Human-Agent Coordination in Games under Incomplete Information via Multi-Step Intent

Shenghui Chen*
University of Texas at Austin
Austin, United States
shenghui.chen@utexas.edu

Ruihan Zhao*
University of Texas at Austin
Austin, United States
ruihan.zhao@utexas.edu

Sandeep Chinchali
The University of Texas at Austin
Austin, United States
sandeepc@utexas.edu

Ufuk Topcu
The University of Texas at Austin
Austin, United States
utopcu@utexas.edu

ABSTRACT

Strategic coordination between autonomous agents and human partners under incomplete information can be modeled as turn-based cooperative games. We extend a turn-based game under incomplete information, the shared-control game, to allow players to take multiple actions per turn rather than a single action. The extension enables the use of multi-step intent, which we hypothesize will improve performance in long-horizon tasks. To synthesize cooperative policies for the agent in this extended game, we propose an approach featuring a memory module for a running probabilistic belief of the environment dynamics and an online planning algorithm called IntentMCTS. This algorithm strategically selects the next action by leveraging any communicated multi-step intent via reward augmentation while considering the current belief. Agent-to-agent simulations in the Gnomes at Night testbed demonstrate that IntentMCTS requires fewer steps and control switches than baseline methods. A human-agent user study corroborates these findings, showing an 18.52% higher success rate compared to the heuristic baseline and a 5.56% improvement over the single-step prior work. Participants also report lower cognitive load, frustration, and higher satisfaction with the IntentMCTS agent partner. [1]

KEYWORDS

Human-Agent Interaction; Online Planning; MCTS

ACM Reference Format:
Shenghui Chen, Ruihan Zhao, Sandeep Chinchali, and Ufuk Topcu. 2025. Human-Agent Coordination in Games under Incomplete Information via Multi-Step Intent. In *Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Detroit, Michigan, USA, May 19 – 23, 2025*, IFAAMAS, 9 pages.

*Equal contribution.
[1]Project page: https://vivianchen98.github.io/mhri_intent_aamas_page/.

Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Y. Vorobeychik, S. Das, A. Nowé (eds.), May 19 – 23, 2025, Detroit, Michigan, USA. © 2025 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

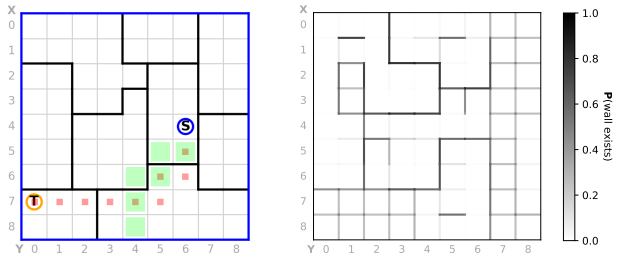


Figure 1: Snapshot of Gnomes at Night gameplay from the agent's perspective. *Left*: The agent's side of the maze, with the human's intent marked in large green squares and its own intent in small red squares. *Right*: The agent's probabilistic belief of the wall layout in the human's maze.

1 INTRODUCTION

Developing autonomous agents that can strategically coordinate with human partners under incomplete information, especially in long-horizon tasks, is important in the field of human-agent interaction. These agents hold the potential to improve interaction across various contexts, ranging from virtual applications like strategy board games [2] and action video games [7] to physical domains like assistive technologies such as wheelchairs [13].

A key challenge in achieving human-agent coordination is multi-step planning and strategizing. Humans naturally rely on multi-step planning [20] and frequently engage in multi-step intent communication, essential for deliberate coordination in long-horizon tasks. Therefore, for autonomous agents to be effective collaborators, they should also understand and perform multi-step reasoning.

An existing model for human-agent cooperation under incomplete information, the *shared-control game* introduced in [8], studies scenarios where each player has access only to its own transition function but not its partner's. However, this model is limited to a single action per turn, preventing the realization of more intricate, multi-step strategies. Although recent work in [9] partially mitigates this issue by allowing one of the players to take multiple actions in its turn, a full extension that supports multi-action dynamics for both players remains unexplored.

We extend the shared-control game by allowing both players to take multiple actions per turn, capturing more fluid interactions and bringing the model closer to natural human-agent coordination.

This extension enables the agent to leverage *multi-step intent*, which was previously unattainable due to the limitation of single-action turns. In this work, we define a multi-step intent as a variable-length sequence of states a player wishes its partner would visit next. The goal is to develop an agent, as the ego player in this game, that infers partner dynamics through their transition histories and coordinates effectively by exchanging multi-step intents.

The proposed approach, inspired by the Belief-Desire-Intention model [25], consists of a memory module and a planning module. The memory module enables the agent to continuously update its belief about unknown game dynamics by integrating new information inferred from the partner's histories. The planning module allows the agent to devise optimal actions by considering its current belief, prior knowledge, and any communicated multi-step intent. This approach exemplifies zero-shot online planning, requiring no prior training or data, yet it ensures adaptability to accumulating environment information and strategic interaction with the partner.

For the memory module, prior work has relied on belief updates based on natural language communication, which can introduce errors caused by human subjectivity or large language model parsing inaccuracies [8]. In contrast, our work uses a probabilistic belief model that performs Bayesian updates based on more objective information, like the partner's transition histories. The key idea is that for belief updates where accuracy matters, the agent should rely on what the partner *did*, not what they *said*. Additionally, we allow the Bayesian update to be weighted by the agent's confidence in both positive and negative evidence, ensuring a more accurate belief revision process.

For the planning module, we propose IntentMCTS, an intent-aware algorithm based on multi-action Monte Carlo tree search (MCTS), which allows players to take multiple actions per turn. IntentMCTS integrates multi-step intent by augmenting the environment reward with a bonus when a transition lands in a desired state included in the intent. This approach directly influences the search tree statistics, making it more natural than using intent for tie-breaking as done in [8]. In addition, reward augmentation helps densify the typically sparse reward structure in coordination tasks. Unlike reward shaping based on domain knowledge [11, 24] or potential functions [22], our approach relies on time-sensitive information gathered from the partner during the interaction. We evaluate four bonus schemes and find that a *discounted* reward bonus leads to the most effective coordination, measured by average steps and control switches taken.

