Graph-based Task Sequencing Framework for Curriculum Learning

Xiaohui Chen **Ramtin Hosseini** **Xu Han**

Abstract

Learning through curriculum has shown promising results in different domains. The idea of curriculum learning has been largely applied in both animal training and pedagogy. Motivated by this, several recent researches have studied using curriculum learning in a reinforcement learning setting and have proved success in this domain too.
By doing transfer learning, the performance could be improved on a final target task. However, there is a limitations in the existing works when applied into reinforcement learning: the method generally needs to define the transfer potential function on feature space; However, when the model adopts neural network to estimate the return function, it is hard to define the freedom of the features. To address this limitation, we introduce a method to define the complexity vector of the task whose dimension is much smaller than the input of the model. Based on the transfer potential, we construct a curriculum graph and generate curriculum learning sequence on it. The results show that our method outperforms the direct learning method.

1 Introduction

Reinforcement learning (RL) has been proved a successful way for addressing sequential decision tasks in which the agent has only limited environmental feedback. Curriculum learning in reinforcement learning is used to shape exploration by presenting the agent with increasingly complex tasks. In recent years, research has shown that transfer learning methods can be leveraged to construct curricula that sequence a series of simpler tasks such that performance on a final target task is improved. A major limitation of existing approaches is that such curricula are handcrafted by humans that are typically domain experts.

There exists a few other previous studies which have explored automatic task generation ways. Task generation is the process of creating a good set of intermediate tasks from which to obtain experience samples. Most of previous works have used a linear sequencing curriculum. Sequencing examines how to create a partial ordering over the set of experience samples.

The most general form of a curriculum is a directed acyclic graph (DAG) over subsets of samples. However, in practice, simplified versions of this representation are often used. In the simplest case, a curriculum is an ordering over samples from a single task. When multiple tasks can be used in a curriculum, curricula are often created at the task-level. These curricula can be represented as a linear chain, or sequence. In this case, there is exactly one source for each intermediate task in the curriculum. It is up to the transfer learning algorithm to appropriately retain and combine information gathered from previous tasks in the chain. More generally, they can be represented as a full directed acyclic graph of tasks.

Recent researches using graph to represent tasks usually adopt some very simple metrics which only concerns the feature domain. We will improve it by coming up with a better transfer function and, what's more, we will combine several measure metrics to decide a bunch of relevant tasks and how to

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

connect them to the final task. More importantly, these works rely on creating clustering tasks based on a simple heuristic function; however, in the presented work, we use k-means approach in order to cluster different tasks. In this process, we will produce a generating curriculum graph.

Contribution. Our contribution in this work are in 4 folds:

- Deploying k-means in order to cluster different tasks rather than using a simple heuristic function.
- Transfer Learning in the graph space
- An automatic way to produce a curriculum graph based on several metrics
- A weight value on edges based on the performance of source tasks (nodes)

2 Related Work

Curriculum Learning. Lazaric et al. (2008) verified that target task can be learned fast through a sequence of transfer learning. Aiming to introduce the idea of transfer learning in to the area of reinforcement learning, Thrun (1998) proposed task generation to create a set of tasks related to each other for the learning of target task. Narvekar et al. (2016) introduces a number of methods to create intermediate tasks for a specific final task. The methods hinge on a definition of a domain as a set of MDPs identified by a task descriptor, which is a vector of parameters specifying the degrees of freedom in the domain.

Task Selection has been studied for general transfer learning, and presents common aspects with the task selection that is part of sequencing in curriculum learning. Several approaches consider learning a mapping from source tasks to target tasks, and estimating the benefit of transferring between the tasks: Ammar et al. (2014), Sinapov et al. (2015), and Sukhbaatar et al. (2017).Curriculum learning has been used to guide the exploration by generating a sequence of goals or initial states within the same environment where all tasks have the same dynamics: Asada et al. (1996), and Sukhbaatar et al. (2017).

Given a set of tasks, the first thing to do is sequencing them. The goal of sequencing is to order them in a way that facilitates learning. Most of existing sequencing method manage curriculum as a unidirectional sequence based on the environment applicability. They order experience progressively containing more properties shared with the final target task.Generally. the target task can benefit from transfer learning in two ways: policy or value functions. Some works adopt policy based transfer method: Bachman et al. (2019).

