Learning to Play Hard Exploration Games Using Graph-guided Self-navigation

Enmin Zhao[1,2*], Renye Yan[2,1*], Kai Li[1], Lijuan Li[1], Junliang Xing[1,2†]
[1]*Institute of Automation, Chinese Academy of Sciences*
[2]*School of Artificial Intelligence, University of Chinese Academy of Sciences*
Beijing 100190, P. R. China
{zhaoenmin2018, yanrenye2018, kai.li, lijuan.li}@ia.ac.cn, jlxing@nlpr.ia.ac.cn

2021 International Joint Conference on Neural Networks (IJCNN) | 978-1-6654-3900-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/IJCNN52387.2021.9534251

***Abstract*—This work considers the problem of deep reinforcement learning (RL) with long time dependencies and sparse rewards, as are found in many hard exploration games. A graph-based representation is proposed to allow an agent to perform self-navigation for environmental exploration. The graph representation not only effectively models the environment structure, but also efficiently traces the agent state changes and the corresponding actions. By encouraging the agent to earn a new influence-based curiosity reward for new game observations, the whole exploration task is divided into sub-tasks, which are effectively solved using a unified deep RL model. Experimental evaluations on hard exploration Atari Games demonstrate the effectiveness of the proposed method. The source code and learned models will be released to facilitate further studies on this problem.**

***Index Terms*—Reinforcement Learning, Hard Exploration Games, Self-navigation, Graph-guided**

I. Introduction

Reinforcement learning (RL) of game playing with long time dependencies and sparse rewards remains a very challenging problem. During the early stage of the game, a game playing agent needs to explore the environment without any rewards. The first reward may occur at the middle or even the end of the game. On exploring the environment with a very long discrete sequence of actions, existing RL algorithms like DQN [1] and its extended variants [2]–[8] may not produce a good strategy if the rewards are sparse. One main reason is that most of these algorithms aim to explore for a single reward of a specific type and need the environments to provide sustained feedback to guide the exploration process. When the game reward is missing or is of an unusual type, these algorithms often struggle to find the correct direction to explore.

To address the sparse reward problem, some existing methods [9]–[12] learn from demonstrations using game replays of human replays [9]. Since the demonstrations from human replays are usually noisy, discrete, and asynchronous with the current game, new learning models built upon DQN [11] or policy optimization [12] have been developed. They yield impressive results on hard exploration Atari Games [1]. Considering the fact that human demonstrations are often not available in practical RL applications, it would be very useful to build upon the basic principles of human players when playing hard exploration games, instead of learning from human game replays.

Suppose that a man is trapped in a labyrinth. The only thing he can do to reach the exit is to explore the labyrinth using trial and error. To perform more efficient and effective exploration, he needs to remember the environment of the labyrinth as well as his position within it while moving around. He should also backtrack to a previous path if a specific exploration path fails. Inspired by these observations, we are motivated in this work to design an agent with these abilities.

In particular, the visual input of the game can be viewed as a grid map, which is similar to a map of an area using longitude and latitude. Different locations of the agent on the grid map represent different states of the game, and successive states establish an exploration path. To remember the environment, the agent needs to locate its position in the map, which can be learned by the agent from consecutive visual inputs [13]. To help the agent track the exploration paths, a graph-based representation is built by denoting different game states as graph nodes and successive state pairs as graph edges. This graph-based model has a much smaller size than the grid map, and is useful for following exploration processes.

To encourage the agent to explore the environment, we propose a new curiosity model in which intrinsic rewards drive the agent to learn. The basic idea behind existing curiosity based models [14]–[17], [17]–[20] is to give a reward for novel observations. In contrast, our curiosity model guides the agent to pay more attention only to those new observations that contribute to the search for the game extrinsic rewards. This model is inspired by human behavior which focuses on observations that directly affect the person [16]. With this new curiosity model and the graph representation, the agent explores the environment and keeps a record of its exploration. Through self-navigation of the agent in the environment, a complex exploration task is divided into several small and simple sub-tasks through the connectivity of the graph-based representation. Then a deep RL model with a unified model architecture is designed to learn policies for each sub-task. These policies can be easily learned without using much computation resource. To summarize, this work makes the following three main contributions:

- We introduce an effective exploration strategy for hard exploration games by encouraging an agent to navigate

*Equal contribution,† Corresponding author.

978-0-7381-3366-9/21/$31.00 ©2021 IEEE

using a graph to record the environment and agent's positions. The graph enables the tracing of the state-action history.
- We propose a new influence based curiosity model in which the agent pays more attention to new observations that guide the search for future game rewards.
- We design a new method for dividing a complex exploration task into several small and simple sub-tasks, and a unified architecture to perform efficient and effective reinforcement learning for policy updating.

