State-Temporal Compression in Reinforcement Learning With the Reward-Restricted Geodesic Metric

Shangqi Guo, Qi Yan, Xin Su, Xiaolin Hu, *Senior Member, IEEE*, and Feng Chen, *Member, IEEE*

Abstract—It is difficult to solve complex tasks that involve large state spaces and long-term decision processes by reinforcement learning (RL) algorithms. A common and promising method to address this challenge is to compress a large RL problem into a small one. Towards this goal, the compression should be state-temporal and optimality-preserving (i.e., the optimal policy of the compressed problem should correspond to that of the uncompressed problem). In this paper, we propose a reward-restricted geodesic (RRG) metric, which can be learned by a neural network, to perform state-temporal compression in RL. We prove that compression based on the RRG metric is approximately optimality-preserving for the raw RL problem endowed with temporally abstract actions. With this compression, we design an RRG metric-based reinforcement learning (RRG-RL) algorithm to solve complex tasks. Experiments in both discrete (2D Minecraft) and continuous (Doom) environments demonstrated the superiority of our method over existing RL approaches.

Index Terms—Semi-Markov decision process (SMDP), reward-restricted geodesic (RRG) metric, option, state compression, state-temporal compression, reinforcement learning (RL)

1 Introduction

Reinforcement learning (RL) has made significant progress in various tasks [1], [2]. However, when solving complex tasks involving large state spaces and long-term decision processes, traditional RL algorithms may encounter difficulties. First, large state spaces lead to high sample complexity [3], [4], [5] which represents the number of samples required to obtain a near-optimal solution with high probability. Second, long-term decision processes make exploration and temporal credit assignment quite inefficient [6], [7]. For instance, some tasks in Doom (a first-person video game) reward the agent only when it accomplishes a series of complex behaviors, such as finding a key and then opening a door in a large room. The reason for the two difficulties is that an RL problem is typically formalized as a fine-grained Markov decision process (MDP) in which the agent is assumed to execute one action in a raw state at each time step [8]. One approach to tackling these difficulties is to compress a large and fine-grained MDP into a small and coarse-grained one [9], [10], [11], [12].

Many works use compression in RL and they fall into three categories. The first category performs compression in the time domain with temporally abstract actions. Each temporally abstract action is associated with a policy over primitive actions and enables the agent to make decisions at a long time span instead of at a single time step [13]. RL with temporally abstract actions has been applied to solve tasks with long-term sparse rewards [6], [14], [15]. However, it remains a challenging problem to optimize the policy over temporally abstract actions for large-scale state spaces because the space size of temporally abstract actions scales with the state space. The second category performs compression in the state space by aggregating similar states [16], [17]. Nevertheless, state compression does not tackle the difficulty induced by long-term decision processes and sparse rewards. The third category performs compression in both the state space and the time domain [12], [18], [19], i.e., state-temporal compression. These methods, however, are either impractical or not optimality-preserving. Thus, state-temporal compression is still an open problem.

Since an MDP with temporally abstract actions is formulated as a semi-MDP [13, SMDP], state-temporal compression in an MDP can be transformed into state compression in an SMDP. The core of state compression is to measure the similarity between states [9]. Therefore, state-temporal compression can be carried out by measuring a state metric in an SMDP. Ferns *et al*. [20] proposed a bisimulation metric to measure the similarity between states in an MDP. They proved that the compression based on the bisimulation metric approximately preserves the optimality of the uncompressed

- *Shangqi Guo, Qi Yan, Xin Su, and Feng Chen are with the Department of Automation, Tsinghua University, Beijing 100086, China. E-mail: {gsq15, q-yan15, suxin16}@mails.tsinghua.edu.cn, chenfeng@mail.tsinghua.edu.cn.*
- *Shangqi Guo, Qi Yan, Xin Su, and Feng Chen are with the Beijing Innovation Center for Future Chip, Beijing 100086, China, and also with the LSBDPA Beijing Key Laboratory, Beijing 100084, China. E-mail: {gsq15, q-yan15, suxin16}@mails.tsinghua.edu.cn, chenfeng@mail.tsinghua.edu.cn.*
- *Xiaolin Hu is with the Department of Computer Science and Technology, Institute for Artificial Intelligence, Beijing National Research Center for Information Science and Technology, State Key Laboratory of Intelligent Technology and Systems, Tsinghua University, Beijing 100084, China. E-mail: xlhu@mail.tsinghua.edu.cn.*

Manuscript received 13 March 2020; revised 11 March 2021; accepted 21 March 2021. Date of publication 25 March 2021; date of current version 4 August 2022.
(Corresponding authors: Xiaolin Hu and Feng Chen.)
Recommended for acceptance by L. Li.
Digital Object Identifier no. 10.1109/TPAMI.2021.3069005

0162-8828 © 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: BEIJING UNIVERSITY OF TECHNOLOGY. Downloaded on December 05,2024 at 00:21:13 UTC from IEEE Xplore. Restrictions apply.

RL problem. This metric can be extended to an SMDP by incorporating multi-step transition models and rewards, but it is computationally difficult.

In this paper, we present a novel geodesic metric to measure the similarity between states in an SMDP. It is defined as the average minimum number of steps required by an agent to transition from one state to another. One of its advantages is that it can be learned by a deep neural network efficiently. We prove that compression based on the geodesic metric induces an approximately optimal policy of the uncompressed SMDP. To improve the accuracy of the approximation, we add a reward similarity restriction and name it the reward-restricted geodesic (RRG) metric. With compression based on the RRG metric, we design an RRG metric-based RL (RRG-RL) algorithm. Our experiments on complex tasks, where only specific complex sequences of actions were rewarded, demonstrated the superiority of our framework over existing RL algorithms. In particular, RRG-RL successfully solved a complex task in the Doom environment that cannot be solved by existing RL algorithms. In summary, our contributions are as follows:

- We propose an RRG metric to measure the similarity between states in an SMDP and prove that compression based on this metric induces an approximately optimal policy of the uncompressed SMDP.
- We propose a neural network to learn the RRG metric.
- We design an RRG metric-based RL (RRG-RL) algorithm to successfully solve some complex tasks.

The rest of our paper is organized as follows. In Section 2, we discuss the background and related work. In Section 3, we introduce the mathematical description of state-temporal compression and the state-temporal compression-based RL framework. In Section 4, we propose several metrics to perform state-temporal compression. In Section 5, we introduce an RRG-RL algorithm. In Section 6, we present the results of our algorithm in two environments. Finally, Section 7 concludes and discusses the paper.

2 Background and Related Work

An RL problem is typically formalized as an MDP $\mathcal{M}$, which is defined as a tuple $\langle S, A, p, r, \gamma\rangle$ consisting of a state space S, an action set A, one-step transition probabilities $p(s' \mid s, a)$, one-step rewards $r(s,a) = \mathbb{E}[r_{t+1} \mid s_t = s, a_t = a]$, and a discount factor $\gamma \in [0,1]$. The objective of RL is to maximize the expected cumulative reward $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_{t+1}]$ by optimizing a policy $\pi_A : S \times A \to [0,1]$ over actions.

Some difficulties exist in an MDP. First, the sample complexity of RL algorithms scales with the state space and the action space [3], [4], [5]. Second, long-term credit assignment remains a major challenge, especially in environments with sparse rewards, such as complex tasks in Minecraft [21], [22] and Doom [18], [23]. These difficulties lie in the fact that MDPs are fine-grained decision-making processes [9], [10], while real-life tasks involve large state spaces and long-term decision-making processes. One way to tackle these difficulties is to compress a large and fine-grained MDP into a small and coarse-grained one. Existing compression methods fall into three categories: temporal compression, state compression, and state-temporal compression.

2.1 Temporal Compression

Many researchers have used temporally abstract actions to solve complex tasks, which are difficult to solve by flat policies [6], [18]. There are many variants of temporally abstract actions, such as options [13], skills [22], subtasks [24], and sub-policies [21]. We adopt the term "option". In an MDP endowed with options, an agent is assumed to output an option associated with a policy over ground states and actions [13], [25]. The agent makes decisions over options at a larger span of time steps instead of a single time steps, which can be taken as temporal compression.

An option o is defined as a tuple (I_o, ω_o, β_o) consisting of three components: an initiation set $I_o \subseteq S$, an option policy $\omega_o : S \times A \to [0,1]$, and a termination function $\beta_o : S \to [0,1]$. Option o is available in state s if and only if $s \in I_o$. When option o is selected, ω_o produces actions until o terminates at state s according to $\beta_o(s)$. We assume that all options terminate in finite time with probability 1. Particularly, we introduce one important type of options—the subgoal option [6], [12], [13], [19]. It is defined as a tuple $g \triangleq (I_g, \omega_g, \beta_g)$ along with a subgoal state $s_g \in S$, where I_g is a set of initial states ensuring that option g can reach s_g; $\omega_g : S \times A \to [0,1]$ is a subgoal option policy (that is to reach s_g as quickly as possible); $\beta_g : S \to [0,1]$ is a termination function (that is to terminate g once s_g is reached).

Let $O(s)$ be the set of available options for each state $s \in S$ and $O \triangleq \cup_{s \in S} O(s)$ be the set of all options. An MDP endowed with O is modeled by an SMDP [13] defined as a tuple $\mathcal{N} = \langle S, O, P, R, \gamma\rangle$, where $P : S \times O \times S \to [0,1]$ and $R : S \times O \to \mathbb{R}$ denote multi-step transition models and rewards for options, respectively. R and P indicate the outcomes of an option at many different time steps instead of one time step, and are defined as [13]

$$P(s' \mid s, o) \triangleq \sum_{k=1}^{\infty} \gamma^k P(s', k \mid s, o), \tag{1}$$

$$R(s,o) \triangleq \mathbb{E}\left[\sum_{i=0}^{k-1} \gamma^i r_{t+1+i} | s_t = s, o_t = o\right], \tag{2}$$

where t denotes the time at which o is initialized; k denotes a random number of steps needed for executing option o; $P(s', k \mid s, o)$ denotes the probability that option o terminates at s' after k time steps. As the multi-step transition model incorporates a discount factor, $P(s' \mid s, o)$ is a sub-probability (i.e., $\sum_{s' \in S} P(s' \mid s, o) < 1$) [25], [26], [27]. The SMDP objective is to optimize a policy $\pi : S \times O \to [0,1]$ over options to maximize the value function defined as [13]

$$V_\pi(s) = \sum_{o \in O(s)} \pi(o \mid s) \left(R(s,o) + \sum_{s'} P(s' \mid s, o) V_\pi(s') \right). \tag{3}$$

The option framework is a hierarchical reinforcement learning (HRL) framework because it consists of two

Authorized licensed use limited to: BEIJING UNIVERSITY OF TECHNOLOGY. Downloaded on December 05,2024 at 00:21:13 UTC from IEEE Xplore. Restrictions apply.

policies: policy $\pi(o \mid s)$ over options and option policy $\omega_o(a \mid s)$ over actions. This framework aims at solving tasks with long-term sparse rewards but does not reduce the complexity of state spaces. Since the option policy is in terms of primitive states and actions, the space of option policies is vast when state space S is very large, which makes the option framework difficult to optimize. Thus, compressing the state space in an SMDP is important.

2.2 State Compression

The core of state compression is to aggregate similar states into an abstract state. State compression can be modeled as a function $\phi : S \rightarrow X$ mapping from a state space S to an abstract state space X, which converts an MDP $\mathcal{M} = \langle S, A, p, r, \gamma \rangle$ to an abstract MDP $\mathcal{M}_\phi = \langle X, A, p_\phi, \ r_\phi, \gamma \rangle$. Let $\mathbb{S}$ be the power set of S and $\phi^{-1} : X \rightarrow \mathbb{S}$ be the inverse map of ϕ; then, $r_\phi(x, a)$ and $p_\phi(x' \mid x, a)$ are respectively defined as [9]

$$r_\phi(x, a) \triangleq \sum_{s \in \phi^{-1}(x)} w(s \mid x) r(s, a), \tag{4}$$

$$p_\phi(x' \mid x, a) \triangleq \sum_{s' \in \phi^{-1}(x')} \sum_{s \in \phi^{-1}(x)} w(s \mid x) p(s' \mid s, a), \tag{5}$$

where $w(s \mid x)$ measures the contribution of state s to abstract state x and satisfies the normalization condition $\sum_{s \in \phi^{-1}(x)} w(s \mid x) = 1$. This constraint ensures that $\mathcal{M}_\phi$ can be studied in a Markovian way [9].

Ferns *et al.* [20], [28] proposed the bisimulation metric to measure the similarity between states. Given MDP $\mathcal{M}$, for any $s_1, s_2 \in S$, the bisimulation metric is defined as [20]

$$\begin{aligned} d_{\mathrm{B}}(s_1, s_2) \triangleq & \max_{a \in A} \{ c_r | r(s_1, a) - r(s_2, a) | \\ & + c_p d_p(p(\cdot \mid s_1, a), p(\cdot \mid s_2, a)) \}, \end{aligned} \tag{6}$$

where c_r and c_p are two positive constants; d_p is a probability metric. c_r and c_p are the weights of the distance between rewards and the distance between transition probabilities, respectively [20]. There are many probability metrics [29]; two of the most important ones are the Kantorovich metric and the total variation metric [20]. Ferns *et al.* [20] proved that d_{B} induces a bisimulation metric when d_p is the Kantorovich metric.[1] Evidently, the bisimulation metric satisfies the non-negativity, symmetry, and triangle inequality conditions but not the identity condition ($d_{\mathrm{B}}(s_1, s_2) = 0$ does not entail $s_1 = s_2$). Therefore, d_{B} is a pseudo-metric.

Ferns *et al.* [20] proved that compression based on the bisimulation metric approximately preserves the optimal policy of the uncompressed MDP. Also, bisimulation metrics can be used to construct options automatically [26]. Castro [30] proposed a scalable method for computing bisimulation metrics with a neural network in a deterministic MDP.

1. If $d_{\mathrm{B}}(s_1, s_2) = 0$ is a necessary and sufficient condition for s_1 to be bisimilar to s_2, d_{B} is a bisimulation metric [20], [26], [27]. When d_p is the total variation metric, $d_{\mathrm{B}}(s_1, s_2) = 0$ is a sufficient but not necessary condition for s_1 to be bisimilar to s_2, and thus d_{B} is not a bisimulation metric.

There also exist other state compression methods. Some researchers have proposed approximate MDP homomorphisms which aggregate states with similar one-step transition probabilities and rewards into an abstract state [12], [16], [17]. Taylor *et al.* [31] proposed a lax version of bisimulation metrics to bridge approximate MDP homomorphisms and bisimulation metric-based state compressions. However, state compression in MDPs does not consider temporally abstract actions.

