Task Factorization in Curriculum Learning

Reuth Mirsky [*1 2] **Shahaf S. Shperberg** [*2] **Yulin Zhang** [2] **Zifan Xu** [2] **Yuqian Jiang** [2] **Jiaxun Cui** [2] **Peter Stone** [2 3]

Abstract

A common challenge for learning when applied to a complex "target" task is that learning that task all at once can be too difficult due to inefficient exploration given a sparse reward signal. Curriculum Learning addresses this challenge by sequencing training tasks for a learner to facilitate gradual learning. One of the crucial steps in finding a suitable curriculum learning approach is to understand the dimensions along which the domain can be factorized. In this paper, we identify different types of factorizations common in the literature of curriculum learning for reinforcement learning tasks: factorizations that involve the agent, the environment, or the mission. For each factorization category, we identify the relevant algorithms and techniques that leverage that factorization and present several case studies to showcase how leveraging an appropriate factorization can boost learning using a simple curriculum.

1. Introduction

Curriculum learning (CL) is a research area that deals with the challenge of generating and sequencing training tasks for a learner to facilitate gradual learning (Bengio et al., 2009). The desired outcome of this challenge can be a sequence of **source tasks** for the learner to train on which improve the learner's performance on a **target task** or a distribution of target tasks. The performance of a learner can be evaluated with respect to different objectives, such as the time required to reach a certain average return, or the asymptotic return achieved by the learner after converging to a policy. The set of source tasks selected for the construction of the curriculum is key for the success of the learning process, and the learner should be able to learn from these source tasks if sequenced properly. Within the context of reinforcement learning (RL), Narvekar et al. (2020) recently presented a framework for CL in RL, and used it to survey and classify existing CL methods in terms of their assumptions, capabilities, and goals. In this survey, we introduce an additional perspective that can inform CL practitioners when seeking a CL approach that will suit their problem domains: the **factors** that can be changed in the target task in order to generate and select appropriate source tasks. In particular, the factors which define a task are the **agent**, the **environment**, and the **mission**. We call the process of generating source tasks by altering a factor as **factorization**, thus each of the above factors induces a different factorization type.

By way of example, consider a piano teacher whose goal is to teach a student a new, complex piece. While playing the piece slowly is clearly easier than playing it up to tempo, typically the teacher does not tell the student to start by playing the whole piece slowly. Speed is not the most relevant factor for a curriculum. Instead, the teacher may ask the student to focus on a particularly difficult passage (trimming the task into a sequence of subtasks) or new technique that is needed to play the piece (acquiring an abstract skill). In this case, the source tasks are generated by changing the **mission** that the student needs to accomplish. Alternatively, source tasks can be created by altering the **student's abilities** or the **environment**. An example for the former is having the student train blindfolded to improve muscle memory, while an example for the latter is a piano with colored stickers on the keys, which makes the state of the environment easier for new learners to process.

In this paper, we posit that each possible CL approach leverages one or more factorization types. If a target task is too simple along one of these dimensions, then a factorization along that dimension won't generate source tasks that improve learning. On the other hand, if a curriculum works, then some factorization was leveraged and identifying it explicitly might help researchers provide an even better curriculum. With this conjecture in mind, the main contribution of this paper is a set of factorization types, describing how a task can be factorized to enable the construction of an insightful curriculum. For each factorization type, we provide an overview of the relevant approaches that can be used

[*]Equal contribution [1]Department of Computer Science, Bar Ilan University, Israel [2]Department of Computer Science, The University of Texas at Austin, USA [3]Sony AI, USA. Correspondence to: Reuth Mirsky <mirskyr@cs.biu.ac.il>, Shahaf S. Shpergerg <shperbsh@post.bgu.ac.il>.

Decision Awareness in Reinforcement Learning Workshop at the 39th International Conference on Machine Learning (ICML), Baltimore, Maryland, USA, 2022. Copyright 2022 by the author(s).

to construct a curriculum. In addition, each factorization type is illustrated using test cases from experiments in two CL domains: `ReachNinja` (Ghonasgi et al., 2021) and `GridWorld` (Chevalier-Boisvert et al., 2018).

2. Definitions

A target task can be represented using three **factors**: the **environment** in which the task is executed, the **agent** that acts in the environment, and the **mission** to accomplish.

Definition 2.1 (Task)**.** A task T is a 3-tuple $\langle a, e, m\rangle$ composed of an agent a, an environment e, and a mission m.

Depending on the choice of representation for a task, some of these factors might be coupled such that one of the factors cannot be represented independently of the others. One common example of this coupling is the state space of a Markov Decision Process (MDP), which we formally define and use as an example in the next section, as both the agent and the environment are often represented using the same state space using this model.

Definition 2.2 (Task Factorization)**.** A factorization of a task $T = \langle a, e, m\rangle$ is any modification of one or more of the three task factors a, e, and m that leads to a new task $T^{'} = \langle a^{'}, e^{'}, m^{'}\rangle$.

