Improving Deep Q-Network Adaptability to Distributional Shift through Distributed Case-Based Experience Sharing

Ardianto Wibowo[1,3,4], Paulo E. Santos[2], Amer Baghdadi[1], Matthew Stephenson[3], Karl Sammut[3], Jean-Philippe Diguet[4]
[1]IMT Atlantique, Lab-STICC, UMR CNRS 6285, F-29238 Brest, France
[2]PrioriAnalytica, Adelaide, Australia
[3]College of Science & Engineering, Flinders University, Adelaide, Australia
[4]CNRS IRL Crossing, Adelaide, Australia

***Abstract*—Reinforcement Learning (RL) deployments often face distributional shifts that can undermine the performance of learned policies. This challenge becomes more severe in decentralized multi-agent systems, where agents operate without a central controller and under conditions of limited observability. We propose D²CBRL-DQN, a hybrid learning approach that integrates Distributed Case-Based Reasoning (CBR) with Deep Q-Networks (DQN) in a Decentralized Multi-Agent Reinforcement Learning (MARL). To the best of our knowledge, D²CBRL-DQN is the first approach to integrate Distributed CBR with DQN in a Decentralized MARL setting. By augmenting DQN with distributed case storage and sharing of successful experiences, D²CBRL-DQN enables agents to adapt collectively to distributional shifts without requiring full retraining of the value function. We evaluate D²CBRL-DQN in a decentralized multi-agent gridworld environment with local, partial observations, where obstacle configurations change episodically to induce distributional shifts. Experimental results demonstrate that D²CBRL-DQN consistently outperforms vanilla DQN in terms of adaptation speed, cumulative reward, and task success rate. These findings show that integrating Distributed CBR with value-based RL can significantly accelerate policy adaptation under distributional shift while promoting effective experience reuse in decentralized multi-agent systems.**

***Index Terms*—Distributional Shifts, Decentralized Multi-Agent Reinforcement Learning, Distributed Case-Based Reasoning**

I. Introduction

Reinforcement Learning (RL) agents are often trained in fixed environment configurations, yet real deployments frequently differ from training conditions or evolve over time, causing distributional shifts [1]–[3]. In value-based methods such as Deep Q-Networks (DQN), shifts can substantially degrade performance because the learned Q-function may overfit to training layouts and dynamics [4]–[6]. As a result, a policy derived from a Q-function optimized on a training state distribution may become unreliable in shifted settings, often requiring costly retraining [2], [3], [7], [8].

In decentralized Multi-Agent Reinforcement Learning (MARL), distributional shifts are amplified by partial observability and the lack of a central controller [9]. Each agent relies on local observations and its own value estimates, making centralized updates impractical. These challenges arise in real-world applications such as Search and Rescue (SAR), where teams of robots must operate under dynamic conditions and unreliable communication [10].

Existing methods mitigate distributional shifts via environment randomization [5], [6], meta-RL [11], or behavior-aware detection and adaptation [8]. These approaches typically assume single-agent settings, centralized training, or explicit retraining phases. Case-Based Reasoning (CBR) has been shown to improve learning efficiency by accelerating convergence through the reuse of past experiences structured into cases (e.g., QCBRL [12]). Our hypothesis is that such experience reuse, when selectively transformed into reusable cases and distributed across agents, can also accelerate adaptation under distributional shifts. Nevertheless, prior CBR-based implementations rely on centralized memory architectures, limiting their applicability to decentralized MARL.

Our Contribution: We introduce D²CBRL-DQN, a hybrid framework that integrates Distributed CBR with DQN in decentralized MARL to improve collective adaptation under distributional shifts. We: (i) formalize decentralized adaptation as selective reuse of locally stored and shared successful experiences; (ii) propose a distributed memory and sharing mechanism that preserves agent autonomy while enabling collective adaptation; and (iii) empirically validate the approach in a multi-agent gridworld with episodically shifting obstacle layouts, showing faster recovery and higher success than vanilla DQN and a centralized shared-memory baseline.

II. Related Work

Generalization under distributional shift has been addressed via environment randomization and procedural generation to reduce layout memorization [5], [6], but these methods do not directly support online adaptation. Meta-RL (e.g., MAML) enables rapid adaptation across task distributions [11], [13], yet often assumes centralized training and task boundary awareness. Behavior-aware methods detect shifts via behavioral deviations and trigger policy adjustments [8], and performative RL models environments that evolve in response to agent actions [7], but such approaches typically rely on centralized feedback or retraining.