Sampling sequencing is a very popular method, which sample sub-tasks from the final target task, without changing the domain. Schaul et al. (2015) proposed to prioritize and replay important transition more. While Ren et al. (2018) proposed to sort samples using a complexity index.They provide one specific instantiation of these functions, which are used in experiments on the Arcade Learning Environment.

Curriculum Learning in graph space. Another important sequencing method class connects tasks with edges into directed acyclic task graph. The nodes in the graph represent a certain task. A directed edge implied one task is a source task for another. The key of such method is how to represent a task feature for generate curriculum graph. Svetlik et al. (2017) encodes features as the properties of the domain. And Silva and Costa (2018)represents the task feature by an object-oriented approach. Both works adopted transfer potential to create curriculum graph.

The first formalization of the sequencing problem is as a Markov Decision Process. These methods formulate curriculum generation as an interaction between 2 types of MDPs. The first is the standard MDP, which models a learning agent (i.e., the student) interacting with a task. The second is a higher level meta-MDP for the curriculum agent (i.e., the teacher), whose goal is to select tasks for the learning agent. Most of prior works, have used a sequence curriculum from low-level to high-level task, but Svetlik et al. (2017), and Rogers et al. (2012) create a curriculum graph based on task similarity.

3 Learning Methodology

We use Reward Shaping as a means of transferring knowledge from source tasks to a target. In other words, we modified The reward signal by adding additional reward, driving the agent towards the desired behavior. By doing so, the reward function becomes:

$$R'(s, a, s') = R(s, a, s') + F(s, a, s') \tag{1}$$

where F is the shaping function. The advantage of using a reward function is that, if the reward is sparse, the agent may have no feedback for many actions, making learning more difficult. The shaping function can fill these gaps, providing advice on the value of the actions.
In the following subsections, we introduce the model along with problem definition and notation and the defined measured metrics that we used for evaluation.

3.1 Problem Definition and Notation

Let $\mathcal{V}$ be a set of MDPs, and $v_i \in \mathcal{V}$ represent the MDP task i. The goal is that given a target MDP $v_t \in \mathcal{V}$, we want to find a suitable curriculum path $\mathcal{T} = \{v_1, v_2, \ldots, v_t\}$, where $\mathcal{T} \subseteq \mathcal{V}$, such that the agent can learn v_t more efficiently through the curriculum $\mathcal{T}$.

Let $C = (V, E)$ be a directed graph, where $v_i \in V$ is an MDP in a domain $\mathcal{V}$, $e_{ij} \in E$ represent the transfer learning direction from MDP task v_i to v_j. Let $F_t$is a vector of degrees of freedom for task v. Each element of F_t is a feature, representing a property of the domain. These properties are:

- Number of grid (width * height)
- Size of agent view
- Number of rooms
- Number of objects (can be repeated)
- Type of object (number of unique types)

Besides, we define a $f(v_i)$ that describes the property of the MDP domain v_i (state space, action space, etc). And we also define a transfer potential function $\phi(e_{ij})$ that evaluates the potential of transferring task i to task j. We consider $\phi(e_{ij})$ as the weight value for edge e_{ij}. In the following subsection, we will how to find an efficient curriculum T given a graph $C = (V, E)$ and the corresponding function f and ϕ.

3.2 Model Architecture and Policy Search Algorithm

Model. The proposed network architecture shown in Figure 1 is consisted of 3 convolution layers followed by a long-short term memory (LSTM) unit. We used the Actor Critic methods, where:

- The "Critic" estimates the value function. This could be the action-value (the Q value) or state-value (the V value).
- The "Actor" updates the policy distribution in the direction suggested by the Critic (with policy gradients).

where both the Critic and Actor functions are parameterized with neural networks.