2.3 State-Temporal Compression

Abel *et al.* [32] presented four state-abstraction-option classes that give rise to suboptimality bounds relative to the environmental MDP, but they did not focus on state similarity measurement. Ravindran and Barto [12] proposed an SMDP homomorphism that aggregates only the states with identical multi-step transition models and rewards. Although this approach preserves the optimality of the raw SMDP, few states are identical in an environment [10] and the SMDP homomorphism is computationally intractable [11]. Castro and Precup [25], [27] extended the bisimulation metric to an SMDP and introduced the notion of option-bisimulation metric. However, this metric is difficult to calculate because its computational cost scales exponentially with the maximum number of steps required for executing an option. Several studies proposed heuristic compression approaches in the option framework [33], [34], but they are challenging to be applied to complex and continuous environments. Thus, these approaches are impractical in realistic scenarios.

Function approximation can be seen as a form of state compression that maps the raw space to a lower-dimensional space, which is learned by optimizing the RL objective in an end-to-end manner. Some studies perform both state and temporal compression through combining neural networks and HRL, such as feudal HRL (and its variants) [18], [19] and option-critic with neural networks (and its variants) [14], [35], [36]. Two problems exist in these algorithms. First, the function that applies dimension reduction to the raw state space is learned through maximizing the expected cumulative reward. However, this could be very difficult for tasks with sparse rewards. Second, dimension reduction compresses the representation of states but does not necessarily compress the number of states. Consequently, the space of state representation may still be large. It is also unclear whether and how the optimal solution in the compressed state representation corresponds to that in the raw space. In summary, how to perform practical and optimality-preserving state-temporal compression is still an open question.

3 State-Temporal Compression-Based RL Framework

Since an MDP with options is formulated as an SMDP [13], state-temporal compression in an MDP can be transformed into state compression in an SMDP. State compression can compress an SMDP into an abstract SMDP. Therefore, state-temporal compression converts an MDP to an abstract SMDP, as shown in Fig. 1a. Aggregating similar states is more practical than aggregating only identical states

Authorized licensed use limited to: BEIJING UNIVERSITY OF TECHNOLOGY. Downloaded on December 05,2024 at 00:21:13 UTC from IEEE Xplore. Restrictions apply.

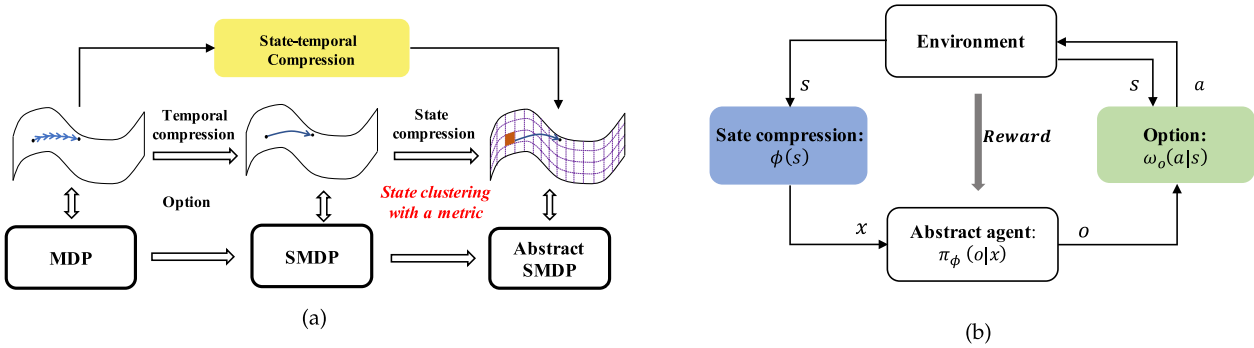


Fig. 1. (a) State-temporal compression. It converts an MDP to an abstract SMDP. (b) State-temporal compression-based RL framework.

because few states are identical in environments [10], [16], [37]. Therefore, state compression can be carried out by measuring the similarity between states in an SMDP.

As shown in Fig. 1b, we introduce a state-temporal compression-based RL framework that consists of three modules: a state compression module, an option module, and an abstract agent module. The state compression module maps a raw state to an abstract state with function $\phi(s)$. The abstract agent module receives an abstract state and produces an option o according to policy $\pi_\phi(o \mid x)$. The option module takes a raw state and produces an action according to option policy $\omega_o(a \mid s)$. As discussed above, the core of the framework is to find an appropriate state metric to measure the similarity between states in an SMDP.

We adopt the definition of metric-based state compression in an MDP [38]. Let d denote a metric and ϵ denote a compression threshold. We introduce the definition of (ϵ, d)-compression.

Definition 1. *(ϵ, d)-compression is defined as a surjection $\phi_{\epsilon,d} : S \to X$ that satisfies*

$$d(s_1, s_2) \leq \epsilon, \forall x \in X \text{ and } \; s_1, s_2 \in \phi_{\epsilon,d}^{-1}(x), \tag{7}$$

where X denotes an abstract state space and $\phi_{\epsilon,d}^{-1}(x)$ denotes the inverse image of abstract state x.

(ϵ, d)-compression can be considered as ϵ-neighborhood clustering that clusters the states if the metrics between them are less than ϵ. It converts an SMDP $\mathcal{N} = \langle S, O, P, R, \gamma \rangle$ to an abstract SMDP $\mathcal{N}_{\phi_{\epsilon,d}} = \left\langle X, O, P_{\phi_{\epsilon,d}}, R_{\phi_{\epsilon,d}}, \gamma \right\rangle$. Similar to Eqs. (4) and (5), multi-step reward $R_{\phi_{\epsilon,d}}$ and transition model $P_{\phi_{\epsilon,d}}$ in $\mathcal{N}_{\phi_{\epsilon,d}}$ are respectively defined as

$$R_{\phi_{\epsilon,d}}(x, o) \triangleq \sum_{s \in \phi_{\epsilon,d}^{-1}(x)} w(s \mid x) R(s, o), \tag{8}$$

$$P_{\phi_{\epsilon,d}}(x' \mid x, o) \triangleq \sum_{s' \in \phi_{\epsilon,d}^{-1}(x')} \sum_{s \in \phi_{\epsilon,d}^{-1}(x)} w(s \mid x) P(s' \mid s, o), \tag{9}$$

where $w(s \mid x)$ measures the contribution of state s to abstract state x and must satisfy the normalization condition $\sum_{s \in \phi_{\epsilon,d}^{-1}(x)} w(s \mid x) = 1$. If (ϵ, d)-compression approximately guarantees the optimal policy of the raw SMDP, it is approximately optimality-preserving state-temporal compression. In the following sections, we discuss several metrics to make (ϵ, d)-compression approximately optimality-preserving.

4 Metrics for State-Temporal Compression

In this section, we propose several metrics to perform state compression in an SMDP. The basic idea is to aggregate states within a short distance (small metric) in an appropriate space such that the compressed SMDP in that space approximately preserves the optimality of the raw SMDP.

4.1 Multi-Step Metric

As mentioned above, the core of state-temporal compression is to measure the similarity between states in an SMDP. Similar states in an SMDP should satisfy two conditions: (1) the available option sets for similar states should be the same; (2) the multi-step transition models and rewards for similar states should be similar. We then introduce the multi-step metric as follows.

Definition 2. *For any $s_1, s_2 \in S$, the multi-step metric is defined as*

$$\begin{aligned} d_{\mathrm{M}}(s_1, s_2) \triangleq & \max_{o \in O(s_1) \cap O(s_2)} \{ c_R |R(s_1, o) - R(s_2, o)| \\ & + c_P d_p(P(\cdot \mid s_1, o), P(\cdot \mid s_2, o)) \} \\ & + c_{\mathbb{H}} \mathbb{H}[O(s_1), O(s_2)], \end{aligned} \tag{10}$$

where c_R, c_P, and $c_{\mathbb{H}}$ are positive constants. The function $\mathbb{H}[x, y] = 0$ if $x = y$, and 1 otherwise. $c_{\mathbb{H}}$ is a sufficiently large constant such that $d_{\mathrm{M}}(s_1, s_2) \leq \epsilon$ implies that $O(s_1)$ equals $O(s_2)$.

We assume without loss of generality that intersection $O(s_1) \cap O(s_2)$ is nonempty in Definition 2. One can easily extend the notion of multi-step metric to the situation, where intersection $O(s_1) \cap O(s_2)$ is empty, by defining metric value $d_{\mathrm{M}}(s_1, s_2)$ as infinity. All our theories can be applied to the extended definition because (ϵ, d)-compression only aggregates states with metric values smaller than an $\epsilon < \infty$. In the rest of the paper, we set $c_R = 1/(1-\gamma)$ and $c_P = 1$. Same as the bisimulation metric, the multi-step metric is a pseudometric: it satisfies the non-negativity, symmetry, and triangle inequality conditions but not the identity condition.

There are two differences between the option-bisimulation metric and the multi-step metric. First, the option-

Authorized licensed use limited to: BEIJING UNIVERSITY OF TECHNOLOGY. Downloaded on December 05,2024 at 00:21:13 UTC from IEEE Xplore. Restrictions apply.

TABLE 1
Error Bounds for Different Compression Types

Compression type	Measurement for two states	Error bound complexity
State compression for an MDP		
Model similarity [37]	$\forall a, x : \lvert r(s_1,a) - r(s_2,a)\rvert \le \overline{\epsilon}_1, \lvert \sum_{s' \in \phi^{-1}(x)} (p(s' \mid s_1,a) - p(s' \mid s_2,a))\rvert \le \overline{\epsilon}_1$	$\mathcal{O}(2\overline{\epsilon}_1 \lvert S\rvert \gamma/(1-\gamma)^3)$
Model similarity [17]	—	$\mathcal{O}(2\overline{\epsilon}_1 \lvert X\rvert \gamma/(1-\gamma)^2)$
Homomorphism [16]	$\forall a : \lvert r_\phi(\phi(s),a) - r(s,a)\rvert \le \overline{\epsilon}_2, \sum_{x' \in X} \lvert p_\phi(x' \mid \phi(s),a) - \sum_{s' \in \phi^{-1}(x')} p(s' \mid s,a)\rvert \le \overline{\epsilon}_2$	$\mathcal{O}(\gamma\overline{\epsilon}_2/(1-\gamma)^2)$
Homomorphism [39]	—	$\mathcal{O}(\gamma\overline{\epsilon}_2/(1-\gamma)^3)$
Q-values [10]	$\forall a : \lvert q^*(s_1,a) - q^*(s_2,a)\rvert \le \overline{\epsilon}_3$ (q^* denotes the optimal value function in an MDP)	$\mathcal{O}(2\overline{\epsilon}_3/(1-\gamma)^2)$
Q-values [10]	$\forall a : \lceil q^*(s_1,a)/\overline{\epsilon}_4 \rceil = \lceil q^*(s_2,a)/\overline{\epsilon}_4 \rceil$ ($\lceil\cdot\rceil$ denotes an inter function)	$\mathcal{O}(2\overline{\epsilon}_4/(1-\gamma)^2)$
Bisimulation metric [20]	$d_{\mathrm{B}}(s_1,s_2) \le \overline{\epsilon}_5$	$\mathcal{O}(2\overline{\epsilon}_5/(1-\gamma))$
State compression for an SMDP		
Multi-step metric	$d_{\mathrm{M}}(s_1,s_2) \le \epsilon$	$\mathcal{O}(2\epsilon/(1-\gamma)^2)$
Geodesic metric	$d_{\mathrm{G}}(s_1,s_2) \le \epsilon\ (\forall \epsilon \in [0, 1/(1-\gamma)))$	$\mathcal{O}(2\delta(\epsilon)/(1-\gamma)^3)$
RRG metric	$d_{\mathrm{RRG}}(s_1,s_2) \le \epsilon (\forall \epsilon \in [0, 1/(1-\gamma)))$	$\mathcal{O}(2\epsilon/(1-\gamma))$

Some studies [17], [20], [37] assume the range of one-step rewards to be $[0,1]$ while others [10], [16], [39] do not. We assume the range of one-step rewards to be $[0,1]$ in this table. Note that $\overline{\epsilon}$ with a subscript represents a compression threshold for the corresponding compression type in an MDP.

bisimulation metric [25], [27] does not contain $\mathbb{H}[O(s_1), O(s_2)]$ because the available option set for each state is assumed to be the same. Second, the option-bisimulation metric uses the Kantorovich metric as d_p. By contrast, we employ the total variation metric as d_p because it is easy to analyze.[2] The choice of d_p does not alter the fact that calculating the multi-step metric is expensive. Its computational cost scales exponentially with the maximum number of steps for executing an option since the multi-step transition model and reward are the outcomes of an option at many different steps.

Suppose two states have the same available option sets and similar multi-step transition models and rewards. In that case, the expected returns for the two states are similar according to the Bellman equation (i.e., Eq. (3)) in an SMDP. Therefore, $(\epsilon, d_{\mathrm{M}})$-compression can approximately preserve the value function over options in an SMDP. Like previous state compression studies [17], [20], [31], [37], we assume without loss of generality that the value range of one-step rewards is normalized to $[0,1]$ in the following theorems.

Let $\widehat{\pi}_{\phi_{\epsilon,d_{\mathrm{M}}}}(o \mid x)$ be the optimal policy for abstract SMDP $\mathcal{N}_{\phi_{\epsilon,d_{\mathrm{M}}}}$, $\widehat{\pi}^{\mathrm{e}}_{\phi_{\epsilon,d_{\mathrm{M}}}}(o \mid s) \triangleq \widehat{\pi}_{\phi_{\epsilon,d_{\mathrm{M}}}}(o \mid \phi_{\epsilon,d_{\mathrm{M}}}(s))$ be the extended policy (not necessarily optimal) of $\widehat{\pi}_{\phi_{\epsilon,d_{\mathrm{M}}}}$ on SMDP $\mathcal{N}$, and $\widehat{\pi}(o \mid s)$ be the optimal policy for SMDP $\mathcal{N}$. We have the following theoretical result.

Theorem 1. *The policy $\widehat{\pi}^{\mathrm{e}}_{\phi_{\epsilon,d_{\mathrm{M}}}}(o \mid s)$ induced by $(\epsilon, d_{\mathrm{M}})$-compression approximates the optimal policy $\widehat{\pi}(o|s)$ for SMDP $\mathcal{N}$ with an error bound of*

$$\| V_{\widehat{\pi}} - V_{\widehat{\pi}^{\mathrm{e}}_{\phi_{\epsilon,d_{\mathrm{M}}}}} \|_\infty \le E_V(\epsilon, d_{\mathrm{M}}) \triangleq 2\epsilon e(H), \tag{11}$$

where $e(H) = \frac{1-\gamma^H}{1-\gamma}\left(\frac{1}{c_R} + \frac{1}{c_P(1-\gamma)}\right)$ and H denotes the horizon of SMDP $\mathcal{N}$ (i.e., the maximum number of decisions that the agent makes over options).