CROP [3] improves zero-shot generalization to structurally shifted environments via compact representations, but does not

address Decentralized MARL. Experience reuse via CBR integrated with RL has been explored in QCBRL [12]; however, existing designs commonly assume centralized memory architectures and are restricted to tabular, Q-table–based methods.

To the best of our knowledge, this work presents the first CBR learning framework deployed in a fully decentralized MARL setting. Building on this foundation, we demonstrate how distributed case memories and selective experience sharing can be leveraged to address adaptation under distributional shifts in decentralized MARL while maintaining independent value functions.

III. Background

Distributional Shift: Distributional shift is commonly expressed as a mismatch between training and test input distributions [1]:

$$\frac{Q(X)}{P(X)} \neq 1,$$

where $P(X)$ and $Q(X)$ denote training and test distributions, respectively. In RL, trajectory-level shift can be characterized as [14]:

$$\frac{q^{\pi_c}(\tau)}{p^{\pi}(\tau)} = \prod_t \underbrace{\frac{q(s_{t+1} \mid s_t, a_t)}{p(s_{t+1} \mid s_t, a_t)}}_{\text{Model Bias}} \cdot \underbrace{\frac{\pi_c(a_t \mid s_t)}{\pi(a_t \mid s_t)}}_{\text{Policy Shift}} \tag{1}$$

In our setup, shifts arise episodically by re-randomizing obstacle layouts every n episodes while learning continues, producing continual non-stationarity and requiring repeated adaptation.

Distributed CBR: CBR addresses new problems by reusing past experiences [15]. Each case comprises a problem description, a solution, and an outcome, and reasoning follows the *Retrieve–Reuse–Revise–Retain* cycle. Distributed CBR extends this process to multi-agent systems with decentralized reasoning and storage, which can be organized as Single-agent/Multi-case-base, Multi-agent/Single-case-base, or Multi-agent/Multi-case-base architectures [16]. We adopt the Multi-agent/Multi-case-base design, where agents maintain local case bases and coordinate indirectly via selective experience sharing.

Decentralized MARL: MARL settings are commonly categorized as centralized, decentralized networked, or fully decentralized [17]. Centralized approaches rely on global information, while fully decentralized ones prohibit explicit communication. We adopt the decentralized networked setting, which enables localized communication and coordination while preserving agent autonomy and avoiding the scalability limitations of centralized control.

IV. Problem Formulation

We consider a decentralized MARL setting with a set of agents $\mathcal{A} = \{1, 2, ..., N\}$, each operating under partial observability. At each timestep t, agent i observes a local state o_i^t and selects an action $a_i^t \in \mathcal{A}_i$ using a policy π_i that reasons over both its own experience $\mathcal{C}_i$ and shared experiences $\mathcal{C}_{-i}$ from other agents.

The environment is episodic and evolving, exhibiting distributional shifts across episodes such that $\mathcal{P}_{env}(e) \neq \mathcal{P}_{env}(e+1)$, where $\mathcal{P}_{env}(e) \in \mathcal{E}$ represents the environment distribution at episode e, and $\mathcal{E}$ is the set of possible environment configurations. Each distribution $\mathcal{P}_{env}(e)$ captures the randomness or variability in the environment at a given episode e.

Let τ_i denote the trajectory of agent i over an episode e, r_i^t be the reward received by agent i at timestep t, and T the episode horizon. The objective of MARL is to adaptively maximize the expected cumulative reward across shifting environments using shared experiential knowledge without retraining. We formalize it as:

$$\max \mathbb{E}_{\mathcal{P}_e \sim \mathcal{E}} \left[\mathbb{E}_{\tau_i \sim \pi_i, \mathcal{P}_e} \left[\sum_{t=0}^{T} r_i^t \,\middle|\, \pi_i(o_i^t, \mathcal{C}_i, \mathcal{C}_{-i}) \right] \right] \tag{2}$$

V. The D^{2}CBRL-DQN Framework

This section introduces the proposed D^{2}CBRL-DQN framework, which integrates Distributed CBR with decentralized multi-agent DQN learning. Figure 1 provides an overview of the per-agent architecture.

