Deceptive Path Planning via Reinforcement Learning with Graph Neural Networks

Extended Abstract

Michael Y. Fatemi
University of Virginia
Charlottesville, VA, United States
gsk6me@virginia.edu

Wesley A. Suttle
U.S. Army Research Laboratory
Adelphi, MD, United States
wesley.a.suttle.ctr@army.mil

Brian M. Sadler
U.S. Army Research Laboratory
Adelphi, MD, United States
brian.m.sadler6.civ@army.mil

ABSTRACT

Deceptive path planning (DPP) is the problem of designing a path that hides its true goal from an outside observer. Existing methods for DPP rely on unrealistic assumptions, such as global state observability and perfect model knowledge, and therefore do not generalize to unseen problem instances, lack scalability to realistic problem sizes, and preclude both on-the-fly tunability of deception levels and real-time adaptivity to changing environments. In this paper, we propose a reinforcement learning (RL)-based scheme that overcomes these issues. Through extensive experimentation we show that, without additional fine-tuning, at test time the resulting policies successfully generalize, scale, enjoy tunable levels of deception, and adapt in real-time to changes in the environment.

KEYWORDS

reinforcement learning; deception; graph neural networks

ACM Reference Format:
Michael Y. Fatemi, Wesley A. Suttle, and Brian M. Sadler. 2024. Deceptive Path Planning via Reinforcement Learning with Graph Neural Networks: Extended Abstract. In *Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), Auckland, New Zealand, May 6 – 10, 2024*, IFAAMAS, 3 pages.

1 INTRODUCTION

A challenging problem at the intersection of deception and autonomy is that of deceptive path planning (DPP): designing a path from a starting location to a goal location that hides the agent's true goal from an external, potentially adversarial observer. The community has produced a range of methods for solving this problem, including classical planning- and control-based methods [13, 15, 19] as well as efforts at reinforcement learning-based approaches [10, 12]. Unfortunately, these methods all suffer from some combination of the following: the need for perfect knowledge of the environment, lack of scalability to realistic problem sizes, excessive computational overhead, lack of generalizability to unseen problems, and/or lack of deceptiveness tunability. These drawbacks are primarily due to the model knowledge requirements of classical planning-based methods and model-based RL methods, and the inflexible and problem-specific perception models assumed by previous works.

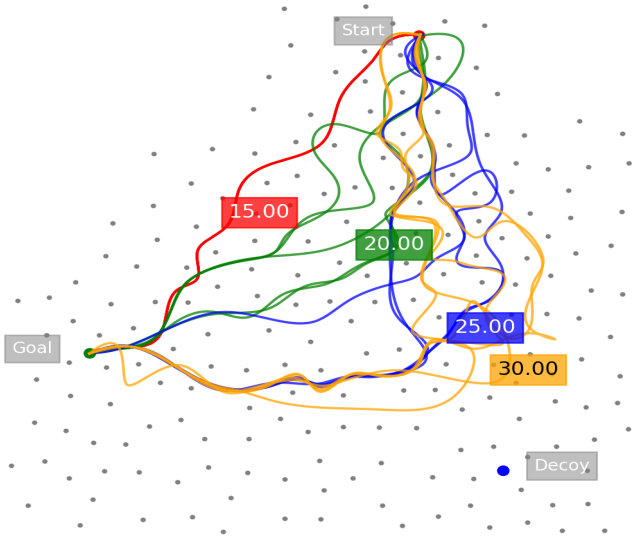

Figure 1: After training on only six small gridworld DPP problems, our GNN-equipped RL agent performs tunably deceptive navigation through a never-before-seen, continuous forest environment using only local perception. Deceptiveness is achieved through exaggeration towards a decoy goal, tuned by allowing $T_{max} = 15, 20, 25, 30$ additional steps of "time-to-deceive" before reaching the goal.

The field of model-free reinforcement learning (RL) [21] has seen incredible growth in recent years. Methods such as deep Q-learning (DQN) [14], deep deterministic policy gradient (DDPG) [11], proximal policy optimization (PPO) [20], and soft actor-critic (SAC) [7] have achieved impressive performance on a wide range of challenging control problems. Graph neural networks (GNNs) are a class of neural network architectures consisting of repeated composition of graph convolutions and pointwise nonlinearities. Due to their invariance, stability, and transferability properties [6, 17, 18], GNNs are particularly well-suited to problems where generalization and scalability to unseen, large graphs are critical, and have notched impressive practical successes [1, 3, 9].

In this work, we propose a novel RL scheme leveraging GNNs that, after training on a small set of simple problems, produces agents that can perform tunably deceptive DPP in complex, previously unseen environments. This addresses an open problem in the DPP literature and lays the groundwork for performing DPP in real-world scenarios. The core of our approach is the introduction of a local perception model for the agent, a new state space representation of the DPP problem, the use of GNN-based policies for

Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), N. Alechina, V. Dignum, M. Dastani, J.S. Sichman (eds.), May 6 – 10, 2024, Auckland, New Zealand. © 2024 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

generalization and scaling, and the introduction of new deception bonuses to the RL setting. Complete details are provided in the full paper [5] and the code used for our experiments is available at [4].

