This article has been accepted for publication in IEEE Intelligent Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MIS.2024.3469574

A Q-learning Novelty Search Strategy for Evaluating Robustness of Deep Reinforcement Learning in Open-world Environments

Shafkat Islam*, Min-Hsueh Chiu†, Trevor Bonjour*, Ruy de Oliveira*, Bharat Bhargava*, Mayank Kejriwal†
*Purdue University, West Lafayette, USA
†University of Southern California/ISI, USA

Abstract—Despite substantial progress in deep reinforcement learning (DRL), a systematic characterization of DRL agents' robustness to unexpected events in the environment is relatively under-studied. Such unexpected events ("novelties"), especially those that are more structural than parametric, may significantly deteriorate the performance of DRL agents, leading them to be unfit for open-world environments and applications. However, not all novelties affect an agent's performance equally. Unfortunately, even with reasonable and constrained definitions of the problem, the space of all novelties can be (at least) exponential. Hence, an effective search strategy is required to find novelties that can adversely affect the agent. This paper presents a formalism for this problem and proposes a deep Q-learning-based novelty-search strategy that efficiently and systematically finds candidate (potentially complex) novelties with significant negative impact on a DRL agent. We conduct a detailed set of experiments in a stochastic multi-agent game environment (Monopoly) with complex decision-making properties.

While deep reinforcement learning (DRL) has led to enormous practical advances, it has also been prone to problems such as lack of explainability, inaccurate confidence estimations, bias, and adversarial attacks that humans are normally robust to. Although some of these problems continue to be heavily researched, a practical problem that has not received adequate attention is *open-world robustness*. This problem arises because the DRL agents are frequently applied 'outside the lab' in domains where unexpected events and violations of assumptions can occur with non-trivial frequency. While 'novelties' [1], [2] that are distributional (e.g., *concept drift*, where the test distribution diverges from the training distribution to such an extent that the system's performance becomes unreliable) or anomalous (e.g., anomaly and outlier detection, common in time-series applications) have been widely studied, far less study has been devoted to novelties that are *structural*.

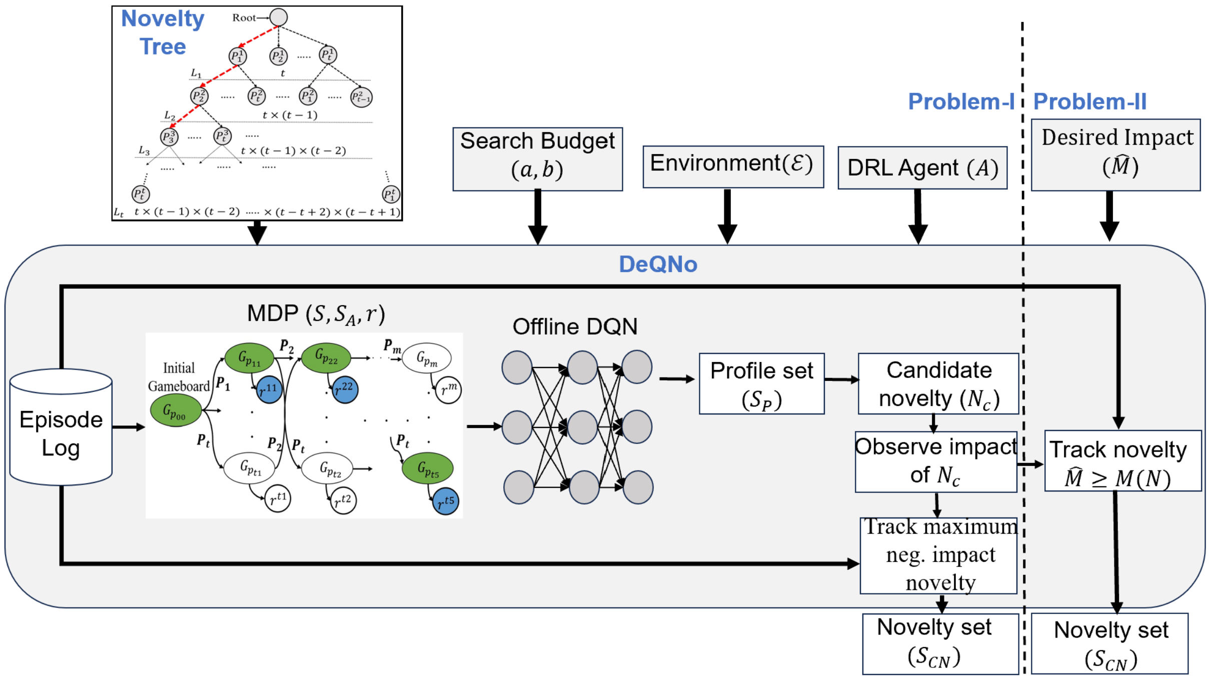
XXXX-XXX © IEEE
Digital Object Identifier 10.1109/XXX.0000.0000000

Structural novelties, and other such open-world elements, can degrade the performance of DRL agents deployed in critical operational domains, such as autonomous driving, military planning, and industrial IoT. In the extreme case, such performance degradation can even cost human lives [3]. Hence, evaluating the performance of DRL in the presence of structural novelties is crucial to characterize its robustness in the event of an unexpected change during its operation. However, such performance evaluation may consume substantial resources because the evaluation of each novelty requires the DRL agent to interact with the environment. Moreover, the novelty space of an environment can become exponentially large even with reasonable constraints. Intuitively (and as formalized subsequently), given n such *novelty primitives* that can be composed into arbitrarily more complex novelties, the space of all possible novelties, even without repetition, is combinatorially explosive.