Proof. The proof is provided in Supplementary Materials, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2021.3069005. □

In the rest of the paper, we call policy $\widehat{\pi}^{\mathrm{e}}_{\phi_{\epsilon,d_{\mathrm{M}}}}$ the $E_V(\epsilon, d_{\mathrm{M}})$-optimal policy. The complexity of error bound $E_V(\epsilon, d_{\mathrm{M}})$ is $\mathcal{O}(2\epsilon/(1-\gamma)^2)$ when H is infinite. If RL tasks are composed of several sequential sub-tasks (i.e., H is small), $E_V(\epsilon, d_{\mathrm{M}})$ is scaled down by a factor of $1/(1-\gamma)$. There exist a variety of state compression criteria in an MDP that enable state compression to approximately preserve the optimality of the uncompressed MDP. We summarize the properties of some important compression types in Table 1.

Computational Complexity. Let τ denote a state-action trajectory and $P_{\mathcal{T}}(\tau \mid s, o)$ denote the distribution over trajectories taken by option o at state s. As options are assumed to terminate in finite time with probability 1, the maximum length (denoted by $T_{\max}$) of trajectories is finite but can be very large. We then rewrite Eqs. (1) and (2) as

$$P(s' \mid s, o) = \sum_{\tau \in \mathcal{T}_{s'}} P_{\mathcal{T}}(\tau \mid s, o), \tag{12}$$

$$R(s, o) = \sum_{\tau \in \mathcal{T}} P_{\mathcal{T}}(\tau \mid s, o) \sum_{k=0}^{T_{\max}-1} \gamma^k r(s_{t+k}, a_{t+k}), \tag{13}$$

where $\mathcal{T}$ represents the set of all trajectories; t is the time at which option o is initialized; $\mathcal{T}_{s'}$ represents the set of the trajectories whose end states are s'. Substituting Eqs. (13) and (12) into Eq. (10), one can find that the computation of metric d_{M} enumerates the trajectory set $\mathcal{T}$ with the number $\mathcal{O}((\lvert S\rvert^2 \lvert A\rvert)^{T_{\max}})$ of trajectories. Thus, the computational complexity of metric d_{M} is $\mathcal{O}((\lvert S\rvert^2 \lvert A\rvert)^{T_{\max}})$. To compute $(\epsilon, d_{\mathrm{M}})$-compression, we must calculate metric d_{M} between any states $s_1, s_2 \in S$. As $T_{\max}$ can be very large, it is very difficult to calculate the multi-step metric.

Metric-Convertibility Condition. To find a practical metric to replace the multi-step metric, we propose a metric-convertibility condition:

Proposition 1. *If two metrics d_1 and d_0 satisfy the inequality*

$$d_1(s_1, s_2) \le B_{d_0}(d_0(s_1, s_2)), \forall s_1, s_2 \in S, \tag{14}$$

2. The total variation metric is usually defined as half of the $L1$-norm or the $L1$-norm of two probability distributions' difference [29]. We adopt the second definition: given two probability distributions $\mu(s)$ and $\nu(s)$, the total variation metric $TV(\mu, \nu)$ is defined as $\sum_{s \in S} \lvert \mu(s) - \nu(s)\rvert$.

Authorized licensed use limited to: BEIJING UNIVERSITY OF TECHNOLOGY. Downloaded on December 05,2024 at 00:21:13 UTC from IEEE Xplore. Restrictions apply.

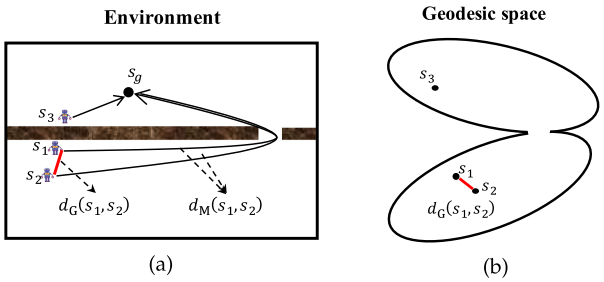


Fig. 2. (a) The relationship between the relative distance and the multi-step metric d_{M}. (b) The geodesic space which incorporates the environmental structures shown in (a).

where B_{d_0} is a monotonically increasing function and $B_{d_0}(0) = 0$, then (ϵ, d_0)-compression is also $\big(B_{d_0}(\epsilon), d_1\big)$-compression.

Proposition 1 is easy to prove. The metric-convertibility condition originates from the idea that (ϵ, d)-compression is a neighborhood clustering. The neighboring states obtained by metric d_0 are also close in metric d_1 that is upper-bounded by B_{d_0}. If metric d_0 upper-bounds multi-step metric d_{M} as in Eq. (14), (ϵ, d_0)-compression induces an $E_V\big(B_{d_0}(\epsilon), d_{\mathrm{M}}\big)$-optimal policy of SMDP $\mathcal{N}$. We then start to look for an appropriate metric d_0 and an appropriate function B_{d_0} to upper-bound multi-step metric d_{M}.

4.2 Geodesic Metric

We model an option by a subgoal option g. Let $\mathcal{G}(s)$ be the available subgoal option set for state s, and $\mathcal{G} \triangleq \cup_{s\in S}\mathcal{G}(s)$ be the set of all subgoal options. Subgoal option g has two properties: (1) g is available for any state from which the agent starts and can reach subgoal state s_g; (2) ω_g is the optimal policy for reaching subgoal s_g in the shortest time. The first property implies that if two states s_1 and s_2 can communicate with each other (i.e., state s_2 is reachable from state s_1 and vice versa), their available subgoal option sets $\mathcal{G}(s_1)$ and $\mathcal{G}(s_2)$ are the same. The second property implies that the closer two states s_1 and s_2 are, the more similar the two trajectories (starting respectively from s_1 and s_2 to s_g) are, as shown in Fig. 2a. In summary, if two states can communicate with each other and are close in an environment, the multi-step metric value is small.

According to our observations, we first consider the euclidean metric as d_0 to measure how close two states are. However, this metric does not consider the geometric structures of the environment, thus it cannot measure the relative distance between two states. As shown in Fig. 2a, although s_1 and s_3 are close in the euclidean metric space, s_1 and s_3 are far away in the environment because a wall separates them. States can be considered to lie in a manifold (for continuous environments) or a graph (for discrete environments), where each state is a node and weighted edges connect states and encode one-step transition probabilities [40]. Compared with the euclidean metric, the geodesic metric, which measures the shortest path between two points, is a more appropriate metric for a manifold [41]. When adapted to an MDP, the geodesic metric can be defined as the average minimum number of steps for transition from one state to another.

Definition 3. *For any $s_1, s_2 \in S$, the geodesic metric is defined as*

$$d_{\mathrm{G}}(s_1, s_2) \triangleq \min_{\pi_{s_1 s_2}} \sum_{t=0}^{\infty} \Gamma(t) P_{s_1 s_2}\big(t \,|\, \pi_{s_1 s_2}\big), \tag{15}$$

where

$$\Gamma(t) = \begin{cases} \sum_{k=1}^{t} \gamma^{k-1}, & t \geq 1 \\ 0, & t = 0, \end{cases}$$

and $P_{s_1 s_2}\big(t \,|\, \pi_{s_1 s_2}\big)$ denotes a probability distribution over $t \in [0, \infty) \cup \infty$. Note that $t < \infty$ represents the first hit time at which the agent reaches s_2 starting from s_1 under policy $\pi_{s_1 s_2}$ over actions, and $t = \infty$ represents that the agent never reaches s_2 starting from s_1 under $\pi_{s_1 s_2}$. If s_2 is not reachable from s_1 under any policy (i.e., $P_{s_1 s_2}(\infty \,|\, \pi_{s_1 s_2}) = 1$, $\forall \pi_{s_1 s_2}$), $d_{\mathrm{G}}(s_1, s_2)$ achieves the maximum value $\Gamma(\infty) = 1/(1-\gamma)$.

We make a state reversibility assumption that any adjacent states can access each other through a one-step reversible action (that is, states lie in an undirected graph). This assumption usually models the state space in an MDP [35], [40]. We then prove that $d_{\mathrm{G}}(s_1, s_2)$ is a metric.

Theorem 2. *Under the state reversibility assumption, $d_{\mathrm{G}}(s_1, s_2)$ is a metric.*

Proof. The proof is provided in Supplementary Materials, available online. □

In the rest of the paper, we consider ϵ to be much smaller than $1/(1-\gamma)$. $d_{\mathrm{G}}(s_1, s_2) \leq \epsilon$ implies that s_1 and s_2 can communicate with each other and are close in an environment, thus $d_{\mathrm{G}}(s_1, s_2)$ can be used as d_0. If the reversibility assumption is not satisfied, $d_{\mathrm{G}}(s_1, s_2)$ is asymmetric (i.e., $d_{\mathrm{G}}(s_1, s_2)$ may not equal to $d_{\mathrm{G}}(s_2, s_1)$). One can adopt a symmetric geodesic metric to measure the distance between states, i.e., $d_{\mathrm{G}}(s_1, s_2) + d_{\mathrm{G}}(s_2, s_1)$; this will be considered in future work.

As the geodesic metric represents the minimum distance between two states, it depends on the environmental dynamics but not on a specific policy. Successor representation [42] and successor feature [43] can also measure the relative distance between states, but they rely on both the environmental dynamics and a specific policy.

We define the geodesic space as shown in Fig. 2b, where the distance between states is measured by the geodesic metric. We assume that the change in trajectory distribution $P_{\mathcal{T}}$ is smooth in the geodesic space.

Assumption 1. *For any two states $s_1, s_2 \in S$ that can communicate with each other and any their available subgoal option g, a non-negative and monotonically increasing function δ exists such that the following inequality holds:*

$$\left| \sum_{\tau \in \mathcal{T}} \big(P_{\mathcal{T}}(\tau \,|\, s_1, g) - P_{\mathcal{T}}(\tau \,|\, s_2, g)\big) \right| \leq \delta(d_{\mathrm{G}}(s_1, s_2)).$$

State spaces in real-life environments are usually smooth. A small change in the starting point does not cause a drastic change in the trajectory for reaching a subgoal. The more

Authorized licensed use limited to: BEIJING UNIVERSITY OF TECHNOLOGY. Downloaded on December 05,2024 at 00:21:13 UTC from IEEE Xplore. Restrictions apply.

distant two starting states are, the more different the trajectories are for transition respectively from the two states to a subgoal. Thus, function δ probably exists, which depends on the environmental dynamics and can be estimated with multi-step transition models.

Theorem 3. *Under Assumption 1, the following inequality holds for any* $s_1, s_2 \in S$

$$d_{\mathrm{M}}(s_1, s_2) \leq B_{d_{\mathrm{G}}}(d_{\mathrm{G}}(s_1, s_2)) \triangleq c_R \frac{\delta(d_{\mathrm{G}}(s_1, s_2))}{1-\gamma} + c_P(1-\gamma) d_{\mathrm{G}}(s_1, s_2), \tag{16}$$

when $d_{\mathrm{G}}(s_1, s_2) < 1/(1-\gamma)$.

Proof. The proof is provided in Supplementary Materials, available online. □

Combining Theorem 1, Proposition 1, and Theorem 3, one can obtain:

Corollary 1. *For any* $\epsilon \in [0, 1/(1-\gamma))$, $(\epsilon, d_{\mathrm{G}})$*-compression can induce an* $E_V(\epsilon, d_{\mathrm{G}})$*-optimal policy for SMDP* $\langle S, \mathcal{G}, P, R, \gamma \rangle$ *with an error bound of*

$$E_V(\epsilon, d_{\mathrm{G}}) = 2e(H)\left(\frac{c_R \delta(\epsilon)}{1-\gamma} + c_P(1-\gamma)\epsilon\right).$$

As we set $c_R = 1/(1-\gamma)$ and $c_P = 1$, the complexity of error bound $E_V(\epsilon, d_{\mathrm{G}})$ is $\mathcal{O}(2\delta(\epsilon)/(1-\gamma)^3)$. Although $E_V(\epsilon, d_{\mathrm{G}})$ may be large in practical applications, it can highlight the properties ensuring a tighter bound, which will benefit further researches.

Computational Complexity Analysis. The geodesic metric $d_{\mathrm{G}}(s_1, s_2)$ requires the optimal policy for transition from state s_1 to state s_2, which can be considered as an RL problem. Since the transition model and reward functions are unknown, the computation of metric $d_{\mathrm{G}}(s_1, s_2)$ is both a learning problem and a computational one, which is difficult to solve in environments with large-scale and high-dimensional state spaces. We analyze this problem from a computational view in the following.

Since metric d_{G} is the average minimum number of steps for transition from one state to another, $d_{\mathrm{G}}(s_1, s_2) \leq \epsilon$ is equivalent to be able to transition from one state to another (and vice versa) within a certain number of steps on average. Let K be this average number that depends on ϵ, and $F(\cdot)$ be a quantile function

$$F(y) \triangleq \inf_{k \geq 0}\left\{\sum_{t=0}^{k} P_{s_1 s_2}\left(t | \pi^*_{s_1 s_2}\right) \geq y\right\}, \forall y \in [0, 1], \tag{17}$$

where $\pi^*_{s_1 s_2}$ denotes the optimal policy for transition from state s_1 to state s_2. Let η be a high probability value, and K_η be $F(\eta) - K$; then, $K + K_\eta$ represents the number of steps within which the transition from s_1 to s_2 can be done with a high probability of η. We require $(K + K_\eta)$-step transitions to approximately compute the average minimum step, leading to a computational complexity of $\mathcal{O}((|S|^2|A|)^{K+K_\eta})$. Note that K_η is zero for $\eta = 1$ in deterministic environments. Even for high probability η in stochastic environments, K_η may not be large because the probability distribution over the first hit time in a stationary Markovian system has a very habitual form: it goes through a single maximum and then decays either exponentially or as a power-law [44], [45]. Additionally, $(\epsilon, d_{\mathrm{G}})$-compression only requires an indicator of whether the geodesic metric between two states is less than ϵ instead of the exact metric value, which inspires us to explore computationally cheap methods to compress the state space.