Policy. In this work, we used Proximal Policy Optimization (PPO) algorithm. This algorithm is a first-order approximation of the expected return while carefully ensuring that the approximation does not deviate too far from the underlying objective. PPO adds a soft constraint that can be optimized by a first-order optimizer and the main reason for using it is that, vanilla policy gradient methods have poor data efficiency and robustness; and trust region policy optimization (TRPO) is relatively complicated, and is not compatible with architectures that include noise (such as dropout) or parameter sharing (between the policy and value function, or with auxiliary tasks).

3.3 Measure Metrics

The characterization of an MDP task is always heuristic. In this work, we first consider using a simple proxy for experience, based on one of these factors: the size of the state space $|S_t|$ and the size of

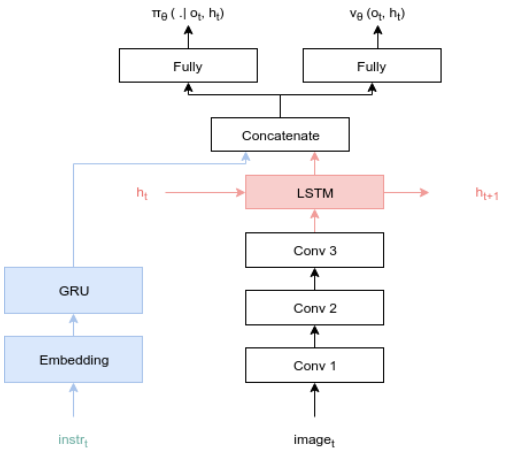


Figure 1: The architecture of the proposed model

action space $|A_t|$.

$$f(v_i) = |S_t||A_t| \tag{2}$$

The Transfer potential can be considered as a short term Curriculum Measurement, in which the transfer learning evaluation methods have been used in a wide range. We list four possible cases here:
Regret (Reg). N is the number of episodes executed.

$$\phi_1(v_i, v_j) = -(Ng - \sum_{n=1}^{N} G_j^n) \tag{3}$$

Jumpstart (JS). G_j^n is the return obtained during episode n in task, D represent the first D episodes a agent excutes.

$$\phi_2(v_i, v_j) = \frac{1}{D} \sum_{d=1}^{D} G_j^d) \tag{4}$$

Max-return (MR). $G_j^I = \frac{1}{Q} \sum_{i=1}^{Q} G_f^i$ is the average return over the Q episodes in the evaluation step I on task v_j.

$$\phi_3(v_i, v_j) = \max_{I \in \mathcal{E}} G_j^I \tag{5}$$

Time-to-threshold (TTT). $a(v_i)$ be the number of actions the agent executed in task v_i before moving on to task v_j in the curriculum, and $a_g(v_j)$ be the number of actions the agent executed in task v_j until the evaluation step in which the policy achieves an average return of g

$$\phi_4(v_i, v_j) = -(a_g(v_j) + a(v_i)) \tag{6}$$

The estimation of the potential transfer is expensive since it need to consider branch pruning first, leaving at most K candidate MDPs for each $v_{i,j}$, this can be simply achieved by using a cheap estimation, we define it as ϕ_0

$$\phi_0(v_i, v_j) = \frac{|D_{Q_i}(j)|}{1 + f(v_j) - f(v_i)} \tag{7}$$

where the definition of $D_{Q_i}(j)$ can be found in Svetlik et al. (2017). For each v_i, We can select the top K neighbours and thus reduce the complete graph into a K-regular graph, and perform the curriculum searching.

