F. Cusumano-Towner and Vikash K. Mansinghka. Encapsulating models and approximate inference programs in probabilistic modules. *CoRR*, abs/1612.04759, 2016. URL http://arxiv.org/abs/1612.04759.

7 List of Symbols, Abbreviations, and Acronyms

BYU	Brigham Young University
DARPA	Defense Advanced Research Projects Agency
DNN	Deep Neural Network
ESS	Effective Sample Size
GPU	Graphics Processing Unit
ICML	International Conference on Machine Learning
IS	Importance Sampling
MAGICC	Multiple AGent Intelligent Coordination and Control Lab
MCMC	Markov Chain Monte-Carlo
MH	Metropolis-Hastings
PPAML	Probabilistic Programming for Advancing Machine Learning
PPL	Probabilistic Programming Language
RL	Reinforcement Learning
RRT	Rapidly-exploring Random Tree
SLAM	Simultaneous Localization and Mapping
SMC	Sequential Monte Carlo
TBB	Theory-based Bayesian
TF	TensorFlow
TOM	Theory of Mind
UAV	Unmanned Aerial Vehicle