2 PROBLEM FORMULATION

Graph-based Environment Model: We represent the environment to be navigated as an undirected, weighted graph $\mathcal{H} = (\mathcal{S}, \mathcal{A}, c)$, where $\mathcal{S}$ is the set of states, or nodes, in the graph, $\mathcal{A} \subset \mathcal{S} \times \mathcal{S}$ is the set of edges representing accessibility between nodes, and $c : \mathcal{A} \to \mathbb{R}$ is the edge weight mapping. For a fixed, prespecified integer $k \geq 0$, define the k-hop neighborhood of $s \in \mathcal{S}$ by $\mathcal{N}_k(s) = \{s' \in \mathcal{S} \mid d_{\mathcal{H}}(s, s') \leq k\}$, where $d_{\mathcal{H}}(s, s')$ is the shortest number of edges that must be traversed to move from s to s' in $\mathcal{H}$. These local neighborhoods form the basis for our agent's perception model: when the agent is at state s, the region $\mathcal{N}_k(s)$, or visibility graph, is visible to it. We associate with each s_t the vector v_t of node attributes, $\left[\mathbf{1}_{visited}(s_t) \;\; d_c(s_t, G_1) \;\; \ldots \;\; d_c(s_t, G_{|\mathcal{G}|}) \;\; T_{max} - t\right]^T$, where $\mathbf{1}_{visited}(s_t) = 1$ if the agent has previously visited s_t and 0 otherwise, $d_c(s_t, G_k)$ is the minimum distance path from s_t to goal $G_k \in \mathcal{G}$, for $k \in \{1, \ldots, |\mathcal{G}|\}$, and T_{max} is a user-specified maximum number of allowable steps by which the agent should reach G^* during an episode. A policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ maps states to probability distributions over $\mathcal{A}$. Let $\mathcal{G} \subset \mathcal{S}$ denote the set of potential goals and let $G^* \in \mathcal{G}$ denote the agent's true goal. Let $\zeta = (s_1, a_1, s_2, a_2, \ldots)$ denote a trajectory of state-action pairs and $\zeta_{1:T}$ a partial trajectory of length $T \in \mathbb{N}$. Building on the foregoing, [19] proposes an observer model enabling the agent to predict the observer's belief about its true goal, given its partial trajectory $\zeta_{1:T}$ up to time T. Specifically, the model provides a probability distribution $P(G|\zeta_{1:T})$ over all possible goals $g \in \mathcal{G}$. We adopt this observer model in our work. See [19] and [5] for details.

Deception Bonuses: The two most common types of deception considered in the DPP literature are exaggeration and ambiguity [13, 16, 19]. In the exaggeration setting, the agent misleads the observer about its true goal by navigating towards a decoy goal instead. In the ambiguity setting, the agent misleads the observer by selecting paths that remain noncommittal regarding the true goal for as long as possible. As in previous works, we consider the exaggeration and ambiguity notions of deception. We define our exaggeration bonus to be $r_e(\zeta_{1:t}) = \max_{G \in \mathcal{G} \setminus G^*} \Pr(G|\zeta_{1:t}) - \Pr(G^*|\zeta_{1:t})$. For ambiguity, we consider the bonus $r_a(\zeta_{1:t}) = \sum_{G \in \mathcal{G}} \left(1 - \frac{|d_c(s_t,G) - d_c(s_t,G^*)|}{d_c(G,G^*)}\right)$.

To address the DPP problem using RL, we define a reward function that balances operating deceptively with respect to the deception bonuses with reaching G^* in a timely manner. For $r \in \{r_e, r_a\}$, we achieve this using reward function R_t defined by: $R_t = r(\zeta_{1:t})$ if $s_t \notin \zeta_{1:t-1}$, $R_t = 1$ if $s_t = G^*$, $R_t = -1$ if $t > T_{max}$, and $R_t = 0$ otherwise. Using this reward, we will train our agents to find a policy π maximizing the objective $J(\pi) = \mathbb{E}_\pi \left[\sum_{t=1}^{T} \gamma^{t-1} R_t\right]$, where $\gamma \in (0, 1)$ is a user-specified discount factor and the horizon T is the first timestep at which the agent reaches G^*.

3 METHOD

GNN Architecture: We use GNNs to infer the best deceptive action given the subgraph $\mathcal{N}_k(s_t)$ of the environment visible to the agent at each timestep t, as they exhibit high generalizability and applicability to complex environments [2, 22]. We use a k-layer GraphSAGE network [8], for various k, which is preceded by a linear layer to project the 4-dimensional node attribute vector associated with each $s \in \mathcal{N}_k(s_t)$ into a 64-dimensional intermediate feature space. The feature vector associated with state s is then updated by sampling several neighbors from $\mathcal{N}_k(s)$, and the process repeats for each layer of the network. See [5] for further discussion and ablation studies concerning the specific GNNs used.

