Adapting the Behavior of Reinforcement Learning Agents to Changing Action Spaces and Reward Functions

Raul de la Rosa
Universidad de los Andes
Bogotá, Colombia
c.delarosap@uniandes.edu.co

Ivana Dusparic
Trinity College Dublin
Dublin, Ireland
ivana.dusparic@tcd.ie

Nicolas Cardozo
Universidad de los Andes
Bogotá, Colombia
n.cardozo@uniandes.edu.co

***Abstract*—Reinforcement Learning (RL) agents often struggle in real-world applications where environmental conditions are non-stationary, particularly when reward functions shift or the available action space expands. This paper introduces MORPHIN, a self-adaptive Q-learning framework that enables on-the-fly adaptation without full retraining. By integrating concept drift detection with dynamic adjustments to learning and exploration hyperparameters, MORPHIN adapts agents to changes in both the reward function and on-the-fly expansions of the agent's action space, while preserving prior policy knowledge to prevent catastrophic forgetting. We validate our approach using a Gridworld benchmark and a traffic signal control simulation. The results demonstrate that MORPHIN achieves superior convergence speed and continuous adaptation compared to a standard Q-learning baseline, improving learning efficiency by up to 1.7x.**

***Index Terms*—Reinforcement Learning, Continual Reinforcement Learning, Q-learning, Concept Drift Detection, Adaptive Systems, Traffic Signal Control**

I. Introduction

Traditional Reinforcement Learning (RL) algorithms assume stationary Markov Decision Processs (MDPs), where transition probabilities and reward functions remain constant [1]. However, real-world environments often exhibit non-stationary characteristics like evolving reward dynamics and changing action spaces [2], limiting the effectiveness of standard approaches.

This paper introduces MORPHIN, a self-adaptive Q-learning algorithm that addresses two key challenges in non-stationary environments: (1) adapting to changing reward functions (goals), and (2) incorporating new actions into the agent's behavior. Our approach integrates concept drift detection using the Page-Hinkley test (PH-test)[3] with dynamic adjustment of learning (α) and exploration (ε) rates, enabling agents to explore for enough time in order to preserve prior knowledge [4] while adapting to environmental changes.

MORPHIN operates through two main mechanisms: *environment monitoring* for drift detection, and *adaptive learning* that dynamically adjusts parameters based on temporal difference errors. When concept drift is detected, the agent increases exploration until the agents stabilize over the new configuration while maintaining its Q-table structure, allowing rapid adaptation without catastrophic forgetting. For new actions, the agent extends its Q-table dimensions and applies targeted exploration to integrate new capabilities.

We validate our approach on Gridworld benchmarks with shifting goals and expanding action spaces, and demonstrate practical applicability in traffic signal control scenarios. Results show superior convergence speed and adaptation efficiency compared to standard Q-learning baselines, positioning MORPHIN as a suitable approach for self-adaptive systems operating in dynamic environments.

II. Background

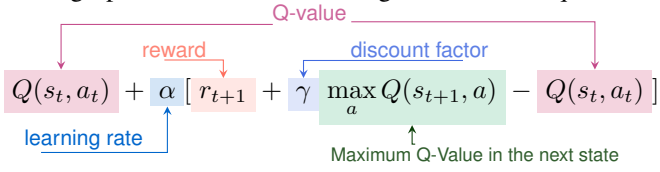
A. Reinforcement Learning in Non-Stationary Environments

RL agents learn optimal policies by interacting with environments formulated as MDPs $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle$, where $\mathcal{S}$ and $\mathcal{A}$ are the state and action spaces, P represents transition probabilities, R is the reward function, and γ is the discount factor [1].

Real-world environments often violate stationarity assumptions, leading to *non-stationary* MDPs represented as temporal sequences:

$$\{\mathcal{M}_t\}_{t=1}^{\infty} \quad \text{with} \quad \mathcal{M}_t = \langle \mathcal{S}, \mathcal{A}_t, P_t, R_t, \gamma \rangle$$

where changes in R_t and $\mathcal{A}_t$ constitute *concept drifts*, which refer to shifts in the environment's underlying dynamics. This implies that the agent's previously learned policy and Q-values may become suboptimal for maximizing future rewards. Q-learning updates action values using the Bellman equation:



While effective in stationary settings, standard Q-learning struggles when R_t or $\mathcal{A}_t$ change over time. Our work addresses this limitation through adaptive mechanisms that detect and respond to environmental changes in the form of modifications to the reward distribution function and action space expansions. We focus on rewards and action spaces, as changes in the state space has been addressed in the literature [5].