This paper surveys the different types of such modifications, and categorizes them into one of three factorization types: factors that modify the abilities of the agent (a), the complexity of the environment (e), or the structure of the mission itself (m).

Definition 2.3 (CL Problem)**.** A **CL problem** is a tuple $\langle L, T, O\rangle$ where L is a learner, T is a distribution of target tasks that the learner needs to learn, and O is the objective of the curriculum learning process.

The objective of the CL problem is different from the mission of a specific task. Consider a case in which the learner L needs to learn how to navigate in a specific maze T. The mission of T is reaching the end of the maze, while the objective of the CL problem could be to teach the learner to accomplish that mission optimally, or to reach an adequate solution as quickly as possible, even if this solution is not optimal. Different objectives have been discussed in the CL literature. For example, Taylor & Stone (2009) provide an extrinsic set of metrics to evaluate whether the objective of the curriculum was achieved, such as asymptotic performance or time-to-threshold. Alternatively, Romac et al. (2021b) define the objective of their curriculum selection as being to help learners to be able to generalize to multiple tasks. Given these different objectives, the designer of a curriculum should reason about the effects of different factors of the target task on the learner. For example, if the objective O is to generate a curriculum that can help a learner handle a variety of target tasks, then the source tasks of the curriculum should focus on varying the environment. If O is improving the asymptotic performance on some task, then the source tasks should vary in terms of the tasks' mission.

The output of a CL problem is either a **curriculum**, which is a static sequence of source tasks, or a **curriculum policy**, which can dynamically change the sequence of source tasks given observations of the student and its performance. Identifying an effective curriculum (policy) requires the selection of source tasks, which can be given beforehand with a set of predefined tasks, or could be generated dynamically (Narvekar et al., 2016). For brevity, we will use curriculum in the paper also to describe curriculum policy, unless explicitly mentioned otherwise.

The distribution of target tasks T might be a delta distribution that contain only one task (e.g., a specific instance of minesweeper) or it can represent a (possibly unknown) distribution of potential target tasks (e.g., all possible board setups of minesweeper with a fixed board size and number of mines, *or* varying instances of minesweeper using different board sizes and number of mines).

With these definitions in hand, the aim of this paper is **to identify the modifiable factors of a target task that can be used to generate source tasks for a curriculum for the learner which achieves the objective of the CL problem.**

3. Background and Related Work

CL for RL is a well-established research area, encompassing a variety of challenges and perspectives. To enable a clear discussion about the factors of a task, we use Markov Decision Process (MDP) as a representation that can be used for a task, as often used in RL literature. These definitions are then followed by a short review of papers which defined these CL terms slightly differently than we do in Section 2, or that explore other dimensions of the CL problem which complement the factors presented in this paper.

3.1. Reinforcement Learning

An RL problem is a problem in which an **agent** acts in the **environment** in order to optimize some accumulated reward (Sutton, 2022). A **learner** in this context is the "brain" of (or a separate decision maker for) the agent, and a common way for a learner to model an RL problem is as a **Markov Decision Process** (MDP), which is defined as a tuple $\langle S, A, T, R, s_0, g\rangle$, where S is the space of states of the task, A are the actions that the learner can execute, $T : S \times A \times S \rightarrow [0, 1]$ is a transition function that provides the distribution of states the agent can reach when executing action a in a specific state s, $R : S \times A \times S \rightarrow \mathbb{R}$ is the reward the agent can get from executing action a in a state s and reaching state s', s_0 is an initial state of the MDP,

and g is an optional parameter used in episodic tasks and represents the terminal state (or a set of terminal states). Using this description, we focus on scenarios with a single learning agent, though many of the conclusions may apply to multi-agent learning as well, as discussed in Section 4.

Consider the task of a robot navigating in a room. It is conceptually easy to consider changes that affect only the room (e.g., placing a chair in the middle of the room) or only the agent (e.g., adding another set of wheels). However, it is difficult to distinguish between the robot and the room in an MDP, in which the agent and the environment are strongly coupled. On the other hand, in an MDP the mission of the target task is clear — it is represented using the reward function R. The agent and the environment are both using S, A, T, s_0 and g when it is given, and often they are both represented using the same *instance* of an MDP.

In an attempt to decouple the agent and the environment in an MDP, we refer to the state space S as a powerset of state variables $V = v_1, \ldots, v_k$. Then each state $s \in S$ is an instantiation of these variables, i.e. $s = v_{s_1}, \ldots, v_{s_k}$. We then partition V into subsets of state variables, $V = V_a \cup V_e$, such that V_a corresponds to variables related to the learning agent, whereas V_e corresponds to variables related to the environment. For example, when considering a robot that operates in a physical environment, the position of the robot (e.g., the position of its joints, end effectors, etc.) is encoded as part of V_a, while the layout of the environment is captured by V_e. In essence, V_a is the part of the state that describes the configuration of the agent, thus it changes when the agent is reconfigured (e.g., putting a different robot in the same room), while V_e is changed when the environment changes (e.g., putting the same robot in a different room). Similarly, the action space can be divided into actions of the agents A_a and actions of the environment A_e, such that $A = A_a \cup A_e$. The concept of actions of the environment is not intuitive, as the environment has no agency and cannot act. Yet, sometimes actions are used to encode environmental rules instead of the actual agent's capabilities.

