UKRAINIAN CATHOLIC UNIVERSITY

MASTER THESIS

Reinforcement Learning Agents in Procedurally-generated Environments with Sparse Rewards

Author:
Oleksii NAHIRNYI

Supervisor:
Dr. Pablo MALDONADO

A thesis submitted in fulfillment of the requirements for the degree of Master of Science

in the

Department of Computer Sciences
Faculty of Applied Sciences

Lviv 2022

Declaration of Authorship

“All that is gold does not glitter,
Not all those who wander are lost...”

J.R.R. Tolkien

UKRAINIAN CATHOLIC UNIVERSITY

Faculty of Applied Sciences

Master of Science

Reinforcement Learning Agents in Procedurally-generated Environments with Sparse Rewards

by Oleksii NAHIRNYI

Abstract

Solving sparse-reward environments is one of the most considerable challenges for state-of-the-art (SOTA) Reinforcement Learning (RL). Recent usage of sparse-rewards in procedurally-generated environments (PGE) to more adequately measure agent's generalization capabilities via randomization makes this challenge even harder. Despite some progress of newly created exploration-based algorithms in MiniGrid PGEs, the task remains open for research in terms of improving sample complexity. We contribute to solving this task by creating a new formulation of exploratory intrinsic reward. We base this formulation on a thorough review and categorization of other methods in this area. Agent that optimizes an RL objective with such a formulation performs better than SOTA methods in some small or medium sized PGEs.
Keywords: reinforcement learning, exploration, sparse rewards, procedurally-generated environment, intrinsic reward

Acknowledgements

I am thankful to my supervisor Pablo Maldonado for all patience and helpful feedback during the process of writing diploma.

I am also very grateful and feel highest respect to Oleksii Molchanovskyi for his major contribution into a golden mean of challenge and friendship inside this educational program. It is truly unique. It goes without saying that all my life will be followed by great memories and knowledge from study and cooperation inside such a great collective of UCU lecturers and students.

I express a deep gratitude to my all closest ones for being with me all this time and to Armed Forces of Ukraine for the ability to express at all.

Contents

List of Figures

List of Tables

List of Abbreviations

RL	**R**einforcement **L**earning
PGEs	**P**rocedurally **G**enerated **E**nvironments
SOTA	**S**tate-**o**f-**t**he-**a**rt (approaches)
MDP	**M**arkov **D**ecision **P**rocess
POMDP	**P**artially **O**bservable **M**arkov **D**ecision **P**rocess

Dedicated to Gosha, Jenny, Masya and Myshka . . . our lovely cats from Bucha.

Chapter 1

Introduction

1.1 Motivation

It is often the case in real-world that a human or animal is seeking to satisfy some prioritized goal. In reinforcement learning framework formulation it can be expressed via sparse reward piecewise function, e.g. 1 if the goal is reached and 0 otherwise ([31]). Also a path to the same goal in real-world can differ dependently on some inner or external context like time, space, brain development etc. Artifical procedurally-generated environments (PGE) attempt to simulate that feature of a reality. It is a well-established to be hard for an RL agent to solve tasks defined in a sparse reward manner and generalize to unknown situations. Combining these two problems into one makes RL research closer to solving real-world tasks. The topic in question exactly concerns such a problem and has recently given appearance to several new benchmarks and algorithms to solve them ([17], [18], [13], [16], [23]).

There is no exclusive list with all possible applications of above-mentioned algorithms, but it may contain a range of industries apart from games. A necessary condition of application is to build a simulation in the form of sparse-reward PGE for industries processes. Solutions then may be may be useful in the following areas:

- **Autonomous car driving**. A car may need to successfully or quickly get some distant goal with practically no reward on the way and different conditions at each step along the route [29]. However, infinite set of combinations of maps, road events, terrains, weather and other conditions is impossible neither to encounter over one's real lifetime nor code it manually. Moreover, field tests with real autonomous cars may be too costly or dangerous in terms of resources and lives. If realistic sparse-reward PGE is created, it may, for instance, cover such simulated scenario as *rare events handling*, when new town intersections and rare event types in different conditions are generated in order to test how system handles them [40]. Reward sparsity in such a case is ensured by giving any reward only on basis of how event was handled. Also the task of *lane keeping* [42] may be tested by agent in PGE as proposed by Gambi et al. [42]. Solutions to such tasks in sparse-reward PGE may be helpful to improve public safety on real roads.

- **Space exploration**. Apart from above-mentioned considerations on public safety, solutions to sparse-reward PGE simulations may be critical for successful unmanned spaceship completion of extraterrestrial missions with previously unseen landscape [41]. In particular, *unknown terrains investigation* in order to create a map of surroundings, cover it and find some goal is similar to hard exploration problem in sparse-reward PGE [41]. Training an agent in such sparse-reward PGE may be helpful afterwards in unpredictable field conditions. Depending on the environment configuration, reward in such case may only be located at one place of the map or just be a negative signal that some resource (like fuel) ran out etc. MarsExplorer environment for this purposes is already proposed by Koutras et al [41].;

- **Healthcare**. Simulated sparse-reward PGE environment in the form of unique patient cases flow may be used for training healthcare professionals [43]. Treatment of patients even with similar deceases can be modelled as sequential decision-making problem with each patient representing a certain context [29]. Thus, a trained RL agent may be used to help *monitoring professional diagnostics*. Reward in this case may be sparse in a sense of only signalling whether patient recovered or not.

1.2 Problem Background

1.2.1 Sparse rewards in RL

RL agent learns to solve tasks by getting positive or negative reward signal from environment. Let us recall that the "value" of state-action pair under policy π is represented by q-value function [1]:

$$q_\pi(s,a) = \mathbb{E}(G_t \mid S = s_t, A = a_t), \tag{1.1}$$

where $G_t = r_{t+1} + r_{t+2} + ... + r_T$ in episodic case, i.e. is a sum of rewards across all possible sequences of states and actions until last timestep T of an episode starting from timestep t. However, in many games and real-world scenarios most states contain no reward at all, i.e. reward function $r(s,a)$ is mostly zero. It may take a long time before RL agent learns how to distinguish valuable actions from whole sequence, given that reward is located far away in time from that action [5]. In such a case agent may get no signal on whether it approaches its goal and not learn optimal policy π^*. In extreme case, reward can be located only in one state ("goal-state") from all state space at some distant timestep. It can be seen from a definion of a sparse reward $r(s,a)$ given by Riedmiller et al. [28]:

$$\begin{cases} \delta_{s_g}(s) & d(s,s_g) \le \epsilon \\ 0 & d(s,s_g) > \epsilon \end{cases}, \tag{1.2}$$

where s_g - goal state; $d(s,s_g)$ - some distance measure between goal state s_g and current state; $\delta_{s_g}(s)$ - reward surface withing epsilon region. In extreme case, mentioned above, ϵ-value may be so small, that a reward surface set will only contain state goal s_g itself, i.e. only $d(s_g,s_g) \le \epsilon$. The problem may be solved by making a reward proportional to $d(s,s_g)$. However, such a reward design would require to build a precise measurement system[27]. Moreover, some environments are partially-observable Markov Decision processes (POMDP), where the state s may contain not all information about the final goal. Classical RL strategies of randomly taking actions with an intent to learn a transition dynamics $p(s_{t+1}, r_t | s_t, a_t)$ (as a way to solve exploration-exploitation dilemma) are generally not convergent to solution in such a case, which makes it a hard exploration problem. It requires from an agent to learn more effective strategies of exploration than in dense rewards environments.

1.2.2 Procedurally-generated environment

The main issue with environments most frequently used in up-to-date RL research (such as Atari-based) is a risk of overfitting: even a slight change of a game level can lead to agent failing with a task at hand. One of the reasons is that those environments have all game levels to be mostly manually designed with gradual transitions from one level to another with agent encountering same states during the

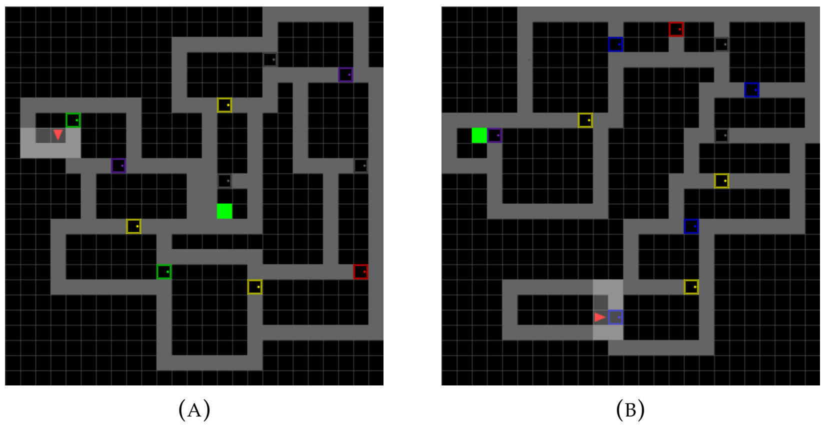


FIGURE 1.0: Two randomly (procedurally) generated levels from MiniGrid env with the same goal: reaching green cell. Source: D.Zha et al. Fig. 9 [15]

training. This scenario contradicts supervised learning methodology, which measures generalisation gap via the following function [29]:

$$GenGap(\phi) := \mathbb{E}_{(x,y)\ D_{test}}[L(\phi, x, y)] - \mathbb{E}_{(x,y)\ D_{train}}[L(\phi, x, y)], \tag{1.3}$$

where ϕ - model parameters; D_{test} - testing data; D_{train} - training data; L - loss function. The need to establish such benchmarks that will be able to measure algorithm's genuine generalization capabilities motivated usage of procedurally-generated environments. Cobbe et. al [18], inspired by Nichol et al. [20], proposed CoinRun, first PGE in RL research. In order to diversify skills needed to be learned, it was expanded to Procgen environment [19], having 16 game environments.

Main features of those environments are the following:

- Usage of random seed to generate game levels. It means that a certain game level seed would generate multivariate distribution where each variable relates to certain feature of game environment, like layout, enemies or entities location etc. At the same time the goal for two levels can be the same (finding some point on the maze) [Figure 1];
- Sequence of generated levels is random,i.e. there is no gradual difficulty increase, however one can switch on such a mode;
- In some environments like ProcGen authors separate/fix certain amount of holdout seeds from training seeds (similar to supervised learning) to measure a level of overfitting as a difference between training and holdout performance, however such methodology is not universal in research [19].

Above configuration makes harder for an agent to learn optimal behavior, given that an agent is highly unlikely to get to exactly the same state twice. PGEs train and test environment elements can be generated from same distribution, from different distribution or from combination of both. PGEs can be formally described in classic MDP-framework terminology via introduction of context space C, where each element of such space, context $c \in C$, may be a seed or any parameter vector defining a level (environment) [44]. Ghosh et al. defined a contextual MDP as an MDP M in which state s can decomposed into tuple $(c, s^{'})$, where $s^{'}$ is an underlying state, while in case of POMDP some emission function also transforms s [44]. If we split C into C_{train} and C_{test} then agent's objective is to maximize expected return on C_{test} with policy π [29]:

$$\underset{\pi}{\operatorname{argmax}} \mathbb{E}[r(\pi, M \mid_{C_{test}})] \tag{1.4}$$

Addition of sparse rewards into PGE setting makes this problem even harder to solve. The most popular benchmark in research among sparse-reward PGEs is MiniGrid [26], which is basically 2D-maze, uses procedure described above. See example in Figure 1 above. Having lower dimensions avoid some issues with computational resources in training an agent and allows to have greater freedom in design of experiments. At the same time, procedurally-generated sparse-reward nature of game allows to make conclusions about generalization capabilities of an agent. There are three main types of MiniGrid PGEs used in benchmarking: MultiRoom, KeyCorridor and ObstructedMaze. In MultiRoom an agent has to navigate through N empty rooms with different door colors. In KeyCorridor an agent needs to find a key via exploring multiple rooms, unlock certain door and take a green ball. ObstructedMaze contains distractor objects, boxes needed to be toggled for key retrieval and randomly connected rooms, making it one of the hardest in MiniGrid [16]. All these PGEs can have small, medium or large sized maps.