B. Motivating Example

We motivate our work using a 9×9 Gridworld benchmark, where agents navigate from the center of the board to a goal state (*i.e.,* a state with reward $+100$) while avoiding negative rewards (states with reward -1). The environment presents two adaptation challenges:

- Goal Relocation: The target switches between corners, from the top-left corner to the bottom right-corner. This requiring agents to adapt to new reward distributions without losing previously learned knowledge, as the goal changes multiple times. In each change, the agent should take less time in learning the behavior to reach the goal, than the required time to learn from scratch.
- Action Space Expansion: New actions (*e.g.,* jumping over obstacles) are introduced, requiring agents to integrate new capabilities into existing policies to further optimize the policy with the new actions.

III. RL Agents with Adaptive Behavior

This section introduces MORPHIN, a self-adaptive Q-learning framework that enables agents to dynamically adapt to non-stationary environments, specifically to changes in reward functions (goals) and on-the-fly expansions of the action space. MORPHIN integrates two core components, detailed in Algorithm 1: proactive environment monitoring using a concept drift detector, and an adaptive learning process that modulates exploration and learning rates to incorporate new knowledge while preserving prior experience. The implementation is publicly available.[1] This approach aligns with the principles of Continual Reinforcement Learning (CRL) [6] and Self-Adaptive Systems (SAS) [7] by enabling agents to autonomously modify their learning strategy at runtime, ensuring resilience in non-stationary contexts.

A. Environment Monitoring and Adaptation

To adapt, an agent must first recognize environmental changes. MORPHIN achieves this by monitoring the stream of cumulative episode rewards (R_{ep}) with the PH-test [3, 8, 9]. A drift is flagged if the cumulative difference between R_{ep} and its running mean exceeds a threshold H. The test's sensitivity is controlled by H and a second hyperparameter, δ, which were selected empirically based on the reward scale of each environment. As shown in Algorithm 1 (lines 21–28), a detected drift triggers an immediate adaptive response: the exploration decay counter e is reset to zero, forcing the exploration rate ε^* to its maximum value. This compels the agent to re-explore the environment and learn the new dynamics, as visualized in Fig. 1a.

Once a drift is detected, MORPHIN employs a two-points strategy. The first part is the mandatory re-exploration described above. The second is a dynamic adjustment of the learning rate α^* to control the speed of knowledge acquisition. This dual mechanism is a cornerstone of MORPHIN, designed to preserve prior knowledge and prevent catastrophic forgetting. Unlike approaches that retrain from scratch, MORPHIN never resets the Q-table; existing Q-values serve as an informed starting point for learning the new policy. This reuse of knowledge enables faster adaptation, as evidenced by the shorter re-learning periods showed in Fig. 1a.

[1] Available at: https://anonymous.4open.science/r/morphin_rl

Algorithm 1 MORPHIN: Adaptive Q-Learning with Concept Drift Detection

1: **Initialize** parameters:
2: Base learning rate α, max learning rate $\alpha_{\max}$, discount factor γ
3: TD-error sensitivity k, exploration decay parameters $\varepsilon_{\min}$, decay_rate
4: Page-Hinkley parameters: sensitivity δ, threshold H
5: **Initialize** Q-table $Q(s, a) \leftarrow 0$ for all $s \in S, a \in A$
6: **Initialize** drift detector $PH\_Test(\delta, H)$
7: **Initialize** exploration decay counter $e \leftarrow 0$
8: **for** episode = 1 to N **do**
9: Reset state $s_t \leftarrow s_{\text{initial}}$
10: Reset cumulative episode reward $R_{ep} \leftarrow 0$
11: **while** s_t is not terminal **do**
12: $\varepsilon_t \leftarrow \varepsilon_{\min} + (1 - \varepsilon_{\min}) \cdot \exp(-\text{decay\_rate} \cdot e)$
13: $a_t \leftarrow \text{choose\_action}(s_t, \varepsilon_t)$ ▷ ε-greedy selection
14: Execute a_t, observe r_{t+1} and s_{t+1}
15: $R_{ep} \leftarrow R_{ep} + r_{t+1}$
16: ▷ Adaptively update Q-value with eq. (1) & (2)
17: $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha^* \cdot TD_{error}$
18: $s_t \leftarrow s_{t+1}$
19: **end while**
20: ▷ Detect Concept Drift at the end of the episode
21: **if** $PH\_Test.update(R_{ep})$ is True **then**
22: ▷ Drift detected: reset exploration schedule
23: $e \leftarrow 0$
24: $PH\_Test.reset()$
25: **else**
26: ▷ Stable environment: continue exploration decay
27: $e \leftarrow e + 1$
28: **end if**
29: **end for**