We evaluate our approach using the Gnomes at Night testbed [8], where two players coordinate to move a single token from an initial position through a maze to a goal position. Each player sees a different maze layout and can only move the token along their visible routes. We modify the testbed so both players always see the goal, focusing on coordination rather than goal communication. We test all possible configurations across three mazes, running 10 trials per configuration, to compare IntentMCTS with a shortest-path-based heuristic controller, no-intent MCTS, and single-step intent MCTS. Agent-agent simulation results show that IntentMCTS outperforms the other MCTS methods, taking fewer steps and control switches, validating that reward augmentation via partner intent improves coordination. The heuristic controller performs well in simple cases but struggles in more complex ones due to its lack of adaptability, achieving an overall success rate of 88%, while MCTS-based agents succeed over 99% of the time. Finally, we conduct a human-agent user study where each participant plays with three agents using different planning algorithms: heuristic, single-step intent MCTS, and IntentMCTS. Gameplay with the IntentMCTS agent achieves an 18.52% higher success rate than the heuristic baseline and a 5.56% improvement over single-step MCTS, along with fewer median steps, fewer control switches, and smaller interquartile ranges for both metrics. Participants also report lower cognitive load, frustration, and higher satisfaction when partnered with the IntentMCTS agent compared to the other two agents.

2 INTENT-AWARE COORDINATION PROBLEM

We extend the shared-control game introduced in [8] by allowing both players to take multiple actions per turn. We formally define the game as follows:

Definition 1. A *multi-action shared-control game* is played between player E and player H, defined by a tuple $(\mathcal{X}, \mathcal{S}, \mathcal{A}, \mathcal{T}^E, \mathcal{T}^H, \mathcal{R})$. $\mathcal{X}$ is a finite set of environment states, with the initial state x_{init} and goal state x_{goal} known to both players. $\mathcal{S} = \mathcal{X} \times \{E, H\}$ is the controller state space, consisting of tuples (x, c) where $x \in \mathcal{X}$ is the environment state and $c \in \{E, H\}$ indicates which player is currently in control. The notation $-c$ denotes the other player without control. $\mathcal{A}$ is a finite set of actions available to both players including an action for switching control to the other player. $\mathcal{T}^i : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the deterministic transition function for player $i \in \{E, H\}$. $\mathcal{R}^{\text{env}} : \mathcal{S} \to \mathbb{R}$ is a function that assigns real-valued environment rewards to both players.

At each timestep t, when player c is in control, we define the *history* of the other player $-c$, denoted as $h_t^{-c} \in (\mathcal{X} \times \mathcal{A})^+$, as the sequence of state-action pairs taken by the other player up to timestep t. We assume that at timestep t, player i can communicate a *multi-step intent*, $\zeta_t^i \in \mathcal{X}^+$, representing a variable-length sequence of states that the player wishes the other player to visit next.

In this game, we aim to compute cooperative policies for the autonomous agent, E, when partnering with a human player, H. We formalize this problem as follows:

Problem 1. Given a human player whose behavior is denoted as π^H, the task is to compute the ego player's policy π^E that maximizes the expected total discounted reward:

$$\max_{\pi^E} \quad \mathop{\mathbb{E}}_{a_0, a_1 \cdots} \left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}^{\text{env}}(s_{t+1}) \right] \tag{1a}$$

$$\text{s.t.} \quad s_{t+1} = \mathcal{T}^{c_t}(s_t, a_t) \tag{1b}$$

$$a_t \sim \pi^{c_t}(s_t, h_t^{-c_t}, \zeta_t^{-c_t}) \tag{1c}$$

where $s_0 = s_{\text{init}}$ and $\zeta_t^c = \mathsf{IntentModel}^c(s_t, h_t^{-c})$.

3 RELATED WORKS

Intents for better coordination. The use of intent for improving coordination between agents can generally be categorized into two approaches: implicit *inference* or explicit *communication*. Intent *inference*, often explored through theory of mind [5, 21, 31] and opponent modeling [15, 30], allows agents to predict the goals and actions of others by modeling their beliefs, desires, and intentions.

However, these methods rely on inferring hidden states from noisy or incomplete observations, often resulting in inaccurate intent models and coordination inefficiencies. Intent *communication*, on the other hand, involves agents sharing explicit messages, such as future trajectories, to align their actions. Recent work has explored imagined trajectories [17], where agents share plans during training to improve coordination. Our method, however, focuses on zero-shot, real-time planning, where agents must coordinate without prior training. Our approach also differs by generating intent as a request for *what the sender player desires the other to do*, rather than predicting *what the other will do*. Despite progress in intent communication in multi-agent reinforcement learning, few approaches ensure that the communicated intent is semantically meaningful with respect to the task at hand, which is essential for effective human-agent coordination. Our approach, inspired by the Belief-Desire-Intention model [3, 25], continuously updates the agent's beliefs based on new observations made in the environment and models intents in a multi-step fashion. Building on recent work that uses single-step intents with non-probabilistic belief updates [8], our work allows multi-step intents and probabilistic belief modeling, supporting more dynamic and adaptive coordination in tasks over a longer horizon.

MCTS-based planning techniques. Monte Carlo tree search (MCTS) is an algorithm that combines tree-based search with Monte Carlo random sampling to explore and evaluate vast decision spaces efficiently [4] and has proven effective in strategic games [6, 26–28], making it ideal as the foundation of our planning module. Standard MCTS works best for turn-based games with perfect information. Several extensions have been developed to handle incomplete information. For example, *partially observable Monte Carlo planning* is designed to tackle environments modeled as *partially observable Markov decision processes* using particle filtering for state estimation [29]. *Information-set MCTS* tackles imperfect information games by maintaining perspective-based trees for each player, whose nodes correspond to players' information sets and edges correspond to moves from that player's viewpoint [10]. However, these methods only handle partial observations of states, not transitions as we do in our problem. To extend beyond traditional turn-based games, previous research [1, 16, 23] has introduced methods for handling *multi-action games*. As summarized by [12], tree search in these games can either create nodes for each individual action or for the entire action sequence within a turn. In this paper, we adopt the former approach for the benefit of a lower branching factor.

4 METHOD

This section presents an approach for an autonomous agent, acting as the ego player, to solve Problem 1. The agent's decision-making process is compartmentalized into a *memory module* (see Section 4.1) and a *planning module* (see Section 4.2).

4.1 Memory Module: Probabilistic Belief over Unknown Dynamics

In Problem 1, the agent player E must coordinate effectively without direct knowledge of its human partner's dynamics, $\mathcal{T}^H$. The memory module maintains a probabilistic belief over these unknown dynamics, updating it throughout the game via Bayesian inference based on the partner's transition history. We hypothesize that the memory module will improve coordination performance as the game progresses.