Based on the above contributions, we achieve an effective algorithm for hard exploration game playing. On the *Montezuma' Revenge*, the algorithm obtains a score of 29,400, which surpasses all previous pure-exploration algorithms without using expert demonstrations. To facilitate further studies of this problem, we will release the experimental results, trained models, and source code of the proposed algorithm.

II. Related Work

The sparse reward phenomenon is common in many hard exploration tasks. Most previous works focus on improving the effectiveness of exploration. Simple heuristic methods such as ϵ-greedy [1] or Gaussian control noise [6] work well on a wide range of tasks, but they are inefficient in sparse reward environments such as those found in the Atari games like *Montezuma's Revenge* and *Venture*. For these games, standard RL algorithms perform poorly without finding even one reward. Policy gradient RL algorithms such as A2C/A3C [6], TRPO [4] and PPO [5] are also inefficient in exploration.

To tackle the exploration problem with sparse rewards, a natural idea is to use human replays or expert demonstration trajectories to guide the learning procedure [9]. One typical class of methods [21]–[23] formulate policies using demonstrations collected passively (behavioral cloning). Another class of methods [24]–[27] assumes an interactive expert to provide demonstrations in response to actions taken by the current policy. Researchers from OpenAI and DeepMind propose different methods based on learning from demonstration to devise policies for the *Montezuma's Revenge* with good results. The two-stage learning algorithm from DeepMind [9] extracts game features using two types of self-supervised objectives, and uses videos on YouTube to learn the game to 45,000 points. OpenAI applies imitation learning from the end of the video and progressively moves back in time as training proceeds [10]. This method turns an imitation learning problem into a sub-goals reinforcement learning problem and achieves 74,000 points.

Since the expert demonstrations are often not available, there is a need for RL algorithms that can learn from some internal motivation or rewards during game playing. Recently, curiosity-driven learning for environment exploration has yielded promising results for the sparse reward problem [16]–[20]. Different kinds of curiosity models have been proposed, however the basic idea behind these models is to give novel observations a reward bonus to encourage the agent to explore. The random network distillation method [28] employs the prediction error of a fixed randomly initialized neural network as an exploration bonus and outperforms average human performance on the *Montezuma's Revenge*.

Recently, Ecoffet [29] proposed the Go-Explore algorithm for the hard exploration Atari games, and obtained amazing high scores in the *Montezuma's Revenge*. Our algorithm shares the same idea which is to encourage the agent to explore environment using an intrinsic representation. However, our algorithm does not read the agent position from the game memory directly and the proposed processing pipeline is completely different from Go-Explore.

III. Graph-guided Self-navigation

For many labyrinth-like Atari games, environmental exploration is of crucial importance. In fact, environmental exploration is sometimes even more important than obtaining the rewards. Imagine that a man is locked in a very complex labyrinth and that the exit is a dark door. To get out of the labyrinth, the man needs to remember the structure of the labyrinth as well as his location within it. To reach the exit, he has to explore the path to it first. With the environmental structure in mind, he can backtrack to previous crossroads and continue exploration each time he meets a dead end.

Our model is inspired by the above human exploration procedure. Given a sequence of game playing images, the proposed approach consist of three steps. It first represents the game environment using a graph model defined over the game grid. The agent position is automatically located using the motion information of the target. Then the agent performs self-navigation using the graph-representation to explore the game and learns to obtain both the game extrinsic reward and the intrinsic rewards derived from an influence based curiosity model; Finally, the whole complex exploration task is decomposed into multiple simpler sub-tasks and a unified deep RL model is trained to deal with each sub-task.

A. Graph-based game representation

The input of many labyrinth-like games is a set of visual images $\mathcal{I} = \{I_1, I_2, \dots, I_t, \dots\}$, each of which can be viewed as a dense rectangular grid with the width W and height H. The agent chooses a set of predefined actions $\mathcal{A} = \{a_k\}_{k=1}^{K}$ to get through different game levels, where K is the number of different actions.

To perform game exploration, the agent needs to remember the structure of the game environment and its position within it. However, the agent does not know its position at first, but has to "guess" it while moving around the game environment. Unlike previous works [20], [29] that read the agent position from the game memory directly, in this work, a simple agent detection model is developed to learn the agent position using the agent motion information. When the agent moves around the game environment using different actions, the motion information in two consecutive frames is obtained by simple frame subtraction [30]. The detected motion areas may also contain some other parts of the game environment, but by sequential analysis of the motion area and agent actions, those

areas can be easily removed. Although more sophisticated approaches [31] can be used to learn the agent position, we find in the experiments this simple approach works well.