Note that the impact of such a compound novelty is not (necessarily) the sum of the impacts of primitives, although there is likely some association that could

Authorized licensed use limited to: Purdue University. Downloaded on July 13,2025 at 23:05:16 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

(a)



(b)

FIGURE 1. (a) Wild boars are crossing a street in a city during Covid-19 pandemic (left), traffic light melted in heat wave (right); (b) An illustration of the two novelty search problems described in the main text, and the key elements in the proposed DeQNo approach for addressing them in a unified environment-agnostic manner.

be learned. Another key element here is that not all novelties are of interest: rather, we seek novelties that have significant adverse impact on the DRL agent, to better understand the safety-envelope and robustness properties of the agent, when navigating that environment under open-world assumptions. Finally, any such novelty search strategy should be conceptually agnostic of the particulars of the environment, although the implementation might depend on such particulars. Hence, the goal is to develop a systematic yet efficient strategy to find distinct complex structural novelties that can negatively impact the performance of an interactive state-of-the-art agent (typically trained using DRL) in an open world version of the environment. Fig. 1a illustrates that wild boars are crossing a street[1](left) and traffic light melted in heat wave[2] (right). These are novel incidents that occurred during the covid pandemic and summer heat wave accordingly. An autonomous intelligent vehicle trained before the covid era (or without summer heat wave data) may strug-

[1]https://www.cbc.ca/news/world/photos-wildlife-animals-take-to-streets-as-people-take-shelter-indoors-1.5519538
[2]https://www.dailyo.in/news/britain-is-melting-and-the-pictures-prove-it-36672

Authorized licensed use limited to: Purdue University. Downloaded on July 13,2025 at 23:05:16 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Intelligent Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MIS.2024.3469574

gle to identify this novel incident accurately since the training data may not have such samples. Therefore, analyzing the robustness of AI agents in the presence of novelty is imperative.

This paper proposes a novel deep Q-network-based novelty search strategy (called `DeQNo`) for DRL agents that addresses the challenges noted above. The search strategy is presented as a solution to a formalized version of the structural novelty search problem, under reasonably defined 'budget' constraints, that is subsequently detailed. Specific contributions that we make in this paper include:

- We formalize the novelty search problem by depicting the novelty space as a multi-level tree (to capture realistic open-world 'wilderness') and imposing a search budget to make the search problem practical.
- We propose a deep Q-network (DQN) based systematic novelty search strategy called `DeQNo` that can learn from the novelty encoding in the environment (e.g., a gameboard) as well as individual novelty impacts on the DRL agent. `DeQNo` is designed to be generalizable (with appropriate modification) to any environment or agent, as it does not conceptually rely on the particulars of either.
- We conduct a detailed set of experiments by comparing the performance of `DeQNo` with four other baseline methods, including another deep neural network approach, on the complex multi-agent game of Monopoly. We also conduct rigorous statistical significance testing, to demonstrate the competitiveness of `DeQNo`.

Motivation

Given the recent multi-domain successes of deep reinforcement learning (DRL), assessing the robustness of DRL under open-world assumptions (where unexpected events or novelties can occur with non-trivial frequency) has been recognized as important [4]. Following this path, [5] discuss both important challenges and potential solutions to be addressed by RL systems aiming for real-world deployment. [6] surveyed the literature on more robust approaches to RL, whereas [7] implement benchmarks to evaluate the performance of DRL agents in autonomous driving applications. However, none of these works study the robustness of DRL when novelty is permitted during or after deployment (but are unseen during training).

Novelties that are structural in scope and may have high impact on an agent are hard to anticipate in advance since they may arise from multiple open-world causes. On the other-hand, adversarial attacks are deliberate modifications in the system or environment that degrade an agent's performance. Both intent and outcome distinguish novelties from adversarial attacks; particularly, novelties can arise from exogenous causes in the environment, and their impact can be uncertain (some novelties may have no impact, or in the case of 'bonus' novelties, even lead to better performance).

There has been much focus on improving an agent's performance, as well as on adversarial attacks, and assessing adversarial robustness in literature. Our work serves as a useful complement to these works. However, our focus in not adversarial intent or attacks but on changes that could reasonably occur in the open-world. Such changes are known to happen in complex applications and, as noted in *Introduction*, can even lead to loss of lives. Yet, systematic discovery of structural changes with high negative impact under practical constraints is an important and under-addressed problem in the literature.

In most instances in the literature, novelties assumed by the agents tend to be more distributional than structural [8], [9]. Even if this were not the case, adapting an agent to novelty requires having control over the agent and its training process [10]. This is an unrealistic assumption in most industrial (or even military) applications, where the agent is treated as a black box and cannot be modified due to trade secrets or other security concerns [11]. This paper treats the agent as a black box and instead focus on understanding which novelties can affect its performance, rather than modifying or adapting it. However, our work is complementary to the novelty detection and adaptation literature, since having a better understanding of these novelties can then be used to better re-train the agent.

The experiments in this paper use the game of Monopoly [12], which is a hard game to model both because of its complexity and its stochasticity, which makes the state and action space high-dimensional. While not much work has been done in this direction [13], [14] compared to some other domains, smart DRL-based agents to play Monopoly were recently proposed by [15]. We use an open-source Monopoly gameboard [12] and DRL agent [15] as our environment and agent, respectively, to facilitate replication. However, the novelty search strategy proposed in this paper can be used in diverse applications, and is conceptually agnostic of the particulars of the environment and agent as long as some standard (subsequently described) assumptions are known to hold. The other single or multi-agent open-world environments, such

Authorized licensed use limited to: Purdue University. Downloaded on July 13,2025 at 23:05:16 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Intelligent Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MIS.2024.3469574

as MineDojo[16], MAESTRO[17], or Craftax[18], are customizable for diverse goals. The proposed novelty search strategy can be applied to any of these environments with a fixed goal setting, a well-defined goal evaluation metric, and a consistent definition of novelty (according to the subsequent section) for the environment.

Problem Formulation

Before describing the novelty search problem, we begin by introducing the key terminologies used in this paper. As is standard in the DRL literature, we assume that a DRL *agent* $\mathcal{A}$ interacts with an *environment* $\mathcal{E}$. Given a state s of $\mathcal{E}$, the agent $\mathcal{A}$ performs actions to maximize its reward or to reach the goal state. We assume that the interaction between $\mathcal{E}$ and $\mathcal{A}$ has a predefined endpoint (or goal state).