Algorithm 1 Memory Module: Belief Update

Input: action space $\mathcal{A}$, histories $h^H = \{(x, a), (x, a), \dots\}$
Memory: $\alpha, \beta : \mathcal{X} \times \mathcal{A} \to \mathbb{R}^+$, initialized to 1 $\forall x \in \mathcal{X}, a \in \mathcal{A}$.
for each (x, a) in h^H **do**
  for each $a' \in \mathcal{A}$ **do**
    if $a' = a$ **then**
      $\alpha(x, a) \leftarrow \alpha(x, a) + c^+$
    else
      $\beta(x, a') \leftarrow \beta(x, a') + c^-$
    end if
    update belief $b(x, a') \leftarrow \frac{\alpha(x,a')}{\alpha(x,a')+\beta(x,a')}$
  end for
end for
return updated belief $b : \mathcal{X} \times \mathcal{A} \to [0, 1]$,

In the game defined in Definition 1 with discrete state space $\mathcal{X}$ and action space $\mathcal{A}$, the presence of each deterministic state-action transition is modeled as a binary random variable $y \in \{0, 1\}$. The belief over these transitions is represented by a Bernoulli distribution $b : \mathcal{X} \times \mathcal{A} \to [0, 1]$, indicating the likelihood of a transition's existence. The probability mass function f is given by:

$$f(y|\theta) = \theta^y (1 - \theta)^{1-y}, \quad \theta = b(x, a), \tag{2}$$

where $y = 1$ when $\mathcal{T}^H(x, a)$ is defined and $y = 0$ otherwise.

This belief is initialized uniformly and updated dynamically during gameplay as new evidence is gathered from the partner's movement history h^H, indirectly revealing the presence or absence of transitions. Unlike prior work that relies on non-probabilistic updates based on potentially erroneous communication, this approach updates based on what the partner *did* over what it *said*. The belief update also incorporates positive and negative evidence with different confidence factors c^+ and c^-, respectively. We ensure $c^+ > c^-$ with the intuition that the presence of a transition in the history (positive evidence) confirms its existence, but the absence of a transition (negative evidence) does not necessarily indicate its nonexistence—it may simply reflect the partner's lack of attempt.

Theorem 1 derives the Bayesian update rule and confidence factor constraint. Algorithm 1 summarizes the belief update procedure.

Theorem 1. Let the prior belief about $\theta = b(s, a)$ follow a Beta distribution, $\theta \sim \text{Beta}(\alpha, \beta)$, i.e.,

$$f(\theta \mid \alpha, \beta) = \theta^{\alpha-1}(1 - \theta)^{\beta-1}. \tag{3}$$

We use a weighted likelihood for positive ($y = 1$) and negative ($y = 0$) evidence, with confidence factors $c^+, c^- \in \mathbb{R}^+$, where $c^+ > c^-$:

$$f(y \mid \theta) = \theta^{c^+ y}(1 - \theta)^{c^-(1-y)}. \tag{4}$$

The confidence factors must satisfy

$$c^+ = \frac{\log\left(1 - (1 - \theta)^{c^-}\right)}{\log(\theta)}. \tag{5}$$

Upon observing new evidence y, the posterior expectation of θ is:

$$\mathbb{E}(\theta \mid y) = \frac{\alpha + c^+ y}{\alpha + c^+ y + \beta + c^-(1-y)}. \tag{6}$$

Proof. The weighted Bernoulli likelihood is defined as $f(y \mid \theta) = \theta^{c^+ y}(1-\theta)^{c^-(1-y)}$ for $y \in \{0,1\}$. To ensure $f(y \mid \theta)$ is a valid probability distribution, it must satisfy $\sum_{y\in\{0,1\}} f(y \mid \theta) = 1$, thus $(1-\theta)^{c^-} + \theta^{c^+} = 1$. Solving for c^+ gives eq. (5) as desired.

Applying Bayes' theorem, the posterior distribution of θ given y is proportional to the product of the likelihood and the prior, i.e.,

$$\begin{aligned} f(\theta|x,\alpha,\beta) &\propto f(y \mid \theta) \cdot f(\theta \mid \alpha, \beta) \\ &= \theta^{\alpha + c^+ x - 1}(1-\theta)^{\beta + c^-(1-x) - 1}, \end{aligned}$$

by substituting in the weighted likelihood from eq. (4) and the Beta prior from eq. (3). This is the kernel of a Beta distribution with updated parameters, $\text{Beta}(\alpha + c^+ y, \beta + c^-(1-y))$. The posterior belief is therefore given by eq. (6) as desired. □

4.2 Planning Module: Multi-Step Intent as Reward Augmentation

The planning module is the core decision-making component of the autonomous agent. We introduce an online planning algorithm called IntentMCTS, which considers the current state, the multi-step intent communicated by the partner, and the current belief from its memory module to compute the most appropriate action. To accommodate the dynamics of players taking multiple actions per turn, this algorithm builds on multi-action Monte Carlo tree search (MCTS) [1, 16, 23], growing trees with nodes representing each atomic action within a turn. Additionally, IntentMCTS selects actions differently based on the available information at self-controlled nodes versus partner-controlled nodes, following the same idea as [8] in maximizing information use and growing the tree as efficiently as possible.

Notations: We denote v as a node in the search tree, with $s(v)$, $r(v)$, and $\delta(v)$ representing the state, reward, and transition feasibility at node v, respectively. Concretely, δ is the probabilistic belief of the action that leads to node v being valid. The visit count of node v is denoted by $N(v)$, while $Q(v)$ represents the total discounted return at node v. The set of child nodes of v is represented as $\mathbb{C}(v)$.

Algorithm. We present IntentMCTS for the ego player in Algorithm 2, which performs the following four phases iteratively:

Selection (lines 4-10): Starting from the root node, the agent recursively selects child nodes that maximize the Upper Confidence Bound value [18] until a terminal state or an expandable node is reached. A node is considered fully expanded if child nodes corresponding to all feasible actions are present.