Given the agent position $p = (x, y)$ on the grid map, the game state is denoted as $\mathbf{s}$. By connecting different game states with an edge when the agent moves around the grid, a graph-based game representation is obtained for the game playing process. Denote this graph as $\mathcal{G} = \{V, E\}$, where $V = \{\mathbf{s}_n\}_{n=1}^{N}$ is the graph node set with N elements, each of which is one specific game state, and $E = \{(\mathbf{s}_i, \mathbf{s}_j), \dots\}$ is the directed graph edges connecting the adjacent game states when the agent moves from s_i to s_j by a specific action a. In a deterministic game like most hard exploration Atari games, executing a specific action in a given game state takes the agent to a deterministic, non-random, state. This graph-based game representation enables the agent to remember the structure of the game environment and game playing history. It will facilitate the game exploration and learning process.

B. Influence-based curiosity model

In the above graph-based game representation, different kinds of reward signals can be associated to the graph nodes and/or graph edges. In hard exploration tasks like *Montezuma' Revenge*, the game reward associated to the graph node is obtained only when the agent successfully passes a game level, which is not sufficient to drive the agent to perform game exploration. To deal with this problem, many existing works [16], [17], [19], [20], [28] define different kinds of intrinsic rewards on the graph node to drive the agent to explore. Among these works, the curiosity-based models [16], [17], [19], [20] generate additional rewards when the agent makes a novel observation on a graph node. Inspired by human behavior which shows more curiosity about changes that may affect the person, instead of defining the reward signal on the graph node, this paper proposes a new influence based curiosity model on the graph edges to ensure that the agent explores the environment efficiently.

Given an initial game state $\mathbf{s}_i$, if the game proceeds to state $\mathbf{s}_j$ after executing a specific action a, the intrinsic reward for the edge from state $\mathbf{s}_i$ to state $\mathbf{s}_j$ is denoted as $r_{\mathbf{s}_i \to \mathbf{s}_j}$, which is set in the following two situations:

- $r_{\mathbf{s}_i \to \mathbf{s}_j} > 0$: executing an action a at state $\mathbf{s}_i$ to proceed to state $\mathbf{s}_j$ results in positive influence or new environmental observations.
- $r_{\mathbf{s}_i \to \mathbf{s}_j} < -R$: executing an action a at state $\mathbf{s}_i$ to proceed to state $\mathbf{s}_j$ results in negative influence.

The above curiosity model encourages the agent not only to explore new game states as previous methods do, but also pay more attention to game states that will affect the agent itself. In Atari games, the influence to agent can be naturally defined using the remaining lives of the agent. Fig. 1 shows an example of an intrinsic reward for each state-action.

With the definition of the influence-based reward $r_{\mathbf{s}_i \to \mathbf{s}_j}$, the agent can perform game exploration using reinforcement

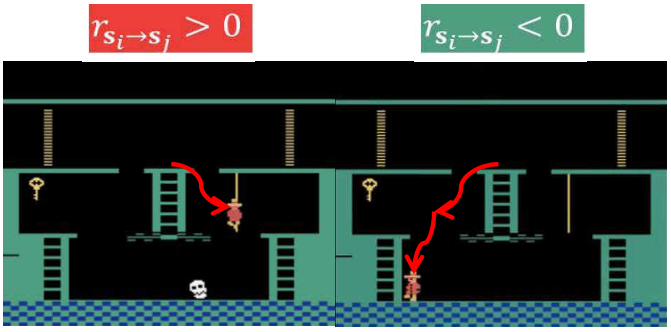


Fig. 1: An example of intrinsic rewards used in defining the curiosity model in the *Montezuma's Revenge*. The agent gets a positive intrinsic reward with a single action 'right jump' and a negative intrinsic reward with an action 'left jump'.

learning with the hybrid reward r, which is the sum of the intrinsic reward and the game extrinsic reward r_e,

$$r = r_e + r_{\mathbf{s}_i \to \mathbf{s}_j}, \tag{1}$$

From the above definition of the hybrid reward, it can be seen that the game reward item corresponding to a reward at the node of the graph representation, and the intrinsic reward corresponds to the reward item defined in the edge of the graph representation. The proposed curiosity model not only produces more game rewarding signals to make RL based game exploration feasible, but also points out the possible exploration directions for an RL algorithm. These benefits result in a new game playing paradigm. The agent can be guided more efficiently to the final game objective.