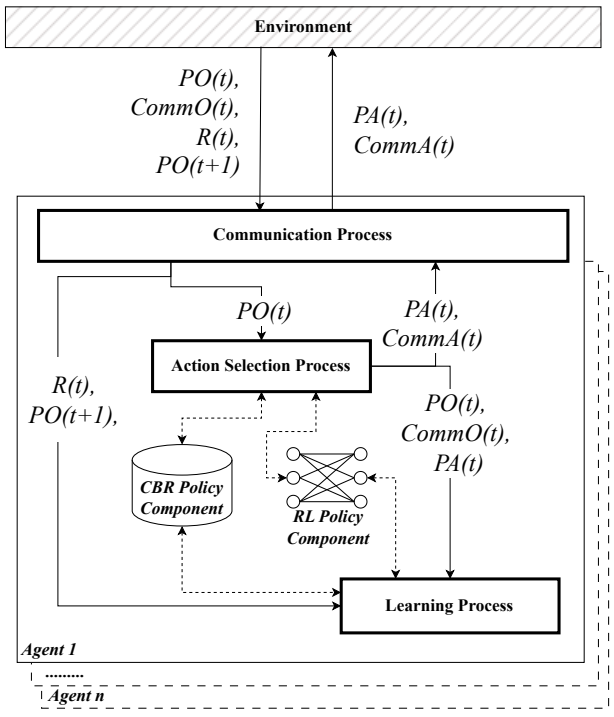


Fig. 1: D^{2}CBRL-DQN framework in each agent.

As shown in Figure 1, at each timestep t, an agent receives a partial observation $PO(t)$ and a communication observation $CommO(t)$ containing experience messages from other agents, and prepares a communication action $CommA(t)$ for experience sharing. Based on these inputs, the agent selects and executes a physical action $PA(t)$. The environment returns a reward $R(t)$ and the next observation $PO(t+1)$, which are used to update both policy components. Learning is fully

local and asynchronous, with agents maintaining independent models and case bases while coordinating indirectly through experience sharing.

Each agent operates autonomously in a closed-loop system composed of two internal policy components—the *CBR Policy Component* and the *RL Policy Component*—and three core processes: *Communication*, *Action Selection*, and *Learning*, described as follows:

A. CBR Policy Component

The CBR Policy Component enables experience reuse and sharing through a distributed CBR mechanism. Each agent maintains a structured case base and operates according to the classical four-phase CBR cycle: *Retrieve*, *Reuse*, *Revise*, and *Retain*.

1. Case Structure:

$$\text{Case}_j = (p_j, s_j, tv_j, d_j) \tag{3}$$

where p_j denotes the problem state, s_j the associated action, $tv_j \in [0, 1]$ the trust value reflecting reliability, and d_j additional performance-related data (e.g., steps to reach the target). The set of all cases forms the agent's local case base $\mathcal{CB}_i$.

To support both local learning and experience sharing, each agent maintains three memories per episode: a persistent case base for long-term knowledge, an *own temporary case base* for locally generated experiences, and an *others' temporary case base* for received cases. Temporary memories are cleared at episode start, while the persistent case base is updated during the *Retain* phase.

2. CBR Cycle: The CBR cycle governs how cases are accessed, updated, and retained.

a) Retrieve.: The agent compares its current observation PO_t with stored cases. In the with-global-coordinate setting, retrieval requires exact state matching. Without global coordinates, a trust-filtered nearest-neighbor retrieval is applied, selecting the most reliable case within a distance threshold:

$$\text{Retrieve}(CB, PO_t) = \begin{cases} \arg\min_{c \in CB,\ tv_c \geq \theta_{\min}} \|PO_t - p_c\| & \text{if } \|PO_t - p_c\| < \epsilon_d \\ \text{None} & \text{otherwise} \end{cases} \tag{4}$$

b) Reuse.: Executed actions and received communications are stored in temporary memories. Local experiences are added to the agent's own temporary case base, while successful cases received via communication are stored separately:

$$\begin{aligned} \text{OwnTCB}_i &\leftarrow \text{Case}(p_t^i, s_t^i), \\ \text{OthersTCB}_i &\leftarrow \text{CommO}_i(t) \end{aligned} \tag{5}$$

c) Revise.: After each episode, trust values are updated based on case usage and episode outcome:

$$tv_j \leftarrow \begin{cases} tv_j + \delta^+ & \text{if reused and episode successful,} \\ tv_j - \delta^- & \text{if reused and episode failed,} \\ tv_j - \delta^0 & \text{if not reused and episode successful.} \end{cases} \tag{6}$$

The decay term δ^0 is introduced to gradually remove rarely used cases, controlling case-base growth.

d) Retain.: Temporary memories are merged and filtered:

$$\text{TCB}_i = \text{OwnTCB}_i \cup \text{OthersTCB}_i \tag{7}$$

Only one case per state is retained. A new case replaces an existing one if it yields better performance, while cases with trust below $\theta_{\min}$ are discarded.