Expansion (lines 15-26): Upon reaching an expandable node, the agent expands the tree by adding new child nodes based on its perception of available actions. When player E is in control, the true transition function is known, so only feasible actions are considered. However, when player H is in control, all actions are considered, with their feasibility δ estimated by the memory module in Section 4.1. Formally, we define the feasible action set as follows:

$$U(s) = \begin{cases} \{a \in \mathcal{A} | \mathcal{T}^E(s,a) \text{ is defined}\} & \text{if } c = E \\ \mathcal{A} & \text{if } c = H. \end{cases} \tag{7}$$

For a chosen unexplored action, we construct a child node that records this action a, the step reward r', the resulting state s', and its feasibility δ'. When the agent E is in control, s' is computed by its forward dynamics $\mathcal{T}^E$ with $\delta' = 1$. At a human-controlled node, s' is computed from $\mathcal{T}^+(s,a)$, a transition function that returns the next state assuming the applied action a is valid at state s, and δ' is retrieved from the memory module:

$$g(s,a) := (s', \delta') = \begin{cases} \mathcal{T}^E(s,a), 1 & \text{if } c = E \\ \mathcal{T}^+(s,a), b^H(s,a) & \text{if } c = H. \end{cases} \tag{8}$$

Since switching control is part of the action space, the search tree can expand multiple actions from one player or switch to another anytime, accommodating the game's multi-action dynamics.

Simulation (lines 27-39): A rollout from the expanded node simulates future transitions until a terminal state or a predefined depth T is reached. Actions are selected randomly among $U(s)$. At an agent-controlled state, the transition follows $\mathcal{T}^E$. In a human-controlled state, the agent imagines the effect of an action with its memory module. The transition is executed when a random number drawn uniformly between 0 and 1 is smaller than the feasibility belief. Otherwise, the state remains unchanged. The simulation accumulates discounted rewards into the outcome q and increments the depth d at each step.

Backpropagation (lines 40-51): The rewards obtained from the simulation phase are backpropagated up the tree, updating $Q(v)$ and $N(v)$ for all nodes along the path from the expanded node back to the root. Due to the uncertain feasibility of human-controlled nodes, the rollout return depends on the feasibility of the child node from which backpropagation comes. Suppose the child node v' gets a sample return q'_{sample}. Its parent node's return q_{sample} should consist of three terms: (1) the step reward r, (2) with probability δ', the transition is feasible, and the discounted future return is $\gamma q'_{\text{sample}}$, (3) with probability $(1-\delta')$, the transition is invalid (action has no effect) and the discounted future return is the discounted value estimate at the current state:

$$q_{\text{sample}} = r + \gamma \left[\delta' q'_{\text{sample}} + (1-\delta') \frac{Q(v)}{N(v)} \right]. \tag{9}$$

Reward Design. IntentMCTS performs reward augmentation to encourage the agent to follow the partner's intent trajectory. We denote the environment reward function as $\mathcal{R}^{\text{env}} : \mathcal{S} \to \mathbb{R}$. Aside from the base reward, we define an intent bonus $\mathcal{R}^{\text{int}} : \mathcal{S} \times \mathcal{X}^+ \to \mathbb{R}$. Since the algorithm tries to find an optimal action for player E, reward augmentation is applied only to its transitions:

$$\mathcal{R}(s, \zeta^H) = \mathcal{R}^{\text{env}}(s) + \mathbb{I}[c = E] \cdot \mathcal{R}^{\text{int}}(s, \zeta^H). \tag{10}$$

Specifically, given an intent trajectory $\zeta^H = \{x_1^H, x_2^H, \ldots, x_m^H\}$, we assign a *discounted* intent bonus to provide a smooth gradient of rewards along the intent trajectory:

$$\mathcal{R}^{\text{int}}((x,c), \zeta^H) = \begin{cases} \lambda^{m-i} & \text{if } x = x_i^H \in \zeta^H \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

Algorithm 2 Planning Module: IntentMCTS for player E

Input: current state s_t, intent-augmented reward $\mathcal{R}(s_t, \zeta_t^H)$, forward dynamics $g(s,a)$ defined in (8), feasible action set $U(s)$ defined in (7)
Parameters: maximum number of iterations $n = 100$, exploration constant $k = \sqrt{2}$, planning discount factor $\gamma = 0.99$, horizon $T = 100$

Create root node v_0 with state s_t
for n iterations **do**
  Set $v_l \leftarrow v_0$
  while v_l is not terminal **do**
    if v_l is fully expanded **then**
      $v_l \leftarrow \arg\max_{v' \in \mathbb{C}(v_l)} \left(\frac{Q(v')}{N(v')} + k\sqrt{\frac{\log N(v_l)}{N(v')}} \right)$
    else
      $v_l \leftarrow \textsc{Expand}(v_l)$
    end if
  end while
  $q \leftarrow \textsc{Rollout}(s(v_l))$
  $\textsc{BackProp}(v_l, q)$
end for
return action of best child $c^\star = \arg\max_{c \in \mathbb{C}(v_0)} N(c)$
procedure $\textsc{Expand}(v)$
  Choose untried action a from possible actions $U(s(v))$
  if a is switch turn **then**
    $x, c \leftarrow s(v)$
    $s' = (x, -c), \quad \delta' = 1, \quad r' = -1$
  else
    $s', \delta' \leftarrow g(s(v), a), \quad r' \leftarrow \mathcal{R}(s(v'), \zeta_t^H)$
  end if
  Create node v' with s', δ', r'
  $\mathbb{C}(v) \leftarrow \mathbb{C}(v) \cup \{v'\}$
  return v'
end procedure

procedure $\textsc{Rollout}(s)$
  Initialize $q = 0$
  while depth $d < T$ and s not terminal **do**
    Choose $a \in U(s)$ uniformly at random
    $s', \delta \leftarrow g(s, a)$
    if uniform-rng$(0,1) < \delta$ **then**
      $s \leftarrow s'$
    end if
    $q \leftarrow q + \gamma^d \mathcal{R}(s, \zeta_t^H)$
    $d \leftarrow d + 1$
  end while
  return q
end procedure

procedure $\textsc{BackProp}(v, q)$
  Initialize sample return $q_{\text{sample}} = q$ and $\delta = 1$
  while v is not null **do**
    Current value estimate $w = \frac{Q(v)}{N(v)}$ if $N(v) > 0$ else 0
    $q_{\text{sample}} \leftarrow r(v) + \gamma \left[\delta q_{\text{sample}} + (1 - \delta) w \right]$
    $Q(v) \leftarrow Q(v) + q_{\text{sample}}$
    $N(v) \leftarrow N(v) + 1$
    $\delta \leftarrow \delta(v)$
    $v \leftarrow$ parent of v
  end while
  return
end procedure