An *episode* d is a sequence of states and actions that begins from the initial state of $\mathcal{E}$ and terminates when $\mathcal{A}$ reaches the goal state in $\mathcal{E}$. Note that if d does not terminate after a finite pre-specified amount of time, it is forcibly terminated. A *trial* D is defined as a set of episodes. Finally, each episode is characterized by a *performance metric*. In games, for example, the performance metric can be set to 1 if the target agent wins, and 0 otherwise. The performance metric for a trial D can be similarly defined as the *win ratio* w or fraction of episodes ('games') in D that the target agent won. Unlike the binary episode-based performance metric, the trial-based win ratio can therefore take values in $[0, 1]$. In this paper, when computing w for a trial, we consider both the forcibly[3] and the conventionally terminated episodes. We assume that there is a way to decide the winner even when the game has been forcibly terminated. In a game like Monopoly, for example, the agent with the highest net-worth (right before forcible termination) is the winner.

A *novelty primitive* (henceforth, *primitive*) is a function $\mathcal{P}$ that, when *instantiated* and applied to E, results in a changed environment $\mathcal{E}'$. By instantiation, we mean that $\mathcal{P}$ must take one of the finite[4] C_P parameters (where $C_P \geq 1$ and depends on $\mathcal{P}$) before it can be applied. For instance, in Monopoly, an example of $\mathcal{P}$ would be changing the mortgage interest rate of all properties from 0.1 to one of $\{0.2, 0.3, \ldots, 1.0\}$. Once instantiated with one of these values, and applied to the default Monopoly gameboard ($\mathcal{E}$), the instantiated primitive P_n will alter the gameboard to reflect the new mortgage rate ($\mathcal{E}'$). For ease of terminology, we assume that each of the (finite) C_P parameters is indexed by an integer[5] n and denote the instantiated version of a primitive $\mathcal{P}$ as P_n.

Given a pre-defined set $\mathcal{P}$ of H primitives $\{\mathcal{P}^1, \ldots, \mathcal{P}^H\}$, each with its own C_{P^i} instantiations, we can expand $\mathcal{P}$ into a (still finite) set $\mathcal{P}'$ consisting of $\sum_i C_{P^i}$ instantiated primitives. Suppose that the total number of such instantiated primitives (i.e., the size of $\mathcal{P}'$) is denoted as t. To take a simple example, given 10 primitives, each with 15 possible instantiations, $\mathcal{P}'$ would consist of 150 (= t) instantiated primitives. Using this terminology, we define a *novelty* ($\mathcal{N}$) as a non-empty sequence of instantiated primitives, without any instantiated primitive being repeated in the sequence. Note, however, that different instantiations of the same primitive are permitted. A single instantiated primitive is also a novelty, as it can be placed in a sequence of length 1.

Because of finiteness, and the non-repetition constraint noted above, the set or 'space' of all possible novelties is necessarily finite. We can represent this space as a *novelty tree* $\mathcal{T}$ (shown in the top-left of Fig. 1b), constructed as follows.

The tree begins with an artificial root node at *level* L_0. Each node in the next level (L_1) is an instantiated primitive. For example, if there are 10 primitives, each with 10 instantiations, then L_1 would contain 100 nodes. Since we impose the repetition constraint on novelty formation, in level L_2, each node $\mathcal{P}^i_j$ in level L_1 becomes the parent of $t-1$ nodes, each of which represents an instantiated novelty primitive *except* $\mathcal{P}^i_j$. Hence, the total number of nodes (using the previous running example) in L_2 would be $100 \times 99 = 9900$. Similarly, higher level ($L_3, L_4, L_5, \ldots. L_t$) nodes in $\mathcal{T}$ are created. By construction, if there are t total instantiations, the last level in $\mathcal{T}$ would be L_t. By the definition of $\mathcal{N}$ (defined earlier), a path from the root to each unique node (except the root) in $\mathcal{T}$ represents a unique novelty.

Technically, two novelties or sequences of instantiated primitives $\mathcal{N}_1$ and $\mathcal{N}_2$ that are only different up to a permutation i.e., $Set(\mathcal{N}_1) = Set(\mathcal{N}_2)$, will likely show

[3]Removing episodes with forcible termination from a trial will apply in some situations. The treatment in this paper would not change substantially.

[4]We leave an exploration of infinitely parameterized primitives for future work, as it would require a fundamentally different (non-discrete) search framework.

[5]Although the example just used one 'argument', the same formalism applies to arbitrary but finite arguments, each taking finite values. C_p is then determined by the cross-product of these argument-sets. Furthermore, in the special case where the primitive only takes one argument (including the empty set), $n = 1$.

Authorized licensed use limited to: Purdue University. Downloaded on July 13,2025 at 23:05:16 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Intelligent Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MIS.2024.3469574

no practical difference in most real-world domains, as the order in which an instantiated primitive is injected is unlikely to matter (if they are all injected at the beginning of an episode, as is assumed here). Hence, we consider such novelties to be *invariant* with respect to each other. To ensure that novelty search is non-trivial, we henceforth represent the novelties as sets rather than sequences.