B. RL Policy Component

The RL Policy Component is implemented using a standard DQN, trained via temporal-difference learning with experience replay and a target network [18]. The DQN serves as a fallback policy when no suitable case is retrieved, ensuring continued exploration and adaptation to previously unseen or shifted states.

C. Communication Process

The communication process enables decentralized and asynchronous experience sharing. Communication is triggered once per episode, when an agent reaches the objective. At that moment, the agent broadcasts its own temporary case base $\text{OwnTCB}_i(t)$ as a communication action $CommA_i(t)$. This information is received by other agents at the next timestep as a communication observation $CommO_j(t+1)$, for all $j \neq i$.

The broadcast and reception mechanisms are formalized as follows:

$$CommA_i(t) = \begin{cases} \text{OwnTCB}_i(t) & \text{if ObjectiveReached}_i(t), \\ \textit{null} & \text{otherwise.} \end{cases} \tag{8}$$

$$CommO_j(t+1) = \begin{cases} CommA_i(t) & \text{if } CommA_i(t) \neq \textit{null}, \\ \textit{null} & \text{otherwise.} \end{cases} \tag{9}$$

This mechanism allows agents to exchange successful experiences without centralized coordination or synchronization. Communication complements local decision-making and learning while preserving full agent autonomy when no communication is available.

D. Action Selection Process

Action selection follows a hierarchical strategy combining exploration, case reuse, and RL. The agent first applies an ε_1-greedy rule to choose between random exploration and consulting its case base. If a matching case is found, the stored action is reused. Otherwise, the agent falls back to its RL policy using an ε_2-greedy strategy. This design balances exploitation of reliable experiences with continued exploration and learning. The full process is illustrated in Figure 2.

E. Learning Process

Learning proceeds through both policy components. In the CBR component, trust values are updated during the *Revise* and *Retain* phases, reinforcing reliable cases and removing ineffective ones. In the RL component, training extends the standard DQN loss by weighting case-based samples:

$$\mathcal{L}_{\text{CBR}} = \frac{1}{B} \sum_{i=1}^{B} \left(1 + f_i \left(w_{\text{CBR}} - 1\right)\right) \left(Q(s_i, a_i) - y_i\right)^2 \tag{10}$$

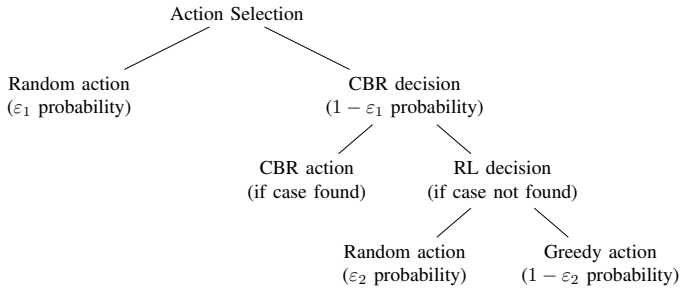


Fig. 2: Action selection tree

where f_i indicates whether a sample originates from the case base. This prioritization accelerates learning by emphasizing trusted experiences.

F. Pseudocode

The pseudocode summarizes the overall D^{2}CBRL-DQN framework as presented in Algorithm 1. The next section presents experimental results demonstrating the effectiveness of the proposed approach under distributional shifts.

Algorithm 1 D^{2}CBRL-DQN Algorithm

for each agent i **do**
  Initialize DQN Q_i, target Q'_i, and empty case base $\mathcal{CB}_i$
end for
for episode $= 1$ to N **do**
  Reset environment; get o_i^{phys}, o_i^{comm} for all i; clear temp $\mathcal{TCB}_i$
  while not all agents done AND step $\leq$ max_steps **do**
    for each agent i **do**
      if random() $< \epsilon_1$ **then**
        $a_i^{phys} \leftarrow$ random, $a_i^{comm} \leftarrow$ null
      else
        CBR: Retrieve matching case from $\mathcal{CB}_i$
        if case found **then**
          $a_i^{phys}, a_i^{comm} \leftarrow$ case actions
        else if random() $< \epsilon_2$ **then**
          $a_i^{phys} \leftarrow$ random, $a_i^{comm} \leftarrow$ null
        else
          $a_i^{phys} \leftarrow \arg\max_a Q_i(o_i^{phys}, o_i^{comm}, a)$
          $a_i^{comm} \leftarrow$ null
        end if
      end if
    end for
    Execute environment step; get $o_i^{phys}, o_i^{comm}, r_i, done_i$
    for each agent i **do**
      CBR: Reuse store $(o_i^{phys}, o_i^{comm}, a_i^{phys}, a_i^{comm})$ in $\mathcal{TCB}_i$
      Store transition in replay buffer
      Update Q_i using CBR-weighted loss
    end for
  end while
  for each agent i **do**
    CBR: Revise trust values in $\mathcal{CB}_i$
    CBR: Retain shortest successful cases from $\mathcal{TCB}_i$ into $\mathcal{CB}_i$
    Decay ϵ_1, ϵ_2
  end for
end for