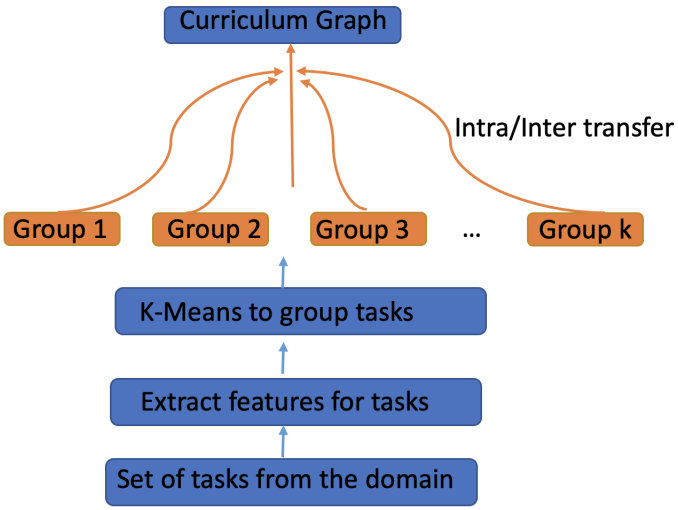


Figure 2: Curriculum graph generation

Algorithm 1 Curriculum Graphs Optimization

Input: seeds, a set of MDPs $\mathcal{V}$, v_t, max_iter
Output: curriculum graph $C = (V, E)$

1: Initialize graph $C_0 = (V, E)$ as a K-regular graph, edge value is evaluated with equation (6)
2: $t \leftarrow 0$
3: **while** the objective not converge and $t <$ max_iter **do**
4: Sample a curricular path $\mathcal{T} = \{v_1, v_2, \ldots, v_t\}$ starting from an random seed $t_0 \in$ seeds based on graph structure information.
5: Evaluate curriculum path $\mathcal{T}$ with equation (2/3/4/5)
6: Optimize Graph structure C
7: **end while**
8: **return** C

3.4 Constructing Curriculum Graph

Curriculum Graph Initialization. We follow the graph generation process defined in Svetlik et al. (2017), where in the Intra-group transfer and Inter-group transfer algorithm, the potential transfer function can be replaced by any measure metrics discussed above during the curriculum graph generation. While the original potential transfer function will be used for edges initialization for graph at the beginning. In our approach, first extract the features, then we use k-means algorithm in order to cluster different tasks. Our curriculum graph generation procedure is shown in figure 2.

Intra group transfer. In intra-group transfer, the transfer happens between tasks within the same group. Intra-group transfer shown in Algorithm 2, takes as input a group $g \in G$, which is then sorted with respect to the transfer potential to the final task.
The algorithm iterates over the group, assigning an edge at line 6 between task j and task i if j has the highest potential among the tasks coming after i in the group, and the potential is above ϵ.
The resulting set of edges is returned.

Inter group transfer. While any given group contains tasks that are themselves related, there may be tasks in separate groups that would also benefit from transfer. This is the goal of Inter-group transfer.
Two task groups are compared directly for transfer. The group g is considered the source group and

Algorithm 2 Intra-group transfer

Given: task group g

$\mathcal{H} = \{\}$, sort g in descending order
for $t_i \in g$ **do**
  $t_m = \arg\max_{t_j \in g \setminus i<j} v(t_j, t_i)$
  if $v(t_m, t_i) > \epsilon$ **then**
    $\mathcal{H} \leftarrow \mathcal{H} \bigcup \langle t_m, t_i \rangle$
  end if
end for
Return $\mathcal{H}$

Algorithm 3 Inter-group transfer

Given: ordered transfer groups g, g'

$\mathcal{H} = \{\}$
for $t_i \in g'$ **do**
  $t_m = \arg\max_{t_j \in g} v(t_j, t_i)$
  if $v(t_m, t_i) > \epsilon$ **then**
    $\mathcal{H} \leftarrow \mathcal{H} \bigcup \langle t_m, t_i \rangle$
  end if
end for
Return $\mathcal{H}$

the other, g', is considered the target group. The algorithm iterates over g, and adds an edge to a task in g' under the same condition as Intra-group transfer.

Curricular Path Sampling. In this section we will discuss the details of sampling a efficient Curricular Path $\mathcal{T}$ utilizing the information in Curriculum graph C. For sampling, we consider topological sort on graph according to Svetlik et al. (2017). The topological sort is defined as follow:

$$\text{sort}(C, v_t) = \{v_{s_1}, v_{s_2}, \ldots, v_t\} \tag{8}$$

where v_t is the target task and the C is the built curriculum graph, we expect the output to be a set of task nodes in C with order.

Combinatorial Optimization in Graph. Combinatorial Optimization (CO) problems are suitable for leaning the best configuration of the graph structure. With CO, We can train to optimize the graph structure by evaluating the sampled path quality. The initial algorithm flow is showed in algorithm 1, and will investigate deeper in the next.