Injecting P_n or $\mathcal{N}$ in $\mathcal{E}$ may cause a change in the performance of $\mathcal{A}$. A central goal of this paper is to discover novelties that can cause significant changes. We term such performance deviation as the novelty's *impact*. While there are several ways to define impact, one reasonable metric is to compute the *relative win-ratio difference* $M(\mathcal{N})$. The $M(\mathcal{N})$ definition depends on the agent's goal. Since the metric measures relative difference, changing $M(\mathcal{N})$ based on the goal will not affect the search strategy. This metric quantifies the *relative* change in win rate as the result of injecting $\mathcal{N}$. We define $M(\mathcal{N})$ at the level of a trial, rather than an episode, to deal with stochastic effects that are usually prevalent in real domains subject to the open-world assumption. Recall that we had earlier defined the notion of a trial-based performance metric (e.g., win ratio w). Let us denote as w_{pre} the win ratio of the DRL agent $\mathcal{A}$ for a trial D where novelty has *not* been injected into any episode in D. Unless the domain or agent is highly unstable, w_{pre} will usually be constant (at least in expectation) for a given environment-agent combination. Next, given a novelty N, we can inject it at the beginning of *every* episode in a trial D, and calculate a 'post-novelty' win ratio w_{post} that represents the performance of $\mathcal{A}$ when exposed to the specific novelty $\mathcal{N}$. Unlike w_{pre}, w_{post} will depend on the novelty $\mathcal{N}$. With these notions in place, the impact $M(\mathcal{N})$ is simply given by $\frac{w_{post} - w_{pre}}{w_{pre}}$. Note that, if the performance metric (w) is constrained to lie between $[0, 1]$, $M(\mathcal{N})$ has lower bound of -1, which would be achieved for novelties with extreme negative impact on the performance of $\mathcal{A}$.

To capture the resource-constrained nature of any practical novelty search strategy, we introduce a *search budget* (a, b). The search budget is defined by an interaction budget (a) and an injection budget (b). The *interaction* budget value is the number of trials that $\mathcal{A}$ is allowed to interact with ('play in') the $\mathcal{E}$ during novelty search (e.g., to know the empirical impact of a novelty on $\mathcal{A}$). The *injection* budget value is defined as the total number of instantiated primitives that the search algorithm is allowed to use, regardless of the number of trials. In practice, search budget can be considered as the computing resource (for calculation), and the interaction frequency between $\mathcal{A}$ and $\mathcal{E}$. With this formalism in place, two important novelty search problems, both schematized in Fig. 1b, can be stated:

Problem I (Maximal Negative Impact): Given the DRL agent $\mathcal{A}$, the environment $\mathcal{E}$, a search budget (a, b) and the novelty tree $\mathcal{T}$, find a set S_{CN} within the search budget such that each novelty $\mathcal{N} \in S_{CN}$ has maximal (and equal) negative impact on $\mathcal{A}$.

Problem II (Desired Impact Novelty): Given the DRL agent $\mathcal{A}$, the environment $\mathcal{E}$, a desired impact $\hat{M}$, a search budget (a, b) and the novelty tree $\mathcal{T}$, find a set S_{CN} of novelties within the search budget such that each novelty $\mathcal{N} \in S_{CN}$ has an equal or higher negative impact $M(\mathcal{N})$ than $\hat{M}$ i.e., $\hat{M} \geq M(\mathcal{N})$.

Both the problems are expected to output a novelty set S_{CN} comprising distinct novelties, where each novelty satisfies the desired impact constraint. S_{CN} can be empty if none of the novelties satisfies the constraints; an issue that can arise for Problem II. The *maximal negative impact* problem is important because it challenges the search algorithm to find the 'safety envelope' of an $\mathcal{A}$. Different search strategies can be evaluated on Problem I by assessing the maximal impacts of the novelties they were able to find within the budgetary constraints. However, in some applications, a user may want to specify their own desired impact, rather than leave it to the search algorithm to find extreme cases. We formulate the *desired impact novelty* problem to address that issue. The value of the desired impact ($\hat{M}$) can vary based on $\mathcal{A}$'s objective or application. Since it is not possible to observe the impact of each specific novelty within a reasonable search budget, it is important for a search strategy to find as many novelties as possible that satisfy the constraints.

Deep Q-network with novelty impact and encoding (DeQNo)

In this section, we describe the deep Q-learning based systematic novelty search approach DeQNo for solving the two problems stated earlier. Earlier, Fig. 1b illustrated the key elements of the approach. Before describing these elements, we note that the DQN used in DeQNo is different from the DQN used in training the (DRL) agent $\mathcal{A}$. The latter yields an (e.g., gameplaying) agent that is trained to play well against other agents in the 'default' environment ($\mathcal{E}$), one that does not have novelties. The former is our proposal to efficiently address the novelty search problems. We use DQN for novelty search since the budget constrain limit the amount of training data for the search, and the convergence of DQN requires less amount of data compared to other methods (i.e., actor-critic, Muzero or EfficientZero).

Authorized licensed use limited to: Purdue University. Downloaded on July 13,2025 at 23:05:16 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Intelligent Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MIS.2024.3469574

Q-learning [19] is a model-free reinforcement learning which can be used to search for instantiated primitives that compose novelties with a substantial negative impact on agent $\mathcal{A}$. In Q-learning-based search, we define each novelty formation as an epoch. We define the state, action and reward of DeQNo as follows.

State Space: Injecting an instantiated primitive causes a change in the gameboard; thus, we represent each such modification in the gameboard as an instance of the state. Therefore, we can define the state space (S) of Q-learning as, $S = \{G_{p_{00}}, G_{p_{11}}, G_{p_{22}}, ..., G_{p_{t5}}\}$ where $G_{P_{ns}}$ represents gameboard with P_n injected at the s^{th} state and thus $G_{p_{00}}$ represents the default gameboard with no P_n injected. In this paper, we only consider novelties with maximum level of five (denoted as L_5) in $\mathcal{T}$. Under this constraint, the state space for DeQNo also reduces till the $G_{P_{t5}}$. Here, $G_{P_{ns}}$ is a tuple of instantiated primitives from P_1 to P_t. The value of P_n can either be 0 or 1. We term such type of state definition as novelty encoding. Here, the value $P_n = 1$ represents that the P_n^{th} instantiated primitive is injected in the game-board and 0 represents the opposite. Hence, at the initial state s_0, the value of all P_n is 0 which represents the original gameboard ($G_{p_{00}}$). Since we are limiting the novelty search up to L_5 in the novelty tree (fig. 1b), we define the goal state for DeQNo as when any five of the instantiated primitive's value will become 1.