VI. Experiments

This section presents the experimental setup and results used to evaluate the proposed hybrid D^{2}CBRL-DQN framework. We first describe the environment shift scenarios used to simulate distributional changes, followed by parameter settings. Adaptation performance is then evaluated across three experimental categories: (1) with and without global (x, y) coordinates, (2) with and without trust decay in the CBR *Revise* phase, (3) with and without experience sharing.

A. Environment Layout & Parameter Setting

All experiments were conducted using a custom Multi-Agent Grid World environment and a set of predefined parameters, detailed as follows:

- *Environment:* The environment consists of a 30×30 grid, with the number of agents $N_{\text{ag}} \in \{2, 3, 4, 5\}$, each having a 9×9 local field of view ($F = 9$). The environment contains 20 static obstacles and one target. A distributional shift is introduced every 100 episodes ($n = 100$) by resampling both obstacle and target positions uniformly at random over valid grid cells, subject to non-overlap constraints with agents, obstacles, and the target.
- *DQN-related parameters:* learning rate $\alpha_1 = 0.001$, discount factor $\gamma = 0.9$, initial exploration rates $\varepsilon_1 = 0.9$, $\varepsilon_2 = 0.9$, exploration decay rate $\varepsilon_{\text{decay}} = 0.95$, minimum exploration rate $\varepsilon_{\text{min}} = 0.01$, using a CNN with two 3×3 conv layers (16, 32 filters) and a 256-unit fully connected layer.
- *CBR-related parameters:* initial trust value $t_{\text{init}} = 0.5$, trust increment $\delta^+ = 0.1$, trust decrement $\delta^- = 0.1$, trust decay $\delta^0 = 0.1$, and minimum trust threshold $\theta_{\text{min}} = 0.1$.
- *Reward structure:* Agents receive +50 for reaching the target, -50 for colliding with an obstacle, and -1 per step taken.

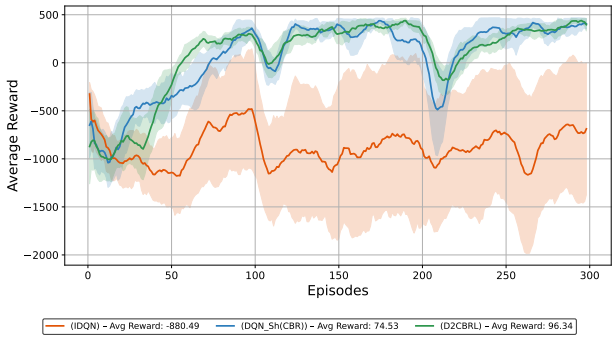
To evaluate and compare performance, three evaluation metrics are used in the experiments: (i) *Average Rewards:* Measures the cumulative rewards obtained per episode; higher values indicate better overall performance. (ii) *Average Steps to Convergence:* Measures the number of steps needed to achieve stable performance, defined by consistently low episode lengths over a fixed window; fewer steps indicate faster convergence. (iii) *Average Success Rate:* Measures the proportion of successful episodes relative to the total number of episodes; higher success rates are better.

B. Experiment 1: With and Without Global Coordinates

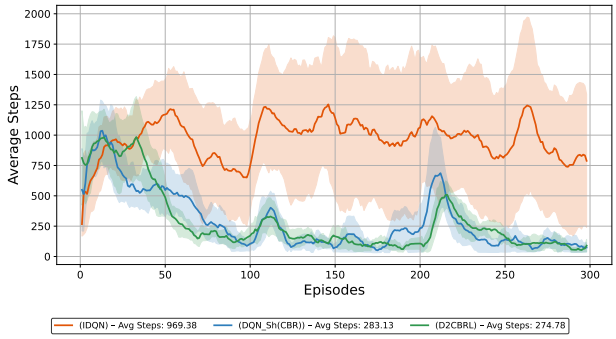
This experiment compares adaptation behavior under different algorithmic configurations. Three approaches are evaluated: *IDQN*—a baseline Independent DQN without enhancements; *DQN_Sh(CBR)*—DQN augmented with a centralized CBR mechanism across all agents (as used in [12]); *D^{2}CBRL-DQN*—our proposed decentralized approach that integrates DQN with distributed CBR. Each algorithm was run five times with different seeds.

