Methods and Mechanisms for Interactive Novelty Handling in Adversarial Environments

Extended Abstract

Tung Thai
Tufts University
Medford, MA, United States
tung.thai@tufts.edu

Mudit Verma
Arizona State University
Tempe, AZ, United States
mverma13@asu.edu

Utkarsh Soni
Arizona State University
Tempe, AZ, United States
usoni1@asu.edu

Sriram Gopalakrishnan
Arizona State University
Tempe, AZ, United States
sgopal28@asu.edu

Ming Shen
Arizona State University
Tempe, AZ, United States
mshen16@asu.edu

Mayank Garg
Arizona State University
Tempe, AZ, United States
mgarg20@asu.edu

Ayush Kalani
Arizona State University
Tempe, AZ, United States
akalani2@asu.edu

Nakul Vaidya
Arizona State University
Tempe, AZ, United States
nvaidya7@asu.edu

Neeraj Varshney
Arizona State University
Tempe, AZ, United States
nvarshn2@asu.edu

Chitta Baral
Arizona State University
Tempe, AZ, United States
chitta@asu.edu

Subbarao Kambhampati
Arizona State University
Tempe, AZ, United States
rao@asu.edu

Jivko Sinapov
Tufts University
Medford, MA, United States
jivko.sinapov@tufts.edu

Matthias Scheutz
Tufts University
Medford, MA, United States
matthias.scheutz@tufts.edu

ABSTRACT
Learning to detect, characterize and accommodate novelties is a challenge that agents operating in open-world domains need to address to achieve satisfactory task performance. We sketch general methods for detecting and characterizing different types of novelties, and for building an appropriate adaptive model to accommodate them utilizing logical representations and reasoning methods in stochastic partially observable multi-agent environments. We also briefly report results from evaluations of our algorithms in the game domain of Monopoly. The results show high novelty detection and accommodation rates.

KEYWORDS
Open-world AI, Agent Architecture, Adaptive Multiagent Systems

1 INTRODUCTION: OPEN-WORLD AI
Many classical AI tasks take place in closed-world domains where the types of entities, their actions, and the overall domain dynamics are known. In contrast, open-world domains allow for novel entities, actions, etc. to arise anytime unbeknownst to the task-performing agent who needs to handle them (cp. to [1, 18]). Especially interactive novelties where agents interact with each other and with the environment in novel ways present a challenge for agents departing from a closed-world assumption (e.g., [2, 6, 9]. This is different from the agent being unaware about certain parts of the world, like other agent’s rewards [11, 14–17] or reasoning mechanisms [3–5, 12], as opposed to the world-changing without emitting explicit signals to the agent.

To tackle the challenges of interactive novelties in open-world environments, we developed a general novelty-handling framework that uses symbolic logical reasoning to detect, learn, and adapt to novelties in open-world environments. The results suggest that our agent can detect novelty with a high accuracy rate while maintaining a dominant performance against other game-playing agents. For a more complete description of our work, please see our full paper on arXiv [13].
2 METHODS AND MECHANISMS

We will use the multi-player adversarial board game Monopoly to briefly describe our methods for detecting, characterizing, and accommodating novelties, as it was also used for the evaluation. In Monopoly, up to four players roll dice to make moves and take actions on the game board with the goal of being the last player standing after bankrupting other players. This objective can be achieved by buying properties, monopolizing color sets, and developing houses on properties. The game includes different surprise factors such as chance cards, community cards, jail, auction, and trading ability between agents. Hence, any action in the game needs to be adapted to dice rolls, community cards, chance cards, and the decisions of other players. Unlike traditional Monopoly, where one can fully observe all the states and actions of other agents, the “Open-world Monopoly” version is only partially observable, i.e., it does not allow us to monitor all the actions and interactions on our turn [7].

2.1 Novelty Detection

We record the information of the game as provided by the Monopoly simulation (“game environment”) and compare it with our “expectation” state of the game board. This “expectation” state is derived from the agent’s knowledge base of the game, including expected states, actions, action preconditions, and end effects. Then, the game environment provides us with the actual game board states and actions that have occurred between the current time step and the previous time our agent performed an action. When we notice a discrepancy between our expected state and the actual state, we surmise that something must have changed within the game.