The learning rate adaptation is driven by the Temporal Difference (TD) error, which quantifies the discrepancy between the predicted and actual outcomes of an action:

$$TD_{error} = r_{t+1} + \gamma \cdot \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \quad (1)$$

A concept drift leads to large TD-errors, signaling a mismatch between the agent's knowledge and the new reality. MORPHIN harnesses this signal to compute a dynamic learning rate α^*:

$$\alpha^* = \alpha + (\alpha_{\max} - \alpha) \cdot \frac{1}{1 + e^{-(|TD_{error}| - k)}} \quad (2)$$

Here, the hyperparameter k controls the sensitivity of α^* to the TD-error and was empirically tuned. High TD-errors yield a large α^*, accelerating learning. As the policy stabilizes, TD-errors diminish and α^* decays towards its base value α, ensuring convergence.

This framework also handles an expanding action space. When a new action a_{new} becomes available, the Q-table (a matrix of size $|A| \times |S|$) is dynamically augmented by adding a new row for a_{new}, initialized to zero. This event, detected as a drift, triggers the same unified response: ε^* is reset to explore the new action's utility, and α^* adapts to integrate it into the policy.

The effectiveness of MORPHIN becomes evident when contrasted with a standard Q-learning agent (Fig. 1b). The baseline agent's static exploration schedule prevents adaptation to new goals, causing a sustained drop in performance. In sharp contrast, MORPHIN (Fig. 1a) explicitly detects changes and coordinates exploration (ε^*) with learning speed (α^*) to rapidly converge to a new optimal policy after each drift.

IV. Validation

We evaluate MORPHIN in two scenarios: a canonical Gridworld benchmark for a clear proof-of-concept, and a more real application over traffic-signal control simulation to validate its adaptability in a scenario representative of real-world self-adaptive systems.

A. Evaluation Scenarios

Gridworld: In a 9×9 grid, we run two experiments for 1,000 independent trials each. First, in a 1,500-episode run, the high-reward goal state is swapped between opposite corners every 300 episodes. Second, in a 400-episode run, a new "jump" action is introduced at episode 300, requiring the agent to find a more optimal path.

Traffic-Signal Control: We model a two-lane intersection using a custom extended Gym environment [10], a well known application for SAS [11]. The state $s_t = (c_1, c_2)$ is the number of queued vehicles per lane. Actions are signal phases with different service capacities. The reward function penalizes both congestion (queues exceeding a threshold) and inefficiency (allocating green time to empty lanes). Concept drift is induced by changing vehicle arrival rates (λ_1, λ_2) at predefined episodes. Upon drift detection, the action space is expanded with more aggressive signal phases to manage heavier traffic.

B. Experimental Setting

Experiments were run using Python 3.12 and Gym 0.26.2. We compare MORPHIN against a Standard Q-learning (Baseline) agent with a fixed learning rate ($\alpha = 0.1$) and a single, non-resetting exponential exploration decay. MORPHIN uses the PH-test to trigger exploration resets ($\varepsilon \to 1$) and a dynamic learning rate α^* modulated by the TD-error. Hyperparameters for both scenarios (e.g., $k = 5, \delta = 0.5, H = 300$) were determined empirically to suit the reward scale of each environment.

C. Evaluation Results

Gridworld: MORPHIN demonstrates superior adaptation and knowledge retention. As illustrated in Fig. 2a, after 1,500 episodes, MORPHIN (left) successfully retains high Q-values for both current and previously learned goals, effectively reducing catastrophic forgetting effects. In contrast, the baseline agent (right) overwrites past knowledge and only remembers the most recent goal. This enhanced adaptability translates to significant efficiency gains, as shown in TABLE I. MORPHIN improves overall learning efficiency by a factor of 1.7x (measured in total steps) and successfully converges after each induced drift. The baseline agent, however, fails to converge after the first 300-episode interval. Furthermore, Fig. 2b shows that when the action space is expanded, MORPHIN effectively incorporates the new "jump" action to discover a more optimal policy, whereas the baseline agent struggles to update its established policy and remains suboptimal.