TABLE I: Performance metrics with global-coordinate

Algorithm	Avg Reward	Avg Steps	Success Rate
IDQN	-880.49 ± 284.8	969.38 ± 248.8	0.96%
DQN_Sh(CBR)	74.53 ± 133.2	283.13 ± 84.9	68.82%
D^{2}CBRL-DQN	$\mathbf{96.34} \pm 188.1$	$\mathbf{274.78} \pm 124.5$	**74.34%**

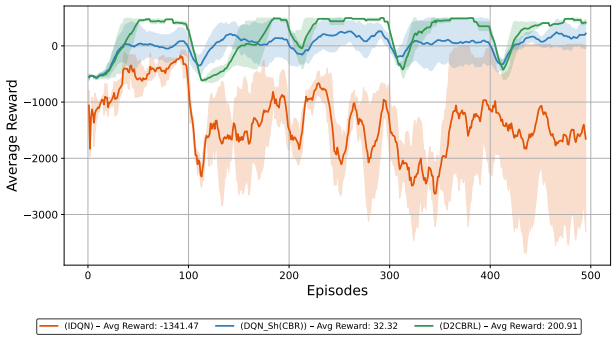


(a) Average Rewards with global coordinate

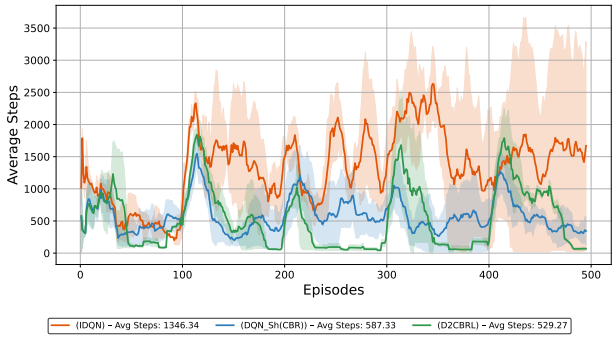


(b) Average Steps with global coordinates

Fig. 3: Performance comparison between *IDQN*, *DQN_Sh(CBR)*, and *D²CBRL-DQN* for 2 agents with global coordinate.



(a) Average Rewards without global coordinate



(b) Average Steps without global coordinate

Fig. 4: Performance comparison between *IDQN*, *DQN_Sh(CBR)*, and *D²CBRL-DQN* for 2 agents without global coordinate.

Figure 3 shows the adaptation performance in the with-global-coordinate setting, with environment shifts at episodes 100 and 200. D^{2}CBRL-DQN achieves the highest rewards and quickly adapts after each shift, while IDQN fails to recover. As shown in Table I, D^{2}CBRL-DQN reduces the average number of steps by 71.7% compared to IDQN, flips the average reward from –880.6 to +96.8, and increases the success rate by 73.3 percentage points.

To enhance generalization ability and scalability, global coordinate information is excluded from both agent observations and case problems. Figure 4 presents the performance in this setting with 500 episodes. In Figure 4(a), D^{2}CBRL-DQN consistently yields the highest average rewards and quickly adapts to shifts at episodes 100, 200, 300, and 400. Figure 4(b) shows it also converges faster with fewer steps. Table II confirms that D^{2}CBRL-DQN reduces the average number of steps by 60.5% compared to IDQN, flips the average reward from –1340.6 to +195.2, and increases the success rate by 54.3 percentage points.

TABLE II: Performance metrics without global-coordinate

Algorithm	Avg Reward	Avg Steps	Success Rate
IDQN	-1341.47 ± 241.5	1346.34 ± 243.1	10.71%
DQN_Sh(CBR)	32.32 ± 183.7	587.33 ± 338.6	30.62%
D^{2}CBRL-DQN	$\mathbf{200.91} \pm 65.3$	$\mathbf{529.27} \pm 120.9$	**65.05%**

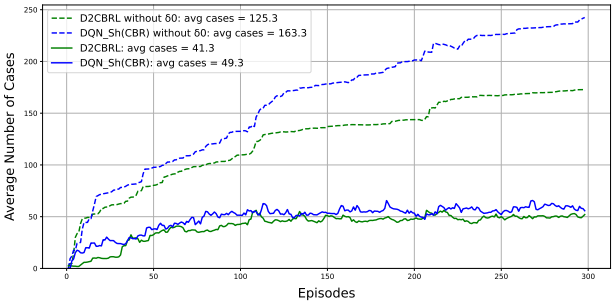
The performance degradation of *DQN_Sh(CBR)* relative to D^{2}CBRL-DQN is attributed to mutual interference among agents, stemming from its unified trust value decrement strategy (as defined in Equation 6). To better understand this effect, the following experiment investigates the role of the decayment rule in the case retention mechanism.