Action Space: At each state DeQNo chooses one of the instantiated primitives to inject in the gameboard. By the definition of novelty (defined in 'problem formulation'), repetition of an instantiated primitive is not allowed within a particular novelty ($\mathcal{N}$). Thus, the action space of DeQNo at the initial state is t which reduces by 1 at each next state until the agent reaches the goal state. Hence, we define the action space ($\mathcal{S}_\mathcal{A}$) as, $\mathcal{S}_\mathcal{A} = \{P_1, P_2, P_3, ..., P_t\}$ where the search agent chooses any of these instantiated primitives (without repetition during a novelty formation) at each state and inject it in the game-board. Since the size of state-action pair ($S \times \mathcal{S}_\mathcal{A}$) increases considerably with the increase in t we use a deep neural network-based Q-function approximator.

Reward: We define the reward (r) as, $r = \alpha(-\beta M(\mathcal{N}) - M(P_n))$ where $M(\mathcal{N})$ and $M(P_n)$ are the win ratio differences after injecting the novelty ($\mathcal{N}$) and the instantiated primitive (P_n) accordingly. Here α and β are the reward scaling and reward balancing parameters accordingly. Reward motivates the DeQNo to select the P_n that itself and as the component of a $\mathcal{N}$ poses a negative impact on $\mathcal{A}$. Since we are interested in finding novelties (instead of each instantiated primitive) that have a negative impact we set the value of β at 2. This inspires DeQNo to give more emphasis on choosing instantiated primitives that as a component of novelty poses a negative impact. Since the value of win ratio difference can be within the range of fractions (depending on applications), irrespective of injecting any P_n or $\mathcal{N}$, we set the value of α at 10 to make the reward at each state distinctive.

Fig. 1b also showed the Markov decision process (MDP) for DeQNo. The green ovals represent observed states whereas blue circles represent the observed rewards from taking the actions. Initially, the environment does not have any instantiated primitive (P_n) injected. The DeQNo method chooses P_1 instantiated primitive as the action at its current state and then transitions to state $G_{P_{11}}$ (and gets reward r^{11}). At that state, the DeQNo has $t-1$ actions available to it, of which it takes action P_2 and transitions to $G_{P_{22}}$ (getting reward r^{22}).

Because the definitions of the novelty search problems impose a budgetary limit on the interaction between $\mathcal{A}$ and $\mathcal{E}$, we adopt an offline training strategy for DeQNo. Specifically, DeQNo collects training data by taking random actions at each state, and logging the episode results by injecting randomly sampled novelties. It is allowed to spend at most 50% of the budget value to collect this data. Following this step, the deep neural network-based Q-function approximator is trained offline[20], using the training samples (by extracting state, action, and reward values from the logged episodes), to avoid using up the budget. As shown in Fig. 1b, DeQNo prepares a profile set (S_P) of instantiated primitives (approximately top 50%) at each level as candidate novelties up to a certain level.

To discover a good set of novelties required for the two problems, DeQNo also trains a linear regression model to predict the win ratio difference of a particular novelty ($\mathcal{N}$) while the input to the model is the individual win ratio difference of each instantiated primitive (P_n) that forms the novelty. The regression model uses the episode logs collected initially as the training samples. DeQNo forms a candidate novelty set (N_C) by taking instantiated primitives from the profile set of S_P, and querying the impact of that novelty in the regression model. Finally, DeQNo observes the impact of each candidate novelty on $\mathcal{A}$ and tracks the novelty that satisfies the search condition. The final output considers both the observed results as well as the initial episode logs and outputs the final novelty set (S_{CN}).

Experiments

We conducted a detailed set of experiments to evaluate the merits of DeQNo on both problems formalized

Authorized licensed use limited to: Purdue University. Downloaded on July 13,2025 at 23:05:16 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Intelligent Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MIS.2024.3469574

earlier in *Problem Formulation*. We begin by describing the environment, agent, and novelty primitives that serve as inputs to any solution to these problems (as also shown on the top in Fig. 1) and that form a common basis to all of the results. We follow this with a brief description of the baselines used to evaluate `DeQNo` on both problems.

Environment (Monopoly): We use an open-source Monopoly gameboard [12] as the environment for all experiments. There are several reasons why we chose Monopoly for these experiments. First, it is one of the few open-source implementations of a complex game that supports novelty. Second, it supports multi-agent plugins. Third, it is a stochastic game where the action space for an agent is significantly large, allowing us to test the proposed framework in a rich enough environment.

Target agent (based on DRL): As our primary objective is novelty search and not the design of an 'optimal' game-playing agent, we use a state-of-the-art, previously published DRL agent [15] already known to work well with the Monopoly gameboard described above, and that can be used to replicate these experiments. Each Monopoly episode is played among four players of which one is the DRL agent and the rest of the three players are rule-based agents. The rules underlying these agents are based on documented tournament winning strategies[6] from the Monopoly community, e.g., preferring railroads over utilities due to their multiplicative reward structure. The win ratio of the DRL agent ($\mathcal{A}$) in the default gameboard (determined by computing the win ratio for a 100-episode trial without novelty) was found to be 0.698. Hence, the *pre-novelty* win-ratio w_{pre} for the DRL agent is set to this constant value when calculating the impact metric $M(\mathcal{N})$, the formula for which was provided earlier in *Problem Formulation*. The impact metric $M(\mathcal{N})$, was defined earlier as well and is motivated by the desired goal to test an agent's robustness by finding novelties that would significantly affect its performance.

Recall that $M(\mathcal{N})$ can be positive, negative, or zero. A negative $M(\mathcal{N})$ implies that injecting $\mathcal{N}$ into the gameboard negatively affects the performance of DRL relative to w_{pre} (= 0.698) in these experiments. A positive value indicates that $\mathcal{N}$ should be thought of as a 'bonus' novelty, and 0 implies that the novelty had no effect.