A main metrics used in measuring progress in those environments is a standard sample complexity, i.e. number of training steps or frames needed for an agent to solve environment, passing certain mean reward threshold. Also some researchers show 2D visual coverage to compare amount of explored space between algorithms.

There are also other sparse reward PGEs with higher dimensional (Obstacle Tower [30] etc.) or more complex grid dynamics (NLE [17] etc.), however principles of their work remain the same. Given computational simplicity, it is customary in RL research community to first improve results on MiniGrid benchmarks.

1.3 Research Goals

We have multiple goals to pursue in this work:

1) Given that there is no complete survey of methods used for the problem solution, we are aimed to provide it. It will include categorization, overview and comparison;

2) Based on above 1) we also aim to propose a new formulation of intrinsic reward comparable in its performance to at least some SOTA approches. As a "Proof-of-Concept" we will test our approach sample efficiency improvement in smaller size map grid-world PGEs, such as MultiRoom-N2S4,KeyCorridor-S3R1 or ObstructMaze-1Dl. A confirmation of method efficiency via its testing on medium size grid world and comparing it some SOTA algorithm is also one of our goals.

1.4 Structure of Master Thesis

In the light of above-defined goals pursuance we will include following parts into our work:

- Chapter 2 contains existing solutions categorization and sub-categorization with explanations, together with their overview and limitations. It also has a gap analysis, setting foundations for the proposed solution;

- Chapter 3 contains proposed solution description and its architectural components;
- Chapter 4 contains experimental setup, where we describe used metrics, environmental states-actions and how we compare SOTA approaches to our solution;
- Chapter 5 describes results in order to compare proposed solution from Chapter 3 with already existent SOTA described in Chapter 2. It also has a discussion part where some modifications are proposed and limitations considered. Based on discussion, directions of future research are proposed;
- Chapter 6 summarizes obtained results.

Chapter 2

Related Work

2.1 Categorization

Sparse-reward PGE solution approaches have a directed exploration strategy, i.e. in addition to extrinsic reward r^e provided by environment an agent also receives an intrinsic reward r^i, constituting a total reward:

$$r(s,a) = r^e + r^i. \tag{2.1}$$

Intrinsic reward here follows optimism in face of uncertainty principle, by which the higher the measure of state-action pair uncertainty the higher its value estimate [22]. Solutions evolve around the goal to create a "better" formulation of intrinsic reward.

They may be differentiated across several axes of comparison. However, the main categorization criterion is based on what type of uncertainty information about environment the agent uses for intrinsic reward definition itself. The most widespread one which has proponents in reinforcement learning [13] prefers dividing such information into novelty-based and prediction-based. Despite some confusion in literature with regards to usage of terms "prediction", "novelty", "curiosity" and "intrinsic motivation", above-mentioned classification is the most general one:

- Novelty-based approaches concentrate on measuring uncertainty with how "similar" a state-action pair or state is to some reference point. This "similarity" relies on different ideas, that can be divided into agent-centric (COUNT[6], RAPID[15],DoWhaM[16]), environment-centric (RIDE[13]) or some combination of both (C-BET[33]), depending on whether reference point is agent's own experience or some immanent interestingness of the state in question [Appendix A.1].

- Prediction-based approaches ground their intrinsic reward on how uncertain is agent's prediction model about some part of environment transition dynamics. They seek to minimize such model loss. At the same time they either reward agent for maximum deviation of the model predictions from certain ground truth (RND[9], ICM[10]) or reward agent for useful adversarial interaction (AMIGo[14], AGAC[23]) [Appendix A.2].

- Some combination of novelty and prediction based (NoveID[34]) [Appendix A.2].

Notably, this view on intrinsic reward classification is accepted in behavioral sciences research [32]. It is hypothesized that division into novelty-based and prediction-based reward is a real-world consequence stemming from environmental causal structure differences. As was noted by Dubey et al.[32], in environments, where states are not expected to be encountered second time in future, there is no incentive for an agent to learn any environment prediction model, but instead he is induced to find the most novel stimulus nearby. On the other hand, if environment states will be encountered in future multiple times it means that agent is motivated to learn prediction model to be able to navigate in such environment more effectively, being induced to find moderately novel stimulus [32] [Figure 2.1].

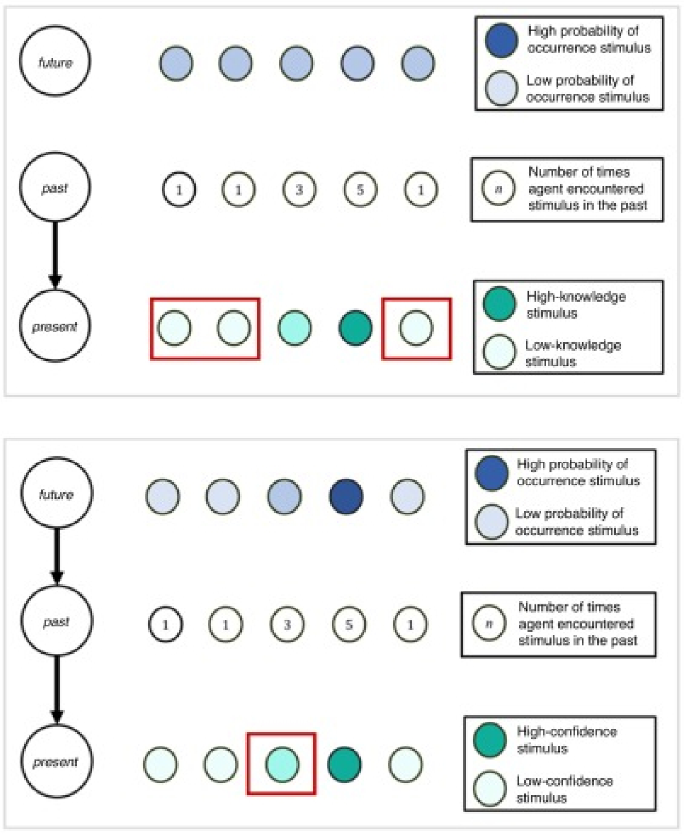

future
High probability of occurrence stimulus
Low probability of occurrence stimulus
past
1 1 3 5 1
n
Number of times agent encountered stimulus in the past
present
High-knowledge stimulus
Low-knowledge stimulus
future
High probability of occurrence stimulus
Low probability of occurrence stimulus
past
1 1 3 5 1
n
Number of times agent encountered stimulus in the past
present
High-confidence stimulus
Low-confidence stimulus

FIGURE 2.1: Top panel: An environment where future is independent of past. Here, it is optimal for the agent to be curious about novel stimuli in the present to maximize future rewards (highlighted in the red box, bottom row). Bottom panel: An environment where future is related to past and present. Here, it is optimal for the agent to be curious about moderately complex stimuli (highlighted in the red box, bottom row). Source: Dubey et al. Fig.2 [32]

During policy training all approaches are built upon some variation of existing SOTA actor-critic architectures, such as A3C or IMPALA, already benchmarked against dense-reward Atari environments.

2.2 Prediction-based approaches

2.2.1 Sub-categorization

Prediction-based approaches can be divided into two main sub-categories, based on two criteria of what is a prediction target of the model built for intrinsic reward and also how such reward is assigned to an agent:

- State-prediction: (prediction target is correct state distribution, while intrinsic reward is for not error in prediction):

  - ICM;

  - RND.

- Adversarial prediction (prediction target are other agent action distribution or skills, while intrinsic reward is for correct prediction) [Figure 2.2]:

  - AGAC;

  - AMIGo.

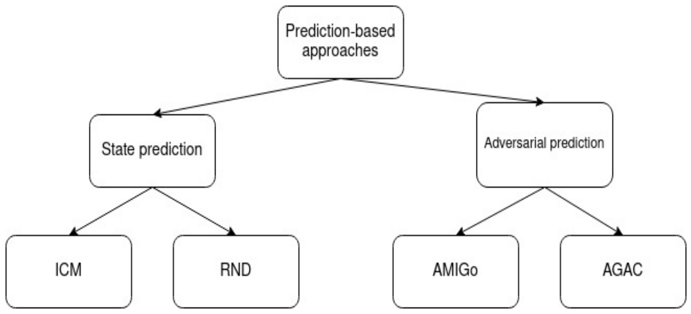


FIGURE 2.2: Sub-categorization of Prediction-based approaches.

2.2.2 ICM

In order to increase agent motivation in exploring different parts of environment one way is to reward it for visiting such states for which it is difficult to learn prediction model, but is simultaneously possible ("learnability"). Schmidhuber et al. [35] proposed an idea of intrinsic reward, that corresponds to the error between actual state s_{t+1} and predicted $\hat{s_{t+1}}$. However, the "learnability" of high-dimensional states is not evident.

To this end Pathak et al. [10] created ICM (Intrinsic Curiosity Model), such that an agent will receive no rewards for getting environmental states that are unpredictable by their nature. Their architecture contains training a sub-module with an embedding model ϕ encoding an arbitrary state s_t into such feature representation $\phi(s_t)$ that is more relevant for agent action a_t than raw data, i.e. decreased state dimensionality to the most important features. Such representation with the most important environment features is used simultaneously to train two other sub-modules [10]:

- Inverse-dynamics function f_{inv} that classifies an action used in transition from state s_t to state s_{t+1}, helping encoder to crystallize such feature encoding of environment that is important for actions of an agent:

$$f_{inv}(\phi(s_t), \phi(s_{t+1})) = \hat{a}_t; \tag{2.2}$$

- Forward-dynamics function f_{forw} that helps encoder to learn feature representation relevant for predictability of environment transition dynamics, by predicting feature encoding of the next environment state:

$$f_{forw}(\phi(s_t), a_t) = \hat{\phi}(s_{t+1}). \tag{2.3}$$

Intrinsic reward is defined using feature representation of next state prediction generated from forward-dynamics model [10]:

$$r_t^i = \frac{\eta}{2} \|\hat{\phi}(s_{t+1}) - \phi(s_{t+1})\|_2^2, \tag{2.4}$$

where $\eta > 0$ - scaling factor. Thus, the agent receives reward only for states that are valuable for his interaction with environment.

Limitation. In ICM you may encounter famous noisy TV issue: an agent get stuck in region where it sees random TV-noise due to huge intrinsic reward it gets [10].

2.2.3 RND

Burda et al. [9] made a progress in solving noisy TV issue with proposing quiet simple approach RND (Random Network Distillation) approach. Its idea is to learn predictor network to predict state embedding generated from random fixed target network. Then it gives an intrinsic reward for getting into unpredictable states to induce effective exploration and, at the same time, to solve above-mentioned problem. It works in the following way [9]:

1) Creating a randomly initialized fixed target network which outputs state embedding $f(s)$ at each time-step, given raw state s:

$$f : O \rightarrow R^k \tag{2.5}$$

2) Training a predictor network to output at each time-step such $\hat{f}(s, \theta)$ that minimizes mean squared error distance from target net output $f(s)$, which also constitutes an intrinsic reward:

$$r_t^i = \|\hat{f}(s, \theta) - f(s)\|^2, \tag{2.6}$$

This way, when target net gets into state representation similar to that it has seen before (even not exactly the same as in TV-noise), a trained predictor network will be able to find this out. It means that all random noise interactions are filtered out and agent is able to concentrate on more fruitful areas to be explored [9].

Limitation. The only issue with RND is that it's intrinsic reward vanishes, making it impossible for an agent to explore after some time, which can be critical in environments with many time-steps [9].

2.2.4 AMIGo

Campero et al. [14] noted that child exploration can be guided by goals generated while playing with the world. Moreover, such goals may be independent to motivation connected to extrinsic reward goals.

In order to reproduce such kind of exploration, the authors of [14] proposed new approach based on AMIGo (Adversarially Motivated Intrinsic Goals). It originally implements an idea of automated curriculum learning, adapting it to sparse PGEs. Agent here has an ability of goal self-proposal and is rewarded each time the goal is reached irrespective of the time passed, while AMIGo alleviates this problem. It also conditions the goal generator on observation in contrast to GoalGAN, given that each episode starts completely anew in PGEs [14].