One of the factorization types in Section 4 pertains to the perception of the agent, which is only relevant in the Partially Observable variant of MDPs (POMDPs). A POMDP is defined as a tuple $M = \langle S, A, T, R, \Omega, O \rangle$, in which the first four elements are the same as in an MDP, Ω is the set of observations, and $O : S \times A \times \Omega \to \mathbb{R}$ is a conditional observation probability function. Ω and O capture the perception capabilities of agents.

3.2. Curriculum Learning

While the term "teacher" is often used in related work, we do not explicitly discuss the role of the teacher as part of the CL problem of this paper, as the factorization of the task is independent of the question of *how* the curriculum (or curriculum policy) is learned and provided to the learner. Deliberation about different types of teachers is beyond the scope of this work and can complement the categorization discussed here. Some examples of teachers are: teacher-student (Matiisen et al., 2019; Torrey & Taylor, 2012; Willems et al., 2020), self-paced learning (Jiang et al., 2015; Kumar et al., 2010), or generative (Wang et al., 2019; Fang et al., 2020).

In addition, as mentioned above, a CL problem does not assume any specific underlying representation for the target tasks or the source tasks. Often in CL contexts the tasks are represented as MDPs in which the learner acts with the objective of maximizing its reward (Narvekar et al., 2020). However, some CL tasks might also be solved using classification (Soviany et al., 2021), generation (Graves et al., 2017), or planning (Chrestien et al., 2021; Shah & Srivastava, 2022) instead of optimizing a policy for an MDP, and for a human learner those tasks might be skills that the human is learning (Gentile, 1987; Ghonasgi et al., 2021).

In addition, it is challenging to predict the effect of a change in a task on the difficulty of the problem, as linearly changing one of the task factors may not necessarily result in a linear effect over the difficulty level. Consider an agent that needs to learn to navigate in complex mazes (Florensa et al., 2017b). Removing walls from the environment can make navigation easier, as the agent is free to move directly to its goal without obstacles, but adding walls might also result in an easier task, as the obstacles will restrict the actions available to the agent. Moreover, some curricula include more challenging tasks than the target tasks, when the objective of the CL problem is to improve the learner's asymptotic performance on these target tasks (Romac et al., 2021b).

Lastly, some CL surveys proposed taxonomies and dimensions to consider in CL, which are orthogonal to the dimensions presented in this work: Narvekar et al. (2020) highlighted three sub-problems that comprise the CL problem (task generation, sequencing, and transfer learning). However, that survey does not include an explicit discussion about the dimensions of the target task which can be modified for generating source tasks. Portelas et al. (2020) identified that controlling the contents of a curriculum is a key task when solving a CL problem (Section 4 of that paper splits curriculum types into two: curricula for data collection, and curricula for data exploitation). All the task factorizations discussed in this paper fall under "curriculum for data collection," as they all focus on how a task can be changed, which in turn determines the data collection process. In addition, Portelas et al. (2020) maps different CL techniques to modification of different components of MDPs, but it is different from the factorizations presented in this paper, which are more abstract.

Table 1. The proposed factorization types. For each type of factorization, we identify the common factors that can vary in a curriculum, detail some examples of instances of these factors, and detail the components that will be modified if an MDP is used to describe the environment.

Type	Factors	Examples	MDP
Agent	State space	Morphology, actuators, joints	S (V_a)
	Action space	Freezing joins, tying fingers	A_a
	Dynamics	Rubber glove, action latency, motor backlash	T
	Observations	Camera resolution and position, filters and noised	O
	Initial state	Initial joint configuration	s_0 ($V_{s_{0_a}}$)
Environment	State space	Chess board configuration, obstacles positions	$S(V_e)$
	Action space	No 'castling' in Chess	A_e
	Dynamics	Wet soccer field	T
	Initial state	Initial door configuration (close/open)	s_0 ($V_{s_{0_e}}$)
Mission	Hierarchical Structure	Different skills in soccer (running, kicking, etc.)	S, A, R, s_0, g
	Sequential Structure	Multiroom (Figure 1)	s_0, g
	Objective Structure	Tolerance for reaching a goal, changing the goal state	R

4. Factorization Types

In theory, any task can be used as a source task for curriculum learning. Yet, there are infinitely many possible source tasks, most of which would not help the agent to improve its performance on the target task (or distribution of target tasks), and in many cases transferring the policy learned on a source task would even hurt the agent's performance on the target task (this phenomenon is commonly known as *negative transfer*). For example, if the agent is trying to solve a complex motion control problem, it is very unlikely that the agent would benefit from training on playing chess.

We now analyze different ways of generating sets of source tasks that can potentially be part of a successful curriculum by factorizing the target task. We use the three main categories of task factorization: agent capabilities, environmental complexity, and mission structure. We review each category and identify the properties that might be modified in order to enable a factorization within that category.