C. Task-decomposition for effective learning

With the hybrid reward signals defined over the graph-based representation, the agent is able to explore more game levels. In complex exploration tasks, it is still very challenging to learn the whole exploration task, as it requires an RL algorithm to explore in all possible directions and trajectories. Since the graph-based representation is very flexible in tracing the game states and exploration trajectories, by incorporating the proposed curiosity model, it can be used to record only the useful exploration history and to avoid multiple exploration of similar tasks. Based on these considerations, we propose a task-decomposition procedure leveraging the graph-based representation and the influence-based curiosity model to further improve the exploration effectiveness.

When exploring the environment, the game states associated with non-zero extrinsic and intrinsic rewards are recorded. In particular, given a game state $\mathbf{s}_i$, a specific action a is executed which causes the game to move to state $\mathbf{s}_j$ and yields a positive intrinsic reward $r_{\mathbf{s}_i \to \mathbf{s}_j} > 0$, or equally denoted as $r_{\mathbf{s}_i, a} > 0$. The game actions are continued until a negative intrinsic reward is obtained. The route is recorded as (p_s, p_e), where p_s is the start position and p_e is the end position. In principle, the agent should search such routes for all the reachable states and the corresponding action sequences. Fig. 2 shows an example

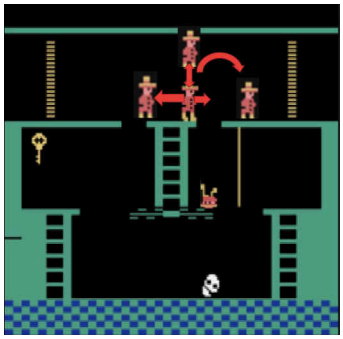


Fig. 2: An example of the route search. The agent will get a negative intrinsic reward if it goes down to the floor.

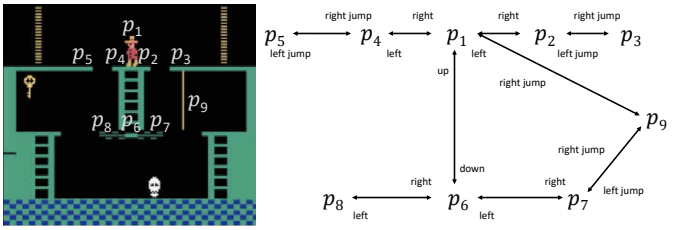


Fig. 3: An example of the graph route search path for *Montezuma's Revenge*. It helps the agent to quickly explore the environment.

of the route search. The agent will obtain a negative intrinsic reward if it goes down to the floor.

However, the structure obtained from this traversal state is too complex for the agent to directly apply it for learning. This work thus proposes an approach for updating the reward to simplify the structure of the graph-based representation. For one p_s, if there is only one p_e for all (p_s, p_e) in $r_{\mathbf{s},a}$ (The out-degree of p_s is only one), we get the sequence $(p_1, p_2, \ldots, p_n)$ which contains (p_s, p_e) and the same action and the out-degree of p_k is one. Then, we update $r_{s,a}$ as follows:

$$r_{\mathbf{s},a} \leftarrow r_{\mathbf{s},a} + \gamma r_{\mathbf{s}-1,a} + \gamma^2 r_{\mathbf{s}-2,a} + \cdots + \gamma^n r_{\mathbf{s}-n,a}, \quad (2)$$

where γ is the discounting factor. Then we can delete all the graph nodes inside the route path $(p_1, p_2, \ldots, p_n)$. The specific methods are described in Algorithm 1. In this way an agent is made to explore all the areas it has reached. After that, the agent finds the turning points in the game and forms a compact graph route search path. One example of graph route search path for *Montezuma's Revenge* is shown in Figure 3.

With the above self-negation process, every time the agent fails in an exploration process, a route path is formed in the graph-based representation. RL model is trained to help the agent to pass the exploration failure. In particular, denote the navigation route as $(p_1, p_2, \ldots, p_{k-1}, p_k)$ and the navigation action as $(a_1, a_2, \ldots, a_k)$. If in a certain step $m (1 \leq m \leq k)$, according to the navigation route (p_{m-1}, p_m) and the navigation action a_m, the agent gets the reward $r_{\mathbf{s}_{m-1} \to \mathbf{s}_m} < 0$, then the agent generates random experimental samples to the position p_m from the position p_{m-1} and trains a deep RL model using these samples with an RL learning algorithms.

Algorithm 1: Graph route path searching.