Traffic-Signal Control: In this scenario, MORPHIN again shows robust adaptation (Fig. 3). When congestion increases at episode 3,000, the PH-test detects the drift, triggering adaptation and enabling a rapid performance recovery. The baseline agent suffers a prolonged degradation. However, the results also highlight a limitation: a second drift at episode 8,000 (lowered traffic) is not detected because the new reward distribution is a subset of previously seen values and does not exceed the PH-test's sensitivity threshold. While both agents performance improves in the easier environment, this failure underscores the challenge of parameterizing drift detectors. Despite this, the case study confirms MORPHIN's ability to react to detected non-stationarity and seamlessly incorporate new actions to manage changing conditions.

TABLE I: Average convergence time (in episodes after drift) and total steps over 1,500 episodes in Gridworld (1,000 runs). Dashes (–) indicate failure to converge within the 300-episode interval. An independent t-test confirms the difference in total steps is statistically significant ($p < 0.05$).

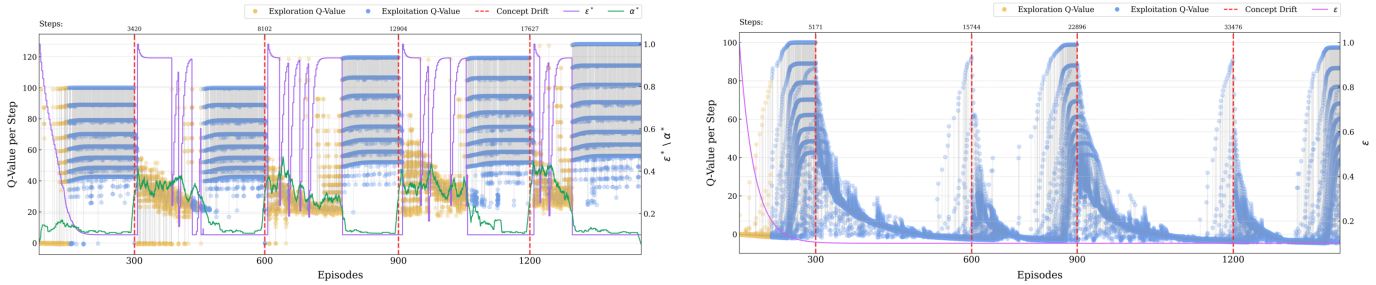
Agent	1st Drift	2nd Drift	3rd Drift	Total Steps
Q-Learning	256.40 ± 4.35%	–	–	40,683.74 ± 1.33%
MORPHIN	135.81 ± 9.31%	175.17 ± 5.80%	167.81 ± 3.13%	23,292.07 ± 6.33%

V. Related Work

Learning in non-stationary environments is a significant challenge in RL, revolving around the exploration-exploitation trade-off [1]. Our work builds on established research lines addressing this issue, comprehensively surveyed by Padakandla [12].

A. Adapting to Environmental Changes

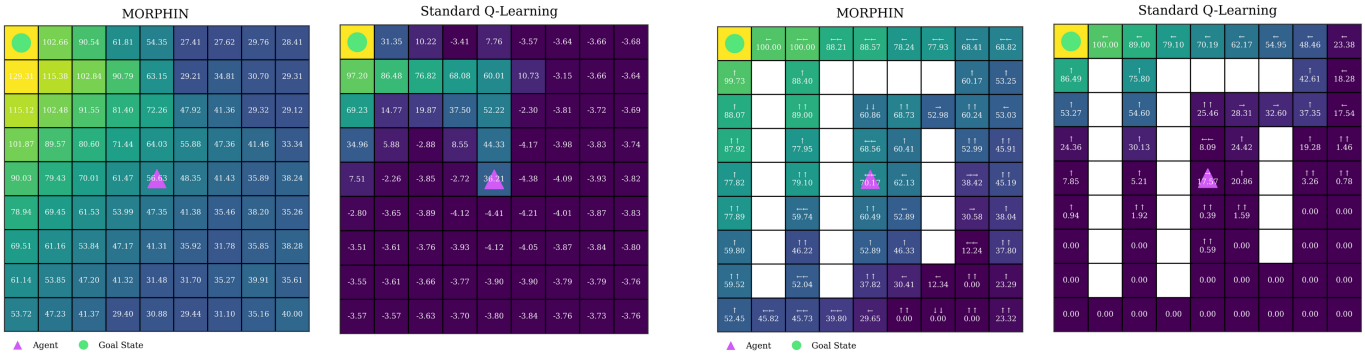
A common strategy in non-stationary environments is adapting learning hyperparameters. Several works adjust the exploration rate, ε, often using a change-point detector to trigger the adaptation. Both Mignon *et al.* [8] and Hartland *et al.* [3] used the PH-test to detect drifts and adapt exploration, a combination our work adopts. Other model-free approaches include Repeated Update Q-learning (RUQL), which addresses policy-bias by repeating updates for less-chosen actions [13], contrasting our method's adaptation of the learning rate α^* based on TD-error magnitude. Proactive methods like RestartQ-UCB periodically reset memory on a fixed schedule [14],