2.2 Novelty Characterization

Next, the agent uses a novelty identification module to characterize the novelty using “Answer Set Programming” (ASP). The resulting program’s answer sets give us the parameter values which reconcile the predicted game board state and the observed game board state. If there is only one answer set and thus a unique parameter value, then if this value is different from the value we had earlier, we have identified a novelty. Now we can update our ASP code that was used for hypothetical reasoning by simply replacing the earlier value of the parameter with the new value.

2.3 Novelty Accommodation

Since novelties in the state (features, dynamics, actions) mean the agent would have to replan often and would have to do so based on the most updated information, we were interested in developing an online planning algorithm to determine the best action. However, with environments that are both long-horizon and stochastic, using online planning approaches like Monte-Carlo tree search, quickly becomes intractable. To address this problem, we formulate a truncated-rollout-based algorithm that uses updated domain dynamics (learned from detected novelties) for a few steps of the rollout and then uses a state evaluation function to approximate the return for the rest of that rollout. In our evaluation function, we use both domain-specific components and a more general heuristic to approximate the return from the state after the truncated rollout.

<table>
<thead>
<tr>
<th>Action Novelties</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPR</td>
</tr>
<tr>
<td>FPR</td>
</tr>
<tr>
<td>NRP</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Interaction Novelties</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPR</td>
</tr>
<tr>
<td>FPR</td>
</tr>
<tr>
<td>NRP</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Relation Novelties</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPR</td>
</tr>
<tr>
<td>FPR</td>
</tr>
<tr>
<td>NRP</td>
</tr>
</tbody>
</table>

Table 1: Evaluation results (see text for details).

Furthermore, to ensure the agent adapts to the detected novelties, we made both the environment simulator used for rollouts and the evaluation function sufficiently flexible and conditioned on the environment attributes; we only used a few tuned constants. Thus, whenever a novelty was detected, we updated the relevant attributes in our simulator and evaluation function before running our algorithm to decide our actions. Using this approach, we are able to incorporate novel information into our decision-making process and adapt efficiently.

3 EVALUATION & RESULTS

In an effort to maintain the integrity of the evaluation, all the information about the novelty was hidden from our team, and all the information about our architecture or methodologies was also hidden from the evaluation team. Our agent was evaluated based on three different metrics: the correctly detect novelties, i.e., true positive rate (TPR), the incorrectly detect novelties, i.e., false positive rate (FPR), and the novelty reaction performance (NRP) after the novelty was introduced (post-novelty) in Table 1. We compute the novelty reaction performance (NRP) of the agent based on the following formula: \[ NRP = \frac{W_{\text{agent}}}{W_{\text{baseline}}} \] where, \( W_{\text{agent}} \) is the win rate of our agent. \( W_{\text{baseline}} \) is 65%.

4 CONCLUSION

Our work presented a new agent architecture for interactive novelty handling in an adversarial environment that can detect, characterize, and accommodate novelties. First, we use ASP to detect and characterize interactive novelties. Then, we update the detected novelties to our agent’s knowledge base. Finally, we utilize the truncated-rollout MCTS agent to accommodate the novelty. The external evaluation results support the cognitive architecture’s effectiveness in handling different levels of interactive novelty. In the near future, we would like to model the opponents’ behavior using reinforcement learning due to its potential to learn opponents’ behavior without knowing opponent’s observations and actions [8, 10]. Ultimately, we believe improving the model’s capability of predicting another agent’s behaviors is the biggest area for growth.

ACKNOWLEDGMENTS

This work was funded in part by DARPA grant W911NF-20-2-0006. We would like to thank Mayank Kejriwal, Shilpa Thomas, Min-Hsueh Chiu and other members of the University of Southern California team for the Monopoly simulator and agent evaluation.

2386
REFERENCES


[19] 2387