Novelty Primitives: The two novelty search problems of interest to us assume novelty primitives and their (finite) instantiations to be provided as inputs. For the Monopoly gameboard, we defined a set of 59 distinct structural primitives, and a total of 1554 instantiations spanning these primitives. Novelty can be formed by combining instantiated primitives from these 1554 instantiations without any repetition. In these experiments, we constrain the search to find novelties between L_2 to L_5. Even with this constraint, the space of all possible novelties is more than millions, demonstrating the difficulty of this search problem. We set the number of episodes in each novelty-injected trial at 30. The novelty is injected into the gameboard at the beginning of every episode in a trial. Recall that each episode is an independent game of Monopoly played from the beginning. Primitives are further described in the Appendix.

Baselines: We used four diverse baselines, each representing an implemented novelty search strategy, for evaluating `DeQNo`:

1) *Linear regression model with novelty impact* `(LiNI)`*:* This search strategy collects training data by randomly injecting novelties in the Monopoly gameboard and logs the win ratio difference of the DRL agent. The linear model uses each individual instantiated primitive's impact as an input feature and the win ratio difference of the novelty as the output label. This search strategy can utilize at most 50% of the budget to collect the training data and uses the rest of the budget to observe interaction based on the novelty set that is created from querying the linear model.
2) *Deep neural network with novelty encoding* `(DeNE)`*:* This strategy uses a deep neural network model to predict the win ratio difference of the DRL agent in the presence of a particular novelty $\mathcal{N}$, by using the novelty's encoding as the input feature. For instance, if there are t distinct instantiated primitives, the feature dimensionality of each data point is t. For each $\mathcal{N}$, the values of those instantiated primitives that form the novelty are 1 while the rest of the feature values remain 0. This strategy also uses 50% of the budget to collect the training data, and uses the rest of the budget for observing interactions.
3) *Random Sampling based Novelty Search* `(Ran)`*:* This method uses the entire budget to inject random novelties in the gameboard and logs the interaction results of the DRL agent. Novelties are output based on the logs.
4) *Stratified Random Sampling based Novelty Search* `(S-Ran)` This strategy also uses ran-

[6]Two resources include http://www.amnesta.net/monopoly/ and https://www.vice.com/en/article/mgbzaq/10-essential-tips-from-a-monopoly-world-champion.

Authorized licensed use limited to: Purdue University. Downloaded on July 13,2025 at 23:05:16 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Intelligent Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MIS.2024.3469574

dom formation of novelties. However, it uses the budget to form twice the L_4 and L_5 novelties compared to the L_2 and L_3 novelties. Since we assume that the complexity of novelty increases with the level, this method was designed to 'try' higher level novelties to better solve the problem.

Results

To evaluate the *maximal negative impact* problem (Problem-*I*), we first compare the performance of `DeQNo` with the four other baselines in finding the novelty set S_{CN} with maximal negative impact, for two different search budget settings: $(2000, 6000)$ and $(1800, 5400)$. In executing the methods, we observed that each of the search strategies was able to find novelties with the *maximum* impact (of -1); however, the size of the set S_{CN} differed for each method.

Fig. 2(a) illustrates the results for finding maximum impact novelty sets, S_{CN}. `DeQNo` outperforms the other search strategies for both budget settings. For instance, `DeQNo` found at least $1.19\times$ and $1.16\times$ as many maximum-impact novelties compared to `LiNI` and `DeNE`, respectively. Apart from `DeQNo`, `LiNI` and `DeNE` found almost equal numbers of maximum impact novelties for the budget value of $(2000, 6000)$, however, for the $(1800, 5400)$ budget `LiNI` found almost $1.73\times$ more maximum impact novelties than `DeNE`. In both cases, `Ran` and `S-Ran` found at most one-third of maximum impact novelties compared to `DeQNo`.

To further compare different strategies, we evaluated their performance in finding the novelty set S_{CN} with a minimum desired negative impact $\hat{M}$ (Problem-*II*). We considered a wide range of values for $\hat{M}$, such as $[-0.4, -0.5, -0.6, -0.7, -0.8, -0.9]$. With decrease in $\hat{M}$, the novelty can have a more significant negative impact on the DRL's performance. Hence, it is important for a search strategy to find as many novelties below the desired impact as possible.

Fig. 2(b) shows the results for different values of $\hat{M}$. We observe that at $\hat{M} = -0.9$ and search budget $(2000, 6000)$, `DeQNo` can find $1.09\times$, $1.45\times$, $3.75\times$, and $4.16\times$ as many novelties compared to `LiNI`, `DeNE`, `S-Ran` and `Ran`, respectively. In addition, for $\hat{M} = -0.7$, `DeQNo` can find $1.02\times$ and $1.2\times$ as many novelties compared to `LiNI` and `DeNE`. We observe that as the value of $\hat{M}$ increases, the performance difference between learning-based search strategies reduces; however, `S-Ran` and `Ran` become significantly worse even at $\hat{M} = -0.4$.

Effect of search budget (a, b)**:** Next, we evaluate the effect of search budgets on different strategies. We conduct experiments on four different search budgets for comparing the performance of `DeQNo`, `LiNI`, `S-Ran`, and `Ran`. Fig. 2(c) and 2(d) illustrate the experimental results for two different desired impacts, such as -0.7 and -0.9 respectively. We observe that in almost all cases the size of S_{CN} increases with the increase in budget value. At search budget of $(1800, 5400)$ and desired impact of -0.9, `DeQNo` can find $1.38\times$ as many novelties compared to `LiNI`, although the performance difference between the two methods reduces to $1.03\times$ when the budget reduces to $(1400, 4200)$. However, the performance of `S-Ran` and `Ran` continue to be noticeably worse compared to the learning-based strategies in all scenarios.