(a) MORPHIN with adaptive ε^* and α^*.

(b) Standard Q-learning with static ε-decay.

Fig. 1: Internal dynamics of MORPHIN versus a standard Q-learning agent, illustrating their contrasting responses to concept drift (red dashed lines) in the Gridworld scenario, the y-axis for the scatter points represents the Q-value of the chosen state-action pair for that step, while their color indicates whether the action was exploratory (yellow) or exploitative (blue). (a) MORPHIN's adaptive response: Upon detecting a drift, the PH-test triggers a reset of the exploration rate ε^* (purple), forcing re-exploration. The resulting high TD-errors cause the dynamic learning rate α^* (green) to increase, accelerating the integration of new knowledge and enabling rapid policy recovery. (b) Standard Q-learning's static behavior: The baseline agent uses a single, exponentially decaying exploration schedule (ε) and a fixed α value. After the initial convergence, the low exploration rate prevents it from adapting to subsequent drifts, causing the agent to remain committed to an obsolete policy and resulting in a sustained performance collapse.



(a) Adaptation to shifting goals (1,500 episodes).

(b) Adaptation to an expanded action space (400 episodes).

Fig. 2: Comparison of Q-value heatmaps demonstrating MORPHIN's superior adaptation over standard Q-learning in two non-stationary scenarios. (a) In a goal-switching environment, MORPHIN (left) preserves high Q-values for both initial and subsequent goals, showcasing effective knowledge retention and preventing catastrophic forgetting, unlike the baseline agent (right). (b) When the action space is expanded, MORPHIN (left) successfully integrates a new "jump" action (indicated by double arrows) to discover a more optimal policy, while the baseline (right) fails to adapt, remaining committed to a suboptimal policy.

whereas MORPHIN is reactive, using an online PH-test to trigger adaptation only when performance changes significantly. Context-based methods like Context Q-learning [15] maintain separate Q-tables for each environmental context, isolating knowledge at the cost of increased memory. MORPHIN follows a more lightweight philosophy, maintaining a single Q-table and adapting in-place by dynamically scaling the learning rate to integrate new experiences.

B. Continual Learning and Self-Adaptive Systems

The aforementioned methods align with the principles of CRL [2], where the goal is for agents to "never stop learning" [6]. Recent studies show that many CL methods struggle when a state-action pair yields different rewards after an environmental shift [16]. MORPHIN addresses this directly: by not resetting its Q-table and forcing re-exploration, an outdated Q-value generates a large TD-error when encountering a new reward, which in turn drives a high learning rate to rapidly overwrite the obsolete knowledge. RL is also a potent tool for building self-adaptive systems [11], with applications in IoT security [17] and network monitoring [9]. Our traffic-control scenario is a canonical example of this paradigm.

C. Positioning of Our Contribution

The main contribution of this paper is not a single algorithmic component, but rather the synthesis and integration of several established techniques into a unified, model-free framework for tabular Q-learning. MORPHIN combines proactive environment monitoring via the PH-test with reactive adaptation of both the exploration rate (ε^*) and the learning