4 Experiments and Results

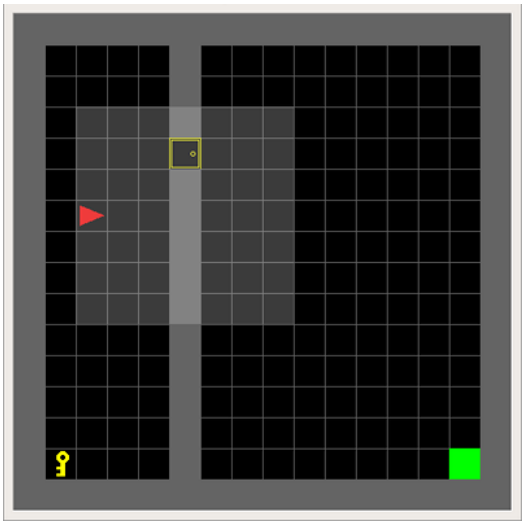
We evaluated our proposed method in a discrete domain: Gridworld. As we mentioned in the previous section, we defined different metrics to evaluate and contrast our method.

4.1 Experiment details

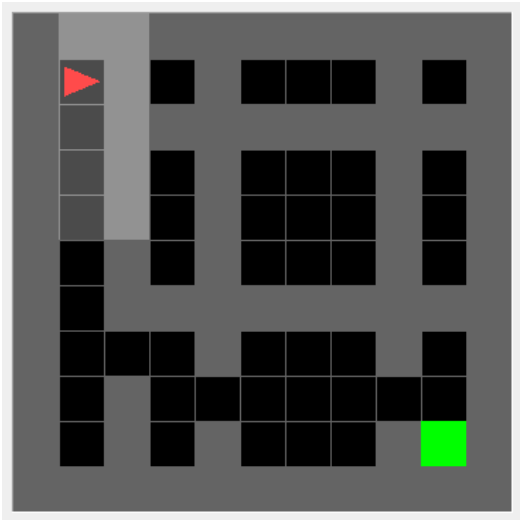
We employ the open source library minigrid Chevalier-Boisvert et al. (2018) to develop our algorithm, and we choose Proximal policy optimization algorithms (PPO) Schulman et al. (2017) as our reinforcement learning policy search. Our goal is to transfer the obtained knowledge between different environments based on our proposed method.

We trained our model using GTX 1070 graphical card, with 30 epochs with or without transfer learning. The learning rate is set to be 0.001 and batch size is set to be 256. The graph generation and task sequencing using topological sort is based on the networkx library, implemented in python Hagberg et al. (2008).