We summarize the factorization types that are presented in this section in Table 1, divided into Agent, Environment, and Mission dimensions. We conjecture that in order for a curriculum to help the learner achieve the task objective, it should utilize one or more of the factorization types presented in this table. Even in cases where the curriculum is constructed one source task at a time, as in the case of reactive or adversarial teachers, that choice of task relies on some factorization that helps the teacher choose it. We also note that there are CL problems in which a factorization using these strict categories will not be straightforward — the identified categories are the dimensions in which factorization can occur, but some factorizations might involve more than one category. For example, learning to perform a task represented as an MDP, in which the initial location of the agent depends both on the morphology of the agent (V_a) and the environmental setup (V_e).

4.1. Agent capabilities

Agent capabilities regard to the perception and physical capabilities of the agent. For example, the set of capabilities of a robot that interacts with an environment is induced by its sensors, actuators, and morphology. This section covers different ways of generating source tasks by modifying the capabilities of agents in ways that affect their interaction with the environment.

One common method for generating source tasks is to control the *initial state* from which the interaction between the agent and the environment begins (corresponding to the values of the agent state-variables in an MDP, often denoted $V_{s_{0_a}}$) (Florensa et al., 2017a; Salimans & Chen, 2018; Ivanovic et al., 2019; Chiu et al., 2021). This method can be used to change the initial state of an agent, e.g., by changing the position of its joints. In many problems, especially goal-oriented tasks such as path finding, motion control, etc., the initial state can dramatically affect the problem difficulty if provided with a promising start (Narvekar et al., 2016). In these cases, a diverse set of source tasks can be achieved by simply considering different initial state distributions, which can be used to design curricula for achieving a variety of objectives. A notable example for changing the initial state of the agent was presented by (Florensa et al., 2017a). In that work, the agent's objective is to put a disc on a peg. To achieve that task, the authors relied on the fact that the task becomes easier when the agent is positioned closer to the goal. Thus, they considered a set of source tasks defined by the different initial position of the agent and developed a curriculum in which the agent starts close to the goal and moves further away in each task until reaching the desired starting position as defined in the target task.

Some methods for task generation rely on changing the *transition dynamics*. Domain randomization (DR) (Tobin et al., 2017) is a distinguished approach, which is based on chang-

ing elements in the task, including the transition dynamics. In DR, an agent faces various perturbation of the target task in order to learn robust policies, which in some cases can even be successfully transferred from simulation to the real-world (also known as sim2real transfer). Such perturbations can be based on modifying the capabilities of the agent. For example, some of the perturbations used in (OpenAI et al., 2019) for evaluating the ability of a robotic hand to solve a Rubik's cube rely on changing the agent's capabilities to result in different transition function. These perturbations include controlling of the agent's motor backlash, the noise of actions, and the friction of the agent.

Task generation can also be accomplished by changing the *action space* A. By limiting the available set of actions, or by considering additional actions, it is possible to create a diverse set of tasks with the potential of improving learning. For example, in human motor learning, freezing some degrees of freedom at the initial stages of acquiring new skills was shown to simplify control (Guimarães et al., 2020; Hodges et al., 2005). Similarly, an artificial agent's capabilities can be limited in different ways, e.g., by locking some of the joints, setting limits on joint and tendon ranges, or tying some of the fingers of a robotic arm using a rubber band (OpenAI et al., 2019). This approach, often referred to as Dynamic Difficulty Adjustment (DDA), is also common in tutorials of video games in which the player gradually gains access to new actions in order to avoid the complexity of learning to use all actions at once (Zohaib, 2018).

An aspect unique to agent factorization is the alteration of the agent's observations (changes to Ω in a POMDP). The modifications of observations can be achieved in many ways, such as, increasing/decreasing image resolution, changing camera's position, controlling the frequency of the actuators and/or sensors, and adding random noise. In addition, by changing the observation space the agent can mimic changes in the environment, even when such changes cannot be directly controlled. For example, by applying filters on camera images the agent can simulate night conditions, even if it is limited to training during the day.

Lastly, factorization can be also achieved by changing the *space of agent state variables* (V_a in an MDP). In order to generate a new state space, it is common to define parameters which control aspects of the state variables. For example, in the TeachMyAgent environment (Romac et al., 2021a) the embodiment of agents can be controlled by setting the value of control parameters which affect different properties of the agents, for example, the number of legs, the number of joints, and the leg size. Each instantiation of these control parameters corresponds to a different state variables, which induce different spaces of states that can be used for defining source tasks. These parameters can be controlled either by using explicit heuristics (Narvekar et al., 2016) or by using black-box procedural generation methods (Fang et al., 2020) for adaptively generating and selecting the next task.