AMIGo divides learning system into a student and teacher policies, where "student" makes actions in environment and "teacher" provides goals to "student". They are operated and optimized at different temporal frequency [14]:

- Student policy is stochastic and works at each time-step to sample action a_t via $\pi_\theta(a_t \mid s_t, g)$ conditioned with g and parameterized by θ_π, given state s_t. At each time step it receives intrinsic reward:

$$r_t^\pi(s_t, g) = \begin{cases} +1 & s_t = g \\ 0 & s_t \neq g \end{cases}, \tag{2.7}$$

  where g - goal state, generated by teacher policy and pursued by student policy. A student policy gets reward of one in an episode only if it reaches a goal at any moment during the episode. In the worst case scenario student will receive zeroes at all time-steps of an episode and in the best case it will be able to reach multiple goals during one episode.

- Teacher policy $G(s_0)$ parameterized by θ_g generates new goal (works) only at time-steps when the episode ends or student policy reaches old goal. It has a following intrinsic reward formulation:

$$r^G(t^*, t, r_t^S) = \begin{cases} +\alpha & t \geq t^* \\ -\beta & t < t^* \end{cases}, \tag{2.8}$$

  where t^* - time threshold such that the teacher policy is rewarded only when student makes more steps than a time threshold established in order to reach a set goal; t - current time index; α and β - hyperparameters, while t^* shall generally be fixed for particular environment. The rationale is that a "teacher" receives positive intrinsic reward α if generated task was not to easy or too hard for "student", otherwise it receives negative intrinsic reward β. In this context "too easy" means that goal was reached at timestep t before some fixed threshold t^*, while "too hard" means it was not reached at all by the end of an episode. This configuration has "constructively adversarial" nature, given that for success teacher has to make sure student success is possible.

In order to show "student" progress in goals "difficulty" the authors additionally decided to linearly increase t^* after some fixed amount of times when student reaches intrinsic goal. If "student" fulfills the goal before t^* it gets an intrinsic reward of one and otherwise it gets zero [14].

Limitation. The biggest problem with AMIGo is that it works only in fully-observable environments [14], as this goal from teacher takes a form of (x,y) coordinates.

2.2.5 AGAC

Adversarially Guided Actor-Critic (AGAC) was proposed by Flet-Berliac et al. [23], where a protagonist was added into the scheme of standard actor-critic architecture. Such protagonist goal is to mimic behavior of an actor, while goal of the actor is to make its behavior more unpredictable.

This scheme works using discrepancy between protagonist policy and actor policy action distributions. Discrepancy takes a form of Kullback-Leibner divergence [23]:

$$D_{KL}(\pi(\cdot \mid s) \parallel \pi_{adv}(\cdot \mid s)) = \mathbb{E}_{\pi(\cdot|s)}[\log \pi(\cdot \mid s) - \log \pi_{adv}(\cdot \mid s)], \tag{2.9}$$

where π - actor policy, π_{adv} - protagonist policy. The protagonist's mimicking goal is satisfied via minimizing discrepancy quantity, while actor's unpredictability goal is reached via maximizing it.

This discrepancy is used as an intrinsic reward to modify generalized advantage estimator A for each state-action pair in a trajectory used in actor loss computation [23]. Hyperparameter c is used to control dependence of advantange function on log probability difference. It is also added to critic and protagonist loss functions respectively. Thus, AGAC minimizes the following loss [23]:

$$L_{AGAC} = L_{actor} + \beta_{critic} L_{critic} + \alpha_{adv} L_{adv}, \tag{2.10}$$

where The terms βcritic and βadv are fixed hyperparameters for critic and protagonist losses.

Limitation. It appears that convergence of this method is highly sensitive to linearly annealed hyperparameter c, so that more robust process is needed.

2.3 Novelty-based approaches

2.3.1 Sub-categorization

Novelty-based approaches can be divided into two main sub-categories (plus combination of those), based on whether novelty of the state is computed based on some measure of inherent state attractiveness or agent own experience, which can also be measured in different manner given particular environment [Figure 2.3]:

- Agent-centric (based on agent own experience):
  - COUNT (based on state-action pair visitation count);
  - RAPID (based on episode exploration score);
  - DoWhaM (based on counting action impact);
- Environment-centric (based on state interestingness): RIDE;
- Combination of environment and agent centric (include both agent and environment reference point): C-BET.

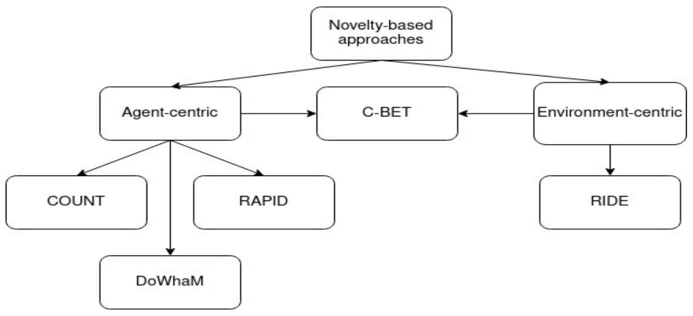


FIGURE 2.3: Sub-categorization of Novelty-based approaches.

2.3.2 COUNT

Main idea of visitation count is to keep track if each state-action occurrence count in order to encourage an agent in future visiting states with smaller amount of visits. It can be done via keeping track of quantity $N_n(s)$, i.e. number of occurences of state s in sequence of states $s_{1:n}$. In algorithmic setting intrinsic reward based on $N_n(s)$ quantity takes mostly one of the following forms [22]:

$$r^i = \begin{cases} \sqrt{\frac{\ln n}{N_n(s)}} \\ \frac{1}{\sqrt{N_n(s)}} \\ \frac{1}{N_n(s)} \end{cases}, \tag{2.11}$$

An idea of $N_n(s)$ square root can be traced back to Upper Confidence Bounds algorithm from Multi-armed bandits setting and Dyna-Q algorithm from model-based RL, where in both cases having an upper bound for deviation from mean extrinsic reward is intended [22].

One issue with pure visitation count is that it is applicable in practice only in low-dimensional tabular settings. In high-dimensional states probability of visiting particular state more than once is almost zero. Bellemare et al. [6] solve such an issue by using a density model which outputs pseudo-count $\hat{N_n}(s)$ per state in replace of pure state visitation count $N_n(s)$. Thus, variation of Markov model was applied. This approach is shortly referred to as COUNT in different benchmark comparisons. It works in the following order [6]:

1) treats state s as a factored state in which each (i, j) pixel is a factor $s^{i,j}$;

2) trains a location dependent model to output probability (conditioned on corresponding pixel upper-left neighbors) for $s^{i,j}$;

3) outputs a probability of state s as a product of all state factors $s^{i,j}$ probabilities.

In order to make density modelling more efficient one approach was proposed to replace Markov model by CNN [7].

Simplicity of pseudo-count $\hat{N_n}(s)$ and its asymptotic relationship with $N_n(s)$ [6] makes its usage in sparse-reward PGEs widespread. It is also used as a intrinsic reward discount in other approaches like RIDE.

Limitation. The issue with COUNT is its reward vanishing problem for visited states without knowing exactly whether such states are explored enough. As was noted by Burda et al. [9] there also exists an issue with scalability. In particular, it is not trivial to use such methods in the setting where multiple agents are learning from different environments in parallel.

2.3.3 RIDE

RIDE[13] learns forward-inverse dynamics and trains embedding network ϕ for feature extraction, similar to ICM [10]. However, prediction error of forward net is not used by RIDE in intrinsic reward formulation. Instead, it is defined in numerator as an $L2$-norm between successive (at timesteps t and $t + 1$) state embeddings $\phi(s)$, i.e. between important features of next and current state [13]:

$$\|\phi(s_{t+1}) - \phi(s_t)\|_2. \tag{2.12}$$

It is in contrast to ICM, where intrinsic reward r^i is based on $L2$-norm between predicted and actual state representations at timestep $t + 1$. The idea of exploration bonus in the form of consecutive states difference idea was already used in RL research, but contribution here is in using embeddings instead of raw data. Thus, RIDE stimulates agent to take actions that change important features of agent's environment surroundings, while ICM stimulates agent to reach more complex (unpredictable) subspace of environment with important features [13].

Formal definition of RIDE intrinsic reward also includes uses above-mentioned visitation counts idea as a discount to ensure that agent does not go back an forth between consecutive states in one episode [13]:

$$r^i_t(s_t, a_t) = \frac{\|\phi(s_{t+1}) - \phi(s_t)\|_2}{\sqrt{N_{ep}(s_{t+1})}}, \tag{2.13}$$

where $N_{ep}(s_{t+1})$ - number of times state s_{t+1} was encountered until time-step $t + 1$.

Limitation. One of RIDE constraint is that it receives intrinsic rewards for impact only one state ahead, however action at t can influence states that are far ahead. As was noted by Zhang et al.[34] there is also a problem with asymptotic inconsistency of RIDE because after training embedding network always some type of change can occur in the environment which can cause a environment change and bring intrinsic reward, so it never vanishes when number of training samples approaches infinity. It means that extrinsic reward r^e is not maximized, which contradicts main goal of

RL. One of the reasons is that visitation count value $N_{ep}(s_{t+1})$ is calculated without any regard to long-term novelty [34].

2.3.4 RAPID

Zha et al. [15] noticed that a human judges whether an agent explored environment sufficiently by looking at agent's behavior from an episode perspective. For example, you can tell how many rooms through which paths were visited and compare this behavior to another agent. It contradicts how most approaches (both novelty and prediction) assesses its own exploration quality, looking from a narrower perspective of whether particular state visitation was useful for environment space exploration. It motivated the authors of [15] to propose RAPID (Ranking the Episodes), a new method of encouraging exploration in environment which from more general perspective works in the following three steps [15]:

- Finding exploration score S per whole episode (instead of intrinsic reward per timestep);
- Recording in small buffer state-action pairs of best episodes ranked by their exploration score S;
- Using imitation learning to reproduce best exploration behavior from above-mentioned buffer.

Exploration score S of episode, being in essence episodic intrinsic reward $r^i_{episode}$, is computed via equation of three separate scores [15]:

$$r^i_{episode} = S = w_0 S_{ext} + w_1 S_{local} + w_2 S_{global}, \tag{2.14}$$

where S_{ext} - total extrinsic reward from environment, S_{local} - score from novelty of states encountered during episode with respect to each other, S_{global} - score for novelty of states encountered during episode with respect to states encountered during whole training process.

In order to make novelty assessment of an episode as a whole (instead of timestep-by-timestep basis) and extend novelty to continuous state space authors define episode exploration score as a mean standard deviation across all states in the episode [15]:

$$S_{local} = \frac{\sum_{i=1}^{l} std(s_i)}{l}, \tag{2.15}$$

where l - dimension of the state, $std(s_i)$ - standard deviation along the i-dimension of the states in the episode.

In contrast to S_local global exploration score should be computed per each state s_t during episode separately as this value is updated across all training and serves to establish how much more exploratory rich was this particular episode in the context of all previous experience [15]:

$$S_{global} = \frac{1}{N_{total}} \sum_s \frac{1}{\sqrt{N_{total}}}, \tag{2.16}$$

where l - dimension of the state, $std(s_i)$ - standard deviation along the i-dimension of the states in the episode. RAPID basically tries to overcome the constraint of RIDE rewarding agent for whole episode instead of impact for next state.

Limitation. A core issue with RAPID is introducing a bias on iterations by using state-actions from previous ranking, that were neither selected nor forgotten [15].

2.3.5 DoWhaM

Seurin et al. [16] provided following situation to motivate their new formulation of intrinsic reward and new exploration method DoWhaM (i.e. Don't Do What Doesn't Matter). Consider an infant that discovered a button that switched on a light. Authors of [16] hypothesized that such child will likely start to instinctively push everywhere in order to switch on new lights. In its turn, it can lead child to discover other buttons, learn common features between buttons and associate state "button" with action "push". Thus, it may be useful to encourage visiting states which allow to perform actions that can only be performed in rare occasions.