Effect of novelty levels: Finally, we evaluate both the performance of different search strategies in finding novelties at different levels, and the impact of these different-level novelties on the DRL agent $\mathcal{A}$. Fig. 2(e-h) illustrates the frequency of novelty levels in different novelty sets for search budget: $(2000, 6000)$. We observe that each learning-based search strategy (e.g., `DeQNo`, `LiNI`, `DeNE`) can find substantial novelties across levels L_2 to L_5, with $\mathcal{A}$ performance degrading below a specified desired impact $\hat{M}$. We also observed that L_4 novelties were most frequently the root cause of performance degradation for different desired impacts and budgets. We also observe that `DeQNo` found the most numbers of L_4 and L_5 novelties with minimum desired negative impact of -0.7, -0.9 and -1.0 for both search budget settings (i.e., $(2000, 6000)$ and $(1800, 5400)$). However, `LiNI` can find the most L_2 and L_3 novelties for all three desired impacts when using the $(2000, 6000)$ budget. For the $(1800, 5400)$ budget, `DeQNo` finds the most number of L_2 and L_3 novelties compared to others. With the increase in novelty primitives and their instantiations, the novelty level increases. We observe that increment in novelty level L_2 to L_5 does not affect the efficacy of the algorithm in finding novelties. Though we limit the search in the novelty tree at level 5, the algorithm can be effectively pluggable for finding novelties at higher levels.

Statistical Analysis: To statistically compare the performance of different search strategies we conducted both ANOVA and post hoc analyses among `DeQNo`, `LiNI`, `Ran`, and `S-Ran`. We collected six different data points for each of these methods (using different budget values) to conduct the tests. The degree of freedom for each test is 23. In the tests, the null hypothesis was that the means of two given methods are not different. We conduct the tests for three different desired impacts, e.g., $-1, -0.9$, and -0.7. For each of these three tests, we observed that the ANOVA p-value is less than 0.05 which implies that one or more groups in the tests are significantly

Authorized licensed use limited to: Purdue University. Downloaded on July 13,2025 at 23:05:16 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Intelligent Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MIS.2024.3469574

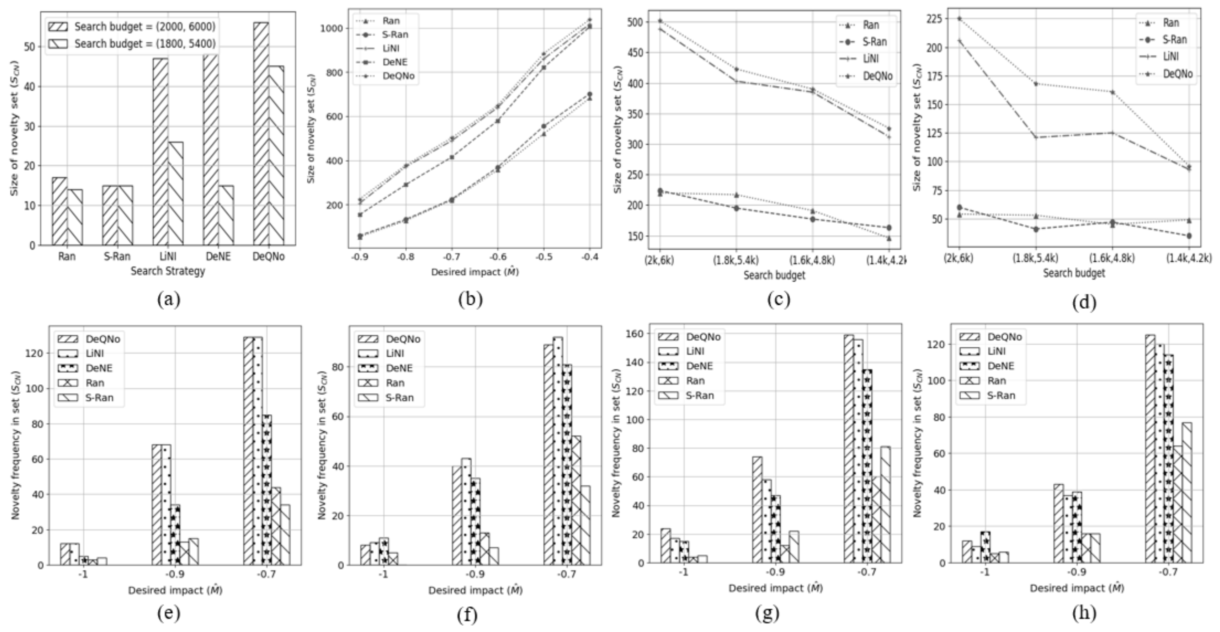


FIGURE 2. Performance of different search strategies, with different search budget, in finding (a) maximal impact novelties (*Problem* $-$ *I*), (b) novelties with a minimum desired negative impact (*Problem* $-$ *II*) at search budget $(2000, 6000)$. Performance of different search strategies, for different budget values, in finding novelties with a minimum desired negative impact (*Problem*$-$*II*) of (c) -0.7, (d) -0.9. Performance of different search strategies in finding novelty at different levels such as (e)L_2, (f)L_3, (g)L_4, and (h)L_5 at the search budget of $(2000, 6000)$.

different. Additionally, in the post hoc Tukey HSD test we found that `DeQNo` is significantly different than `Ran` and `S-Ran` with Tukey HSD p-values of 0.007 and 0.005 accordingly at the desired impact of -1. In the other two tests, we found that `DeQNo` and `LiNI` are both significantly different from `Ran` and `S-Ran`.

Conclusion

Structural novelties can significantly degrade the performance of a DRL agent, by surprising the agent with unexpected events during operation. Novelty can have many causes, making it difficult to fully anticipate, and even under reasonable constraints, the space of all possible novelties is combinatorially explosive. To better understand a DRL agent's robustness operating under open-world assumptions, it is important to discover high-impact novelties from the space in a systematic and efficient way. In this paper, we presented an environment-agnostic formalism for the novelty search problem, and a deep Q-network-based novelty search approach (`DeQNo`) for efficiently finding candidate sets of high-impact novelties. Experiments conducted against four baselines, using a complex Monopoly gameboard and a recently published DRL agent, demonstrate both the practical utility of the problem formulation and the efficacy of `DeQNo` in finding more novelties with greater impact. In future, we will investigate the impact of training data characteristics on the proposed `DeQNo`.