The intent discount factor $\lambda \in (0,1)$ adaptively trades off intent trajectory following and final intent state reaching. By design, $\mathcal{R}^{\text{int}} \in [0,1]$ adheres to the general principle that auxiliary objectives should not outweigh the task signals, which is a -1 control cost at each step in this paper. Moreover, considering a planning discount factor γ, we can compare two extreme cases: (1) The agent follows the intent trajectory meticulously to reach the final state and (2) the agent finds a shorter path with length $n < m$ to reach the final state without visiting any intermediate intent states:

$$J_{\text{follow}} = \sum_{t=0}^{m-1} \gamma^t (\lambda^{m-t-1} - 1). \qquad \text{(Case 1)}$$

$$J_{\text{skip}} = \gamma^{n-1}\lambda^0 + \sum_{t=0}^{n-1} \gamma^t (-1). \qquad \text{(Case 2)}$$

As shown in the equations above, the agent receives all intent bonuses in (Case 1) but only receives the last intent bonus in (Case 2). We analyze the benefit of skipping to the final state as follows:

$$\begin{aligned} J_{\text{diff}} := J_{\text{skip}} - J_{\text{follow}} &= \sum_{t=n-1}^{m-1} \gamma^t - \sum_{t=0}^{m-1} \gamma^t \lambda^{m-t-1} \\ &= \frac{\gamma^n - \gamma^{m+1}}{\gamma(1-\gamma)} - \frac{\gamma^m - \lambda^m}{\gamma - \lambda}. \end{aligned} \qquad (13)$$

When we choose $\lambda < \gamma$, we can compute a threshold for n that makes one case more preferable:

$$J_{\text{diff}} \begin{cases} > 0 & \text{if } n < \log_\gamma \left[\gamma^{m+1} + \frac{\gamma(1-\gamma)(\gamma^m - \lambda^m)}{\gamma - \lambda} \right] \\ \leq 0 & \text{otherwise.} \end{cases} \qquad (14)$$

As shown in the inequality above, choosing the shorter path (Case 2) is preferable to following the intent trajectory (Case 1) only for small enough n. Changing the hyper-parameter γ smoothly adjusts the decision boundary between the two cases and results in different levels of intent-following behavior in IntentMCTS.

5 AGENT-TO-AGENT SIMULATION

We evaluate our approach on the Gnomes at Night testbed from [8], a board game where two players coordinate to move a single token through a maze. The two players see different maze layouts and can only move the token along the paths within their respective mazes. Each game configuration $(x_{\text{init}}, x_{\text{goal}})$ pairs an initial and goal position, and a round is considered successful if the players reach the goal within 1000 steps. To focus on coordination rather than goal discovery, we modify the testbed to reveal the goal position x_{goal} to both players from the start, eliminating the need for communication about the goal's location and isolating the core coordination challenge. The state space includes all maze grid cells.

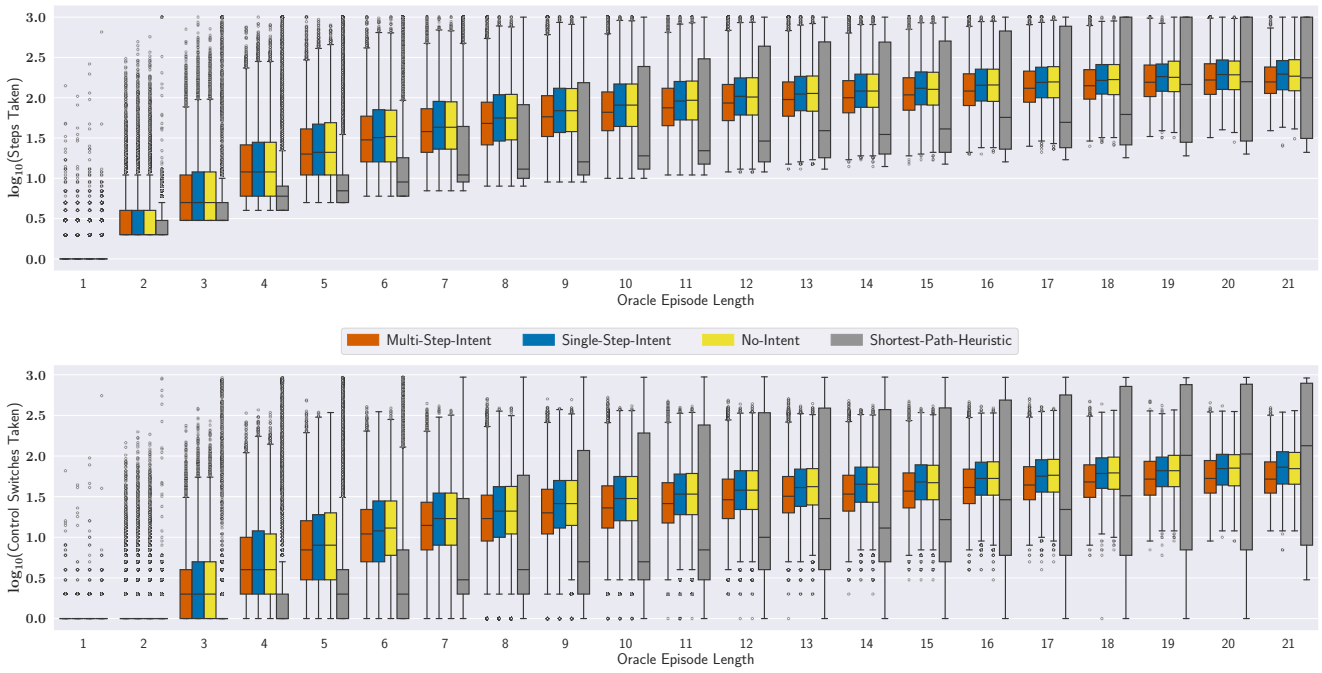


Figure 2: Performance comparison of our method (orange) with three baselines, measuring steps taken (top) and control switches (bottom). The Y-axis is on a log-10 scale, and the X-axis represents the oracle episode length, indicating task difficulty.

Players can move right, up, left, down, or switch control. The transition function is deterministic, allowing actions unless blocked by a wall. The reward is 100 for reaching the goal, with a penalty of −1 per step to encourage efficient coordination.

5.1 Intent Model

The agent-to-agent simulation aims to anticipate the agent's performance when playing with humans. Hence, IntentMCTS agents should communicate and plan with human-like intent trajectories. In the problem formulation, player c's intent at timestep t, ζ_t^c, is retrieved from IntentModel^c given the current state s_t and the partner's history h_t^{-c}. In our experiments, we instantiate both agents' intent models with the belief-conditioned shortest path toward the goal. Concretely, before switching control, each agent plans the lowest-cost path as its intent, factoring in its maze layout and belief over partner transitions: taking a valid transition costs −1, and crossing a wall incurs a penalty of $-10(1-\delta)$. The resulting intent minimizes steps and uncertainty: the agents prefer to traverse on their side of the maze and prioritize more feasible partner transitions when blocked.