C. Experiment 2: Decayment Rule Effect

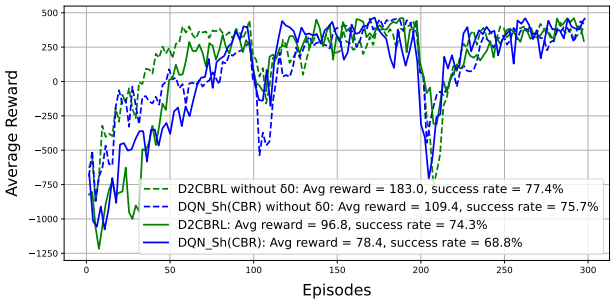
The objective of this experiment is to evaluate the impact of the decayment rule in D^{2}CBRL-DQN and DQN_Sh(CBR) on case retention, as defined in Eq. 6. All results are averaged over 15 independent runs per algorithm.

Figure 5 shows that activating the decayment rule slightly reduces reward and step performance but significantly lowers the number of stored cases. For D^{2}CBRL-DQN, the average drops from 125.3 (without decayment) to 41.3 (with decayment), resulting in a 3.03$\times$ reduction in memory usage with minimal performance loss. We set $\delta^0 = 0$ to disable decayment, and used $\delta^0 = 0.3$ to highlight its impact.

This result suggests that the method effectively filters out low-impact cases while retaining those most beneficial for learning. This selective retention supports scalability in complex environments, where handling large volumes of experience is essential.

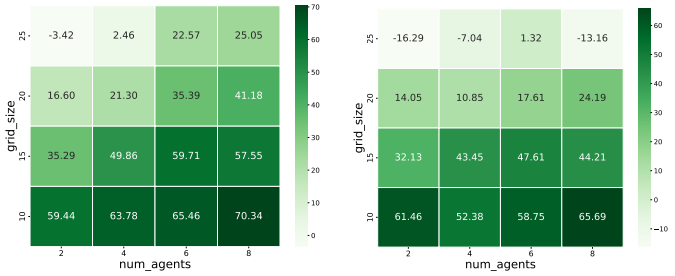


(a) Average number of stored cases

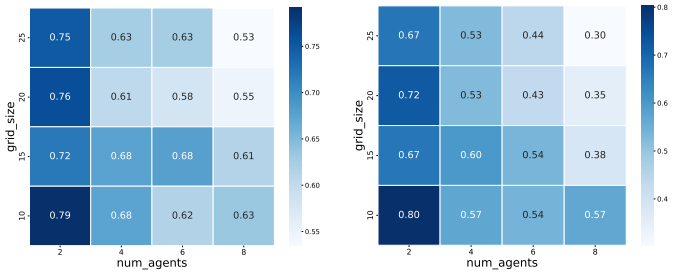


(b) Average rewards and success rate

Fig. 5: Impact of the decayment rule on memory and performance



(a) Average rewards: with experience sharing (left), without experience sharing (right)



(b) Average success rate: with experience sharing (left), without experience sharing (right)

Fig. 6: Heatmaps comparing reward and success rate across communication settings.

D. Experiment 3: Impact of Case Sharing under Varying Team Sizes

We evaluate D^{2}CBRL-DQN with and without case-based experience sharing across different team sizes (N agents) and grid dimensions ($G \times G$). For each combination of $N \in \{2, 4, 6, 8\}$ and $G \in \{10, 15, 20, 25\}$, we run 15 independent trials and report the mean and standard deviation of both average reward and success rate. As shown in Figure 6, enabling case sharing yields consistent performance gains in both metrics, with improvements growing as N and G increase. In contrast, disabling sharing incurs a marked drop in reward and success—especially for larger teams—highlighting the crucial role of shared experiences in scaling coordination and robustness in decentralized MARL.

VII. Conclusion

This paper introduced D^{2}CBRL-DQN, a hybrid framework that combines Distributed CBR and Decentralized MARL to improve adaptability under distributional shifts. By reusing relevant experiences through decentralized case sharing and employing selective case retention based on trust values, the approach enhances coordination, scalability, and memory efficiency. Its reliance on local observations increases robustness to localization uncertainty, making it suitable for real-world applications. Experimental results show that case sharing yields significant performance gains, especially in larger teams and environments. Future work will address resource constraints related to memory and communication bandwidth.