In order to implement this in RL setting, the authors of [16] proposed following intrinsic reward [16]:

$$r^i = \begin{cases} \frac{B(a_t)}{\sqrt{N_n(s)}} & s_t \neq s_{t+1} \\ 0 & otherwise \end{cases}, \tag{2.17}$$

where $N_n(s)$ - discussed above visitation count and used as a discount factor as in RIDE, and $B(a_t)$ is defined as follows:

$$B(a_t) = \frac{\eta^{1-\frac{E^H(at_t)}{U^H(a_t)}} - 1}{\eta - 1}, \tag{2.18}$$

where $\eta(\cdot)$ is an approximation of exponential decay, U is a number of times when action was taken and E is the number of times when action changed environment state, H - whole history of transitions (across all episodes) (s_h, a_h, s_{h+1}).

The main takeaway is that the lesser is ratio $\frac{E(a)_t}{U(a)_t}$ the higher is r^i, i.e. agent is rewarded for actions which cause rare environment state changes. An explanation of this formula is similar to above-mentioned example with infant: it may be useful for an agent to visit states, such that usually ineffective (not changing environment) actions become effective (changing environment). For instance, in most cases an action "grabbing non-existent objects" doesn't change environment state, but cases when it does may be interesting to explore [16].

Limitation. There is a problem with author assumption that only actions with rare environment effect has exploratory value. What if some frequently encountered and environment changing actions constitute a necessary condition for constant faster exploration and that's why they need to be repeated ? For instance, moving can be frequently performed action leading to environment changes and if we stop rewarding it we are hampering progress.

2.3.6 C-BET

Parisi et al. [33] assume that there are some states or scenarios that are inherently interesting, apart from what is perceived as interesting by agent subjectively. Thus, C-BET (Change-Based Exploration Transfer) approach was proposed in order to cover both scenarios. In the spirit of above-described DoWhaM approach, C-BET also favors transitions that produce rare environment change. It borrows pseudo-counts from COUNT approach and environment change idea from RIDE. Intrinsic reward here is the following [33]:

$$r^i = \frac{1}{N(s') + N(c)}, \tag{2.19}$$

where $c(s,s^{'})$ - environment change of transition $(s,a,s^{'})$, N - pseudo(counts) of changes and states. From this formula it is evident that the highest intrinsic reward will get those transitions $(s,a,s^{'})$ that are rarely encountered and rarely change environment.

C-BET training is divided into two stages [33]:

1) Agent learns exploration policy $f_i(s,a)$ across different environments with defined-above intrinsic-reward only ("task-agnostic"):

$$\pi_{exp}(s,a) = \sigma(f_i(s,a)), \tag{2.20}$$

where σ - softmax function;

2) Agent transfers this exploration policy to learn task policy $f_e(s,a)$ per each environment separately to maximize extrinsic reward. Exploration policy here works as a fixed bias for interaction, especially useful at the beginning of task policy training:

$$\pi_{task}(s,a) = \sigma(f_e(s,a) + f_i(s,a)), \tag{2.21}$$

It is assumed that after some training is passed agent will become more greedy to extrinsic reward with f_e having higher influence at final result [33].

In order to avoid situation when initial states of an episode will always get higher reward C-BET introduce random pseudo(count) reset. Hence, state pseudo(count) reset is not done each time when episode starts or ends, but has a probability p at each time-step instead [33].

Limitation. The biggest limitation of C-BET is a possible misalignment between exploration and task policies, as some exploration moves can slow down extrinsic reward discovery, e.g. harmful states encounter [33]. Also the same limitation here as in above-described DoWhaM - assumption that only rarely encountered actions that rarely change environment do have exploratory value.

2.4 Combination of Prediction and Novelty: NovelD

T.Zhang et al.[34] noticed that it is often the case when there are multiple regions of interest for exploration, many SOTA approaches get caught into one area very fast without sufficiently exploring others [Figure 2.4]. It is also known as a detachment problem in RL [12].

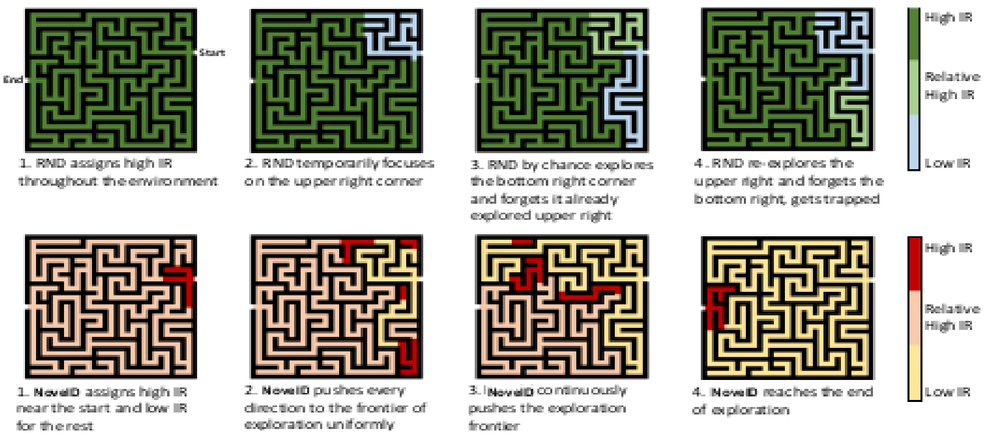

Start
End
1. RND assigns high IR throughout the environment
2. RND temporarily focuses on the upper right corner
3. RND by chance explores the bottom right corner and forgets it already explored upper right
4. RND re-explores the upper right and forgets the bottom right, gets trapped
High IR
Relative High IR
Low IR
1. NovelD assigns high IR near the start and low IR for the rest
2. NovelD pushes every direction to the frontier of exploration uniformly
3. NovelD continuously pushes the exploration frontier
4. NovelD reaches the end of exploration
High IR
Relative High IR
Low IR

FIGURE 2.4: Possible situations on exploration in NovelD (lower row) versus RND (upper row), with regards to intrinsic reward (IR) distribution. NovelD constantly expands the exploration boundary while RND gets caught in already over-explored areas. Source: T.Zhang et al. Fig.1 [34]

One of agent aims may be to explore environment with weighting explored areas equally in order to cover more state space. Thus, the authors of [34] propose new formulation of intrinsic reward which provides a bigger intrinsic reward at the boundary between the explored and the unexplored regions:

$$r_t^i = \max[RND(s_{t+1}) - \alpha \cdot RND(s_t), 0] \cdot \mathbf{1}[N_e(s_{t+1}) = 1], \tag{2.22}$$

where α - scaling factor, $N_e(s_{t+1})$ - episodic state count, RND - prediction from Random Network Distillation, already described previous section. Episodic state count is reset at each episode. From indicator function we can see that intrinsic reward is assigned at time-step t only if state is visited for the first time. In such a way it avoids going back and forth between unpredictable state and previous ones even in stricter way that RIDE.

NovelD also clips negative intrinsic reward if agent went back to same region. Also we can see that the closer is agent to area of it's exploration boundary the lower is reward, i.e. r_i is lower for all familiar states even if these were not visited before [34].

According to this definition next state is rewarded only if its predictability is lower than predictability of current state. From above equation we can derive some explored region as $\{s : RND(s) \leq m\}$ and can say that agent receives reward only if it explores beyond the boundary of explored regions. It becomes even more important across longer trajectories [34].

Limitation. A main limitation of NovelD is similar from RND limitation, which is vanishing intrinsic reward problem. It also more computationally intensive.

TABLE 2.1: Different sizes of MiniGrid environment used by SOTA. Small size environment were excluded from reports apart from one case in AMIGo (underlied by black), however AMIGo works only for fully-observed environment.

	MultiRoom	KeyCorridor	ObstructedMaze
RIDE	N7-S8,N10-S4,N12-S10,N10-S10	S3R3	2Dlh
AMIGo	-	S3R3, S4R3, S5R3	2Dlhb, 1Q, **1Dl**
RAPID	N10-S10, N7-S8, N10-S10, N12-S10, N7-S4	S3-R2, S3-R3, S4-R3	-
DoWhaM	N7-S4, N12-S10	S4R3,S5R3	2Dlh,2Dlhb,1Q,Full
AGAC	N10-S10	S4R3,S5R3	1Q,2Q,2Dlhb,
Other (C-BET,RND,ICM,COUNT)	same as above	same as above	same as above

2.5 Gap Analysis

We found following main gaps in solutions of sparse reward PGEs SOTA approaches :

- We created a Table 2.1 containing all MiniGrid envs tried by SOTA. We can see that SOTA methods are simply not reporting any results from small-sized environments (like KeyCorridor-S3R1, MultiRoom-N2S4 or ObstrMaze-1Dl (1Dl was used only in AMIGo and in not in standard partially-observable setting). However, smaller size of environment should not pose a problem for them if such methods are aimed at generalization capable agent.
- There exist no generally accepted quantitative metrics in PGE-oriented methods that can be used to report improvement in "exploration" per se. The most important part is still a sample complexity, but those methods use qualitative analysis to show increase in agent's state-space coverage, i.e. how much space is visually seen to be explored.

First gap fully corresponds to one of our initial research goals from Chapter 1: trying out improvements on MiniGrid envs with smaller maps, trying to surpass at least some SOTA methods. The second gap may be a subject of a different research, however we will provide our idea with regard to it in Discussion section in the next Chapter.

Chapter 3

Proposed approach: OneRIDE

3.1 Intrinsic reward formulation

Our approach OneRIDE is based on a new intrinsic reward formulation. The latter is derived from the combination of RIDE with a simple regularization idea from NoveID. It was mainly inspired by some limitations of those approaches mentioned in the previous section. OneRIDE benchmark results improvements in MiniGrid environments are discussed in the next section.

The formulation of intrinsic reward is the following:

$$r_t^i = \|\phi(s_{t+1}) - \phi(s_t)\|_2 \cdot \mathbf{1}[N_e(s_{t+1}) = 1], \tag{3.1}$$

where $\phi(\cdot)$ - embedding network for feature extraction (similar to RIDE embedding net) from current and next states, $\mathbf{1}[\cdot]$ - indicator function, $N_e(s_{t+1})$ - episodic count of next state occurrences.

Thus, our formulation inherits a novelty-based numerator from RIDE formulation, i.e. $L2$-norm between features of consecutive states. However, instead of using soft RIDE regularization $\frac{1}{\sqrt{N_e(s_{t+1})}}$ we propose more strict form of $\mathbf{1}[N_e(s_{t+1}) = 1]$ originated from NoveID [Figure 3.1].

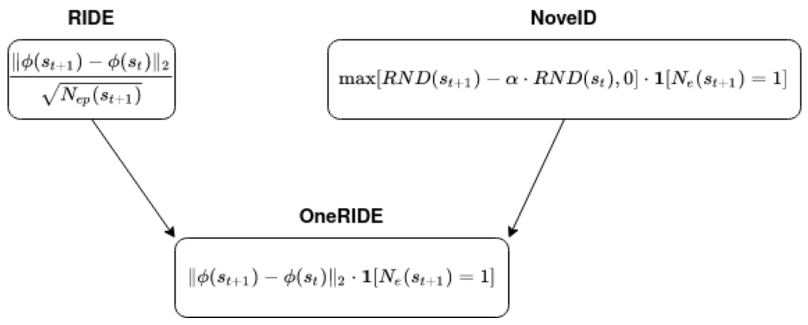

RIDE
$\frac{\|\phi(s_{t+1}) - \phi(s_t)\|_2}{\sqrt{N_{ep}(s_{t+1})}}$
NoveID
$\max[RND(s_{t+1}) - \alpha \cdot RND(s_t), 0] \cdot \mathbf{1}[N_e(s_{t+1}) = 1]$
OneRIDE
$\|\phi(s_{t+1}) - \phi(s_t)\|_2 \cdot \mathbf{1}[N_e(s_{t+1}) = 1]$

FIGURE 3.1: OneRIDE intrinsic reward inheritance from RIDE and NoveID.