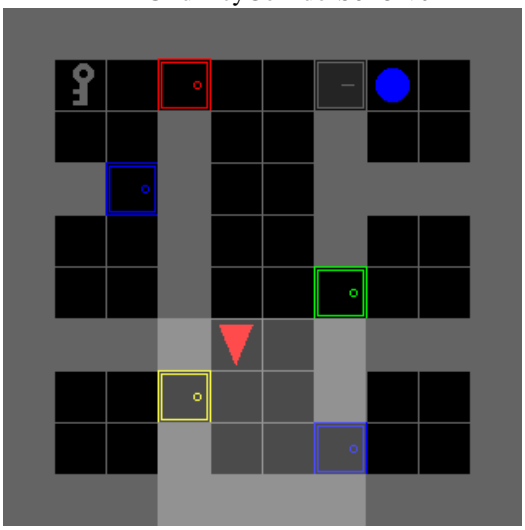
MiniGrid-DoorKey-16x16-v0



MiniGrid-SimpleCrossingS11N5-v0



MiniGrid-KeyCorridorS6R3-v0



MiniGrid-ObstructedMaze-Full-v0

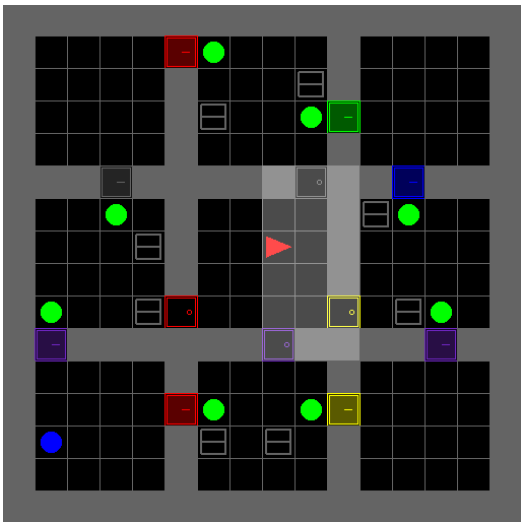


Figure 3: Environments Demonstration

4.2 Gridworld: Minigrid

We have 40 different environments in total, and they might comprise of different tasks and objects. As shown in the table 1, we categorize the environments into 7 categories.

(1) For Empty environment, the agent's goal is to reach the goal square, which provides sparse rewards. For the number of steps to reach the target, a small amount of fines will be subtracted. (2) The DoorKey environment has a key that the agent must pick up to unlock the target, and then reach the target square. (3) This Multiroom environment has a series of connected rooms with doors that must be opened in order to get to the next room. The final room has the goal square the agent must get to. (4) The KeyCorridor environment are multiple registered locked room of increasing size. The key is hidden in another room, and the agent has to explore the environment to find it. The mission string does not give the agent any clues as to where the key is placed. (5) The Unlock series can be considered as a sub-task of many other environments, such as DoorKey, Key Corridor, etc. The goal is simply to get the key and open the door. (6) In ObstructedMaze environment, the agent has to pick up a box which is placed in a corner of a 3x3 maze. The doors are locked, the keys are hidden in boxes and doors are obstructed by balls. (7) In SimpleCrossing, the agent has to reach the goal square on the other corner of the room.

Those environments are extremely difficult to solve using Reinforcement Learning alone. However, by gradually increasing the number of rooms and building a curriculum, the environment can be solved. The random variation of the environment allows the subject to start from a random position in each plot, while the regular variation allows the subject to always start from the corner opposite the target.

Categories	Variations					
Empty	5x5	6x6	8x8	16x16	-	-
DoorKey	5x5	6x6	8x8	16x16	-	-
Multiroom	N2	N4	N6	-	-	-
KeyCorridor	S3R1	S3R2	S3R3	S4R3	S5R3	S6R3
Unlock	Unlock	+Pickup	+Blocked	-	-	-
ObstructedMaze	1Dl	2Dl	1Q	2Q	Full	-
SimpleCrossing	S9N1	S9N2	S9N3	S11N5	-	-

Table 1: Available environments

4.3 Curriculum graph visualization and agent training

We visualized the curriculum graph generated by our method. As it is shown in figure 4 (left), the curriculum graph is a directed disconnected graph without cycle, where not all the tasks are related to each other. For example, we consider environment "ObstructedMaze-2Q" as our target task (node ID 26), and by utilizing topological sort, the curriculum graph allows us to use different task sequence to train the model. One possible output of the topological sort is "28 -> 20 -> 14 -> 26", which gives us "Empty-8x8-v0 -> UnlockPickup -> ObstructedMaze-1D -> ObstructedMaze-2Q".

Figure 4 (right) illustrates that by employing curriculum training, the agent is able to learn and converge faster compared to training directly on the target environment. Here we consider the curriculum sequence mentioned above, and train the model among those task following Svetlik et al. (2017).