5.2 Comparison with Baselines

To verify the effectiveness of our IntentMCTS planning algorithm, we compare with three relevant baselines: a shortest-path-based heuristic controller, MCTS without intent bonus, and MCTS considering single-step intent, all using the same memory module:

- **Shortest-Path Heuristic Controller**: Same as how the intent trajectory is generated, the agent plans the lowest-cost path to the goal. If blocked by a wall, control passes to its partner. To improve the robustness of the controller, a random action is taken instead with 20% chance.
- **No-Intent MCTS**: No reward augmentation regarding partner intent. The agents must plan with environment rewards only.
- **Single-Step-Intent MCTS**: Instead of reward augmentation, the intents are used as a tie-breaker during action selection. At the end of MCTS, if two child nodes are equally preferable and one follows the intent, that node is selected.

We test all configurations across three 9×9 mazes, each containing $81 \times 80 = 6480$ different configurations. For each configuration, we run 10 trials. The oracle episode length for each configuration is calculated as the minimal number of moves plus control switches needed to reach the goal, considering both players' maze layouts. This metric reflects task difficulty. We then plot the number of steps and control switches required to complete the mazes against the oracle episode lengths for all tested trials.

Figure 2 shows that IntentMCTS consistently outperforms the other two MCTS-based methods, taking fewer steps and fewer control switches. The result validates that reward augmentation based on partner intent effectively improves coordination. The heuristic controller performs well in easy configurations. However, it lacks the flexibility to take a detour when both agents are blocked at the same grid and perform poorly in more challenging configurations. Across all game trails, the heuristic controller gets a success rate of 88%, whereas all MCTS-based agents succeed over 99% of the time.

5.3 Ablation Study

We perform an ablation study with three alternative reward schemes to justify the discounted intent reward used in IntentMCTS: fixed

reward, first step only reward, and length inverse reward. Note that these alternatives all range between the same bound $[0, 1]$ as R^{int}, guaranteed not to outweigh the control cost of -1 in our setting.

First, a *fixed* intent bonus assigns a constant reward whenever the agent visits a state appearing in the intent trajectory:

$$\mathcal{R}^{\text{fixed}}((x, c), \zeta^H) = 0.5 \cdot \mathbb{I}[x \in \zeta^H]. \tag{15}$$

The fixed bonus scheme weighs all intent states equally, ignoring the temporal information within the trajectories. Alternatively, analogous to the simple-step MCTS, the *first step only* reward only encourages visiting the immediate next intent state. Given an intent trajectory $\zeta^H = \{x_1^H, x_2^H, \ldots, x_m^H\}$, we have:

$$\mathcal{R}^{\text{fso}}((x, c), \zeta^H) = 0.5 \cdot \mathbb{I}[x = x_1^H]. \tag{16}$$

On the other hand, the *length inverse* bonus scheme prioritizes reaching the final intended state:

$$\mathcal{R}^{\text{linv}}((x, c), \zeta^H) = \begin{cases} 1 & \text{if } x = x_m^H \\ \frac{1}{m} & \text{if } x \in \zeta^H, x \neq x_m^H \\ 0 & \text{otherwise.} \end{cases} \tag{17}$$

By design, the agent receives a big bonus only when reaching the final intent state. Traversing the environment exactly as intended is unnecessary, especially for long trajectories.

As Table 1 shows, the *discounted* reward bonus leads to the most effective coordination quantified by the fewest average steps and control switches taken. We report in geometric mean ± geometric standard deviation across all experiment trials because the data are positive and right-skewed.

Bonus Scheme	Steps Taken	Control Switches Taken
fixed	45.69 ± 3.98	17.45 ± 3.54
first step only	46.31 ± 3.98	18.03 ± 3.55
length inverse	45.98 ± 3.96	18.09 ± 3.58
discounted	**44.21 ± 3.92**	**17.11 ± 3.52**

Table 1: Ablation study results for steps and control switches across reward bonus schemes.

6 HUMAN-AGENT EVALUATION

We conduct a user study where human participants play with different agent partners in the Gnomes at Night testbed.

6.1 User Study Design

Independent Variables. The study varies the agent decision-making algorithms from the proposed algorithm and baseline methods:

Agent Alice: *Shortest-Path-Heuristic* Controller,
Agent Bob: *Multi-Step-Intent* MCTS (IntentMCTS),
Agent Charlie: *Single-Step-Intent* MCTS.

All agents share the same memory module, as detailed in Section 4.1. We exclude No-Intent MCTS, as [8] has already demonstrated that single-step intent performs better than no-intent.

Method	Session 1	Session 2	Session 3
Shortest-Path-Heuristic	0.37	0.37	0.26
Single-Step-Intent	0.32	0.26	0.42
Multi-Step-Intent	0.31	0.37	0.32

Table 2: Even distribution of methods across session orders.

Dependent Variables. We measure the number of steps taken, the number of control switches taken, and whether the token reaches the goal position. Participants also complete a 7-point Likert survey: The first six questions are from the NASA-TLX [14], covering mental demand, physical demand, temporal demand, frustration, effort, and performance, respectively. The final question assesses participant satisfaction with each agent partner.

Hypotheses. We design the experiment with two hypotheses:

(H1) Gameplay with the *Multi-Step-Intent* agent will outperform the other two agents in coordination efficiency.
(H2) Participants will report better subjective survey scores on the *Multi-Step-Intent* agent than other two agents.