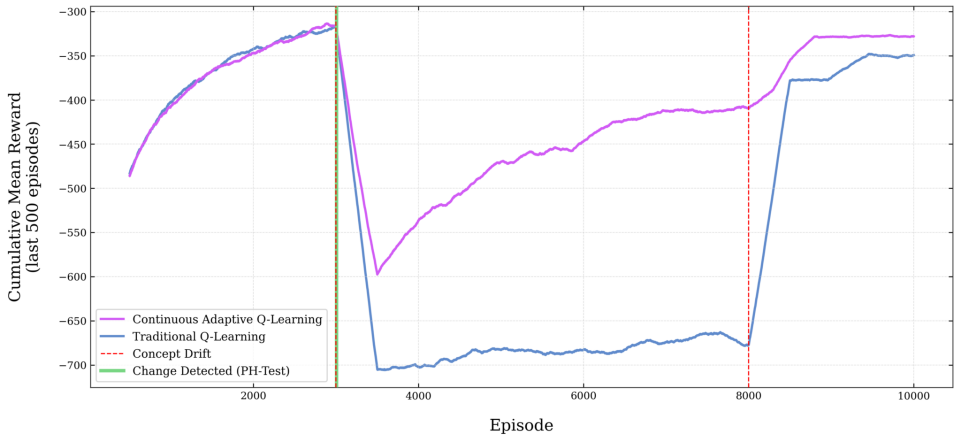


Fig. 3: Learning performance in the traffic scenario. After the first drift (episode 3,000), MORPHIN detects the change (green line) and rapidly recovers. Traditional Q-learning suffers prolonged degradation. The second drift (episode 8,000) is not detected by the PH-tests because the new reward distribution is a subset of previously seen values and does not exceed the sensitivity threshold.

rate (α^*). This coordinated strategy is specifically designed to enable an agent to adapt to two distinct types of non-stationarity simultaneously: changes in the reward function (shifting goals) and on-the-fly expansions of the action space. By preserving and building upon a single Q-table, our approach provides a lightweight solution aimed at preventing catastrophic forgetting and ensuring continual adaptation in dynamic environments, without the need for full retraining or maintaining multiple explicit context models.

VI. Conclusion and Future Work

This paper introduced MORPHIN, a self-adaptive framework for tabular Q-learning agents operating in non-stationary environments. We specifically addressed the challenge of agents adapting to simultaneous changes in their goals (reward functions) and their available capabilities (action-space expansions). Our approach integrates concept drift detection using the PH-test with dynamic adjustments to the exploration (ε) and learning (α) rates. Experimental results in both a Grid-world benchmark and a traffic control simulation demonstrate that this coordinated strategy enables agents to adapt more effectively than a standard Q-learning baseline, achieving a performance increase of up to $1.7\times$ in learning efficiency while successfully reducing catastrophic forgetting effects.

Through this synthesis of established techniques into a unified framework, MORPHIN demonstrates how a lightweight, model-free agent can achieve robust continual learning. By preserving and adapting a single Q-table, it effectively enables knowledge reuse when faced with contradictory environmental changes, a key challenge highlighted by Bagus *et al.* [16], without requiring full retraining or multiple context models. Our work thus presents a practical and resource-efficient method for developing self-adaptive systems capable of real-time learning in dynamic environments.

This work represents an initial validation, and several avenues for future work are evident. The limitations observed in our experiments, such as the failure of the PH-test to detect certain drifts and the need for empirical tuning of hyperparameters, point to clear directions for improvement. Future work should therefore focus on:

1) Generalization and Scalability: Extending the core principles of MORPHIN from tabular methods to deep RL architectures to handle high-dimensional state spaces. This would also involve investigating the use of memory-based plasticity to enhance knowledge transfer across tasks.
2) Robustness and Empirical Analysis: Conducting a rigorous sensitivity analysis of the introduced hyperparameters, particularly the TD-error sensitivity (k) and H threshold for the PH-test, to better understand their impact on performance. Furthermore, we plan to explore more robust drift detection methods to overcome the limitations of the PH-test observed in our traffic scenario, where drifts that result in subsets of known reward distributions can be missed. We also intend to extend the evaluation to include scenarios with action removal and other types of reward function changes, such as those examined by Mignon *et al.* [8].
3) Advanced Application Domains: Applying the framework to more complex distributed multi-agent systems (*e.g.,* bigger street network configuration). Finally, we plan to deploy MORPHIN on resource-constrained edge devices for real-world applications in Internet of Things (IoT) and robotics.

References

[1] Richard S. Sutton and Andrew G. Barto, *Reinforcement Learning, An Introduction*, Second Edition. MIT Press, 2018, p. 550, ISBN: 9780262039246.