Acknowledgment

Co-funded by the European Union under the Marie Skłodowska-Curie Grant Agreement No 101081465 (AUFRANDE). Views and opinions expressed are however, those of the author(s) only and do not necessarily reflect those of the European Union or the Research Executive Agency. Neither the European Union nor the Research Executive Agency can be held responsible for them.

The authors would also like to acknowledge the support and collaboration of Naval Group.

References

[1] J. Quiñonero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, Eds., *Dataset shift in machine learning*, ser. Neural information processing series. Cambridge, Mass: MIT Press, 2010.

[2] T. Haider, F. S. Roza, D. Eilers, and K. Roscher, "Domain Shifts in Reinforcement Learning: Identifying Disturbances in Environments," *CEUR Workshop Proceedings*, vol. 2916, 2021.

[3] P. Altmann, F. Ritz, L. Feuchtinger, J. Nüßlein, C. Linnhoff-Popien, and T. Phan, "CROP: Towards Distributional-Shift Robust Reinforcement Learning Using Compact Reshaped Observation Processing," in *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, Macao, SAR China, Aug. 2023, pp. 3414–3422.

[4] C. Zhang, O. Vinyals, R. Munos, and S. Bengio, "A Study on Overfitting in Deep Reinforcement Learning," Apr. 2018, arXiv:1804.06893.

[5] K. Cobbe, O. Klimov, C. Hesse, T. Kim, and J. Schulman, "Quantifying Generalization in Reinforcement Learning," *Proceedings of the 36th International Conference on Machine Learning*, 2019.

[6] N. Justesen, R. R. Torrado, P. Bontrager, A. Khalifa, J. Togelius, and S. Risi, "Illuminating Generalization in Deep Reinforcement Learning through Procedural Level Generation," Nov. 2018, arXiv:1806.10729.

[7] B. Rank, S. Triantafyllou, D. Mandal, and G. Radanovic, "Performative Reinforcement Learning in Gradually Shifting Environments," in *Proceedings of the 40th Conference on Uncertainty in Artificial Intelligence*, 2024.

[8] Z. Liu, J. Lu, G. Zhang, and J. Xuan, "A Behavior-Aware Approach for Deep Reinforcement Learning in Non-stationary Environments without Known Change Points," May 2024, arXiv:2405.14214.

[9] F. A. Oliehoek and C. Amato, *A Concise Introduction to Decentralized POMDPs*, ser. SpringerBriefs in Intelligent Systems. Cham: Springer International Publishing, 2016.

[10] Harsha Avinash Bhute, "Implementing Swarm Robotics for Coordinated Multi-Agent Systems in Search and Rescue Operations to Improve Efficiency and Success Rates in Disaster Response," *Panamerican Mathematical Journal*, vol. 34, no. 4, pp. 532–549, Oct. 2024.

[11] J. Beck, R. Vuorio, E. Z. Liu, Z. Xiong, L. Zintgraf, C. Finn, and S. Whiteson, "A Survey of Meta-Reinforcement Learning," Aug. 2024, arXiv:2301.08028.

[12] T. P. D. Homem, P. E. Santos, A. H. Reali Costa, R. A. Da Costa Bianchi, and R. Lopez De Mantaras, "Qualitative case-based reasoning and learning," *Artificial Intelligence*, vol. 283, Jun. 2020.

[13] C. Finn, P. Abbeel, and S. Levine, "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks," Jul. 2017, arXiv:1703.03400.

[14] W. Luo, H. Li, Z. Zhang, C. Han, C. Zhou, J. Lv, and T. Guo, "Mitigating Distribution Shift in Model-based Offline RL via Shifts-aware Reward Learning," Oct. 2025, arXiv:2408.12830.

[15] A. Aamodt and E. Plaza, "Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches," *AI Communications*, vol. 7, no. 1, pp. 39–59, 1994.

[16] E. Plaza and L. Mcginty, "Distributed case-based reasoning," *The Knowledge Engineering Review*, vol. 20, no. 3, pp. 261–265, Sep. 2005.

[17] K. Zhang, Z. Yang, and T. Başar, "Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms," in *Handbook of Reinforcement Learning and Control*, K. G. Vamvoudakis, Y. Wan, F. L. Lewis, and D. Cansever, Eds. Cham: Springer International Publishing, 2021, vol. 325, pp. 321–384, series Title: Studies in Systems, Decision and Control.

[18] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," Dec. 2013, arXiv:1312.5602.