4.3 Reward-Restricted Geodesic Metric

Although Corollary 1 shows that compression based on the geodesic metric induces an $E_V(\epsilon, d_{\mathrm{G}})$-optimal policy, $E_V(\epsilon, d_{\mathrm{G}})$ can be very large due to the first term on the right-hand side of Eq. (16). Neighboring initial states induce similar trajectories when executing an option o, but the difference in the cumulative rewards of these similar trajectories can be large. Only for an environment with sparse rewards will the difference be small. Thus we need to reduce the cumulative reward difference by incorporating reward information into the geodesic metric. We then introduce the reward-restricted geodesic (RRG) metric as follows:

Definition 4. *For any* $s_1, s_2 \in S$*, the RRG metric is defined as*

$$d_{\mathrm{RRG}}(s_1, s_2) \triangleq \frac{c_R}{1-\gamma} \max_{g \in \mathcal{G}(s_1) \cap \mathcal{G}(s_2)} |R(s_1, g) - R(s_2, g)| + c_P d_{\mathrm{G}}(s_1, s_2). \tag{18}$$

Same as the geodesic metric, d_{RRG} is a metric under the reversibility condition. We then have a theorem:

Theorem 4. *For any* $s_1, s_2 \in S$*, the following inequality holds:*

$$d_{\mathrm{M}}(s_1, s_2) \leq B_{d_{\mathrm{RRG}}}(d_{\mathrm{RRG}}(s_1, s_2)) \triangleq (1-\gamma) d_{\mathrm{RRG}}(s_1, s_2), \tag{19}$$

when $d_{\mathrm{RRG}}(s_1, s_2) < 1/(1-\gamma)$*. Thus, for any* $\epsilon \in [0, 1/(1-\gamma))$, $(\epsilon, d_{\mathrm{RRG}})$*-compression induces an* $E_V(\epsilon, d_{\mathrm{RRG}})$*-optimal policy for SMDP* $\langle S, \mathcal{G}, P, R, \gamma \rangle$ *with an error bound of*

$$E_V(\epsilon, d_{\mathrm{RRG}}) = 2(1-\gamma)\epsilon e(H). \tag{20}$$

Proof. The proof is provided in Supplementary Materials, available online. □

Note that the above theorem does not rely on Assumption 1. Bound $B_{d_{\mathrm{RRG}}}$ is much tighter than bound $B_{d_{\mathrm{G}}}$ because γ is usually close to 1 and the first term of $B_{d_{\mathrm{G}}}$ is large. However, the estimation of multi-step reward R is computationally expensive. This problem can be avoided when an environment is deterministic or with sparse rewards.

Deterministic Environments. We find that when SMDP $\langle S, \mathcal{G}, P, R, \gamma \rangle$ is deterministic, multi-step reward R can be replaced by reward function R^{st} that only rewards the agent at the initial state in the process for executing subgoal option g

$$R^{\mathrm{st}}(s, g) \triangleq \sum_{a \in A} \omega_g(a \,|\, s) r(s, a), \forall s \in S, g \in \mathcal{G}(s). \tag{21}$$

We call $R^{\mathrm{st}}(s, g)$ the starting-reward function. We then have the following theorem:

Authorized licensed use limited to: BEIJING UNIVERSITY OF TECHNOLOGY. Downloaded on December 05,2024 at 00:21:13 UTC from IEEE Xplore. Restrictions apply.

Theorem 5. *The optimal value in deterministic SMDP* $\langle S, \mathcal{G}, P, R, \gamma\rangle$ *equals to that in deterministic SMDP* $\langle S, \mathcal{G}, P, R^{\mathrm{st}}, \gamma\rangle$.

Proof. The proof is provided in Supplementary Materials, available online. □

This theorem implies that we can calculate the RRG metric by replacing multi-step reward R with R^{st}, which relieves the burden for estimating multi-step reward R. We call it the starting-reward-based calculation (SR-calculation) approach.

Environments With Sparse Rewards. In real-life environments, the agent is often rewarded only in a few states. In this situation, we adopt the following approach to calculate the RRG metric: taking each state with nonzero rewards as an abstract state and then measuring only the similarity between the states with zero rewards. In terms of measuring the similarity between states with zero rewards, the RRG metric is equivalent to the geodesic metric. Thus multi-step rewards for similar states are the same. This approach is practical when the states with nonzero rewards are sparse. We call it the reward-equivalence constraint-based calculation (REC-calculation) approach.

SR-calculation can also calculate the RRG metric approximately for environments with sparse rewards because most of states along the trajectory taken by the option have no reward. As our study focuses on environments with sparse rewards, the SR-calculation and REC-calculation approaches can both be used to calculate the RRG metric. For stochastic environments with dense rewards, however, we have to estimate multi-step rewards using trajectories sampled from those environments.

4.4 Geodesic Metric Learning

One advantage of the geodesic metric is that it can be learned by a neural network. The RRG metric can be calculated with the geodesic metric according to the SR-calculation or REC-calculation approach. Metric learning learns a distance metric from pairs of points that preserve the distance relation among the training data [46]. For this purpose, we need to construct a training dataset consisting of pairwise states that preserve the geodesic metric between them. Since state compression involves clustering states in the neighborhood of certain anchor states, the exact distance between states does not need to be calculated, and we only need to know whether the distance between two states is smaller than compression threshold ϵ.

We set K-step discount accumulation $\Gamma(K) = \sum_{t=1}^{K} \gamma^{t-1}$ as the compression threshold denoted by ϵ_K. The geodesic metric learning goal is to optimize a neural network to map the raw state space to another that retains the state pairs within distance ϵ_K of each other while separating those outside distance ϵ_K. We define a training dataset $D_K = \{(s_i, s_j), z_{ij}\}$, where $z_{ij} = 1$ indicates that the transition from state s_i to state s_j is empirically reachable within K steps, while $z_{ij} = 0$ indicates otherwise. In the following, we discuss how to obtain training samples for geodesic metric learning.

According to Definition 3, calculating the geodesic metric requires the optimal policy for transition from one state to another. However, we do not necessarily use optimal policies to obtain K-step reachable state pairs. We prove that these state pairs can be obtained by any policies as follows.

Theorem 6. *Let* $\Omega_{\mathrm{G}}(s, K) \triangleq \{s' \mid d_{\mathrm{G}}(s, s') \le \Gamma(K)\}$ *be the* K*-step distance region at state* s *in terms of the geodesic metric. Let* π_A *be a policy over actions and* Π *be the set of all possible policies. For any* $s \in S$, *we have*

$$\Omega_{\mathrm{G}}(s, K) = \bigcup\nolimits_{\pi_A \in \Pi} \Omega_{\pi_A}(s, K), \tag{22}$$

where $\Omega_{\pi_A}(s, K) \triangleq \{s' \mid \sum_{t=0}^{\infty} \Gamma(t) P_{ss'}(t \mid \pi_A) \le \Gamma(K)\}$.

Proof. The proof is provided in Supplementary Materials, available online. □

Theorem 6 is based on a basic idea: if the average first hit time for transition from state s_1 to state s_2 is less than K under a non-optimal policy, it is also less than K under the optimal policy. We use a uniform distribution over actions as the sampling policy because it can produce sufficiently diverse action sequences. After drawing state-action trajectories $\{s_0, a_0, \ldots, s_i, a_i, \ldots\}$ from the environment, we construct training dataset D_K as follows:

$$z_{ij} = \begin{cases} 1, & |i-j| \le K \\ 0, & |i-j| \ge K + l, \end{cases} \tag{23}$$

where we set $l > 0$ to create a gap between state pairs within the K-step distance and those beyond.

We build a neural network $\Psi_{\mathbf{w}}$ parameterized with $\mathbf{w}$ to learn the geodesic metric. We call $\Psi_{\mathbf{w}}$ a geodesic-metric network. Fig. 3 intuitively illustrates the learning objective of $\Psi_{\mathbf{w}}$, i.e., keeping all the state pairs within the K-step distance close while separating those beyond the K-step distance. Given training dataset D_K, the neural network $\Psi_{\mathbf{w}}$ is trained by minimizing the loss function:

$$\begin{aligned} &L(\mathbf{w}; D_K) \\ &= \mathbb{E}_{(s_i, s_j, z_{ij}) \sim D_K} \Big[z_{ij} \max\big\{ \| \Psi_{\mathbf{w}}(s_i) - \Psi_{\mathbf{w}}(s_j) \|_2 - \chi, 0 \big\}^2 \\ &\quad + (1 - z_{ij}) \max\big\{ \chi + m - \| \Psi_{\mathbf{w}}(s_i) - \Psi_{\mathbf{w}}(s_j) \|_2, 0 \big\}^2 \Big], \end{aligned} \tag{24}$$

where χ is a threshold to differentiate whether two states are within the K-step distance; m denotes a margin. This loss function can be considered a variant of the contrastive loss function [47].

The geodesic-metric network maps a raw state s to a state representation $\Psi_{\mathbf{w}}(s)$. We call this the geodesic representation. The geodesic metric is represented as the euclidean metric on the geodesic representation

$$d_{\mathrm{G},\mathbf{w}}(s_1, s_2) = \frac{\epsilon_K}{\chi} \| \Psi_{\mathbf{w}}(s_1) - \Psi_{\mathbf{w}}(s_2) \|_2 . \tag{25}$$

Like many studies [6], [21], [48], [49], we consider one-step rewards that are independent of actions (i.e., $r(s) = r(s, a), \forall a \in A$). We adopt the SR-calculation approach to compute the RRG metric, i.e.,

$$d_{\mathrm{RRG},\mathbf{w}}(s_1, s_2) = \frac{c_R}{1-\gamma} |r(s_1) - r(s_2)| + c_P d_{\mathrm{G},\mathbf{w}}(s_1, s_2). \tag{26}$$

Authorized licensed use limited to: BEIJING UNIVERSITY OF TECHNOLOGY. Downloaded on December 05,2024 at 00:21:13 UTC from IEEE Xplore. Restrictions apply.

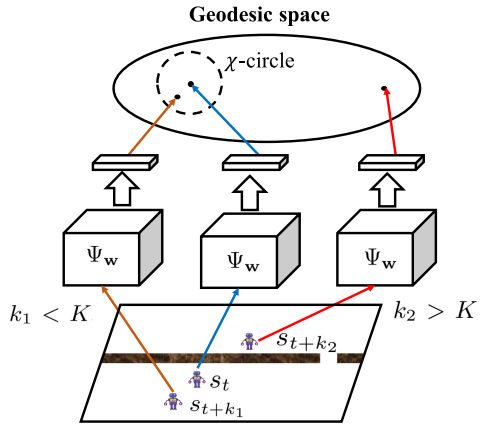


Fig. 3. The geodesic metric learning objective. Geodesic-metric network $\Psi_{\mathbf{w}}$ maps the state pairs within a K-step distance to neighboring points (in a circle of radius χ) and the state pairs beyond the K-step distance to distant points (outside the circle of radius χ).

When reward functions are dependent on actions, RRG metric $d_{\mathrm{RRG},\mathbf{w}}$ can be calculated using the REC-calculation approach which does not require subgoal options.

The successor representation learns continuous similarity between states [42]. By contrast, the objective of geodesic metric learning is to retain binary similarity between states (i.e., whether two states are similar or not), which has less learning burden than the successor representation. Savinov *et al.* [50], [51] proposed a classification network to discriminate whether transition from one state to another is reached within K steps. Compared with their method, the geodesic metric network has two advantages. First, a metric network can be used to partition the state space, which guarantees each state to be mapped to one partition. Thus, it can be used to perform state compression in an SMDP. Second, a metric network can be used to calculate the relative distance between two states. Therefore, a metric network can provide reward shaping for goal-reaching tasks where the agent is rewarded for reaching a goal state. We verified the advantages of the geodesic-metric network for those tasks in Section 6.2.

5 RRG Metric-Based Reinforcement Learning

In this section, we present an algorithm to perform state-temporal compression-based RL (see Fig. 1b) using the RRG metric, called the RRG metric-based reinforcement learning algorithm (RRG-RL). As shown in Fig. 4, this algorithm has two phases: a state compression construction phase and an abstract reinforcement learning phase. The first phase trains a geodesic-metric network to perform RRG metric-based state compression (RRG-compression). With this state compression, we then train the subgoal options and the policy over them.

5.1 Phase 1: RRG-Compression Algorithm

Unlike usual RL settings, our algorithm takes some episodes to learn the geodesic metrics before training the agent, as shown in Fig. 4. To fairly compare our algorithm with many other RL algorithms, we incorporate the metric learning process into the reinforcement learning

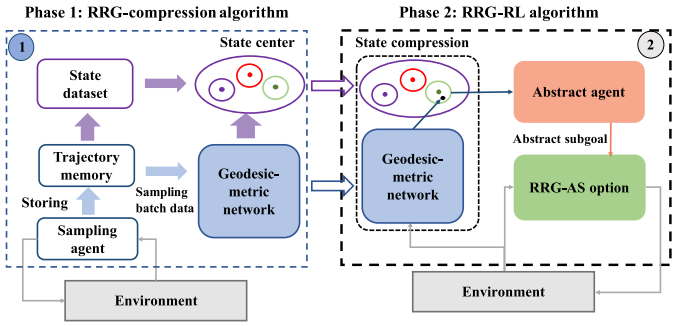


Fig. 4. Two phases in training an RRG-RL algorithm.

process. We train geodesic-metric network $\Psi_{\mathbf{w}}$ in a reinforcement learning style. We use a sampling agent with the uniform policy over actions to interact with the environment and store the sampled trajectories in memory M_{G}. With the trajectories evenly drawn from M_{G}, we construct dataset $D_K = \{(s_i, s_j, z_{ij})\}$ according to Eq. (23). During the construction, we maintain the value of label z_{ij} to be 1 if state s_i and state s_j were ever sampled as a K-step reachable state pair in memory M_{G}. We then use dataset D_K to train geodesic-metric network $\Psi_{\mathbf{w}}$. Lines 1-1 of Algorithm 1 show this metric learning process. After geodesic-metric network $\Psi_{\mathbf{w}}$ is learned, the RRG metric can be easily obtained according to Eq. (26).

Using RRG metric $d_{\mathrm{RRG},\mathbf{w}}$, we can perform (ϵ, d)-compression. It aggregates states into the same abstract state if the RRG metric between them is less than a specified compression threshold. Thus, (ϵ, d)-compression can be considered as threshold-based state clustering [52]. As ϵ is predefined, the maximum size of state clusters is fixed. The number of state clusters should correspond to the state space size. In general, a larger state space entails more state clusters. The state compression algorithm is shown in Lines 1-1 of Algorithm 1, the aim of which is to divide the raw state space into some state clusters where the RRG metric between the central state of each state cluster and any state in this cluster is less than ϵ_K. Let $\{S_1, \ldots, S_x, \ldots\}$ denote these state clusters. We compress each state cluster S_x into one abstract state x. Thus, S_x is the inverse image of abstract state x in the raw space. Let s_x^{c} be the center of state cluster S_x and C be the set of central states; then,we use C to construct a state compression function as follows:

$$\phi_{\epsilon_K, d_{\mathrm{RRG},\mathbf{w}}}(s) = \arg\min_{s_x^{\mathrm{c}} \in C} d_{\mathrm{RRG},\mathbf{w}}\left(s, s_x^{\mathrm{c}}\right). \tag{27}$$

To simplify the notation, we use ϕ_K to denote $\phi_{\epsilon_K, d_{\mathrm{RRG},\mathbf{w}}}$ in the rest of the paper. Eq. (27) means that we need to go through all possible state clusters to compute the abstract state for each new state. In future work, we will consider constructing the topology of abstract states. Using this topology, we can search only the abstract states close to new states, which shrinks the set of possible state clusters for new states.