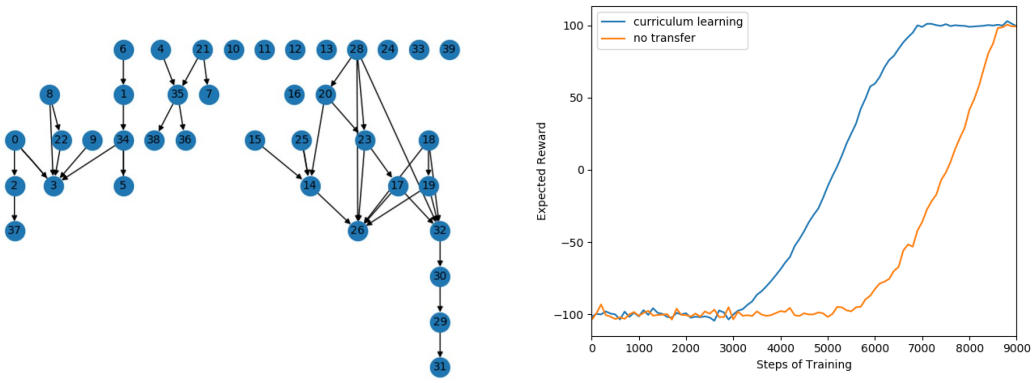


Figure 4: Curriculum Graph and training

5 Conclusion and Future Work

By deploying curriculum learning, our model is able to converge faster and in most cases which higher average obtained reward. In this study, the sets of source tasks are generated automatically by using some heuristic measurements. This way, we no longer require to manually hand-crafting the tasks. For clustering tasks, our experiments show that using k-means is a more accurate way to cluster tasks rather than defining a heuristic threshold.

One limitation of the presented work is that, in general using heuristic feature might not be the ideal metric for measuring the potential transfer value. The reason is that, the real potential transfer value will not be revealed until really trying to transfer from one task to another.

For future work, we would like to propose an optimizing framework on graph, by optimizing the edge value (potential transfer value). By doing this, we might be able to obtain a better curriculum graph.

References

Ammar, H. B., Eaton, E., Taylor, M. E., Mocanu, D. C., Driessens, K., Weiss, G., and Tuyls, K. (2014). An automated measure of mdp similarity for transfer in reinforcement learning. In *Workshops at the Twenty-Eighth AAAI Conference on Artificial Intelligence*.

Asada, M., Noda, S., Tawaratsumida, S., and Hosoda, K. (1996). Purposive behavior acquisition for a real robot by vision-based reinforcement learning. *Machine learning*, 23(2-3):279–303.

Bachman, P., Hjelm, R. D., and Buchwalter, W. (2019). Learning representations by maximizing mutual information across views. In *Advances in Neural Information Processing Systems*, pages 15535–15545.

Chevalier-Boisvert, M., Willems, L., and Pal, S. (2018). Minimalistic gridworld environment for openai gym. `https://github.com/maximecb/gym-minigrid`.

Hagberg, A., Swart, P., and S Chult, D. (2008). Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States).

Lazaric, A., Restelli, M., and Bonarini, A. (2008). Transfer of samples in batch reinforcement learning. In *Proceedings of the 25th international conference on Machine learning*, pages 544–551.

Narvekar, S., Sinapov, J., Leonetti, M., and Stone, P. (2016). Source task creation for curriculum learning. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, pages 566–574.

Ren, Z., Dong, D., Li, H., and Chen, C. (2018). Self-paced prioritized curriculum learning with coverage penalty in deep reinforcement learning. *IEEE transactions on neural networks and learning systems*, 29(6):2216–2226.

Rogers, A., Ramchurn, S., and Jennings, N. R. (2012). Delivering the smart grid: Challenges for autonomous agents and multi-agent systems research.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. (2015). Prioritized experience replay. *arXiv preprint arXiv:1511.05952*.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Silva, F. L. D. and Costa, A. H. R. (2018). Object-oriented curriculum generation for reinforcement learning. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1026–1034.

Sinapov, J., Narvekar, S., Leonetti, M., and Stone, P. (2015). Learning inter-task transferability in the absence of target task samples. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 725–733.

Sukhbaatar, S., Lin, Z., Kostrikov, I., Synnaeve, G., Szlam, A., and Fergus, R. (2017). Intrinsic motivation and automatic curricula via asymmetric self-play. *arXiv preprint arXiv:1703.05407*.

Svetlik, M., Leonetti, M., Sinapov, J., Shah, R., Walker, N., and Stone, P. (2017). Automatic curriculum graph generation for reinforcement learning agents. In *AAAI*, pages 2590–2596.

Thrun, S. (1998). Lifelong learning algorithms. In *Learning to learn*, pages 181–209. Springer.