ACKNOWLEDGMENTS

This research is supported, in part, by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL) under the contract number W911NF2020003.

REFERENCES

1. T. Boult, P. Grabowicz, D. Prijatelj, R. Stern, L. Holder, J. Alspector, M. M. Jafarzadeh, T. Ahmad, A. Dhamija, C. Li, *et al.*, "Towards a unifying framework for formal theories of novelty," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, pp. 15047–15052, 2021.
2. T. Boult and W. Scheirer, *A Unifying Framework for Formal Theories of Novelty: Discussions, Guidelines, and Examples for Artificial Intelligence*. Springer Nature, 2023.
3. F. Siddiqui, "17 fatalities, 736 crashes: The shocking toll of tesla's autopilot," *The Washington Post*, 2023.

Authorized licensed use limited to: Purdue University. Downloaded on July 13,2025 at 23:05:16 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Intelligent Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MIS.2024.3469574

4. M. Kejriwal, A. Shrivastava, E. Kildebeck, B. Bhargava, and C. Vondrick, "Designing artificial intelligence for open worlds," *Proceedings of the AAAI Spring Symposium on Designing Artificial Intelligence for Open Worlds*, 2022.
5. G. Dulac-Arnold, N. Levine, D. J. Mankowitz, J. Li, C. Paduraru, S. Gowal, and T. Hester, "Challenges of real-world reinforcement learning: definitions, benchmarks and analysis," *Machine Learning*, vol. 110, pp. 2419–2468, Apr. 2021.
6. J. Moos, K. Hansel, H. Abdulsamad, S. Stark, D. Clever, and J. Peters, "Robust reinforcement learning: A review of foundations and recent advances," *Machine Learning and Knowledge Extraction*, vol. 4, pp. 276–315, Mar. 2022.
7. A. Sharif and D. Marijan, "Evaluating the robustness of deep reinforcement learning for autonomous policies in a multi-agent urban driving environment," in *2022 IEEE 22nd International Conference on Software Quality, Reliability and Security (QRS)*, IEEE, Dec. 2022.
8. Y.-C. Hsu, Y. Shen, H. Jin, and Z. Kira, "Generalized odin: Detecting out-of-distribution image without learning from out-of-distribution data," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10951–10960, 2020.
9. H. Mirzaei, M. Salehi, S. Shahabi, E. Gavves, C. G. Snoek, M. Sabokrou, and M. H. Rohban, "Fake it until you make it: Towards accurate near-distribution novelty detection," in *The Eleventh International Conference on Learning Representations*, 2022.
10. Z. Wang, L. Liu, Y. Duan, Y. Kong, and D. Tao, "Continual learning with lifelong vision transformer," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 171–181, 2022.
11. Z. Liu, J. Guo, K.-Y. Lam, and J. Zhao, "Efficient dropout-resilient aggregation for privacy-preserving machine learning," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 1839–1854, 2022.
12. M. Kejriwal and S. Thomas, "A multi-agent simulator for generating novelty in monopoly," *Simulation Modelling Practice and Theory*, vol. 112, p. 102364, 2021.
13. A. F. P. Bailis and I. Vlahavas, "Learning to play monopoly: A reinforcement learning approach," in *the 50th Anniversary Convention of The Society for the Study of Artificial Intelligence and Simulation of Behaviour*, AISB, 2014.
14. E. Arun, H. Rajesh, D. Chakrabarti, H. Cherala, and K. George, "Monopoly using reinforcement learning," in *TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON)*, pp. 858–862, 2019.
15. T. Bonjour, M. Haliem, A. Alsalem, S. Thomas, H. Li, V. Aggarwal, M. Kejriwal, and B. Bhargava, "Decision making in monopoly using a hybrid deep reinforcement learning approach," *IEEE Transactions on Emerging Topics in Computational Intelligence*, pp. 1–10, 2022.
16. L. Fan, G. Wang, Y. Jiang, A. Mandlekar, Y. Yang, H. Zhu, A. Tang, D.-A. Huang, Y. Zhu, and A. Anandkumar, "Minedojo: Building open-ended embodied agents with internet-scale knowledge," *Advances in Neural Information Processing Systems*, vol. 35, pp. 18343–18362, 2022.
17. M. Samvelyan, A. Khan, M. Dennis, M. Jiang, J. Parker-Holder, J. Foerster, R. Raileanu, and T. Rocktäschel, "Maestro: Open-ended environment design for multi-agent reinforcement learning," *arXiv preprint arXiv:2303.03376*, 2023.
18. M. Matthews, M. Beukman, B. Ellis, M. Samvelyan, M. Jackson, S. Coward, and J. Foerster, "Craftax: A lightning-fast benchmark for open-ended reinforcement learning," *arXiv preprint arXiv:2402.16801*, 2024.
19. V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.
20. K. Schweighofer, M.-c. Dinu, A. Radler, M. Hofmarcher, V. P. Patil, A. Bitto-Nemling, H. Eghbalzadeh, and S. Hochreiter, "A dataset perspective on offline reinforcement learning," in *Conference on Lifelong Learning Agents*, pp. 470–517, PMLR, 2022.

Shafkat Islam is currently pursuing a Ph.D. in Computer Science at Purdue University.

Min-Hsueh Chiu is working as a research engineer in AICS at the University of Southern California/ISI.

Trevor Bonjour is a PhD candidate in the computer science department at Purdue University.

Ruy De Oliveira is a visiting scholar in the computer science department at the Purdue University.

Bharat Bhargava is a Professor in the computer science department at the Purdue University.

Mayank Kejriwal is a Research Assistant Professor in the Department of Industrial and Systems Engineering, and a Principal Scientist in the USC Information Sciences Institute.

Authorized licensed use limited to: Purdue University. Downloaded on July 13,2025 at 23:05:16 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.