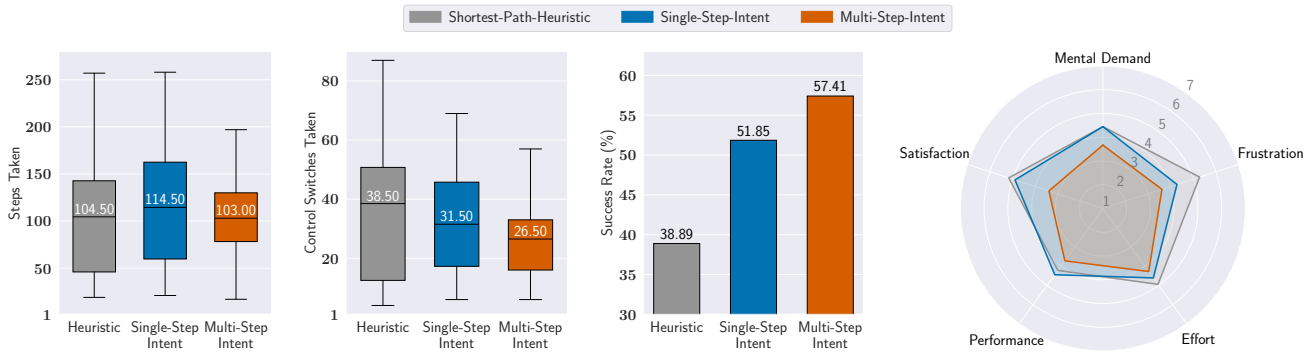
Experiment Design. We use a *within-subject design* where each participant completes three game sessions, each partners with one of the agents—Alice, Bob, or Charlie—as outlined in the independent variables. All sessions use the same maze, and each session includes the same set of three distinct configurations to ensure consistency for comparison. We ask participants to answer the 7-item survey after each session. To control for order effects—such as learning, fatigue, and carryover—we use *counterbalancing* to randomize the order of sessions [19]. The counterbalancing ensures an even distribution of methods across session orders, as shown in Table 2. Within each session, we also randomize the order of configurations to prevent boredom or disengagement over time. This two-level randomization design reduces potential biases, allowing for a more accurate comparison of participant performance across methods.

Participants. The study recruits 18 university students as consenting participants, with an average age of 26.28. The gender distribution was 0.83 male, 0.11 female, and 0.06 non-binary.

6.2 Results and Discussions

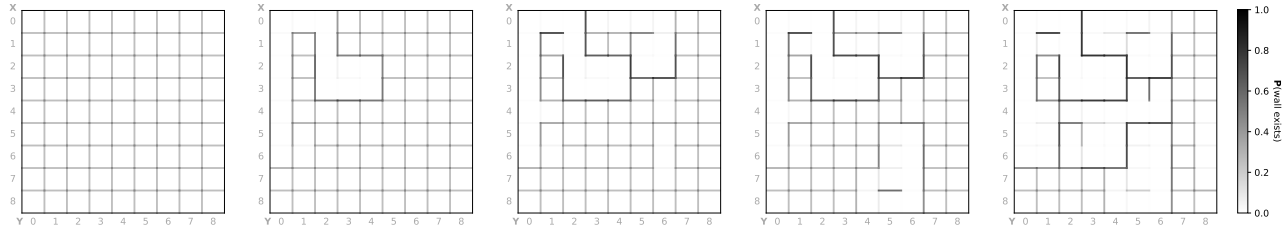
Regarding (H1). We present the distributions of steps and control switches for each method using boxplots, with median values labeled in white text. In Figure 3a, *Multi-Step-Intent* shows a median step count 1.5 steps lower than *Shortest-Path-Heuristic* and 11.5 steps lower than *Single-Step-Intent*, with a significantly smaller interquartile range (IQR), indicating less variation due to randomness or participant differences. Similarly, Figure 3b shows *Multi-Step-Intent* requires a median of 12 fewer control switches than *Shortest-Path-Heuristic* and 5 fewer than *Single-Step-Intent*, again with a much smaller IQR. Additionally, Figure 3c highlights that *Multi-Step-Intent* improves success rates by 5.56% over *Single-Step-Intent* and 18.52% over *Shortest-Path-Heuristic*. These results support (H1), demonstrating improved coordination efficiency.

Regarding (H2). We present average participant ratings for 5 of the 7 survey items in Figure 3d, excluding physical and temporal demand as they are less relevant to the task. Lower scores indicate



(a) Steps Taken (b) Control Switches Taken (c) Success Rates (d) Survey Ratings

Figure 3: (a, b) Box plots for steps and control switches taken, with medians labeled in white text; (c) Bar chart for average success rates; (d) Radar chart for average ratings from 7-point Likert scale survey (Lower values are preferred in all questions, e.g., 1=very satisfied and 7=not satisfied at all).



(a) Step 0 (b) Step 100 (c) Step 250 (d) Step 350 (e) Step 620

Figure 4: Visualization of the agent's belief of the wall layout in human's maze at key steps in gameplay. Darker lines indicate a stronger belief in the presence of walls.

lower cognitive load or higher satisfaction. The *Multi-Step-Intent* agent achieves lower scores in all dimensions than the other two agents, shown as the smallest orange area, supporting (H2) and demonstrating an enhanced user experience.

Effect of Memory Module in Gameplay. In the Gnomes at Night testbed, players' private transition functions reflect the unique wall layouts in their mazes. Figure 4 showcases how the agent's belief about the human's maze walls, stored in its memory module, evolves during a human-agent sample gameplay. Darker lines indicate a stronger belief in the presence of walls. Starting with a uniform belief of 0.5 for all walls, the agent refines its understanding, identifying one room's boundaries by step 100 and inferring the overall maze layout by the end of the game at step 620.

Qualitative Observations. The quantitative results from both objective metrics and subjective ratings clearly indicate that *Multi-Step-Intent* outperforms *Shortest-Path-Heuristic* and *Single-Step-Intent*. Qualitative feedback from participants reveals additional insights. Many participants appreciate the explainability provided by the multi-step intent algorithm, which enhances their satisfaction even when the performance difference is not immediately evident. We also observe that participants use alternative strategies, such as marking cells around closed rooms to signal being trapped rather than moving along the room boundary repeatedly. However, frustration arises when the agent seems to ignore their intended paths, suggesting that incorporating natural language communication or clarification as a valuable next step.

7 CONCLUSION

We extend a turn-based shared-control game under incomplete information to allow multiple actions per turn, enabling the use of multi-step intent to enhance performance in long-horizon tasks. We develop IntentMCTS, an online planning algorithm that incorporates a memory module for probabilistic beliefs and leverages multi-step intent through reward augmentation. Both agent-to-agent simulations and a human-agent user study show that IntentMCTS outperforms baseline methods in terms of steps, turns, and success rates. The user study also highlights improvements in participant cognitive load and satisfaction toward the partner.

For future work, we can explore using multi-step intent as a constraint when feasible, extracting it from natural language for belief updates, and transitioning from a heuristic to a data-driven intent generation model.

ACKNOWLEDGMENTS

This work was supported by NSF CNS-1836900, NSF CPS-2133481, ARL W911NF-23-2-0011 and the Lockheed Martin Corporation. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

REFERENCES

[1] Hendrik Baier and Peter I Cowling. 2018. Evolutionary MCTS for Multi-Action Adversarial Games. In *Proceedings of the Conference on Computational Intelligence and Games (CIG)*. IEEE, Maastricht, 1–8.