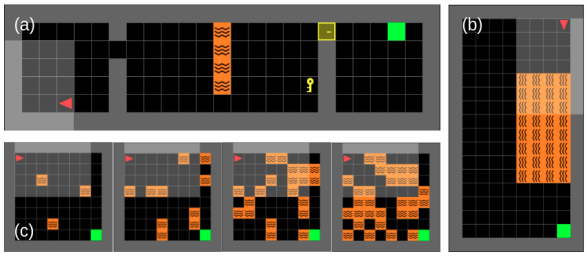

Figure 1. Three examples of tasks in `GridWorld` domain: (a) `MultiRoom`, (b) `Cliff`, and (c) `RandomLava`. In each task, the agent (red triangles) can only see a limited steps ahead (greyed-out squares). To succeed the task, the agent has to navigate to the goal (green tiles) under the challenges of picking up the key, opening the door, avoiding the lava (orange tiles with wave patterns), and crossing the walls (grey tiles).

4.2. Environment Complexity

As opposed to the agent capabilities factorization, which utilizes different ways to modify the agent, the environment complexity factorization is concerned with altering aspects of the environment. Consider, for example, the RoboCup challenge (Kitano et al., 1997), in which teams of autonomous agents compete in the game of soccer. In this example, the environment is the soccer field in which the different team compete. The field can be modified in different ways (e.g., changing the lighting conditions, changing the size of the goal, and coloring the grass) that are agnostic to the morphology and capabilities of the players.

Despite the conceptual difference between the agent and the environment, under some representations, the different factorization can affect the same component. For instance, modifications to the environment can also rely on changing the initial state (e.g., the values of the environment state-variables in an MDP, often denoted $V_{s_{0_e}}$) . A common example can be seen in the game of chess, as chess puzzles, which are based on different initial board configurations, have been designed for the purpose of improving players' skills since the Middle Ages (Murray, 1913).

Similarly, changes to the environment can affect the transition dynamics as well. A prominent example of changing the transition dynamics is AlphaZero (Silver et al., 2017)). In AlphaZero the agent learns to play the games of chess, shogi and Go by competing against itself. In this context, the agent's role as the opponent controls the transitions, i.e., what is the next state of the game reached after making a move. Subsequently, as the agent's policy improves, the transitions will be shifted towards more challenging states. A more straightforward way to control the transitions is by

controlling the conditions of the environment or the conditions of the agents. For example, the dynamic of the ball on a soccer field changes when the field is wet. Moreover, domain randomization can be used to also for perturbating aspects of the environment, e.g., the physics (geometry, friction, gravity) in a simulation (OpenAI et al., 2019).

Finally, control parameters can be used for generating new environments (changing V_e in an MDP). For example (Narvekar et al., 2016) defined different degrees-of-freedom for chess, including the size of the board, number of pieces, etc. Another example is the TeachMyAgent framework (Romac et al., 2021a), a neural network is used for generating environments with different terrains.

4.3. Mission Structure

Complicated task missions often come with inherited structures. One such structure is the modular or compositional structure, where a task can be factorized into multiple primitive behaviors and those primitive behaviors are hierarchically synthesized to solve the original task in a strategic way. These primitives can be an abstractions over the state space S (for example, considering symmetrical states as a single state), over the action space A, (e.g., learning a choreography – a sequence of actions that are often used together). Another common structure is temporal dependency that requires the learner to first execute one sub-task before it has the resources or ability to execute a second sub-task. In this case, the set of source tasks would comprise of sequentially ordered tasks, such that each task t_i offers the learner a different initial state s_0 and a goal state g such that g of t_i is s_0 for the next task in the sequence, t_{i+1}. A third structure that can be leveraged is how a reward, an objective, or a set of objectives is defined. In an MDP, this factorization is implemented as changes to the reward function R for each source task. We detail below on each of these structures and how they might be used.

4.3.1. Hierarchical Mission Structure

Hierarchical factorization breaks a complicated task into subtasks that are organized in a task hierarchy in a way that supports synthesizing the policies or skills for each subtask to solve the original one. A research field that adopts this idea is hierarchical reinforcement learning, which makes abstractions of low-level actions as high-level skills, and solves the problem with those high-level skills (Barto & Mahadevan, 2003). One representation of such skills is called option framework, which is a close-loop policy for taking actions over a period of time. A theory of semi-Markov Decision Process is used to plan with predefined options and improve existing options (Sutton et al., 1999). Similar idea is used in learning motor skills where nonlinear policies are built on top of the canonical systems. A motor primitive framework is proposed to guide a robot to learn a desired complex control policy by transforming from an existing simple canonical control policy (Ijspeert et al., 2002; Kober & Peters, 2009). In a more sophisticated scenario like robot soccer, there are several types of primitive robot behaviors: walk, dribble, and kick. Those behaviors are further strategically assembled into high-level behaviors in the upper layer (MacAlpine & Stone, 2018). Almost all above examples generate subtasks or skills using expert knowledge. Existing techniques also seek to automatically learn the abstract representations directly from low-level environment (Konidaris et al., 2018). Those manually defined or automatically discovered abstractions leverage the hierarchical structure to efficiently solve complicated tasks.