[2] K. Khetarpal, M. Riemer, I. Rish, and D. Precup, *Towards continual reinforcement learning: A review and perspectives*, 2022. arXiv: 2012.13490 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2012.13490.

[3] C. Hartland, N. Baskiotis, S. Gelly, M. Sebag, and O. Teytaud, "Change point detection and meta-bandits for online learning in dynamic environments," Apr. 2011.

[4] B. Norman and J. Clune, *First-explore, then exploit: Meta-learning to solve hard exploration-exploitation trade-offs*, 2024. arXiv: 2307.02276 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2307.02276.

[5] M. Guériau, N. Cardozo, and I. Dusparic, "Constructivist approach to state space adaptation in reinforcement learning," Jun. 2019. DOI: 10.1109/SASO.2019.00016.

[6] D. Abel, A. Barreto, B. V. Roy, D. Precup, H. van Hasselt, and S. Singh, *A definition of continual reinforcement learning*, 2023. arXiv: 2307.11046 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2307.11046.

[7] T. Wong, M. Wagner, and C. Treude, "Self-adaptive systems: A systematic literature review across categories and domains," *Information and Software Technology*, vol. 148, p. 106 934, 2022, ISSN: 0950-5849. DOI: https://doi.org/10.1016/j.infsof.2022.106934. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584922000854.

[8] A. Mignon and R. L. A. Rocha, "An adaptive implementation of -greedy in reinforcement learning," *Procedia Computer Science*, vol. 109, pp. 1146–1151, Dec. 2017. DOI: 10.1016/j.procs.2017.05.431.

[9] S. Wassermann, T. Cuvelier, P. Mulinka, and P. Casas, "Adaptive and reinforcement learning approaches for online network monitoring and analysis," *IEEE Transactions on Network and Service Management*, vol. PP, pp. 1–1, Nov. 2020. DOI: 10.1109/TNSM.2020.3037486.

[10] M. Towers *et al.*, *Gymnasium: A standard interface for reinforcement learning environments*, 2024. arXiv: 2407.17032 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2407.17032.

[11] E. Henrichs, V. Lesch, M. Straesser, S. Kounev, and C. Krupitzer, "A literature review on optimization techniques for adaptation planning in adaptive systems: State of the art and research directions," *Information and Software Technology*, vol. 149, p. 106 940, 2022, ISSN: 0950-5849. DOI: https://doi.org/10.1016/j.infsof.2022.106940. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584922000891.

[12] S. Padakandla, "A survey of reinforcement learning algorithms for dynamically varying environments," *ACM Computing Surveys*, vol. 54, no. 6, pp. 125, Jul. 2021, ISSN: 1557-7341. DOI: 10.1145/3459991. [Online]. Available: http://dx.doi.org/10.1145/3459991.

[13] S. Abdallah and M. Kaisers, "Addressing environment non-stationarity by repeating q-learning updates," vol. 17, Apr. 2016.

[14] W. Mao, K. Zhang, R. Zhu, D. Simchi-Levi, and T. Basar, "Near-optimal model-free reinforcement learning in non-stationary episodic mdps," in *Proceedings of the 38th International Conference on Machine Learning*, M. Meila and T. Zhang, Eds., ser. Proceedings of Machine Learning Research, vol. 139, PMLR, 2021, pp. 7447–7458. [Online]. Available: https://proceedings.mlr.press/v139/mao21b.html.

[15] S. Padakandla, P. K. J., and S. Bhatnagar, "Reinforcement learning algorithm for non-stationary environments," *Applied Intelligence*, vol. 50, no. 11, pp. 35903606, Jun. 2020, ISSN: 1573-7497. DOI: 10.1007/s10489-020-01758-5. [Online]. Available: http://dx.doi.org/10.1007/s10489-020-01758-5.

[16] B. Bagus and A. Gepperth, "A study of continual learning methods for q-learning," in *International Joint Conference on Neural Networks (IJCNN)*, IEEE, Jul. 2022, pp. 19. DOI: 10.1109/ijcnn55064.2022.9892384. [Online]. Available: http://dx.doi.org/10.1109/IJCNN55064.2022.9892384.

[17] A. Kumar and D. Singh, "Adaptive epsilon greedy reinforcement learning method in securing iot devices in edge computing," *Discover Internet of Things*, vol. 4, Nov. 2024. DOI: 10.1007/s43926-024-00080-7.