Algorithm 1 does not require all states in the environment; instead, it requires only the state set $\mathcal{S}$ that covers the environment evenly. For any state $s \notin \mathcal{S}$, ϕ_K can find a central state s_x^{c} closest to s in the RRG metric and maps s to

Authorized licensed use limited to: BEIJING UNIVERSITY OF TECHNOLOGY. Downloaded on December 05,2024 at 00:21:13 UTC from IEEE Xplore. Restrictions apply.

Algorithm 1. RRG-Compression Algorithm: Get$[\phi](K, N_1)$

Require: K and training episodes N_1 of the first phase.

Initialize geodesic-metric network parameters $\mathbf{w}$ and the sampling agent with a uniform distribution over actions.
for $n = 0 : N_1 - 1$ **do**
  Use the sampling agent to sample trajectory $\{(s_t, a_t, r(s_t))\}$ from the environment.
  Store trajectory $\{(s_t, a_t, r(s_t))\}$ in memory M_{G} and store state s_t in set $\mathcal{S}$.
  Construct dataset D_K using the samples from memory M_{G} according to Eq. (23).
  Update parameters $\mathbf{w}$ with loss function $L(\mathbf{w}; D_K)$.
end for
Construct the calculation equation (Eq. (26)) of metric $d_{\mathrm{RRG,w}}$ with the learned geodesic metric $d_{\mathrm{G,w}}$.
Sample state s without replacement from $\mathcal{S}$ and take it as the first central state of set $C = \{s_1^{\mathrm{c}} = s\}$.
while $\mathcal{S}$ is not $\emptyset$ **do**
  Sample state s' without replacement from $\mathcal{S}$.
  if $\min_{s_x^{\mathrm{c}} \in C} d_{\mathrm{RRG,w}}(s', s_x^{\mathrm{c}}) > \epsilon_K$ **then**
    Assign s' as a central state of new cluster and then add s' to set C.
  end if
end while
Construct compression function ϕ_K with set C according to Eq. (27).
Output: compression function ϕ_K.

abstract state x, thus guaranteeing each state to be mapped to an abstract state.

As described in Section 3, function ϕ_K compresses SMDP $\langle S, \mathcal{G}, P, R, \gamma\rangle$ into abstract SMDP $\langle X, \mathcal{G}, P_{\phi_K}, R_{\phi_K}, \gamma\rangle$. According to the triangle inequality, the RRG metric between any two states in the same state cluster is less than $2\epsilon_K$. Therefore, ϕ_K is a $(2\epsilon_K, d_{\mathrm{RRG,w}})$-compression function. According to Theorem 4, this compression induces an $E_V(2\epsilon_K, d_{\mathrm{RRG,w}})$-optimal policy of the raw SMDP with error bound $4\big(1-\gamma^K\big)e(H)$, showing that K is important for the error bound. The larger K is, the more the state space will be reduced, and the larger the value of $E_V(2\epsilon_K, d_{\mathrm{RRG,w}})$ will be. In Section 6, we empirically present the impact of different K values on the performance of RRG-RL.

5.2 Phase 2: RRG-RL Algorithm

In phase 2, RRG-RL learns a two-level policy: subgoal options and a policy over them.

5.2.1 RRG Metric-Based Abstract Subgoal Option

It is difficult to optimize the policy over subgoal options when the set of subgoal states is large. One way to tackle this difficulty is to aggregate all subgoal states into several abstract subgoal states [53], [54]. The option to reach an abstract state is called the abstract subgoal option [53], [54]. Different from previous studies [53], [54], we use a state-similarity metric (RRG) to cluster subgoal states into abstract subgoal states, thus we term our abstract subgoal option as RRG metric-based abstract subgoal option (RRG-AS option).

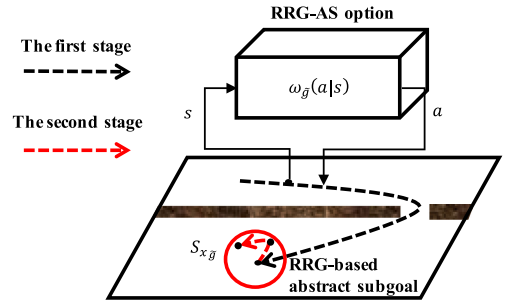


Fig. 5. RRG-AS option.

Let $\tilde{g}$ denote an abstract subgoal option and $x_{\tilde{g}}$ be its abstract subgoal. Similar to subgoal option g, abstract subgoal option $\tilde{g}$ is defined as a tuple $\langle I_{\tilde{g}}, \omega_{\tilde{g}}, \beta_{\tilde{g}}\rangle$, where $I_{\tilde{g}}$ denotes an initiation set, $\omega_{\tilde{g}}(a|s)$ denotes an option policy, and $\beta_{\tilde{g}}(s)$ denotes a termination function to terminate $\tilde{g}$ when $x_{\tilde{g}}$ is reached.

How do we determine whether RRG-AS option $\tilde{g}$ reaches abstract subgoal $x_{\tilde{g}}$? A straightforward idea is that as long as the agent arrives at any state s in inverse image $S_{x_{\tilde{g}}}$ of $x_{\tilde{g}}$, RRG-AS option $\tilde{g}$ is considered to reach abstract subgoal $x_{\tilde{g}}$ and terminated at state s. However, this idea may overlook some important states within $S_{x_{\tilde{g}}}$. To tackle this problem, we design a two-stage process for RRG-AS options. As shown in Fig. 5, each RRG-AS option $\tilde{g}$ consists of two stages: reaching the center of the inverse image $S_{x_{\tilde{g}}}$ of $x_{\tilde{g}}$ and then reaching raw state $s \in S_{x_{\tilde{g}}}$. State s is the end state and determined by function $\beta_{\tilde{g}}(s)$. The second stage enables the states within $S_{x_{\tilde{g}}}$ to be explored in a fine-grained way.

Each RRG-AS option $\tilde{g}$ can represent a cluster of subgoal options whose subgoal states belong to $S_{x_{\tilde{g}}}$. For any state $s \in S_{x_{\tilde{g}}}$, the option whose subgoal state is s can be approximated by RRG-AS option $\tilde{g}$ because the optimal policy for directly reaching $s \in S_{x_{\tilde{g}}}$ can be approximated by the policy first reaching the center of $S_{x_{\tilde{g}}}$ and then reaching s. We theoretically analyze the relationship between subgoal options and RRG-AS options in Supplementary Materials, available online.

Optimization of option policy $\omega_{\tilde{g}}$ corresponds to termination function $\beta_{\tilde{g}}$. We can optimize the termination function using the termination gradient algorithm [14], i.e., both the option policy and termination function require optimization. Whereas, to ease the burden for learning RRG-AS options, we do not optimize $\beta_{\tilde{g}}$ and heuristically design a non-parametric termination function. Our basic idea is to let option $\tilde{g}$ explore abstract area $S_{x_{\tilde{g}}}$ sufficiently instead of terminating at a single end state in the second stage. Option $\tilde{g}$ is terminated in the T_Δth state after arriving at the central state $s^{\mathrm{c}}_{x_{\tilde{g}}}$, where T_Δ is a time window in which the agent explores raw states within $S_{x_{\tilde{g}}}$.

It is efficient to explore the states within $S_{x_{\tilde{g}}}$ since they are close in the environment. Thus, we design an intrinsic reward to learn RRG-AS option $\tilde{g}$

$$r_{\tilde{g}}(s_t, a_t) = (1 - I_{\mathrm{arrive}}) \underbrace{\mathbb{I}\Big[s_t = s^{\mathrm{c}}_{x_{\tilde{g}}}\Big]}_{\text{The first stage}} + \lambda_1 I_{\mathrm{arrive}} \underbrace{\mathbb{I}\big[\phi_K(s_t) = x_{\tilde{g}} \text{ and } \ s_t \notin h_{\tilde{g}}\big]}_{\text{The second stage}} + \lambda_2 r(s_t, a_t), \tag{28}$$

Authorized licensed use limited to: BEIJING UNIVERSITY OF TECHNOLOGY. Downloaded on December 05,2024 at 00:21:13 UTC from IEEE Xplore. Restrictions apply.

where I_{arrive} is 1 after the first time to reach central state $s^{\text{c}}_{x_{\tilde{g}}}$ and I_{arrive} equals 0 before that; $\mathbb{I}[\cdot] = 1$ when its argument is true and zero otherwise; $h_{\tilde{g}}$ denotes the history trajectory when executing $\tilde{g}$; λ_1 and λ_2 denote weighting factors. Before arriving at $s^{\text{c}}_{x_{\tilde{g}}}$, the first term in the right of Eq. (28) drives option $\tilde{g}$ to reach $s^{\text{c}}_{x_{\tilde{g}}}$. In continuous environments, whether the central state is reached can be judged by watching if the geodesic metric between them is less than a small predefined threshold. After arriving at $s^{\text{c}}_{x_{\tilde{g}}}$, the second term in the right of Eq. (28) drives option $\tilde{g}$ to explore the states within the inverse image $S_{x_{\tilde{g}}}$ of $\tilde{g}$. Environmental reward $r(s_t, a_t)$ is taken as an auxiliary reward for learning RRG-AS options, thus we set λ_2 to be smaller than λ_1.

The learning objective for RRG-AS options is to maximize the expected cumulative (intrinsic) reward. A range of techniques can optimize the objective function, such as the Q-learning and A2C algorithms. We adopt the Q-learning algorithm to learn RRG-AS option policy. Let $Q_\theta(s, a, \tilde{g})$ denote the parameterized Q-value function; then RRG-AS option policy $\omega_{\tilde{g}}$ is expressed as $\omega_{\tilde{g}}(a \mid s) = \arg\max_{a' \in A} Q_\theta(s, a', \tilde{g})$. In the training process, each action a is sampled in a greedy policy with exploration probability $\mathcal{E}_1$ or in a softmax policy. $Q_\theta(s, a, \tilde{g})$ is learned through minimizing the loss function

$$J_1(\theta) = \mathbb{E}_{(s,a,s',r_{\tilde{g}},\tilde{g}) \sim M^{\tilde{g}}} \left[\left(r_{\tilde{g}} + \gamma \max_{a' \in A} Q_\theta(s', a', \tilde{g}) - Q_\theta(s, a, \tilde{g}) \right)^2 \right], \tag{29}$$

where $M^{\tilde{g}}$ denotes the replay memory of RRG-AS option $\tilde{g}$.

Algorithm 2. RRG-RL Algorithm

```
Initialize parameters {θ, ϑ}, exploration probabilities E_1 and E_2, K, training episodes N_1 of the first phase, and total training episodes N_episodes.
Obtain compression function: φ_K ← Get[φ](K, N_1).
for n = N_1 : N_episodes − 1 do
  Initialize state s ← s_0 and abstract state x ← φ_K(s).
  while s is not a terminal do
    Choose g̃ from Q^A_ϑ(x, g̃) according to the E_2-greedy policy and initialize t_g̃ = 0.
    while g̃ is not terminated and s is not a terminal do
      Choose a from Q_θ(s, a, g̃) according to the E_1-greedy policy.
      Execute a, obtain environmental reward r and next state s', calculate intrinsic reward r_g̃, and store experience (s, a, s', r_g̃, g̃) in M^g̃.
      Update t_g̃ ← t_g̃ + 1 and s ← s'.
      Update R_φK ← R_φK + γ^{t_g̃} r.
      Update θ with loss function J_1(θ).
    end while
    Store the experience (x, g̃, φ_K(s), R_φK, t_g̃) in M^A.
    Update ϑ with loss function J_2(ϑ).
  end while
end for
```

5.2.2 Abstract Agent

RRG-AS options transform optimization of policy $\pi_{\phi_K}(g \mid x)$ over subgoal options into optimization of policy $\pi^{\text{A}}_{\phi_K}(\tilde{g} \mid x)$. A range of optimization techniques can be used to learn policy $\pi^{\text{A}}_{\phi_K}(\tilde{g} \mid x)$. We adopt the Q-learning algorithm. Let $Q^{\text{A}}_\vartheta(x, \tilde{g})$ denote the parameterized Q-value function for the policy over $\tilde{g}$; then, $\pi^{\text{A}}_{\phi_K}$ is expressed as $\pi^{\text{A}}_{\phi_K}(\tilde{g} \mid x) = \arg\max_{\tilde{g}' \in \tilde{\mathcal{G}}(x)} Q^{\text{A}}_\vartheta(x, \tilde{g}')$. $Q^{\text{A}}_\vartheta(x, \tilde{g})$ can be learned by minimizing the following loss function:

$$J_2(\vartheta) = \mathbb{E}_{(x,\tilde{g},x',R_{\phi_K},t_{\tilde{g}}) \sim M^{\text{A}}} \left[\left(R_{\phi_K} + \gamma^{t_{\tilde{g}}} \max_{\tilde{g}' \in \tilde{\mathcal{G}}(x)} Q^{\text{A}}_\vartheta(x', \tilde{g}') - Q^{\text{A}}_\vartheta(x, \tilde{g}) \right)^2 \right], \tag{30}$$

where $t_{\tilde{g}}$ denotes the time steps needed for executing $\tilde{g}$; $\tilde{\mathcal{G}}(x)$ denotes the set of RRG-AS options available at abstract state x; M^{A} denotes the replay memory of the abstract agent. Each $\tilde{g}$ is sampled in a greedy policy with exploration probability $\mathcal{E}_2$ (or in a softmax policy). Algorithm 2 illustrates the overall RRG-RL algorithm.

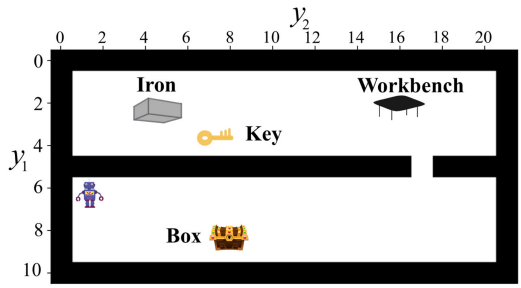


Fig. 6. 2D Minecraft map.

6 Experiments

To evaluate the performance of RRG-RL, we chose two complex tasks in both a discrete environment (2D Minecraft) and a continuous environment (Doom). In the two environments, rewards were given only after accomplishing a specific complex sequence of actions. All the experiments were conducted with PyTorch 0.3.1 and Python 3.5.