Input: The initial graph-based representation $\mathcal{G}$.
Output: Action set A, Route buffer B, Reward set R.
Initialize: $A \leftarrow \emptyset$, $B \leftarrow \emptyset$, $R \leftarrow \emptyset$
for *all states in* $\mathcal{G}$ **do**
  for $k = 1, \ldots, K$ **do**
    Get the position p_s from current state $\mathbf{s}$.
    Give a new action a_k.
    Update the proceeded position p_e.
    Get reward $r_{\mathbf{s},a_k}$ for this action.
    if $r_{\mathbf{s},a_k} > 0$ **then**
      $B \leftarrow \{(p_s, p_e)\}, A \leftarrow \{a\}, R \leftarrow \{r_{\mathbf{s},a_k}\}$
  for *all* p_s *in* B **do**
    if *there is only one* p_e *for all* (p_s, p_e) **then**
      Get the sequence $(p_1, p_2, \ldots, p_n)$ by R_{sr} which contains (p_{i-1}, p_i) and the same action a_k.
    Update $r_{\mathbf{s},a_k}$.
    Delete all the items in B with $r_{\mathbf{s},a_k} < 0$.
  Record A, B, and R in $\mathcal{V}$ and $\mathcal{E}$.

Take the A2C algorithm [6] as an example. The learning objective is,

$$\begin{aligned} \mathcal{L}^{a2c} &= E_{\mathbf{s},a\sim\pi_\theta}[\mathcal{L}^{a2c}_{policy} + \beta^{a2c}\mathcal{L}^{a2v}_{value}], \\ \mathcal{L}^{a2c}_{policy} &= -\log \pi_\theta(a_t|\mathbf{s}_t)(R - V_\theta(\mathbf{s}_t)) - \alpha H^{\pi_\theta}_t, \\ \mathcal{L}^{a2c}_{value} &= \frac{1}{2}\left\|(V_\theta(\mathbf{s}_t) - V^n_t)\right\|^2, \\ H^{\pi_\theta}_t &= -\pi_\theta(a|\mathbf{s}_t)\log \pi_\theta(a|\mathbf{s}_t), \end{aligned} \quad (3)$$

where $(x)_+ = \max(x, 0)$, θ is the model parameters, α and β are the regularizers. Driven by the learning algorithm, the agent can succeed in reaching the position p_m from the position p_{m-1} finding the optimal policies. The agent performs exploration with the navigation routes and actions until obtaining negative internal rewards. Algorithm 2 outlines the above procedure. Through this navigation map, the agent decomposes the complex whole game task into many smaller tasks, which can be easily learned with a light-weight neural network.

IV. Experiments

The proposed approach is evaluated on four hard exploration Atari 2600 games: *Montezuma's Revenge*, *Freeway*, *Hero*, and *Private Eye*. The following parts first introduce the implementation details, and then describe experimental evaluations to verify the effectiveness of the proposed method by ablation studies and comparisons with the state-of-the-art. Some discussions on the learned results are also provided in the last.

A. Implementation details

For the unified deep RL model, we adopt a three-layer convolutional neural network used in DQN [1] with the last

Algorithm 2: Task-decomposition for A2C learning.

Input: Action set A, Route buffer B, Reward set R.
Output: Learned parameter θ.
Initialize θ randomly.
while *NotEmpty(B)* **do**
  Take a sub-task from A, B and R into T.
  Get internal reward $r_{\mathbf{s}_i \to \mathbf{s}_j}$ from T.
  if $r_{\mathbf{s}_i \to \mathbf{s}_j} < 0$ **then**
    while *NotConverge* **do**
      Generate random experimental data.
      Perform one round of the A2C learning:
      $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}^{a2c}$.

4 stacked frames as input. Our basic RL algorithm is based on the A2C algorithm implemented from the OpenAI baselines [32]. The specific parameters of the network architecture and the A2C learning algorithm are shown in Table I. In all the experiments, the reported results of the proposed approach are all averaged over 5 trials of model learning.

TABLE I: Parameter settings in the experiments when learning the RL model using the A2C algorithm.

Parameter	Value
Network Architecture	Input: 210×160×4 Conv (32-8×8-4) Conv (64-4×4-2) Conv (64-3×3-1) FC (512) Output: # of game actions
Learning rate	0.0007
Number of environments	16
Number of steps per iteration	5
Entropy regularization (α)	0.01
Value loss regularization (β)	0.5
Discount factor (γ)	0.01

B. Ablation studies

To verify the effectiveness of the different components in the proposed algorithm, we perform ablation studies on *Montezuma's Revenge*. We compare three different variants of the proposed algorithm, including:

- A2C baseline from OpenAI's implementation (A2C).
- A2C+Graph-based game representation & Influence-based curiosity model (A2C+GDc).
- A2C+Graph-based game representation & Influence-based curiosity model & Task-decomposition for effective learning(Graph-guided Self-navigation) (A2C+GS).