In the context of hierarchical factorization, each skill can be treated as a source task to be used in a curriculum. Moreover, if the skills are complex to master directly, an additional layer of curricula could be constructed for learning each skill. Note that different tasks can be designed for the purpose of learning different skills. For example, learning delicate motion control for object manipulation is significantly different than learning to run. Consequently, the source tasks that correspond to different skills can be different from one another in terms of states, actions and rewards (i.e., S, A, R, s_0, or g in an MDP). Eventually, those skills can be composited in a strategic way to solve the target task.

4.3.2. Sequential Mission

Sequential structure factorization is possible in tasks for which the task can be factorized into sub-tasks, where the sub-tasks are independent from one another with the exception of some partial ordering that indicates that some sub-tasks must be completed before others. For example, consider a Mini-Grid setting (Figure 1 a) in which an agent needs to first find a key, use this key to open a door to another room, and find a goal location in another room. This problem can be factorized into three separate sub-goals: finding the key, unlocking the door, and finding the goal. Each sub-goal corresponds to a source task (or a distribution over source tasks). Note that as opposed to hierarchical structure, the individual components are independent from one another, i.e., it is not necessary be able to solve one sub-goal in order to learn to how to solve another. Furthermore, the objective is not to learn abstract skills which would be useful for solving the target task, but rather to solve different parts of the task and combine the partial solutions to a full solution. In a sequential factorization, tasks differ from one another in their initial state and goal. In the example above, the initial state of the first sub-task is the initial state of the target task, and the goal is to find the key. The second sub-task starts with the key already in possession of the agent and ends when the door is unlocked. Finally, the last sub-task begins when the door is unlocked and ends once

the agent has reaches the goal location.

Factorizing a problem to independent sub-tasks is an old concept in reinforcement learning (Karlsson, 1994), yet a challenging task is to find a problem factorization without needing domain knowledge. Sequential structure is strongly related to the concept of landmarks in planning (Hoffmann et al., 2004). Landmarks are facts that must be true at some point in every valid plan and can be used to decompose the planning task into small chunks. A recent line of work (Eysenbach et al., 2019; Huang et al., 2019; Emmons et al., 2020; Zhang et al., 2021; Savinov et al., 2018; Kim et al., 2021; Liu et al., 2020; Shang et al., 2019) has focused on utilizing planning concepts such as landmarks in order to obtain sub-goals for RL-based agents.

4.3.3. Representation of the Mission

Lastly, the *reward* or the objective(s) of a task can be factorized to assist a learner and aiming its search. A common use-case where such factorization is important is in domains with sparse reward, where reward shaping can be leveraged to guide the learner. For instance, (Fournier et al., 2018) have used a curriculum to control the accuracy requirements in a goal reaching task. Sometimes a task has multiple objectives, even if these objectives are all part of one reward function. Consider a learner that learns to control a vehicle: it needs to learn to drive and reach a goal, while avoiding collisions with pedestrians and other vehicles.

Finally, a case of special interest is when the objective of the agent is to reach a particular *goal* (i.e., the a particular state $g \in S$). In this case, the goal which the agent needs to reach can be altered in order to generate a range of different tasks (Sukhbaatar et al., 2018; Racanière et al., 2019; Pong et al., 2020). By using this method, the agent can focus on training on achievable goals based on its current policy, and as a result to gradually learn to solve the target tasks. In RL, this process of goal selection can be thought of as a form of intrinsic motivation that enables the agent to make progress even if the reward is sparse. One particular form of goal selection is common when the agent need to preform well on a set of target tasks, where each target task is associated with a different goal (goal-conditioned learning), as opposed to a single target task with a fixed goal. In this case, the agent can choose on which target task to train next in order to best improve its progress on learning to preform well on all tasks in the target set (Florensa et al., 2018). Similar to the method of changing the initial state, either of the two factorizations, of the agent or of the environment, can be used as well as their combination.

5. Case Studys

To showcase how different factorizations affect the overall learning process, we use two different domains in which straightforward learning is challenging or even infeasible, and a more elaborate curriculum is needed: the `Gridworld` which is based on the widely used benchmark for RL research by Chevalier-Boisvert et al. (2018), and a more complicated motor control task domain called `ReachNinja` (Ghonasgi et al., 2021). In each of these domains, we use hand-crafted curricula based on different factorizations and compare it to a target task-only learning, to show how each curriculum choice impacts the learning progress of the agent.

5.1. `Gridworld` Domain

As shown in Figure 1, each task in this domain requires an agent to navigate to a goal location (green tiles) by executing a set of discrete actions that change the states of the agent (e.g. moving around) or objects (e.g. pick up keys or open doors). Three types of challenges are presented in this domain: (1) navigate around the blocks (grey tiles) which the agent cannot overlap with; (2) avoid tiles covered with lava (orange tiles with wave patterns) that cause immediate failure of an episode once the agent visit them; (3) pick up the key and open the door that blocks the way to the goal location. A positive reward of $+10$ is assigned for the agent reaching the goal, otherwise, the agent receives an slack reward of -0.005.