6.1 Environments

2D Minecraft [21] is a discrete version of the popular game Minecraft. As shown in Fig. 6, the agent lives in an 11×22 grid world with many raw materials, such as wood, a workbench, a key, and a treasure box. In this environment, the agent uses coordinate (y_1, y_2) as the input. It has four available actions: moving up, down, right, and left. In the environment, an action may fail with a probability of $1/5$, in which case the agent randomly chooses one of the other three actions. The agent can collect raw materials (e.g., a key) only when reaching the corresponding states. Andreas *et al.* [21] used task-specific information to solve complex tasks in a 2D Minecraft environment. In this work, we attempted to solve complex tasks without additional task-specific priors.

The Doom game [2] employs a first-person perspective in a semi-realistic 3D world (Fig. 7), where the agent can perform one of three standard actions: moving forward, turning left, and turning right. The three standard actions can be combined, yielding 4 combination actions: moving forward+turning left, moving forward+turning right, turning left+turning right, and moving forward+turning

Authorized licensed use limited to: BEIJING UNIVERSITY OF TECHNOLOGY. Downloaded on December 05,2024 at 00:21:13 UTC from IEEE Xplore. Restrictions apply.

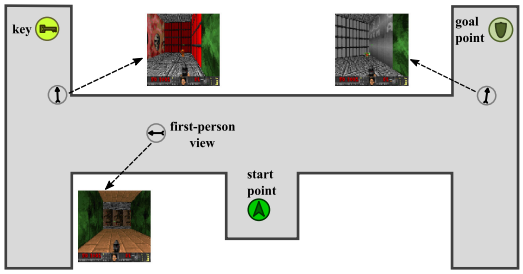


Fig. 7. Doom game map.

left+turning right. In this experiment, we used the 3 standard actions and 4 combination actions as the action set. The input of the agent is a three-channel 84×84 RGB image of the 3D environment. In addition to the high-dimensional state space, Doom tasks are usually complex, and the rewards are very sparse. For example, some doors require a key to open, but the key must be searched for by the agent in a large room. The agent is rewarded only when it opens the door. To accomplish these tasks, the agent needs to not only tackle the complex inputs but also carry out efficient plans in the large-scale state space.

6.2 Results of Geodesic Metric Learning

For the 2D Minecraft task, the geodesic-metric network was a three-layer (128-128-20) neural network, as shown in the dashed box in Fig. 11. The first and second layers were respectively followed by a ReLU and a Tanh activation. As presented in Algorithm 1, we used a uniform distribution over actions to sample trajectories from the environment. We trained the geodesic-metric network with 1000 episodes. For each episode, we evenly drew samples from memory M_{G} to construct batch dataset D_K which consisted of ten mini-batches of 64 state pairs. We adopted the Adam optimizer [55] to optimize the geodesic-metric network.

For the Doom game, the geodesic-metric network was a convolutional neural network (CNN). As shown in the dashed box of Fig. 15, all the filters in this CNN had the same size of 3×3 with stride 2. Each of its first three layers had 64 filters. Its fourth layer had 128 filters and the fifth had 256. The fifth layer was followed by two additional fully connected layers (the sizes of which were both 256). The first and second fully connected layers were followed by a ReLU and Tanh activation, respectively. We used a uniform random policy to sample trajectories. At each episode, we evenly drew samples from memory M_{G} to construct batch dataset D_K which consisted of 10 mini-batches of 128 state pairs. We trained the geodesic-metric network with 1500 episodes. Other parameters can be found in Supplementary Materials, available online.

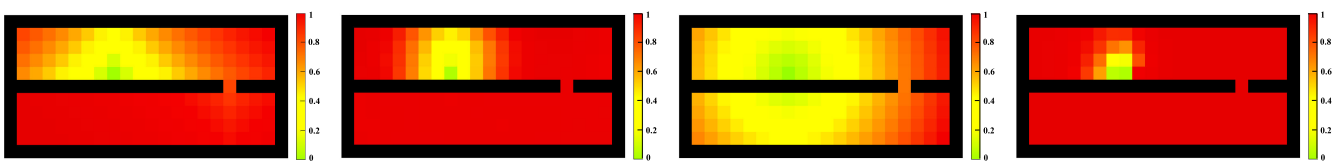
We consider the 2D Minecraft environment to visualize the results of the geodesic metric learning. To compare the learned geodesic metric $d_{\mathrm{G,w}}$ with the ground truth, we chose $s = (4, 8)$ as an example and visualized the learned geodesic metrics and the exact geodesic metrics calculated by a discount of 0.9 in Figs. 8b and 8a, respectively. The two figures showed that $d_{\mathrm{G,w}}$ was a good approximation of the exact geodesic metric. Particularly, the closer the state was to (4,8), the more accurate $d_{\mathrm{G,w}}$ was. This scenario was required by (ϵ, d)-compression that compresses only neighboring states.

We compared the learned geodesic metrics with two baselines: the euclidean metric and the successor representation [42]. We visualized the euclidean metric based on raw observations in Fig. 8c. The euclidean metric did not incorporate the geometric structure of the environment. Like our geodesic-metric network, the successor representation was able to encode environmental topology [42], [56]. To make fair comparison, we also used the uniform distribution over actions to sample trajectories from the environment to learn the successor representation, which also took 1000 episodes. More details can be found in Supplementary Materials, available online.

The geodesic metric can provide reward shaping for goal-reaching tasks where the agent is rewarded for reaching a goal state because it measures the relative distance between two states. These tasks are difficult to solve in a large and high-dimensional environment [57]. Some works [19], [48] used the euclidean metric between the agent's state and the goal state as the reward shaping for the goal-reaching task. Compared with the euclidean metric, our geodesic metric is more appropriate for complex environments. Let s_{g} be the goal state. The metric-based reward shaping is

$$r_{s_{\mathrm{g}}}(s, a) = r(s, a) - \alpha d\big(s, s_{\mathrm{g}}\big), \tag{31}$$

where d is a state metric and α is a weighting constant. Notably, we used this reward shaping function only for goal-reaching tasks (Fig. 9a) and did not use it for 2D Minecraft and Doom tasks. The metric-based reward shaping has been used in many studies [19], [48], [58]. If the metric d properly measures the distance between the current state and the goal state, it will guide the agent to reach the goal. Empirically, the learned metrics might not be exact for all state pairs and possibly provided a few wrong rewards, thus α should not be large and we set it to 0.15.



(a) (b) (c) (d)

Fig. 8. (a) Groud truth. (b) The learned geodesic metrics. (c) Metric based on raw observations. (d) Metric based on successor representation. Subfigures (a) to (d) visualize the four metrics between $(4, 8)$ and the other states, respectively. We normalized all the distance values to $[0, 1]$ with min-max normalization. Since larger values of the successor representation represent closer states, we used the successor representation's negative values and normalized them as the state metrics.

Authorized licensed use limited to: BEIJING UNIVERSITY OF TECHNOLOGY. Downloaded on December 05,2024 at 00:21:13 UTC from IEEE Xplore. Restrictions apply.

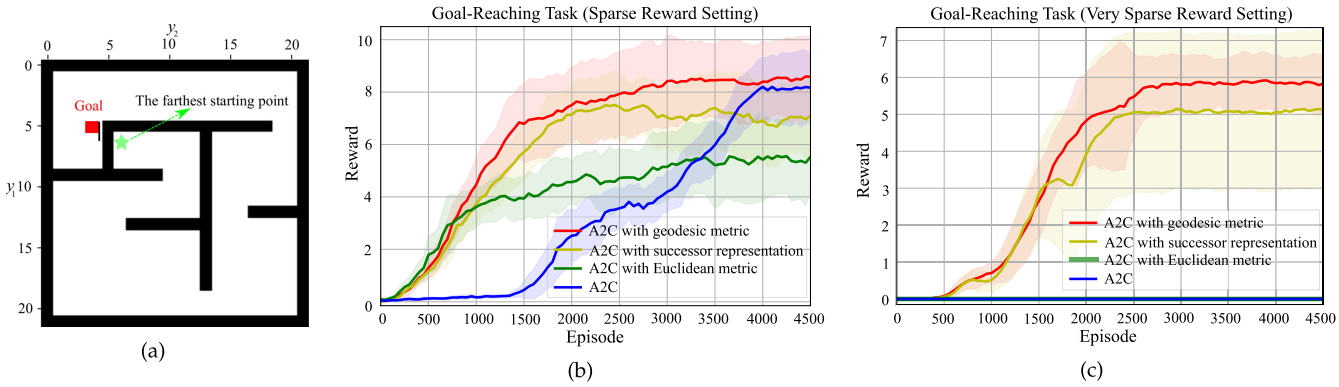


Fig. 9. (a) Complex grid world. Average rewards±std over 10 runs for (b) the goal-reaching task with random starting points and (c) the goal-reaching task with the farthest starting point. The shaded areas represent the standard deviations.

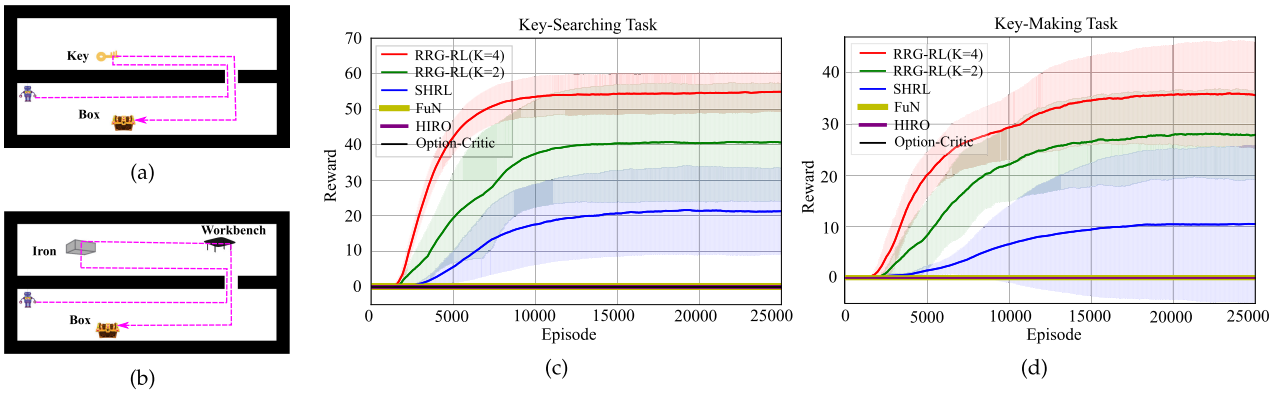


Fig. 10. (a) The key-searching task. (b) The key-making task. (c) Average rewards±std over 10 runs for the key-searching task. (d) Average rewards±std over 10 runs for the key-making task. The shaded areas represent the standard deviation.

To verify the advantages of the geodesic metric, we designed a complex environment where the agent is spawned in a 22×22 grid world (Fig. 9a). We set the goal as state $s = (6, 4)$. We chose two goal-reaching tasks: moving from random starting points to the goal and moving from the farthest starting point $(6, 6)$ to the goal. The agent obtained reward 20 only at the goal state. Thus, extrinsic rewards were sparse, especially for the second task. We evaluated several A2C algorithms with the geodesic metric, with the successor representation, with the euclidean metric, and without reward shaping, respectively. The geodesic-metric network for the complex grid world had the same architecture as that for the 2D Minecraft. All settings of the four algorithms were the same, except for reward shaping function $d(s, s_g)$. Training parameters can be found in Supplementary Materials, available online. Figs. 9b and 9c illustrate that the geodesic metric-based reward shaping achieved substantially better performance than the three baselines.

6.3 Results in 2D Minecraft

Task. In this environment, we designed two tasks: a key-searching task and a key-making task. As shown in Fig. 10a, the key-searching task required the agent to look for a key and then use it to open the treasure box. Without the key, the agent could not open the box or obtain the treasure. In this task, a reward of 100 was provided only when the agent successfully opened the box. As shown in Fig. 10b, for the key-making task, the agent must first collect the iron and the workbench to produce a key, and then used the key to open the box. Similar to the key-searching task, a reward of 100 was provided only when the box was successfully opened. Both tasks had long-term structural rewards [10] because the agent was rewarded only when it accomplished a sequence of specific behavior.

Baseline. Andreas *et al.* [21] first proposed a 2D Minecraft environment and solved tasks in the environment by providing task-specific instruction information. As we attempted to solve tasks without any additional task priors, we compared the performance of our approach against those methods not requiring additional task information. Kulkarni *et al.* [6] used a policy over subgoals and subgoal options to solve complex tasks with long-term structural rewards, which is essentially a subgoal-based HRL (SHRL) framework. This framework took the raw state space as the subgoal space, thus the number of subgoals equaled the number of states. Since RRG-RL consists of a policy over the abstract subgoals and the abstract subgoal options, SHRL can be considered as a special case without state abstraction, i.e., $K = 0$. We also evaluated the RRG-RL algorithm in the cases of $K = 2$ and $K = 4$. Furthermore, we compared our approach against those methods that combine state compression with the subgoal option framework using neural networks, such as FuN [18] and HIRO [19]. In the two algorithms, the (continuous) output of a neural network is used as the subgoal; thus, the high-level policy (the policy over subgoals) is a continuous control policy. We also evaluated

Authorized licensed use limited to: BEIJING UNIVERSITY OF TECHNOLOGY. Downloaded on December 05,2024 at 00:21:13 UTC from IEEE Xplore. Restrictions apply.

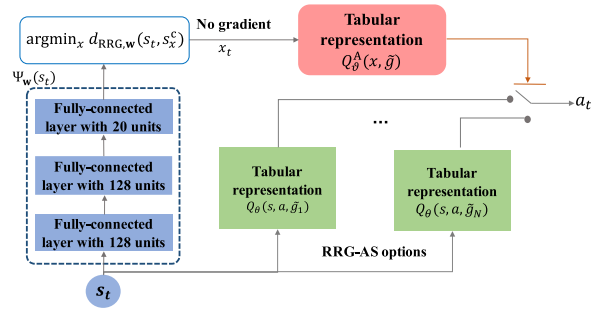


Fig. 11. The abstract reinforcement learning architecture for the 2D Minecraft environment. This was used to learn the agent policy in the second phase of RRG-RL.

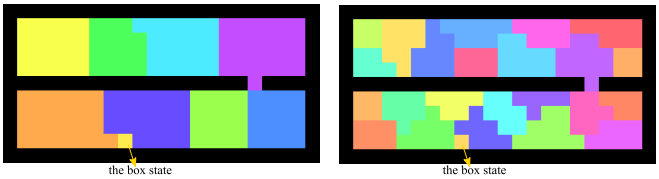


Fig. 12. (a) Visualization of abstract states for $K = 4$. (b) Visualization of abstract states for $K = 2$. One colored block represents an abstract state.

the performance of a well-known HRL method, the option-critic [14].