By designing these three variants, we intend to verify the importance of the influence-based curiosity model and the task-decomposition for effective learning. The learning curves of these variants on *Montezuma's Revenge* are shown in Figure 4. The X-axis and Y-axis represent learning steps and the average game score, respectively. The baseline A2C algorithm failed to obtain even one game reward in this game. With the curiosity model defined over the graph-based representation, the agent succeeded in obtaining some game rewards. By learning a deep RL model for each sub-task, the agent obtained a final score of 24,500. These experiments show that our full model helps the deep reinforcement learning work in long time dependencies with sparse rewards well. The graph representation acts like a map to divide the whole task into small tasks and guide agents to get more rewards.

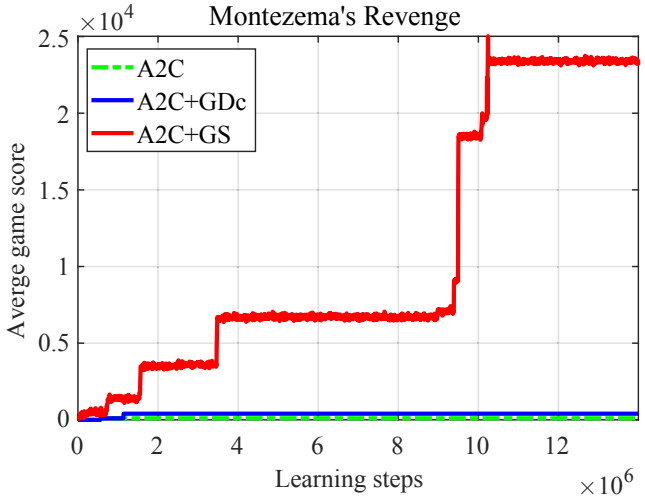


Fig. 4: Learning curves of different variants of the proposed learning algorithm on *Montezuma's Revenge*. Each learning step use 32 Atari frames.

Through the above experimental results, we can draw the following three observations: 1) the graph-based representation provides a general way for hard exploration game modeling; 2) the influence-based curiosity model increases the number of rewards and helps the agent learn better strategies; and 3) the task-decomposition for effective learning helps the agent to form navigation maps and divide a complex exploration task into some small and simple ones. Based on this divide-and-conquer strategy, the agent further effectively utilizes the advantages of the graph-based game representation and greatly simplifies the learning for game playing.

C. Comparisons with the state-of-the-art

To evaluate the overall performance of the proposed model, we first compare it against many state-of-the-art RL algorithms on different hard exploration games under standard game settings. The algorithms to be compared are: 1) the PPO algorithm [5]; 2) the self-imitation learning [33]; 3) the state-of-the-art count-based exploration actor-critic agents A3C-CTS [34]; 4) the Reactor-PixelCNN [34]; 5) the curiosity based method RND [28]; and the unified count and curiosity method UCC [19]. These methods imitate good experience which gets rewards or learns a density model of the observation. Similar to state-of-the-art count-based exploration actor-critic agents A3C-CTS [34] and Reactor-PixelCNN [34], our method has an explicit exploration bonus to encourage exploration. The experiment result is shown in Table II. The reported results of other methods are directly taken from the

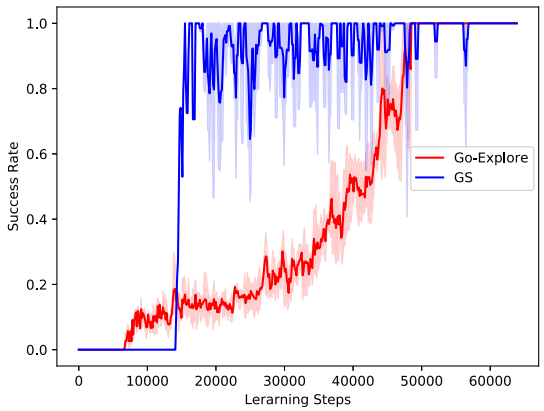


Fig. 5: Comparison with the Go-Explore approach on the first room of *Montezuma's Revenge*.

corresponding papers. The proposed approach obtains the best results among all the methods. It also outperforms the average human performance in all the games except *Private Eye*.

To compare with the recent Go-Explore method [29], we follow the experimental settings in the Go-Explore paper to retrain our model on *Montezuma's Revenge*. Three main experimental tricks are borrowed from Go-Explore to speedup the model training: 1) the agent position is directly read from the game memory; 2) the game running process is controlled to jump from different game states; and 3) simulated samples at different games states are generated to train the model. Figure 5 compares the success rate curves of Go-Explore and our model. Both methods divide the whole task into four subtasks and use the same A2C algorithm to train the strategy. We can see our model learns a good policy with fewer steps compared to Go-Explore.