Training Scheme: Using the reward and GNN architectures detailed above we trained two policies, one each for exaggeration and ambiguity, on a small but representative set of training environments: three 8×8 gridworlds and three 16×16 gridworlds. For simplicity, in this work we focused on the single-decoy goal setting and we assumed that the edge weights c of the graph $\mathcal{H}$ were all 1. We trained our policies using PPO on a variety of randomly generated configurations of start states s_1, true goals G^*, decoy goals G, and time limits T_{max} on each of the six environments. By training across a wide variety of T_{max} values, we found that we could control the level of "urgency" the model used when balancing reaching G^* with behaving deceptively.

4 RESULTS

We evaluated the performance of the agents trained using the methods described in the foregoing section on a variety of DPP problems over graphs. In all cases, the policies we trained were applied to previously unseen problems without any additional training or fine-tuning. We provide a high-level overview in this section; a complete description of the experiments can be found in [5] and the experimental framework is publicly available at [4].

The first set of experiments illustrates the ability of our DPP policies to generalize to previously unseen problems and to scale to larger problems (including 32×32 and 100×100 grids) than those encountered encountered during training. Both policies were able to effectively balance the incentive to move towards the true goal with their respective deception incentives. Importantly, equipped with only local observability of $\mathcal{N}_k(s_t)$ at each timestep, the policies were able to design paths whose deceptiveness is only perceptible at a scale beyond that allowed by the agent's limited, local perception model. Our second set of experiments demonstrates that the deceptiveness level of a policy can be tuned without retraining by dynamically altering the time constraint T_{max}. As T_{max} grows, the policy yielded progressively more deceptive motion. We emphasize that T_{max} may be altered at any time by the agent, enabling online tunability of deception level without the need for replanning. The third set of experiments (see Figure 1) evaluates the performance of our deceptive agents in a continuous "forest" navigation problem, where the objective is to navigate through a continuous, 2-D region populated by "tree" obstacles. These experiments further illustrate the tunability of our agents, as well as their ability to dynamically adapt to time-varying goal and decoy positions.

ACKNOWLEDGMENTS

M. Y. Fatemi gratefully acknowledges the support of the University of Maryland's 2023 National Security Scholars Summer Internship Program, a cooperative agreement with the U.S. Army Research Laboratory.

REFERENCES

[1] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. 2017. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine* 34, 4 (2017), 18–42.

[2] Hanjun Dai, Yujia Li, Chenglong Wang, Rishabh Singh, Po-Sen Huang, and Pushmeet Kohli. 2019. Learning Transferable Graph Exploration. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2019/file/afe434653a898da20044041262b3ac74-Paper.pdf

[3] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems* 29 (2016).

[4] Michael Y. Fatemi. 2023. https://github.com/myfatemi04/rl-deceptive-graph-planning.

[5] Michael Y. Fatemi, Wesley A. Suttle, and Brian M. Sadler. 2024. Deceptive Path Planning via Reinforcement Learning with Graph Neural Networks. arXiv:2402.06552

[6] Fernando Gama, Joan Bruna, and Alejandro Ribeiro. 2020. Stability properties of graph neural networks. *IEEE Transactions on Signal Processing* 68 (2020), 5680–5695.

[7] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*. PMLR, 1861–1870.

[8] William L. Hamilton, Rex Ying, and Jure Leskovec. 2018. Inductive Representation Learning on Large Graphs. arXiv:1706.02216 [cs.SI]

[9] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[10] Alan Lewis and Tim Miller. 2023. Deceptive Reinforcement Learning in Model-Free Domains. *Proceedings of the International Conference on Automated Planning and Scheduling* 33 (2023), 587–595.

[11] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).

[12] Zhengshang Liu, Yue Yang, Tim Miller, and Peta Masters. 2021. Deceptive Reinforcement Learning for Privacy-Preserving Planning. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*. 818–826.

[13] Peta Masters and Sebastian Sardina. 2017. Deceptive path-planning. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. 4368–4375.

[14] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529–533.

[15] Melkior Ornik and Ufuk Topcu. 2018. Deception in optimal control. In *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 821–828.

[16] Adrian Price, Ramon Fraga Pereira, Peta Masters, and Mor Vered. 2023. Domain-Independent Deceptive Planning. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*. 95–103.

[17] Luana Ruiz, Luiz FO Chamon, and Alejandro Ribeiro. 2023. Transferability properties of graph neural networks. *IEEE Transactions on Signal Processing* (2023).

[18] Luana Ruiz, Fernando Gama, Antonio García Marques, and Alejandro Ribeiro. 2019. Invariance-preserving localized activation functions for graph neural networks. *IEEE Transactions on Signal Processing* 68 (2019), 127–141.

[19] Yagiz Savas, Christos K Verginis, and Ufuk Topcu. 2022. Deceptive decision-making under uncertainty. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 5332–5340.

[20] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[21] Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction*. MIT Press.

[22] Chenning Yu and Sicun Gao. 2021. Reducing Collision Checking for Sampling-Based Motion Planning Using Graph Neural Networks. In *Proceedings of the 35rd International Conference on Neural Information Processing Systems*.