Implementation. In the two tasks, the agent was initialized to the same location, as illustrated in Fig. 6. We set the discount γ to 0.995. The experimental details for baselines are in Supplementary Materials, available online. As shown in Fig. 4, the RRG-RL algorithm has a two-phase training. Before the training, our algorithm had no access to any samples. In the first phase, we trained the geodesic-metric network with 1000 episodes (see details in Section 6.2). We represented the abstract agent and RRG-AS options with the tabular method, as shown in Fig. 11. After compressing the state space using Algorithm 1, we took each abstract state as an abstract subgoal of RRG-AS option. All states in 2D Minecraft could communicate with each other, thus all RRG-AS options were available at each state. We assigned a replay memory to each RRG-AS option. According to Algorithm 2, when an RRG-AS option was selected, the agent used the option policy to interact with the environment, stored the sampled trajectory into the replay memory for the selected option, and then updated the option policy. We implemented the policy over RRG-AS options using a greedy policy with the exploration probability 0.95 annealing to 0.05 as the learning proceeds. We implemented RRG-AS option policies using a softmax policy with a temperature of 0.01 annealing to 0.0001. Other parameters are provided in Supplementary Materials, available online.

Results. To perform RRG-RL, we first used the RRG metric $d_{\text{RRG},\mathbf{w}}$ to perform state compression with Algorithm 1. We then obtained 9 and 24 abstract states under $K = 4$ and $K = 2$, respectively, as shown in Figs. 12a and 12b. Since we set $c_R = 1/(1-\gamma)$ and the agent was rewarded only in the box state, the RRG metric between the box state and any other state was very large. Thus, the box state alone made up an abstract state. For the other

TABLE 2
Average Rewards±std for Two 2D Minecraft Tasks

	States	Subgoals	Key-searching	Key-making
Option-Critic [14]	161	-	0±0	0±0
FuN [18]	161	Continuous	0±0	0±0
HIRO [19]	161	Continuous	0±0	0±0
SHRL [6]	161	161	21.0±12.2	10.5 ± 15.3
RRG-RL (K=2)	24	24	40.6 ± 16.7	27.9±8.7
RRG-RL (K=4)	9	9	**54.9±5.7**	**35.6±10.3**

algorithms without state abstraction, the number of states was 161. After reducing the state space, our RRG-RL algorithm could solve the tasks in an abstract SMDP. In Figs. 10c and 10d, we illustrated the curves of average rewards±std of the five algorithms. Table 2 showed the final average rewards±std of the five algorithms. The results demonstrated that our algorithm ($K = 2, 4$) outperformed all the baselines: it not only achieved higher average rewards but also converged more quickly than all the baselines in the two tasks.

By comparing the RRG-RL and SHRL algorithms, we found that state compression significantly accelerated the learning of complex tasks, because the RRG-based state compression reduced the number of subgoals through clustering raw states. As shown in Table 2, when $K = 2$ and $K = 4$, the numbers of RRG-based abstract subgoals were 9 and 24 respectively. In contrast, the number of subgoals for the SHRL algorithm was 161. FuN [18] and HIRO [19] used the output space of the neural network as the subgoal space, resulting in a continuous subgoal space. As the number of RRG-AS options was less than that of subgoal options, it was easier to optimize the policy over RRG-AS options than to optimize the policy over subgoal options. Thus, it was probably easier to find a good solution with our algorithm than the baselines.

The Impact of Different N_1 Values. We empirically studied how the number of metric learning episodes N_1 impacts the overall performance of RRG-RL. As shown in Figs. 13a and 13b, RRG-RL suffered performance degradation when N_1 was small but maintained high performance when N_1 was comparatively large. The possible reason was as follows. A small N_1 led to insufficient training samples that weakened the generalization ability of the geodesic-metric network, but a large N_1 could provide sufficient samples for the geodesic-metric network. As N_1 increased, the marginal improvement from extra training samples got smaller, so the RRG-RL performance converged.

According to the above analysis, we propose a heuristic way to adjust N_1 automatically. We used the geodesic-metric network to predict label z_{ij} of state pair (s_i, s_j) (the true label was given according to Eq. (23)) through judging whether $\|\Psi_{\mathbf{w}}(s_i) - \Psi_{\mathbf{w}}(s_j)\|$ was smaller than χ. Before training the geodesic-metric network at each episode, we recorded the prediction accuracy for labels of state pairs from the new trajectory sampled from the environment. This accuracy reflected the geodesic-metric network's (generalization) capability to some degree. We terminated the metric learning phase when the prediction accuracy converged at a value within an error range of 0.5 percent during the last 50 episodes. More details can be found in

Authorized licensed use limited to: BEIJING UNIVERSITY OF TECHNOLOGY. Downloaded on December 05,2024 at 00:21:13 UTC from IEEE Xplore. Restrictions apply.

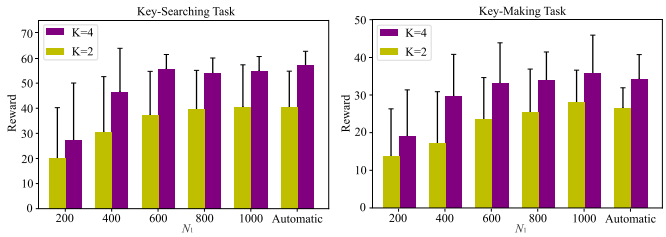


Fig. 13. Average rewards±std of different N_1 values for (a) the key-searching task and (b) the key-making task.

Supplementary Materials, available online. Figs. 13a and 13b demonstrated that the performance of automatic N_1 was similar to that of large N_1 (i.e., 800 and 1000). However, we believe that a better approach is to adjust N_1 according to the sampling sufficiency of the state space, which we leave for future work.

6.4 Results in Doom

Task. To evaluate the performance of our algorithm in a complex environment, we considered the key-searching task in Doom. As illustrated in Fig. 7, the task required the agent to pick up the key and then reach the goal. The agent started at a fixed position and orientation. It received a reward of 100 if it first obtained the key and then reached the goal point. However, it obtained a reward of only 20 if it arrived at the goal point without the key. A reward of 5 was provided if only the key was obtained. The agent had to accomplish the task within 1000 steps. Similar tasks in 2D environments (e.g., the Montezuma game) have been solved [6]. However, such task has not been accomplished in a first-person 3D environment without any task-specific priors.

Baseline. SHRL [6] without task-specific information and HIRO [19] have not been reported to solve RL problems in a high-dimensional state space. Thus, we did not evaluate the two algorithms in Doom. Vezhnevets *et al.* used FuN [18] to solve complex tasks in Montezuma, which is a game environment with high-dimensional images. Pathak *et al.* [23] proposed a curiosity-driven algorithm named intrinsic curiosity module (ICM) to solve tasks with spare rewards in the Doom environment. Therefore, we chose these two algorithms as baselines. We also compared our framework against the well-known option-critic [14].

Implementation. We used repeated actions four times during training in Doom, similar to Mnih *et al.* [1]. We trained the algorithms with 50000 episodes, and any episode requiring more than 1000 steps was terminated. We set the discount γ of each step to 0.995. The details for the baselines were in Supplementary Materials, available online. Like the Minecraft experiment, the RRG-RL algorithm for the Doom environment had no access to any sample before the two-phase training. In the first phase, the training of the geodesic-metric network took 1500 episodes (see details in Section 6.2). Like the 2D Minecraft, after the state space was compressed using Algorithm 1, we took each abstract state as an abstract subgoal of RRG-AS option. All RRG-AS options were available for each state. We assigned a replay memory to each RRG-AS option. We represented the abstract agent with the tabular method and RRG-AS options with CNNs, as shown in Fig. 15. Q-value function Q_θ for each RRG-AS option was represented by a convolution architecture that had a first layer with 32 8×8 filters of stride 4, a second layer with 128 6×6 filters of stride 2, and a third layer with 64 5×5 filters of stride 1. We then added two fully connected layers (with sizes 256 and 7). The first fully connected layer was followed by a ReLU activation function. We implemented the RRG-AS option policy using a greedy policy with exploration probability 1 annealing to 0.1 as the learning proceeds. We used a tabulation (or a matrix) ϑ to represent Q^{A}_ϑ and implemented the policy over RRG-AS options using a greedy policy with exploration probability 1 annealing to 0.0001 as the learning proceeds. Other parameters are provided in Supplementary Materials, available online.

Results. We illustrated the four algorithms' converged trajectories in Fig. 14a and reward curves in Fig. 14b. The optimal path for reaching the goal point from the starting point required approximately 36 steps. However, the optimal path for finding the key and then reaching the goal required approximately 115 steps, which was three times of 36 steps. The three baselines just reached the goal without finding the key. Thus, their discounted cumulative rewards were around $20 \times \gamma^{36} \approx 16$.

Due to the long-term structural exploration, the agent rarely accomplished the key-searching task in the training process. The key-searching task was difficult to solve without efficient long-term exploration. Efficient exploration techniques, such as curiosity-based methods [23], can

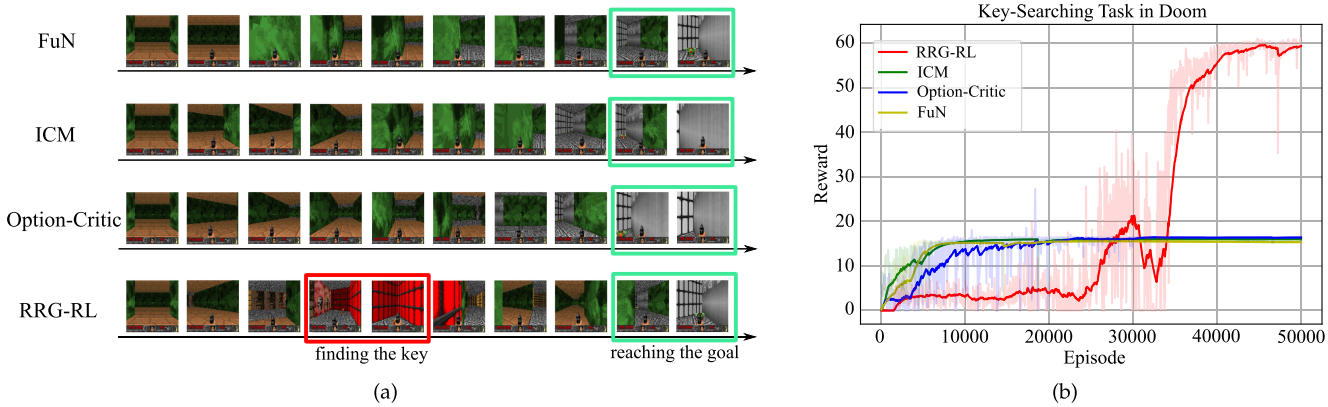


Fig. 14. (a) The learned trajectories of the four algorithms. (b) The reward curves of the four algorithms for the key-finding task in Doom.

Authorized licensed use limited to: BEIJING UNIVERSITY OF TECHNOLOGY. Downloaded on December 05,2024 at 00:21:13 UTC from IEEE Xplore. Restrictions apply.

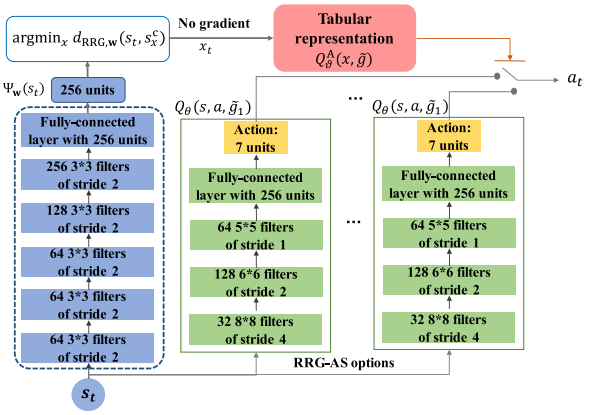


Fig. 15. The abstract reinforcement learning architecture for the Doom environment. It was used to learn the agent policy in the second phase of the RRG-RL algorithm.

explore more unseen states but cannot explore the long-term structural actions. Although FuN can explore long-term actions with subgoals, optimizing the policy over subgoals becomes very difficult due to the large-scale state space. Thus, FuN failed to accomplish the key-searching task.

Compared with the baselines, our state-temporal compression (i.e., RRG-based abstract states and RRG-AS options) enabled the agent to explore the environment in coarse-grained state space and larger span of time steps. This exploration was efficient for the key-searching task. Our algorithm could accomplish the key-searching task and achieved a reward of $100 \times \gamma^{115} \approx 58$. However, RRG-RL had a shortcoming: it required some warming-up episodes to start. In the beginning, RRG-AS option policies were ineffective, leading to a difficult learning problem for the policy over RRG-AS options. Consequently, the learning curve stayed at low values in early episodes, as shown in Fig. 14b. When RRG-AS options were learned relatively well, the learning curve for RRG-RL increased rapidly.

7 Conclusions and Discussions

Many difficulties exist in RL algorithms, such as high sample complexity, inefficient exploration, unstable training, etc. Compression is an efficient way to tackle these difficulties in RL algorithms [59]. Human brains also make decisions in an abstract way [60]. Although compression in state spaces and times has been studied for many years [9], [10], [12], [18], [19], it remains a persistent problem to perform practical state-temporal compression with theoretical guarantees. In this work, we propose a new metric RRG, which can be learned by a neural network, to perform state-temporal compression in RL. Furthermore, we develop a systematic and theoretical analysis for this metric. Our theory and algorithm make a step towards practical and optimality-preserving state-temporal compression in RL. Although our method addresses several open issues for compression in RLs, the following five questions remain unanswered.

First, can Compression in an SMDP Approximately Preserve Optimality for an MDP? We have proved that (ϵ, d)-compression based on three metrics approximately preserves optimality of an SMDP. Nevertheless, the optimal policy for an SMDP may not equal that for an MDP because the former relies on options. Suppose the set of policies induced by subgoal options covers (or approximately covers) the set of policies for an MDP; then, the optimal solution for an SMDP equals (or approximately equals) that for an MDP. Thus, one promising approach for preserving optimality of an MDP is to parameterize and optimize a subgoal space to enable the set of policies induced by subgoal options to cover the whole set of policies for an MDP.

Second, can the RRG-RL Algorithm be Applied to Any RL Tasks? The RRG-RL algorithm depends on subgoals. Although subgoals have been widely used in RL [6], [18], [48], [61], it remains unknown whether subgoal options are feasible for any RL tasks. Essentially, the first and second questions are the same. If subgoal options guarantee (or approximately guarantee) optimality for an MDP, the RRG-RL algorithm can be applied to any RL task. Consequently, the theoretical relation between an SMDP and an MDP is crucial for future studies.

Third, can State Compression be Applied in Nonstationary Environments? Existing compression approaches including ours make an implicit assumption that the environment is stationary. When an environment is dynamic over time, performing state compression is very challenging. We will consider the dynamic construction of state compression in future studies.