D. Discussions

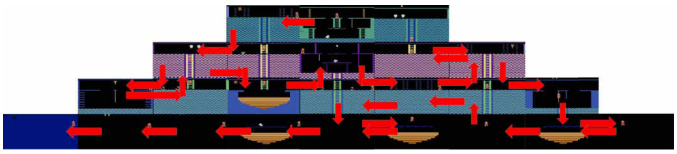
Finally, we analyze some learned results of the proposed model. Take *Montezuma's Revenge* as an example, in most experiments the first level is explored, but with game strategy differing from the one suggested in the game manual[1], in which all 24 rooms in the first level are visited. Our agent finds the quickest way out of the first level , which involves visiting only 15 rooms. This strategy is different from all the previous ones. The agent learned a special way to play *Montezuma's Revenge* and improve its scores over successive trials, in the same way as a human player.

Graph-guided self-navigation also helps the agent divide a complex task into some simple and small ones, which can be directly solved or learned quickly. If the goal of the agent is to get out of the first level rather than get as many rewards as possible, graph-guided self-navigation also helps the agent find the best strategy. Most existing reinforcement learning methods still need many experiments in which the agents traverse the first level. Graph-guided self-navigation helps agents decompose tasks effectively and continues the learning procedure efficiently. The graph-representation helps the agent record every task and self-navigation helps the agent find a solution to the whole task automatically.

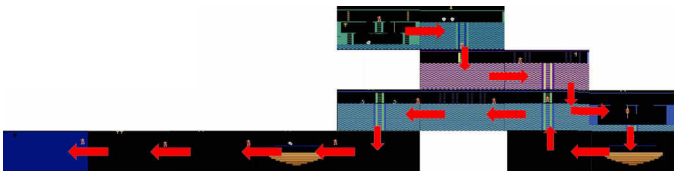
[1]https://en.wikipedia.org/wiki/Montezuma's_Revenge_(video_game)

TABLE II: Comparisons with the state-of-the-arts using averaged obtained scores on hard exploration Atari games.

	Montezuma	Freeway	Hero	Private Eye
PPO	2,497	32.5	-	105
A2C+SIL	2,500	**34**	33,069	8,684
A3C+CTS	400	30	15210	99
Reactor-PixelCNN	100	32	28,000	200
RND	8,152	**34**	-	8,666
UCC	3,439	30	34,880	15,806
Proposed	**24,500**	**34**	**38,604**	**33,628**
Avg. Human	4,753	16	28756	69,571



(a) The suggested strategy from the game manual for getting out the first level in *Montezuma's Revenge*.



(b) The learned sequence of the 15 rooms from the model for getting out the first level in *Montezuma's Revenge*.

Fig. 6: Comparison between the suggested strategy and the learned strategy for getting out the first level in *Montezuma's Revenge*.

V. Conclusions

In this paper we propose a graph-guided self-navigation approach for agent exploration. It contains a graph-based game representation for the game environment, an influence-based curiosity model, and a task-decomposition for effective learning. The graph-based game representation effectively simulates the game environment, influence-based curiosity helps the agent to increase curiosity about some areas of the graph and inspires the agent to explore these areas. The task-decomposition for effective learning divides a complex task into small and simple ones. In this way, the agent forms self-navigation maps for game exploration and learning.

VI. Acknowledgments

This work was supported in part by the Natural Science Foundation of China under Grant No. 62076238 and 61902402, in part by the National Key Research and Development Program of China under Grant No. 2019AAA010340X, in part by the CCF-Tencent Open Research Fund under Grant No. RAGR20200104, and in part by the Strategic Priority Research Program of Chinese Academy of Sciences under Grant No. XDA27000000.

REFERENCES

[1] M. Volodymyr, K. Koray, S. David, R. Andrei A, V. Joel, B. Marc G, G. Alex, R. Martin, F. Andreas K, and O. Georg, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[2] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy, "Deep exploration via bootstrapped DQN," in *Advances in Neural Information Processing Systems*, 2016, pp. 4026–4034.

[3] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, and N. De Freitas, "Dueling network architectures for deep reinforcement learning," in *International Conference on Machine Learning*, 2015, pp. 1995–2003.

[4] J. Schulman, S. Levine, P. Abbeel, M. I. Jordan, and P. Moritz, "Trust region policy optimization," in *International Conference on Machine Learning*, 2015, pp. 1889–1897.

[5] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[6] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Harley, T. Lillicrap, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International Conference on Machine Learning*, 2016, pp. 1928–1937.

[7] Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu, and N. de Freitas, "Sample efficient actor-critic with experience replay," in *International Conference on Learning Representations*, 2017, pp. 1–8.

[8] H. Tang, R. Houthooft, D. Foote, A. Stooke, X. Chen, Y. Duan, J. Schulman, F. DeTurck, and P. Abbeel, "Exploration: A study of count-based exploration for deep rein-forcement learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 2753–2762.