This domain benefits from its simplicity incurred by the limited discrete state and action space, yet also maintains the flexibility of creating diverse and challenging tasks using different task factorizations. In this section, we demonstrate how each type of task factorization can be utilized to facilitate the training of corresponding challenging target tasks in the `GridWorld` domain. The agent is trained by PPO (Schulman et al., 2017) algorithm implemented by `Stable-Baseline3` (Raffin et al., 2021). The hyperparameters are kept close to the default, and are only tuned when necessary if the manually designed curriculum failed to solve the target tasks. During training, the policies are evaluated on the *target tasks* for 12 episodes at every 50k time steps to measure the progress on the target tasks. Figure 2 shows for three target tasks the cumulative episodic rewards during the evaluations at different time steps for two training policies: one that leverages one type of task factorization to generate a manually-crafted curriculum (Task factorization) and one that trains directly on the target tasks without any factorization (target task only).

Mission (Sequence) For this task factorization, we propose a `MultiRoom` task where the agent has to execute a long trajectory that starts from a random location in an

Figure 2. Cumulative Episodic rewards evaluated on target tasks with policies trained on curriculum with task factorization (blue) and on target tasks only (orange) in `MultiRoom`, `Cliff` and `RandomLava` respectively. The evaluated episode rewards are averaged by the results of 5 independent runs, and the shaded area shows the 95% confidence interval. The vertical dashed lines mark the time steps when tasks switch in the curriculum for the factorization based training.

initial room and enters three rooms sequentially to the goal location (see Figure 1 (a)). To enter each room, the agent has to acquire one of the three unique skills (crossing the wall, avoiding the lava, and opening the door). Training for this target task will benefit from decomposing the whole trajectory and learning just one additional skill in each new task in a curriculum. Therefore, we create a four-task curriculum with its first, second, third, and fourth tasks randomly initializing the agent in the fourth, third, second, and first rooms respectively (thus changing s_0). The four tasks are trained sequentially for 50k, 500k, 1000k, 1000k time steps respectively. As shown in Figure 2 on the left, training on the factorized source tasks at the beginning deteriorates the performance on the target task, but equips the agent with the ability to solve the future tasks in the sequence. Equipped with all the skills, the agent learns to solve the target task very quickly in just a few thousands time steps. On the other hand, directly training on the target task does not show any improvement in four of the five independent runs. Notice that, by training on the second task that initializes the agent at the third room, the reward on the target task actually drops. The reason is that learning on the second task will encourage the agent to avoid the lava completely which does not contribute to the target task directly, but will facilitate the learning of the next task (across the lava).

Agent Capability To demonstrate the task factorization in terms of the agent capability, we propose a `Cliff` task in which the agent is subject to an action noise p_a that has a fixed probability of p_a to execute a random action at each time step. To create a challenging target task, the agent with an action noise $p_a = 0.6$ is placed in an environment with a cliff represented by a 4×10 grid of lava (see Figure 1 (b)). The exploration becomes difficult in this task, because bad random actions may easily cause catastrophic failures by making the agent into step the lava. To solve this target task, we create a simple two-task curriculum. The agent is first trained with a source task of action noise $p_a = 0$ for 300k time steps to learn to walk along the cliff, then trained on the target task of $p_a = 0.6$ for 700k time steps to perform safely under the action noise. The result is shown in Figure 2 in the middle. By training on the task without action noise, the agent can easily learn to cross the cliff and to preserve this knowledge when switching to high action noise. By contrast, directly training on the target task leads to failures in two of the five independent runs.

Environment Complexity Factorizing a target task by environmental complexity is especially useful when aiming to learn a policy that solves multiple instances of the same domain. To this end, we propose `RandomLava` tasks, in which the agent has to navigate from the left-top corner to the bottom-right corner in an environment of 8×8 grid randomly filled with a total number of n lava tiles (see Figure 1 (c)). The configuration of lavas is completely random so that the agent cannot solve the task by simply memorizing any specific configuration, but rather forced to learn a general policy over all possible lava configurations. The difficulty of the tasks can be rendered by controlling the number of lava tiles n. We use a target task in which $n = 30$. The target task is factorized into a series of source tasks, in which the number of lava tiles n starts from 3 and incremented by 3 every 600k time steps up to $n = 30$. Then, the agent is trained on the target task for additional 2000k time steps. As shown in Figure 2 on the right, while no observable improvement is shown by directly training on the target task, the curriculum that gradually increases the number of lava leads to a steady improvement on the target task.