Fourth, can the Degree of State Compression be Automatically Adjusted? K represents the degree of state space reduction and is predefined in this work. Experimental results have shown that K is an important factor for the performance of RRG-RL. Thus, the automatic setting of K merits study. The reason for fixing K is that current metric learning approach requires a fixed K. If the value of K is variable in geodesic metric learning, K can be automatically adjusted for the RRG metric-based compression.

Fifth, do More Efficient Methods Exist to Collect Data for Metric Learning? Although it is common to collect data with a uniform random policy for learning state representations or abstractions [35], [40], [62], [63], the uniform random policy may lead to insufficient exploration especially for realistic and big environments [62], [63]. To tackle this problem, we believe that there are two approaches worth studying in the future. The first approach is to collect data using a policy that can explore the entire state space efficiently. For example, the studies [62], [63] attempt to leverage the deep Q-network [1] to collect data such that training data can cover all regions of the state space. Also, one can adopt more efficient exploration policies, such as the curiosity-based exploration policy [23]. The second approach is to use a finite set of policies to collect data. If these policies are diverse enough, they can sample data from all regions of the state space. Note that the agent in an SMDP typically has some diverse option policies. We can use these policies to collect data for learning the geodesic-metric network. It is desirable to incorporate the phase of agent policy learning into the phase of metric learning.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 61671266, Grant

Authorized licensed use limited to: BEIJING UNIVERSITY OF TECHNOLOGY. Downloaded on December 05,2024 at 00:21:13 UTC from IEEE Xplore. Restrictions apply.

61836004, Grant U19B2034, and Grant 61836014 and in part by the Tsinghua-Guoqiang research program under Grant 2019GQG0006.

REFERENCES

[1] V. Mnih *et al*. "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, 2015.

[2] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaśkowski, "ViZDoom: A doom-based AI research platform for visual reinforcement learning," in *Proc. IEEE Conf. Comput. Intell. Games*, 2016, pp. 1–8.

[3] S. M. Kakade, "On the sample complexity of reinforcement learning," PhD dissertation, Gatsby Comput. Neurosci. Unit, Univ. College London, London, U.K., 2003.

[4] A. L. Strehl, L. Li, and M. L. Littman, "Reinforcement learning in finite MDPs: PAC analysis," *J. Mach. Learn. Res.*, vol. 10, pp. 2413–2444, 2009.

[5] R. I. Brafman and M. Tennenholtz, "R-MAX-A general polynomial time algorithm for near-optimal reinforcement learning," *J. Mach. Learn. Res.*, vol. 3, no. 2, pp. 213–231, 2003.

[6] T. D. Kulkarni, K. R. Narasimhan, A. Saeedi, and J. B. Tenenbaum, "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 3675–3683.

[7] J. A. Arjona-Medina , M. Gillhofer, M. Widrich, T. Unterthiner, J. Brandstetter, and S. Hochreiter, "RUDDER: Return decomposition for delayed rewards," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 13 544–13 555.

[8] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[9] L. Li, T. J. Walsh, and M. L. Littman, "Towards a unified theory of state abstraction for MDPs," in *Proc. Int. Symp. Artif. Intell. Math.*, 2006, pp. 531–539.

[10] D. Abel, D. Arumugam, L. Lehnert, and M. L. Littman, "State abstractions for lifelong reinforcement learning," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 10–19.

[11] B. Ravindran, "An algebraic approach to abstraction in reinforcement learning," PhD dissertation, Dept. Comput. Sci., Univ. Massachusetts at Amherst, Amherst, MA, USA, 2004.

[12] B. Ravindran and A. G. Barto, "SMDP homomorphisms: An algebraic approach to abstraction in semi-Markov decision processes," in *Proc. 12th Int. Joint Conf. Artif. Intell.*, 2003, pp. 1011–1016.

[13] R. S. Sutton, D. Precup, and S. Singh, "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning," *Artif. Intell.*, vol. 112, no. 1/2, pp. 181–211, 1999.

[14] P.-L. Bacon, J. Harb, and D. Precup, "The option-critic architecture," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 1726–1734.

[15] Y. Jiang, S. Gu, K. Murphy, and C. Finn, "Language as an abstraction for hierarchical deep reinforcement learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 9414–9426.

[16] B. Ravindran and A. G. Barto, "Approximate homomorphisms: A framework for non-exact minimization in Markov decision processes," in *Proc. 5th Int. Conf. Knowl. Based Comput. Syst.*, 2004.

[17] A. A. Taïga, A. Courville, and M. G. Bellemare, "Approximate exploration through state abstraction," 2018, *arXiv: 1808.09819*.

[18] A. S. Vezhnevets *et al*."Feudal networks for hierarchical reinforcement learning," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 3540–3549.

[19] O. Nachum, S. Gu, H. Lee, and S. Levine, "Data-efficient hierarchical reinforcement learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 3303–3313.

[20] N. Ferns, P. Panangaden, and D. Precup, "Metrics for finite Markov decision processes," in *Proc. 20th Conf. Uncertainty Artif. Intell.*, 2004, pp. 162–169.

[21] J. Andreas, D. Klein, and S. Levine, "Modular multitask reinforcement learning with policy sketches," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 166–175.

[22] J. Oh, S. Singh, H. Lee, and P. Kohli, "Zero-shot task generalization with multi-task deep reinforcement learning," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 2661–2670.

[23] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 2778–2787.

[24] T. G. Dietterich, "Hierarchical reinforcement learning with the MAXQ value function decomposition," *J. Artif. Intell. Res.*, vol. 13, pp. 227–303, 2000.

[25] P. S. Castro, "On planning, prediction and knowledge transfer in fully and partially observable Markov decision processes," PhD dissertation, School Comput. Sci., McGill Univ., 2011.

[26] P. S. Castro and D. Precup, "Automatic construction of temporally extended actions for MDPs using bisimulation metrics," in *Proc. Eur. Workshop Reinforcement Learn.*, 2011, pp. 140–152.

[27] P. S. Castro and D. Precup, "Using bisimulation for policy transfer in MDPs," in *Proc. AAAI Conf. Artif. Intell.*, 2010, pp. 1065–1070.

[28] N. Ferns, P. Panangaden, and D. Precup, "Metrics for Markov decision processes with infinite state spaces," in *Proc. 21th Conf. Uncertainty Artif. Intell.*, 2005, pp. 201–208.

[29] A. L. Gibbs and F. E. Su, "On choosing and bounding probability metrics," *Int. Statist. Rev.*, vol. 70, no. 3, pp. 419–435, 2002.

[30] P. S. Castro, "Scalable methods for computing state similarity in deterministic Markov decision processes," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 10069–10076.

[31] J. J. Taylor, D. Precup, and P. Panangaden, "Bounding performance loss in approximate MDP homomorphisms," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2008, pp. 1649–1656.

[32] D. Abel, N. Umbanhowar, K. Khetarpal, D. Arumugam, D. Precup, and M. L. Littman, "Value preserving state-action abstractions," in *Proc. 23rd Int. Conf. Artif. Intell. Statist.*, 2020, pp. 1639–1650.

[33] S. Mannor, I. Menache, A. Hoze, and U. Klein, "Dynamic abstraction in reinforcement learning via clustering," in *Proc. 21st Int. Conf. Mach. Learn.*, 2004, pp. 560–567.

[34] T. G. Dietterich, "State abstraction in MAXQ hierarchical reinforcement learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2000, pp. 994–1000.

[35] M. C. Machado, M. G. Bellemare, and M. Bowling, "A Laplacian framework for option discovery in reinforcement learning," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 2295–2304.

[36] S. Tiwari and P. S. Thomas, "Natural option critic," in *Proc. 33rd AAAI Conf. Artif. Intell.*, 2019, pp. 5175–5182.

[37] D. Abel, D. E. Hershkowitz, and M. L. Littman, "Near optimal behavior via approximate state abstraction," in *Proc. 33rd Int. Conf. Mach. Learn.*, 2016, pp. 2915–2923.

[38] R. Ortner, "Pseudometrics for state aggregation in average reward Markov decision processes," in *Proc. Int. Conf. Algorithmic Learn. Theory*, 2007, pp. 373–387.

[39] N. Jiang, A. Kulesza, and S. Singh, "Abstraction selection in model-based reinforcement learning," in *Proc. 32nd Int. Conf. Mach. Learn.*, 2015, pp. 179–188.

[40] S. Mahadevan and M. Maggioni, "Proto-value functions: A Laplacian framework for learning representation and control in Markov decision processes," *J. Mach. Learn. Res.*, vol. 8, pp. 2169–2231, 2007.

[41] J. Bouttier, P. Di Francesco , and E. Guitter, "Geodesic distance in planar graphs," *Nucl. Phys. B*, vol. 663, no. 3, pp. 535–567, 2003.

[42] P. Dayan, "Improving generalization for temporal difference learning: The successor representation," *Neural Comput.*, vol. 5, no. 4, pp. 613–624, 1993.

[43] A. Barreto *et al*."Successor features for transfer in reinforcement learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 4055–4065.

[44] T. Verechtchaguina, I. Sokolov, and L. Schimansky-Geier , "First passage time densities in resonate-and-fire models," *Phys. Rev. E Statal Nonlinear Soft Matter Phys.*, vol. 73, no. 3, 2006, Art. no. 031108.

[45] T. G. Mattos, C. Meja-Monasterio, R. Metzler, and G. Oshanin, "First passages in bounded domains: When is the mean first passage time meaningful?," *Phys. Rev. E Statal Nonlinear Soft Matter Phys.*, vol. 86, 2012, Art. no. 031143.

[46] L. Yang and R. Jin, "Distance metric learning: A comprehensive survey," Dept. Comput. Sci. Eng., Michigan State Univ., Tech. Rep., 2006.

[47] R. Hadsell, S. Chopra, and Y. LeCun , "Dimensionality reduction by learning an invariant mapping," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2006, pp. 1735–1742.

[48] O. Nachum, S. Gu, H. Lee, and S. Levine, "Near-optimal representation learning for hierarchical reinforcement learning," in *Proc. Int. Conf. Learn. Representations*, 2019.

[49] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," *J. Mach. Learn. Res.*, vol. 10, no. 1, pp. 1633–1685, 2009.

Authorized licensed use limited to: BEIJING UNIVERSITY OF TECHNOLOGY. Downloaded on December 05,2024 at 00:21:13 UTC from IEEE Xplore. Restrictions apply.

[50] N. Savinov, A. Dosovitskiy, and V. Koltun, "Semi-parametric topological memory for navigation," in *Proc. Int. Conf. Learn. Representations*, 2018.
[51] N. Savinov *et al.*, "Episodic curiosity through reachability," in *Proc. Int. Conf. Learn. Representations*, 2019.
[52] S. K. Bhatia, "Adaptive k-means clustering," in *Proc. Int. Florida Artif. Intell. Res. Soc. Conf.*, 2004, pp. 695–699.
[53] G. D. Konidaris, L. P. Kaelbling, and T. Lozano-Perez , "Symbol acquisition for probabilistic high-level planning," in *Proc. 24th Int. Joint Conf. Artif. Intell.*, 2015, pp. 3619–3627.
[54] G. Konidaris, L. P. Kaelbling, and T. Lozano-Perez , "From skills to symbols: Learning symbolic representations for abstract high-level planning," *J. Artif. Intell. Res.*, vol. 61, pp. 215–289, 2018.
[55] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[56] M. C. Machado, C. Rosenbaum, X. Guo, M. Liu, G. Tesauro, and M. Campbell, "Eigenoption discovery through the deep successor representation," in *Proc. Int. Conf. Learn. Representations*, 2018.
[57] G. Brockman *et al.* "OpenAI gym," 2016, *arXiv:1606.01540*.
[58] V. Pong, S. Gu, M. Dalal, and S. Levine, "Temporal difference models: Model-free deep RL for model-based control," in *Proc. Int. Conf. Learn. Representations*, 2018.
[59] G. Konidaris, "On the necessity of abstraction," *Current Opinion Behav. Sci.*, vol. 29, pp. 1–7, 2019.
[60] M. M. Botvinick, "Hierarchical reinforcement learning and decision making," *Current Opinion Neurobiol.*, vol. 22, no. 6, pp. 956–962, 2012.
[61] J. Rafati and D. Noelle, "Unsupervised subgoal discovery method for learning hierarchical representations," in *Proc. 7th Int. Conf. Learn. Representations*, 2019.
[62] J. Mugan and B. Kuipers, "Autonomous learning of high-level states and actions in continuous environments," *IEEE Trans. Auton. Mental Develop.*, vol. 4, no. 1, pp. 70–86, Mar. 2012.
[63] A. Srinivas, R. Krishnamurthy, P. Kumar, and B. Ravindran, "Option discovery in hierarchical reinforcement learning using spatio-temporal clustering," 2020, *arXiv:1605.05359*.

Shangqi Guo received the BS degree in mathematics and physics basic science from the University of Electronic Science and Technology of China, Chengdu, China, in 2015. He is currently working toward the PhD degree in the Department of Automation, Tsinghua University, Beijing, China. His current research interests include inference in artificial intelligence, brain-inspired computing, computational neuroscience, and reinforcement learning.

Qi Yan received the BS degree from the Department of Automation, Tsinghua University, Beijing, China, in 2015, where he is currently working toward the PhD degree. His current research interests include computer vision, artificial intelligence, and reinforcement learning.

Xin Su received the BS degree in mathematics and physics basic science from the Department of Physics, Tsinghua University, Beijing, China, in 2016. He is currently working toward the PhD degree in the Department of Automation, Tsinghua University, Beijing, China. His current research interests include lifelong learning, reinforcement learning, and general machine learning techniques.

Xiaolin Hu (Senior Member, IEEE) received BE and ME degrees from the Wuhan University of Technology, Wuhan, China, in 2001 and 2004, respectively, both in automotive engineering and the PhD degree in automation and computer-aided engineering from The Chinese University of Hong Kong, Hong Kong, in 2007. He is currently an associate professor at the Department of Computer Science and Technology, Tsinghua University, Beijing, China. His current research interests include deep learning and computational neuroscience. At present, he is an associate editor of the *IEEE Transactions on Image Processing*. Previously, he was an associate editor of the *IEEE Transactions on Neural Networks and Learning Systems*.

Feng Chen (Member, IEEE) received the BS and MS degrees in automation from Saint-Petersburg Polytechnic University, Saint Petersburg, Russia, in 1994 and 1996, respectively, and the PhD degree from the Automation Department, Tsinghua University, Beijing, China, in 2000. He is currently a professor with Tsinghua University. His current research interests include computer vision, brain-inspired computing, and inference in graphical models.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/csdl.

Authorized licensed use limited to: BEIJING UNIVERSITY OF TECHNOLOGY. Downloaded on December 05,2024 at 00:21:13 UTC from IEEE Xplore. Restrictions apply.