[2] Nolan Bard, Jakob N Foerster, Sarath Chandar, Neil Burch, Marc Lanctot, H Francis Song, Emilio Parisotto, Vincent Dumoulin, Subhodeep Moitra, Edward Hughes, et al. 2020. The Hanabi Challenge: A New Frontier for AI Research. *Artificial Intelligence* 280, C (2020), 103216.

[3] Michael Bratman. 1987. *Intention, Plans, and Practical Reason* (1 ed.). Harvard University Press, Cambridge, MA.

[4] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. 2012. A Survey of Monte Carlo Tree Search Methods. *Transactions on Computational Intelligence and AI in Games* 4, 1 (2012), 1–43.

[5] Lindsey J Byom and Bilge Mutlu. 2013. Theory of Mind: Mechanisms, Methods, and New Directions. *Frontiers in Human Neuroscience* 7 (2013), 413.

[6] Murray Campbell, A Joseph Hoane Jr, and Feng-hsiung Hsu. 2002. Deep Blue. *Artificial Intelligence* 134, 1–2 (2002), 57–83.

[7] Micah Carroll, Rohin Shah, Mark K Ho, Tom Griffiths, Sanjit Seshia, Pieter Abbeel, and Anca Dragan. 2019. On the Utility of Learning about Humans for Human-AI Coordination. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Curran Associates Inc., Vancouver, Article 465, 12 pages.

[8] Shenghui Chen, Daniel Fried, and Ufuk Topcu. 2024. Human-Agent Cooperation in Games under Incomplete Information through Natural Language Communication. In *Proceedings of the 33rd International Joint Conference on Artificial Intelligence, Human-Centred AI*. International Joint Conferences on Artificial Intelligence Organization, Jeju, 7833–7841.

[9] Shenghui Chen, Shufang Zhu, Giuseppe De Giacomo, and Ufuk Topcu. 2024. Learning to Coordinate without Communication under Incomplete Information. arXiv:2409.12397

[10] Peter I Cowling, Edward J Powley, and Daniel Whitehouse. 2012. Information Set Monte Carlo Tree Search. *Transactions on Computational Intelligence and AI in Games* 4, 2 (2012), 120–143.

[11] Marco Dorigo and Marco Colombetti. 1994. Robot Shaping: Developing Autonomous Agents through Learning. *Artificial Intelligence* 71, 2 (1994), 321–370.

[12] Tsubasa Fujiki, Kokolo Ikeda, and Simon Viennot. 2015. A Platform for Turn-Based Strategy Games, with a Comparison of Monte-Carlo Algorithms. In *Proceedings of the Conference on Computational Intelligence and Games*. IEEE, Tainan, 407–414.

[13] Aditya Goil, Matthew Derry, and Brenna D Argall. 2013. Using Machine Learning to Blend Human and Robot Controls for Assisted Wheelchair Navigation. In *Proceedings of the 13th International Conference on Rehabilitation Robotics*. IEEE, Seattle, 1–6.

[14] Sandra G. Hart and Lowell E. Staveland. 1988. Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research. *Advances in Psychology* 52 (1988), 139–183.

[15] He He, Jordan Boyd-Graber, Kevin Kwok, and Hal Daumé, III. 2016. Opponent Modeling in Deep Reinforcement Learning. In *Proceedings of The 33rd International Conference on Machine Learning*. PMLR, New York, 1804–1813.

[16] Niels Justesen, Tobias Mahlmann, Sebastian Risi, and Julian Togelius. 2017. Playing Multiaction Adversarial Games: Online Evolutionary Planning versus Tree Search. *IEEE Transactions on Games* 10, 3 (2017), 281–291.

[17] Woojun Kim, Jongeui Park, and Youngchul Sung. 2021. Communication in Multi-Agent Reinforcement Learning: Intention Sharing. In *Proceedings of the 9th International Conference on Learning Representations*. OpenReview, Virtual. https://openreview.net/forum?id=qpsl2dR9twy

[18] Levente Kocsis and Csaba Szepesvári. 2006. Bandit Based Monte-Carlo Planning. In *Proceedings of the European Conference on Machine Learning*. Springer Berlin, Heidelberg, Berlin, Germany, 282–293.

[19] David W Martin. 2007. *Doing Psychology Experiments* (7 ed.). Wadsworth/Thomson Learning, Belmont, CA.

[20] Kevin J Miller and Sarah Jo C Venditto. 2021. Multi-Step Planning in the Brain. *Current Opinion in Behavioral Sciences* 38 (2021), 29–39.

[21] Sahil Narang, Andrew Best, and Dinesh Manocha. 2019. Inferring User Intent using Bayesian Theory of Mind in Shared Avatar-Agent Virtual Environments. *IEEE Transactions on Visualization and Computer Graphics* 25, 5 (2019), 2113–2122.

[22] Andrew Y Ng, Daishi Harada, and Stuart Russell. 1999. Policy Invariance under Reward Transformations: Theory and Application to Reward Shaping. In *Proceedings of the 16th International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, 278–287.

[23] Connor M Pipan. 2021. *Application of the Monte-Carlo Tree Search to Multi-Action Turn-Based Games with Hidden Information*. Master's thesis. Air Force Institute of Technology, Ohio, USA.

[24] Jette Randløv and Preben Alstrøm. 1998. Learning to Drive a Bicycle Using Reinforcement Learning and Shaping. In *Proceedings of the 15th International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, 463–471.

[25] Anand S Rao, Michael P Georgeff, et al. 1995. BDI Agents: From Theory to Practice. In *Proceedings of the 1st International Conference on Multiagent Systems*. AAAI Press, San Francisco, 312–319.

[26] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature* 529, 7587 (2016), 484–489.

[27] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. 2018. A General Reinforcement Learning Algorithm that Masters Chess, Shogi, and Go through Self-Play. *Science* 362, 6419 (2018), 1140–1144.

[28] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. 2017. Mastering the Game of Go without Human Knowledge. *Nature* 550, 7676 (2017), 354–359.

[29] David Silver and Joel Veness. 2010. Monte-Carlo Planning in Large POMDPs. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems*. Curran Associates Inc., Vancouver, 2164–2172.

[30] Michalis Smyrnakis and David S. Leslie. 2010. Dynamic Opponent Modelling in Fictitious Play. *Comput. J.* 53, 9 (2010), 1344–1359.

[31] Wako Yoshida, Ray J Dolan, and Karl J Friston. 2008. Game Theory of Mind. *PLOS Computational Biology* 4, 12 (2008), 1–14.