There are three main advantages of OneRIDE intrinsic reward formulation over RIDE, NoveID or other SOTA approaches:

- OneRIDE inhibits repeated visits to already visited states with a simpler regularization criterion than that of NoveID via removing bounding $max[\cdot, 0]$ from formulation, leaving only indicator function $\mathbf{1}[\cdot]$. In fact indicator function is completely enough to encourage agent visiting not explored areas of space;

- At the same time OneRIDE decreases a level of asymptotic inconsistency, i.e. decreases an average intrinsic reward level in case of convergence to solution;

- It removes a RND neural network training layer from approach, making it less computationally complex.

Limitation. Main limitation concerns its strictness. OneRIDE discourages agent forever from returning to previously seen state withing an episode however sometimes such states or state-actions can be practical to return to in order to get out from environment trap or finding new trajectories. It concerns a problem of detachment described by Ecoffet et al [12].

Importantly, our intention was creating a novel formulation that can be easily combined with other machine learning ideas, such as adversarial agent, time-based bonus or attention. That is why Proof-of-Concept goal from Chapter 1 is regarded as close to satisfaction if the method beat some SOTA on some of smaller maps. We will look into such ideas more carefully in Discussion section.

3.2 Architectural details and scheme of work

OneRIDE follows RIDE and IMPALA in terms of scheme of work and architectural details. Experience collection (performed by Worker) is decoupled from optimization (performed by Learner) [39]. After some predetermined time of Workers using Policy network with old parameters (behavior policy), collected experience (s_t, a_t, r_t, s_{t+1}) goes to Learner in order to update parameters of Embedding, Forward, Inverse and Policy neural (target policy) networks. More detailed separate explanation of those networks will be below. In short, "mechanics' of their learning is the following [39] [Figure 3.2]. Embedding network extracts feature vector $\phi(s)$ from raw observation s_t and s_{t+1}, transmitting it to Inverse and Forward networks. Inverse net identifies what action a has been taken given $\phi(s_t)$ and $\phi(s_{t+1})$. Forward net predicts $\phi(s_{t+1})$ from $\phi(s_t)$ and a. Embedding net feature vectors $\phi(s_t)$ and $\phi(s_{t+1})$ are also used to calculate intrinsic reward. Policy network (target net) loss is calculated with collected experience and intrinsic reward. Networks are updated after total loss is calculated. Following loss function is minimized by gradient descent [13]:

$$L_{OneRIDE} = \omega_\pi L_{RL}(\theta_\pi) + \omega_f L_f(\theta_f, \theta_{emb}) + \omega_{inv} L_{inv}(\theta_{inv}, \theta_{emb}), \tag{3.2}$$

where θ - parameters of a corresponding neural network, ω - coefficient (hyperparameter), π - policy, RL - RL agent, f - Forward network, inv - Inverse network, emb - Embedding network, L - loss function. Each loss component will be explained below.

Embedding network $\phi_{\theta_{emb}}$. Its main aims are the following [13]:

- Reduce raw state space dimensionality via transforming it into feature space;
- Construct feature space in such a way that it contains only features relevant for minimizing loss of both Forward and Inverse dynamics networks as seen from Eq.(3.2).

Given that input state is an image and convolutional neural network (CNN) is a widespread way to process images, Embedding network $\phi_{\theta_{emb}}$ is a CNN that maps such input into a feature vector. It uses three layers with ELU activation after each, kernel size 3x3. First two layers have 32 filters, while the last one has 128. An output of 128-element vector is then separately transmitted into both Forward and Inverse neural networks [13]. Embedding network does not have its own loss function and its weights are updated via gradient of Eq.(3.2): $\nabla L_{OneRiDE}$.

Forward dynamics network f_{θ_f}. Forward network direct aim is to predict feature vector of next state s_{t+1} given current state s_t and action a_t [13]. Forward net is a feed-forward neural networks with one layer and ReLU activation. It gets s_t and action a_t in a concatenated form. Its loss is just an Euclidean distance between actual and predicted feature vector of next state s_{t+1} and is computed in the following manner [13]:

$$L_f(\theta_f, \theta_{emb}) = \|\phi_{\theta_{emb}}(s_{t+1}) - f_{\theta_f}(\phi_{\theta_{emb}}(s_t), a_t)\|_2. \tag{3.3}$$

Inverse dynamics network $inv_{\theta_{inv}}$. Inverse network learns to correctly classify action a_t taken in environment given feature vectors of current state s_t and next state s_{t+1} [13]. Inverse net is also a feed-forward net with one layer and ReLU activation, but with input in the form of two concatenated state feature vectors and output in the form of action value. Given a classification problem, loss function L_{inv} of Inverse network is a negative log-likelihood of true action a_t, which was taken in environment [13]:

$$L_{inv}(\theta_{inv}, \theta_{emb}) = -\ln\left[inv_{\theta_{inv}}(\phi_{\theta_{emb}}(s_t), \phi_{\theta_{emb}}(s_{t+1})) = a_t\right]. \tag{3.4}$$

Policy network. Two-headed policy network is used by RL agent to learn parameters θ_π with the following goals [13]:

1) To learn an optimal policy π^*_θ which outputs action distribution per each state s_t (classification head);

2) To learn an optimal state-value $V^*_\theta(s_t)$ of input state s_t (regression head).

Policy net training in OneRIDE is based on RIDE, which is derived from IMPALA actor-critic minimization of L_{RL} loss function [39]. Its main idea is to provide an off-policy correction to the time lag between learner policy π_θ (which is few steps ahead) and behavior actor policy μ_θ (which is basically an old learner policy taking actions in environment) in the form of V-trace target v_k for state-value approximation $V(s_k)$ [39]:

$$v_k = V(s_k) + \sum_{t=k}^{k+n-1} \gamma^{t-s}\left(\prod_{i=k}^{t-1} c_i\right)\delta_t V, \tag{3.5}$$

where $\delta_t V$ is temporal difference $p_t(r_t + \gamma v_{s+1} - V(s_t))$, p_t and c_i are truncated importance sampling weights equal to $min[1, \frac{\pi(a_t|s_t)}{\mu(a_t|s_t)}]$. This allows to decouple acting and learning more efficiently. For details and proofs see Section 4 and Supplementary material by Espeholt et al [39]). L_{RL} is comprised of three components [13]:

$$L_{RL} = L_{\pi_\theta} + L_{V_\theta} + L_{entropy}, \tag{3.6}$$

where L_{π_θ} - regression loss, L_{V_θ} - classification loss, $L_{entropy}$ - entropy bonus. Each component of L_{RL} is defined in the following manner [13]:

$$L_{\pi_\theta} = -\ln\left[\pi_\theta(a_t^{\mu_\theta} \mid s_t)\right] \cdot \delta_t V, \tag{3.7}$$

$$L_{V_\theta} = \frac{1}{2}(v_s - V_\theta(s_k))^2, \tag{3.8}$$

$$L_{entropy} = -\sum_{i=1}^{|A|} \pi_\theta(a_i \mid s_t) \ln\left[\pi_\theta(a_i \mid s_t)\right], \tag{3.9}$$

where $\pi_\theta(a_t^{\mu_\theta} \mid s_t)$ - probability of action $a_t^{\mu_\theta}$ taken by behavior policy μ_θ in state s_t given learner policy π_θ action distribution, A - action set, i - index of action from A,

$\pi_\theta(a_i \mid s_t)$ - probability of action a_i given the learner policy π_θ in state s_t. Importantly, we are not using Embedding network output to train Policy net, used only for Forward-Inverse networks. Policy net trains with its own separate embedding convolutional neural net to avoid agent intentionally maximizing distance between consecutive state embeddings to receive more intrinsic reward [13]. LSTM is used in RIDE Policy network as well as in OneRIDE in order to take account of time in partially observable MDPs after extraction of features from convolutional net. In comparison to RIDE we decreased number of LSTM layers to 1 from 2 and number of hidden neurons from 1024 to 256. Output of LSTM is transmitted to two Fully-connected layers which constitute a two "heads" of Policy net π. Number of hidden neurons is also decreased for those from 1024 to 256 in comparison to RIDE. Thus, from RIDE we leave only CNN untouched in Policy net. Such a configuration makes training less computationally intensive given limited amount of CPU resources that we had.

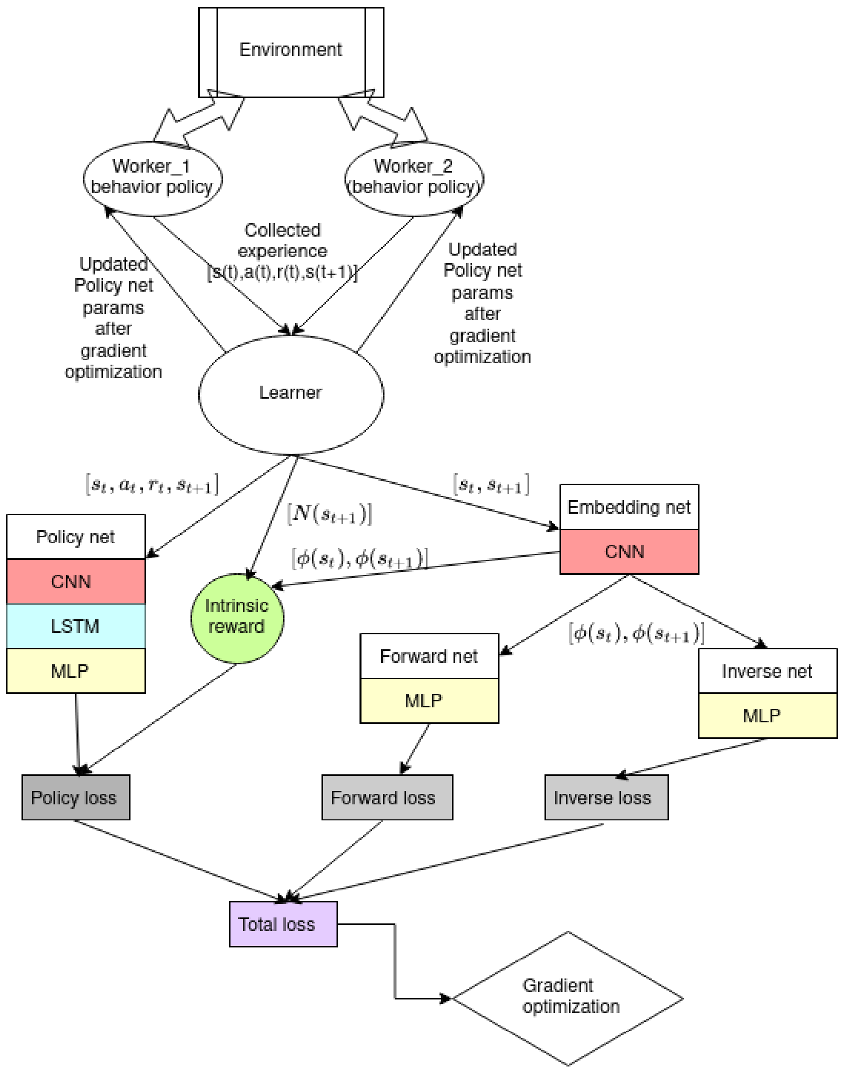

Environment
Worker_1 behavior policy
Worker_2 (behavior policy)
Collected experience [s(t),a(t),r(t),s(t+1)]
Updated Policy net params after gradient optimization
Updated Policy net params after gradient optimization
Learner
$[s_t, a_t, r_t, s_{t+1}]$
$[N(s_{t+1})]$
$[s_t, s_{t+1}]$
$[\phi(s_t), \phi(s_{t+1})]$
Policy net
CNN
LSTM
MLP
Intrinsic reward
Embedding net
CNN
$[\phi(s_t), \phi(s_{t+1})]$
Forward net
MLP
Inverse net
MLP
Policy loss
Forward loss
Inverse loss
Total loss
Gradient optimization

FIGURE 3.2: OneRIDE scheme of work based on IMPALA[39] and RIDE[13], where experience collection (performed by Worker) and optimization (performed by Learner) are separated. Collected experience from Workers (s_t, a_t, r_t, s_{t+1}) transmits to Learner in order to update parameters of Embedding, Forward, Inverse and Policy neural (target policy) networks.

Chapter 4

Experiments

4.1 Environments

We evaluate OneRIDE on such MiniGrid environments that are mostly often used by SOTA methods for benchmarking solution in the world of sparse-reward PGEs: MultiRoom, KeyCorridor and ObstructedMaze.