[9] Y. Aytar, T. Pfaff, D. Budden, T. L. Paine, Z. Wang, and N. de Freitas, "Playing hard exploration games by watching youtube," in *Advances in Neural Information Processing Systems*, 2018, pp. 2935–2945.

[10] T. Salimans and R. Chen, "Learning montezuma's revenge from a single demonstration," *arXiv preprint arXiv:1812.03381*, 2018.

[11] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband, G. Dulac-Arnold, J. Agapiou, J. Z. Leibo, and A. Gruslys, "Deep Q-learning from demonstrations," in *AAAI Conference on Artificial Intelligence*, 2018, pp. 3223–3230.

[12] B. Kang, Z. Jie, and J. Feng, "Policy optimization with demonstrations," in *International Conference on Machine Learning*, 2018, pp. 2474–2483.

[13] H. Kim, J. Kim, Y. Jeong, S. Levine, and H. O. Son, "EMI: Exploration with mutual information," in *International Conference on Machine Learning*, 2019, pp. 3360–3369.

[14] R. M. Ryan and E. L. Deci, "Intrinsic and extrinsic motivations: Classic definitions and new directions," *Contemporary educational psychology*, vol. 25, no. 1, pp. 54–67, 2000.

[15] P. J. Silvia, "Curiosity and motivation," *The Oxford handbook of human motivation*, pp. 157–166, 2012.

[16] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *International Conference on Machine Learning*, 2017, pp. 2778–2787.

[17] N. Savinov, A. Raichuk, D. Vincent, R. Marinier, M. Pollefeys, T. Lillicrap, and S. Gelly, "Episodic curiosity through reachability," in *International Conference on Learning Representations*, 2019, pp. 1–8.

[18] K. Gregor, D. J. Rezende, and D. Wierstra, "Variational intrinsic control," 2017, pp. 1–8.

[19] M. G. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos, "Unifying count-based exploration and intrinsic motivation," in *Advances in Neural Information Processing Systems*, 2016, pp. 1471–1479.

[20] C. Stanton and J. Clune, "Deep curiosity search: Intra-life exploration improves performance on challenging deep reinforcement learning problems," *arXiv preprint arXiv:1806.00553*, 2018.

[21] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *International Conference on Machine Learning*, 2004, pp. 1–8.

[22] U. Syed and R. E. Schapire, "A game-theoretic approach to apprenticeship learning," in *Advances in Neural Information Processing Systems*, 2008, pp. 1449–1456.

[23] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Advances in Neural Information Processing Systems*, 2016, pp. 4565–4573.

[24] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *International Conference on Artificial Intelligence and Statistics*, 2011, pp. 627–635.

[25] S. Ross and J. A. Bagnell, "Reinforcement and imitation learning via interactive no-regret learning," *arXiv preprint arXiv:1406.5979*, 2014.

[26] K.-W. Chang, A. Krishnamurthy, A. Agarwal, H. Daume III, and J. Langford, "Learning to search better than your teacher," in *International Conference on Machine Learning*, 2015, pp. 2058–2066.

[27] W. Sun, A. Venkatraman, G. J. Gordon, B. Boots, and J. A. Bagnell, "Deeply aggrevated: Differentiable imitation learning for sequential prediction," in *International Conference on Machine Learning*, 2017, pp. 3309–3318.

[28] Y. Burda, H. Edwards, A. Storkey, and O. Klimov, "Exploration by random network distillation," in *International Conference on Learning Representations*, 2019, pp. 1–8.

[29] A. Ecoffet, J. Huizinga, J. Lehman, K. O. Stanley, and J. Clune, "First return, then explore," *Nature*, vol. 590, no. 7847, pp. 580–586, 2021.

[30] J. Xing, L. Liu, and H. Ai, "Background subtraction through multiple life span modeling," in *International Conference on Image Processing*, 2011, pp. 2953–2956.

[31] J. Choi, Y. Guo, M. Moczulski, J. Oh, M. N. Neal Wu, and H. Lee, "Contingency-aware exploration in reinforcement learning," in *International Conference on Learning Representations*, 2019, pp. 1–8.

[32] P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, , and Y. Wu, "OpenAI baselines," 2017. [Online]. Available: https://github.com/openai/baselines

[33] J. Oh, Y. Guo, S. Singh, and H. Lee, "Self-imitation learning," in *International Conference on Machine Learning*, 2018, pp. 3875–3884.

[34] G. Ostrovski, M. G. Bellemare, A. van den Oord, and R. Munos, "Count-based exploration with neural density models," in *International Conference on Machine Learning*, 2017, pp. 2721–2730.