Our main metrics will be sample complexity, i.e. how quickly in terms of frames agent reaches a solution or some maximum bound. Solution is indicated by reaching stable flow of certain level episode returns. Average episode return is used as a standard in those PGE benchmarks and we follow the same rule. It is computed as the mean return over past finished episodes in N steps, where N in our case will be equal to 50. Agent receives return only when it reaches certain state. The faster agent reaches such state the bigger is the final return.

We concentrate on approaching our desired Proof-of-Concept from the Chapter 1, which is succeeding in sparse-reward PGEs with smaller maps. Each of above PGEs has some smaller version of its own, which are: MultiRoom-N2S4,KeyCorridor-S3R1 or ObstructMaze-1Dl.

- MultiRoom-N2S4. Solved when average reward is approximately equal to 0.8;
- KeyCorridor-S3R1. Solved when agent reaches mean reward of 0.9;
- ObstructMaze-1Dl. Solved when agent receives 0.9 on average.

Some description of those environments is given in Chapter 1. In general state s_t in MiniGrid world is NxM grid of tiles, where each tile is a three-dimensional tuple that contains object id, color id and state of an object [26]. Action set is discrete and contains the following actions: Turn left, Turn right, Move forward, Pick up an object, Drop the object, Toggle (open doors, interact with objects), Done (task completed, optional). Detailed description of actions and states in MiniGrid environments is provided by Chevalier-Boisvert et al. at [26]. We take into account that we do not need for our research goal satisfaction to be better than all or even most of SOTA approaches, but at least to approximate one of them. However, in order to confirm whether OneRIDE has a potential application in bigger map we will test it in medium-sized environment, such as "MultiRoom-N7S4".

4.2 Approaches comparison

We compare OneRIDE against two SOTA approaches (we do not need to compare to all of them according to our goal) such as RIDE and NoveID for the following reasons:

- OneRIDE intrinsic reward is a combination of those methods and it is natural to test it's performance against "parent" SOTA methods;
- NoveID authors claim that they were able to solve all large and medium-sized environment faster than other methods, being among top SOTA methods;
- RIDE is traditionally a baseline method used in benchmarking all other PGE-oriented methods and is the closest to OneRIDE in terms of the intrinsic reward formula.

We will use NoveID and RIDE neural net architectures and hyperparameters from specifications included into their papers and open-source implementation. RIDE we already described. NoveID, apart from having RND computation, uses additional LSTM layer in Embedding net training. NoveID CNN is bigger, as well as amount of neurons in its MLP.

Our computational constraints are 8 CPUs and 32Gb of memory RAM.

OneRIDE code implementation is open-sourced[1].

[1]*https://github.com/nalexus/OneRIDE*

Chapter 5

Results

5.1 Benchmarking

Smaller maps. Results of benchmarking are present in Figure 5.1 and Table 5.1. It is visible that in for all three small map PGEs OneRIDE approach demonstrates better results in terms of sample complexity than that of RIDE and NoveID.

In case of ObstructedMaze environment no algorithm was able to converge in a limited timeframe, which was expected as it is considered to be the hardest even among larger map variations. It is also clear that OneRIDE progress at the beginning is more substantial that of others. However, until one of the algorithms converges to solution we cannot be sure that such situation will not change in future to the favour of RIDE or NoveID. Taking such theoretical possibility into account, we are also not excluding a hypothesis that OneRIDE simple neural network architecture in combination with intrinsic reward formulation criteria plays some role in such a behavior.

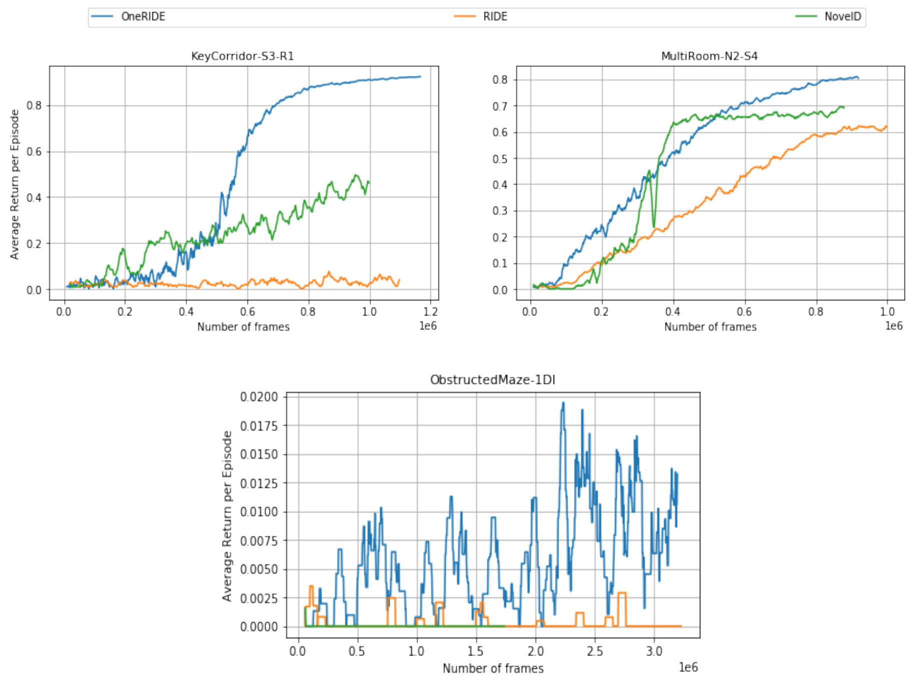


FIGURE 5.1: Sample complexity comparison between OneRIDE, RIDE and NoveID in environments with smaller map

Medium map. In order to confirm that OneRIDE has also ability to solve bigger MiniGrid maps we decided to look at its performance in medium-sized map, MultiRoom-N7-S4. We compare it only with RIDE as it is sufficient to prove above

TABLE 5.1: Compiled results from Comparison with SOTA for Mini-Grid environments

	MultiRoom-N2S4/N7S4	KeyCorridor-S3R1	ObstrucredMaze
OneRIDE	0.8 (Solved) in 800k steps	0.9 (Solved) in 1m steps.	0.01 is 3m steps
RIDE	0.6 in 800k steps	0.01 in 1m steps	0.0012 in 3m steps
NoveID	0.7 in 800k steps	0.45 in 1m steps	0.0 in 3m steps

hypothesis. As we can see from Figure 5.2, OneRIDE works even better in this particular environment than RIDE. It means that OneRIDE has a potential for testing in at least medium-sized MiniGrid environments.

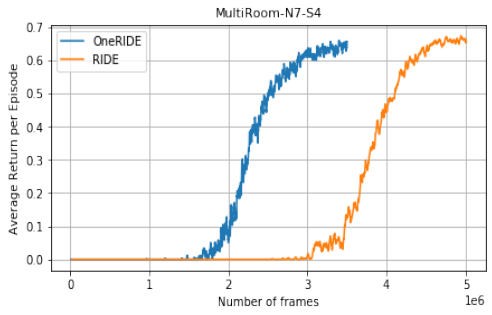


FIGURE 5.2: Sample complexity of OneRIDE vs. RIDE in MultiRoom-N7-S4 medium-sized map

It was established during experiments that asymptotic inconsistency of RIDE has less influence in case of OneRIDE. When average episode return approaches optimal solution average intrinsic reward of OneRIDE is smaller than that of RIDE [Figure 5.2]. It means that optimal solution in RL sense is not that affected by "eternal" flow of intrinsic reward.

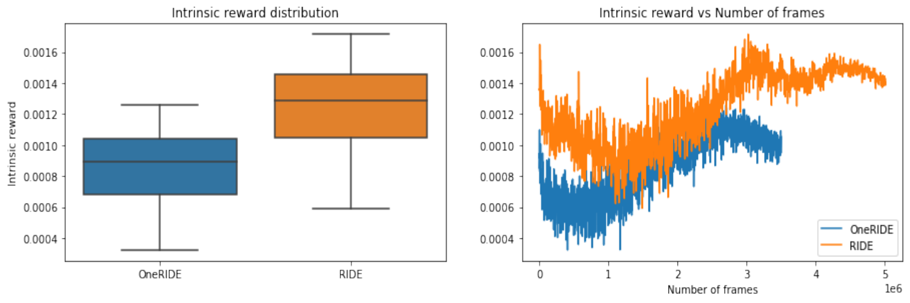


FIGURE 5.3: Intrinsic reward of RIDE and OneRIDE

5.2 Discussions and Future Work

Using benchmarking only against smaller or medium maps may trigger the question of applicability of such methods for large maps. It is an important issue to be resolved in a future work. On the other hand, does an agent really generalize if he is not able to solve smaller maps, but solves only larger ones ? Some SOTA methods in this field progressively increase an amount of computational power needed for an agent training, however one of implicit advantages of grid-world PGE are their simplicity for experiments. Hypothetically, OneRIDE should work in all other MiniGrid environments (apart from KeyCorridor,MultiRoom and ObstructedMaze), including those for which action that produces environment change may be harmful, such as LavaCrossing or Dynamic-Obstacles. Only RIDE among all SOTA methods tested and converged for Dynamic-Obstacles. LavaCrossing and other MiniGrid envs are not used in benchmarking at all by any SOTA approaches. Not benchmarking mentioned envs also constitutes a limitation of our proposed approach. Hence, we plan to include in our future work experimental setup all MiniGrid environments, that are not covered in this work.

During first episodes of training, agent may need to return to previously tried options for disclosing potentially unseen trajectories. Thus, OneRIDE has also a limitation of strict regularization that discourages agent forever from returning to previously seen state within an episode, invoking a detachment problem [12]. It may be useful for an agent to be able to get rewards for returning to states that he visited long time ago. Moreover, an idea of bonus for long tried state-action pair already exist from Dyna-Q+ method [37], which introduced an intrinsic bonus dependent on time passed from last time when certain state-action pair was active. Additionally, some threshold can be defined per areas of space when after certain time pass they all become exploratory attractive again.

Apart from this ideas, an interesting direction of thought could be using some adversary during training, generally comparable to AGAC or AMIGo approaches. For instance we, in one experiment of ours with smaller MultiRoom environment, where number of rooms equal 2 (not 7) an adversarial modification of OneRIDE showed reasonable perspectives in terms of sample complexity [Figure 5.4], however for bigger amount of rooms it did not confirm its applicability as OneRIDE. Such modification (OneAdversarialRIDE) replaces indicator function regularization of OneRIDE with classifier prediction of whether s_{t+1} has previously occured in the episode given only s_t. If from s_t it is possible to predict whether agent will get into already visited state then such state is not "novel" even being practically different in terms of distance. Such classifier $AdvClassifier$ is trained to correctly predict binary output, while policy network is encouraged to act in more unpredictable way from previous state flow:

$$r_t^i = \|\phi(s_{t+1}) - \phi(s_t)\|_2 \cdot \mathbf{1}[AdvClassifier(N_e(s_{t+1})) = 1], \tag{5.1}$$

We hypothesize that for OneAdversarialRIDE more hyperparameter configuration or some modifications of network architecture are needed to approach SOTA in at least medium size map, making it an another perspective direction of future research.

Finally, there was no attention mechanism applied in SOTA approaches for PGE, similar to that of Tang et al. [36], when agent is capable of dynamically using it to base its decision only on those pixel patches which were regarded as important by attention layer. However they report that their agent does not generalize to slightly changed environment, which directly concerns nature of PGEs. Some researchers suggest that attention is important for intrinsic motivation in a living organism [38]. Hence, integrating attention into existing SOTA approaches for solving sparse reward PGEs could be a promising choice of future work.

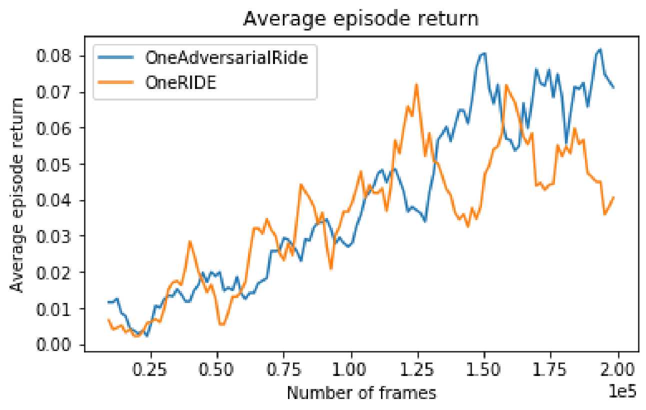


FIGURE 5.4: Average episode return of OneAdversarialRIDE and OneRIDE

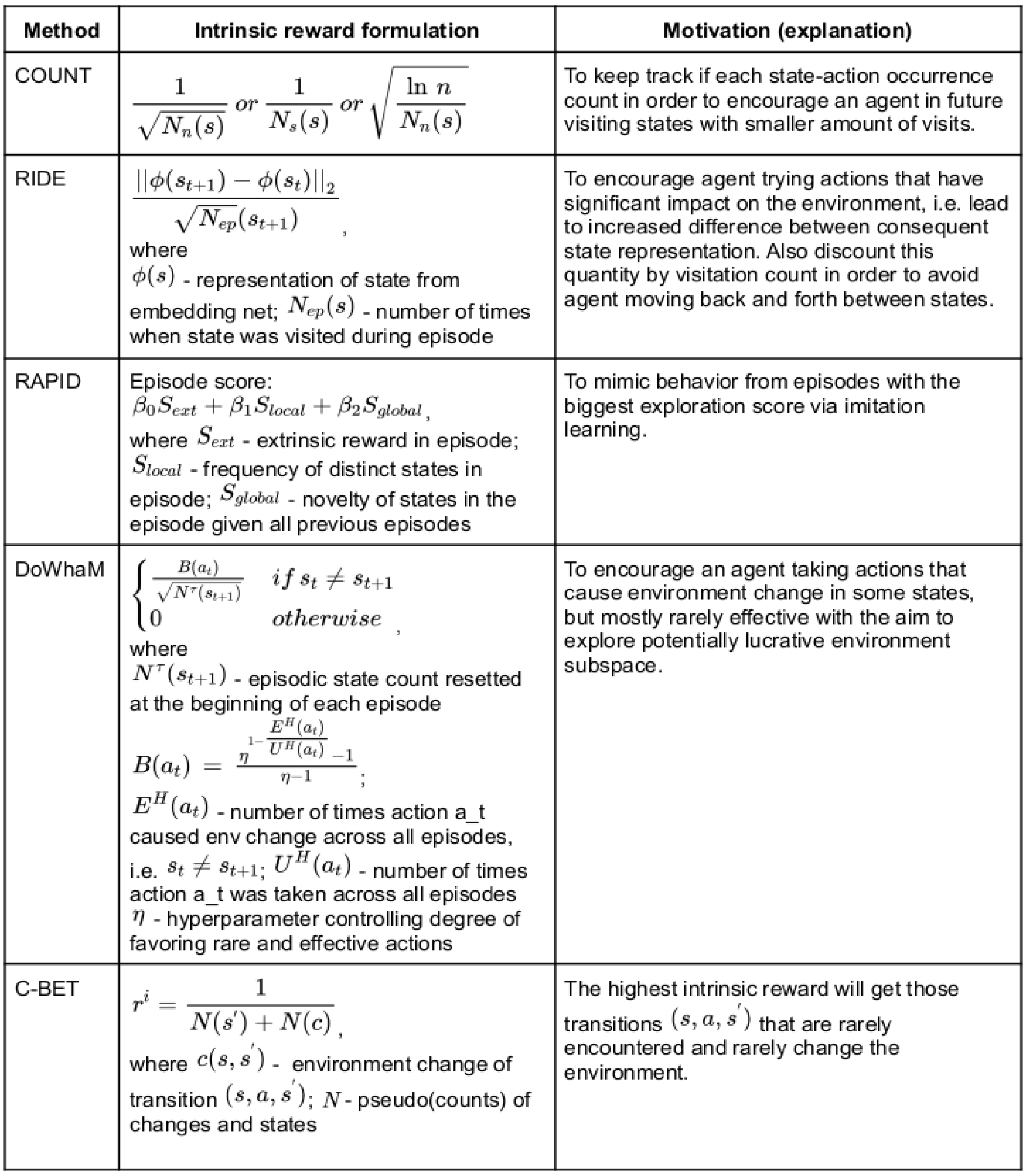
Chapter 6

Conclusions

Given information from previous Chapters, it may be concluded that we achieved the following results:

- We reviewed and categorized all sparse-reward PGE approaches thoroughly, choosing their most general classification based on the type of uncertainty information about environment;
- Proposed new intrinsic reward formulation OneRIDE based on some of the SOTA approaches, such as RIDE and NoveID. We also took into account some limitations of those approaches;
- In a set of experiments we showed OneRIDE agent capability of getting results comparable to SOTA in some small size PGE MiniGrid maps such as MultiRoom-N2-S4, KeyCorridor-S3R1. We also were able to confirm efficiency of our method in a medium size map MultiRoom-N7-S4 comparing it to RIDE;
- Discussed some limitations of our proposed method;
- Proposed potential improvements to OneRIDE intrinsic reward definition for the future work.

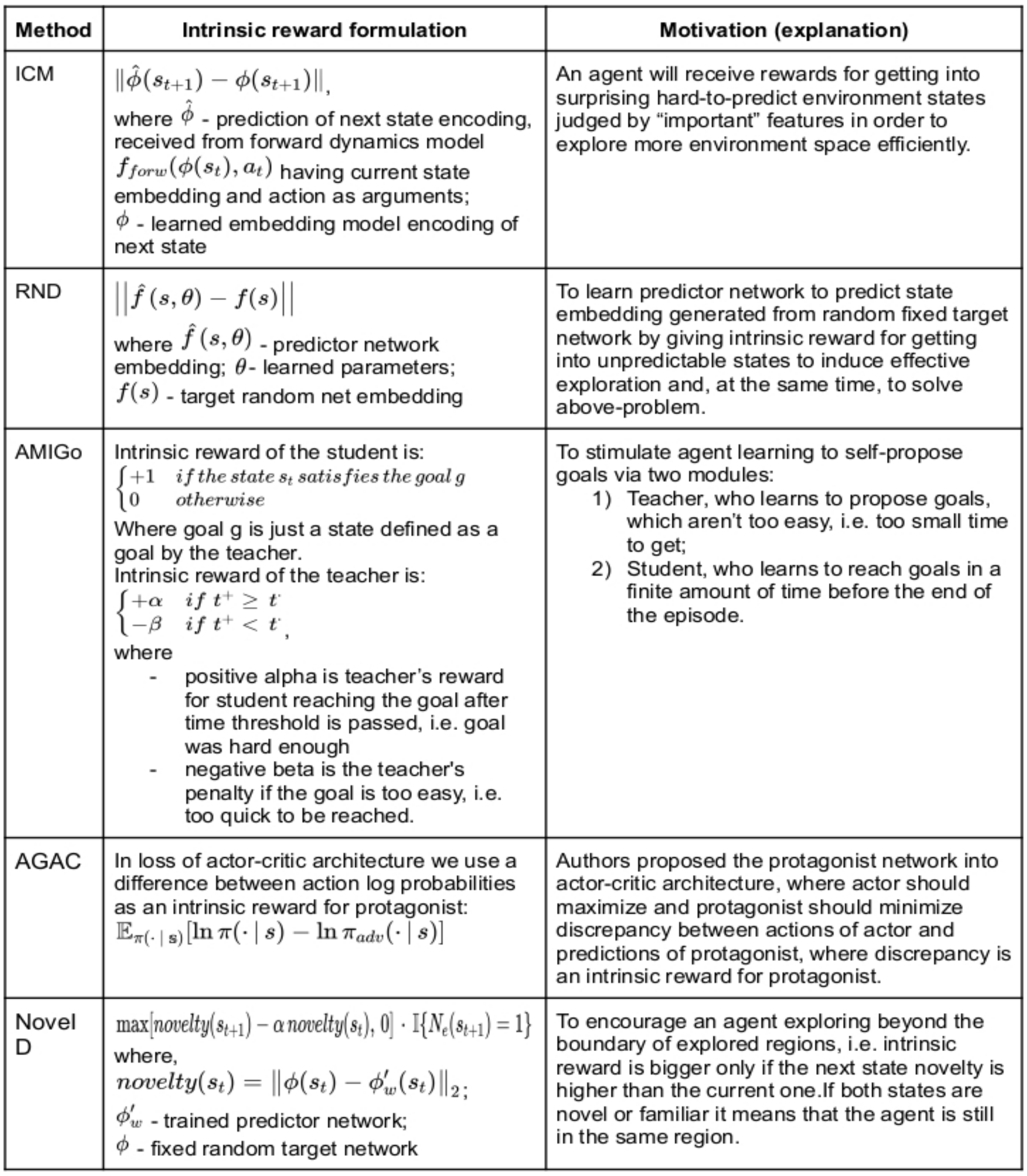
Appendix A



Method	Intrinsic reward formulation	Motivation (explanation)
COUNT	$\frac{1}{\sqrt{N_n(s)}}$ or $\frac{1}{N_s(s)}$ or $\sqrt{\frac{\ln n}{N_n(s)}}$	To keep track if each state-action occurrence count in order to encourage an agent in future visiting states with smaller amount of visits.
RIDE	$\frac{\lVert\phi(s_{t+1}) - \phi(s_t)\rVert_2}{\sqrt{N_{ep}(s_{t+1})}}$, where $\phi(s)$ - representation of state from embedding net; $N_{ep}(s)$ - number of times when state was visited during episode	To encourage agent trying actions that have significant impact on the environment, i.e. lead to increased difference between consequent state representation. Also discount this quantity by visitation count in order to avoid agent moving back and forth between states.
RAPID	Episode score: $\beta_0 S_{ext} + \beta_1 S_{local} + \beta_2 S_{global}$, where S_{ext} - extrinsic reward in episode; S_{local} - frequency of distinct states in episode; S_{global} - novelty of states in the episode given all previous episodes	To mimic behavior from episodes with the biggest exploration score via imitation learning.
DoWhaM	$\begin{cases} \frac{B(a_t)}{\sqrt{N^{\tau}(s_{t+1})}} & if\ s_t \neq s_{t+1} \\ 0 & otherwise \end{cases}$, where $N^{\tau}(s_{t+1})$ - episodic state count resetted at the beginning of each episode $B(a_t) = \frac{\eta^{1-\frac{E^H(a_t)}{U^H(a_t)}} - 1}{\eta - 1}$; $E^H(a_t)$ - number of times action a_t caused env change across all episodes, i.e. $s_t \neq s_{t+1}$; $U^H(a_t)$ - number of times action a_t was taken across all episodes η - hyperparameter controlling degree of favoring rare and effective actions	To encourage an agent taking actions that cause environment change in some states, but mostly rarely effective with the aim to explore potentially lucrative environment subspace.
C-BET	$r^i = \frac{1}{N(s') + N(c)}$, where $c(s, s')$ - environment change of transition (s, a, s'); N - pseudo(counts) of changes and states	The highest intrinsic reward will get those transitions (s, a, s') that are rarely encountered and rarely change the environment.

FIGURE A.1: Novelty-based approaches conceptual table: intrinsic reward definition and explanation per each method.

Appendix B



Method	Intrinsic reward formulation	Motivation (explanation)
ICM	$\|\hat{\phi}(s_{t+1}) - \phi(s_{t+1})\|$, where $\hat{\phi}$ - prediction of next state encoding, received from forward dynamics model $f_{forw}(\phi(s_t), a_t)$ having current state embedding and action as arguments; ϕ - learned embedding model encoding of next state	An agent will receive rewards for getting into surprising hard-to-predict environment states judged by "important" features in order to explore more environment space efficiently.
RND	$\|\|\hat{f}(s,\theta) - f(s)\|\|$ where $\hat{f}(s,\theta)$ - predictor network embedding; θ- learned parameters; $f(s)$ - target random net embedding	To learn predictor network to predict state embedding generated from random fixed target network by giving intrinsic reward for getting into unpredictable states to induce effective exploration and, at the same time, to solve above-problem.
AMIGo	Intrinsic reward of the student is: $\begin{cases} +1 & if\, the\, state\, s_t\, satisfies\, the\, goal\, g \\ 0 & otherwise \end{cases}$ Where goal g is just a state defined as a goal by the teacher. Intrinsic reward of the teacher is: $\begin{cases} +\alpha & if\ t^+ \geq t^* \\ -\beta & if\ t^+ < t^* \end{cases}$, where - positive alpha is teacher's reward for student reaching the goal after time threshold is passed, i.e. goal was hard enough - negative beta is the teacher's penalty if the goal is too easy, i.e. too quick to be reached.	To stimulate agent learning to self-propose goals via two modules: 1) Teacher, who learns to propose goals, which aren't too easy, i.e. too small time to get; 2) Student, who learns to reach goals in a finite amount of time before the end of the episode.
AGAC	In loss of actor-critic architecture we use a difference between action log probabilities as an intrinsic reward for protagonist: $\mathbb{E}_{\pi(\cdot\mid s)}[\ln \pi(\cdot\mid s) - \ln \pi_{adv}(\cdot\mid s)]$	Authors proposed the protagonist network into actor-critic architecture, where actor should maximize and protagonist should minimize discrepancy between actions of actor and predictions of protagonist, where discrepancy is an intrinsic reward for protagonist.
Novel D	$\max[novelty(s_{t+1}) - \alpha\, novelty(s_t), 0] \cdot \mathbb{I}\{N_e(s_{t+1}) = 1\}$ where, $novelty(s_t) = \|\phi(s_t) - \phi'_w(s_t)\|_2$; ϕ'_w - trained predictor network; ϕ - fixed random target network	To encourage an agent exploring beyond the boundary of explored regions, i.e. intrinsic reward is bigger only if the next state novelty is higher than the current one.If both states are novel or familiar it means that the agent is still in the same region.

FIGURE B.1: Prediction-based and Combined (NoveID) approaches conceptual table: intrinsic reward definition and explanation per each method.

Bibliography

[1] R.Sutton and A.Barto. "Reinforcement learning: An introduction (2nd ed.).". The MIT Press. 2020.

[2] R. McFarlane. "A survey of exploration strategies in reinforcement learning". URL: *https://www.cs.mcgill.ca/ cs526/roger.pdf*

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

[4] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, K. Kavukcuoglu. "Asynchronous Methods for Deep Reinforcement Learning". In *arXiv: 1312.5602* Feb. 2016.

[5] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z Leibo, D. Silver, K. Kavukcuoglu. "Reinforcement Learning with Unsupervised Auxiliary Tasks". In *arXiv: 1611.05397* Nov. 2016.

[6] M.Bellemare, S.Srinivasan, G.Ostrovski, T.Schaul, D.Saxton, R.Munos. "Unifying Count-Based Exploration and Intrinsic Motivation". In *arXiv: 1606.01868* Nov. 2016.

[7] Georg Ostrovski, Marc G. Bellemare, Aaron van den Oord, Remi Munos. "Count-Based Exploration with Neural Density Models". In *arXiv: 1703.01310* Dec. 2017.

[8] H.Tang, R.Houthooft, D.Foote, A.Stooke, X.Chen2, Y.Duan2, J.Schulman, F.De Turck, P.Abbeel. "Exploration: A Study of Count-Based Exploration for Deep Reinforcement Learning". In *arXiv: 1611.04717* Dec. 2017.

[9] Y.Burda, H.Edwards, A.Storkey, O.Klimov. "Exploration by Random Network Distillation". In *arXiv: 1810.12894* Oct. 2018.

[10] D.Pathak, P.Agrawal, A.Efros, T.Darrell. "Curiosity-driven Exploration by Self-supervised Prediction". In *arXiv: 1705.05363* May 2017.

[11] A.P.Badia, P.Sprechmann, A.Vitvitskyi, D.Guo, B.Piot, S.Kapturowski, O.Tieleman, M.Arjovsky, A.Pritzel, A.Bolt, C.Blundell. "Never Give Up: Learning Directed Exploration Strategies". In *arXiv: 2002.06038* Feb. 2020.

[12] A.Ecoffet, J.Huizinga, J.Lehman, K. O.Stanley, J.Clune "Go-Explore: a New Approach for Hard-Exploration Problems". In *arXiv: 1901.10995* Jan. 2020.

[13] R.Raileanu,T.Rocktäschel. "RIDE: Rewarding Impact-Driven Exploration for Procedurally-Generated Environments". In *arXiv: 2002.12292* Feb. 2019.

[14] A.Campero, R.Raileanu, H.Küttler, J.Tenenbaum, T.Rocktäschel, E.Grefenstette. "Learning with AMIGo: Adversarially Motivated Intrinsic Goals.". In *arXiv: 2006.12122* Feb 2021.

[15] D.Zha, W.Ma, L.Yuan, X.Hu, J.Liu. "Rank the Episodes: A Simple Approach for Exploration in Procedurally-Generated Environments.". In *arXiv: 2101.08152* Feb 2021.

[16] M.Seurin, F.Strub, P.Preux, O.Pietquin. "Don't Do What Doesn't Matter: Intrinsic Motivation With Action Usefullness.". In *arXiv: 2105.09992* May 2021.

[17] H.Küttler, N.Nardelli, A.Miller, R.Raileanu, M.Selvatici, E.Grefenstette, T.Rocktäschel. "The NetHack Learning Environment.". In *arXiv: 2006.13760* Dec. 2020.

[18] K.Cobbe, O.Klimov, C.Hesse, T.Kim, J.Schulman. "Quantifying Generalization in Reinforcement Learning.". In *arXiv: 1812.02341* Dec. 2018.

[19] K.Cobbe, C.Hesse, J.Hilton, J.Schulman. "Leveraging Procedural Generation to Benchmark Reinforcement Learning.". In *arXiv: 1912.01588* Dec. 2019.

[20] A.Nichol, V.Pfau, C.Hesse, O.Klimov, J.Schulman. "Gotta Learn Fast: A New Benchmark for Generalization in RL.". In *arXiv: 1804.03720* Apr. 2018.

[21] OpenAI, I.Akkaya, M.Andrychowicz, M.Chociej, M.Litwin, B.McGrew, A.Petron, A.Paino, M.Plappert, G.Powell, R.Ribas, J.Schneider, N.Tezak, J.Tworek, P.Welinder, L.Weng, Q.Yuan, W.Zaremba, L.Zhang. "Solving Rubik's Cube with a Robot Hand.". In *arXiv: 1910.07113* Oct. 2019.

[22] Susan Amin, Maziar Gomrokchi, Harsh Satija, Herke van Hoof, Doina Precup. "A Survey of Exploration Methods in Reinforcement Learning.". In *arXiv: 2109.00157* Sep. 2021.

[23] Yannis Flet-Berliac, Johan Ferret, Olivier Pietquin, Philippe Preux, Matthieu Geist. "Adversarially Guided Actor-Critic.". In *arXiv: 2102.04376* Feb. 2021.

[24] A.P.Badia, B.Piot, S.Kapturowski, P.Sprechmann, A.Vitvitskyi, D.Guo, C.Blundell. "Agent57: Outperforming the Atari Human Benchmark.". In *arXiv: 2003.13350* Mar. 2020.

[25] J.Li, W.Monroe, A.Ritter, M.Galley, J.Gao, D.Jurafsky. "Deep Reinforcement Learning for Dialogue Generation.". In *arXiv: 1606.01541* Sep. 2016.

[26] M. Chevalier-Boisvert, L. Willems, and S. Pal. "Minimalistic gridworld environment for openai gym". URL: *https://github.com/maximecb/gym-minigrid*

[27] G.Paolo. "Learning in Sparse Rewards settings through Quality-Diversity algorithms". Sorbonne Université, 2021.

[28] M.Riedmiller, R.Hafner, T.Lampe, M.Neunert, J.Degrave, T. Van de Wiele, V.Mnih, N.Heess, J.T.Springenberg. "Learning by Playing - Solving Sparse Reward Tasks from Scratch". In *arXiv:1802.10567* Feb. 2018.

[29] R.Kirk, A.Zhang, E.Grefenstette, T.Rocktäschel. "A Survey of Generalisation in Deep Reinforcement Learning". In *arXiv:2111.09794* Nov. 2021.

[30] A.Juliani, A.Khalifa, V.-P.Berges, J.Harper, E.Teng, H.Henry, A.Crespi, J.Togelius, D.Lange. "Obstacle Tower: A Generalization Challenge in Vision, Control, and Planning". In *arXiv:1902.01378* Nov. 2021.

[31] M.Vecerik, T.Hester, J.Scholz, F.Wang, O.Pietquin, B.Piot, N.Heess, T.Rothörl, T.Lampe, M.Riedmiller. "Leveraging Demonstrations for Deep Reinforcement Learning on Robotics Problems with Sparse Rewards". In *arXiv:1707.08817* Jul 2017.

[32] R.Dubey, T.Griffiths. "Understanding Exploration in Humans and Machines by Formalizing the Function of Curiosity". In *PsyArXiv* Jul 2020.

[33] S.Parisi, V.Dean, D.Pathak, A.Gupta. "Interesting Object, Curious Agent: Learning Task-Agnostic Exploration". In *arXiv:2111.13119* Nov 2021.

[34] T.Zhang H.Xu1 X.Wang Y.Wu K.Keutzer, J.E.Gonzalez, Y.Tian. "NovelD: A Simple yet Effective Exploration Criterion". In *Proceedings of 35th Conference on Neural Information Processing System: https://proceedings.neurips.cc/paper/2021/file/d428d070622e0f4363fceae11f4a3576-Paper.pdf* 2021.

[35] J.Schmidhuber. "A possibility for implementing curiosity and boredom in model-building neural controllers". In *From animals to animats: Proceedings of the first international conference on simulation of adaptive behavior* 1991.

[36] Yujin Tang, Duong Nguyen, David Ha. "Neuroevolution of Self-Interpretable Agents". In *arXiv:2003.08165* 2020.

[37] R.Sutton. "Integrated Modeling and Control Based on Reinforcement Learning and Dynamic Programming". In *Advances in Neural Information Processing Systems 3: https://proceedings.neurips.cc/paper/1990/file/d9fc5b73a8d78fad3d6dffe419384e70-Paper.pdf* 1990.

[38] Nazmul Siddique, Paresh Dhakan, Inaki Rano, and Kathryn Merrick. "A Review of the Relationship between Novelty, Intrinsic Motivation and Reinforcement Learning". In *https://doi.org/10.1515/pjbr-2017-0004* 2017.

[39] L.Espeholt, H.Soyer, R.Munos, K.Simonyan, V.Mnih, T.Ward, Y.Doron, V.Firoiu, T.Harley, I.Dunning, S.Legg, K.Kavukcuoglu. "IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures". In *arXiv:1802.01561* 2018.

[40] I.Paranjape, A.Jawad, Y.Xu, A.Song, J.Whitehead. "A Modular Architecture for Procedural Generation of Towns, Intersections and Scenarios for Testing Autonomous Vehicles". In *2020 IEEE Intelligent Vehicles Symposium (IV): https://ieeexplore.ieee.org/document/9304625* 2020.

[41] D.Koutras, A.Kapoutsis, A.Amanatiadis, E.Kosmatopoulos. "MarsExplorer: Exploration of Unknown Terrains via Deep Reinforcement Learning and Procedurally Generated Environments". In *arXiv:2107.09996* 2021.

[42] A.Gambi and M.Muller and G.Fraser. "Automatically testing self-driving cars with search-based procedural content generation". In *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis: https://dl.acm.org/doi/10.1145/3293882.3330566* 2019.

[43] J.Duffy and Z.Wang. "Application of Procedural Generation as a Medical Training Tool". In *Int'l Conf. Health Informatics and Medical System: http://worldcomp-proceedings.com/proc/p2015/HIM3115.pdf* 2015.

[44] D.Ghosh, J.Rahme, A.Kumar, A.Zhang, R.Adams, S.Levine. "Why Generalization in RL is Difficult: Epistemic POMDPs and Implicit Partial Observability". In *arXiv:2107.06277* 2021.