5.2. Case Study - `ReachNinja` Domain

The usefulness of task factorization is also tested in `ReachNinja`. This domain is inspired by the popular game "Fruit Ninja" (Halfbrick, 2010), in which the agent has to manipulate a red marker by assigning a two-dimensional

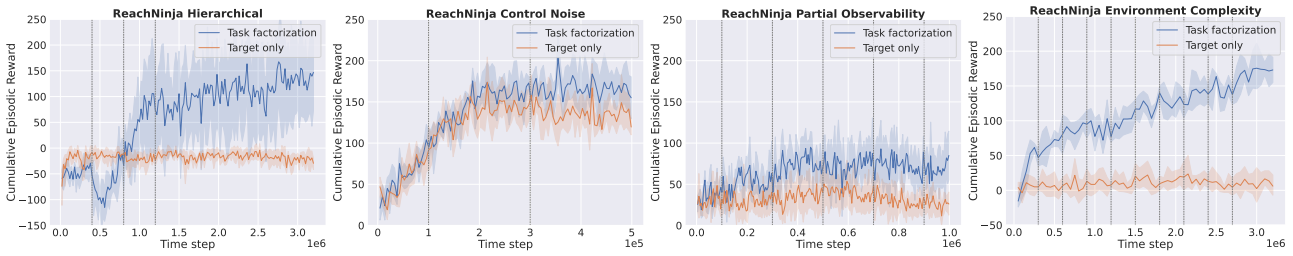

Figure 3. Episode rewards of policies trained on *curriculum with task factorization* (blue) and on *target tasks only* (orange), evaluated on corresponding target tasks. The evaluated episode rewards are averaged by the results of 5 independent random seed, and the shaded area shows the 95% confidence interval. The vertical dashed lines mark the time steps when tasks switch in the curriculum for the task-factorization based training.

continuous vector of acceleration in a screen of 480×640 pixels. At each time step, random blue and black markers will have fixed probabilities to pop out from the bottom of the screen with random velocities and eventually leave the screen due to a constant acceleration of gravity. The mission of the task is to catch as many blue markers as possible while avoiding black markers. Similar to `GridWorld`, we identify one or more dimensions of parameters for each type of task factorization and demonstrate how the tasks generated by modifying the target task along these dimensions can be utilized to build a curriculum that solves the challenging target tasks.

Mission (Hierarchical Structure) In this example of a hierarchical mission structure, the target task includes a maximum of 3 blue markers and 3 black markers at any given time. The penalty of a collision with a black marker is higher than the reward of catching a blue marker. To maximize the performance, the agent has to master two primitive skills: catching blue markers and avoiding black markers. We design source tasks that enable the agent to learn the two skills separately and synthesize their learned policies using Progressive Neural Networks (Rusu et al., 2016). The first source task includes only the black markers and gives extra reward for high velocity, so that the agent learns to avoid black markers while moving fast. The second source task includes only one blue marker, and the third source task includes all three blue markers. After learning each source task, all the network weights are frozen and another column of hidden layers is added. The weights are initialized at random for the first two source tasks to train the primitive skills. For the third task and the target task, the weights of the new columns are initialized to match the last policy, as introduced by (Rusu et al., 2017). This case also demonstrates a factorization which is utilized in an incremental way, where each skill is learned separately and progressively added to the learner's representation.

Agent Capability In the `ReachNinja` domain, we study the task factorization of the agent capability by looking at the Control Noise and Partial Observability. The target task for Control Noise is that the agent's action execution suffers from a zero-mean Gaussian noise with standard deviation of $3 \sim \mathcal{N}(0, 3)$. The learning process suffers from such noise, but the agent has to be aware of the action noise to find a safer trajectory. In light of this, we design a three-stage curriculum so that the learner can learn from the noise-free environment for the first 100k steps, and we switch to two larger control noise levels in the following 400k steps. For Partial Observability, the target task is to have the agent perform well under an 80 % frame-skip rate, without any history augmentation. In this case, we teach the agent to learn from higher observability first and then gradually switch to low observability (frame-skip rate from 0%, to 20%, 40%, 60%, 80%). In both experiments, by first enlarging the agent's capability and then limiting its capability, we are able to learn a better policy under the scenarios where the agent's capability is highly constrained.

Environment Complexity `ReachNinja` has a rich set of parameters to control the game. To demonstrate the usefulness of task factorization in terms of the environment complexity, we chose two continuous parameters: Marker Radius r and Marker Velocity Range v, where r is the radius of blue markers, and the radius of black markers is always set to be $20\text{px} - r$. Marker Velocity Range v sets a range of $[0.5 \times v, v]$ from which the magnitudes of both the blue and black markers are sampled. The challenging target task has a small Marker Radius $r = 5$ and a large Marker Velocity Range $v = 1600$. To solve this target task, the agent is trained with $r = 15\text{px}$ and $v = 400\text{px}/s$ at the beginning and gradually change r and v by -2px and $240\text{px}/s$ at every 300k time steps. In the end, the agent is trained on the target task for another 1000k time steps. As shown in Fig. 3 on the right, the manually designed curriculum with source tasks created from task factorization enables the training of the challenging target task, which cannot be learned directly.

6. Conclusions

We identified factors of a target task that can be modified to generate source tasks for the learner, which can then be used to reach the objective of the CL problem. For each category of factors — agent, environment, and mission — we identify the commonly used algorithms that leverage that factorization. We also presented two domains to showcase how leveraging an appropriate factorization type can boost learning using a simple curriculum. The presented case studies are based on manually designed curricula, and the policies are transferred by copying all the parameters of policy. In the future, better automatic curriculum learning algorithms Portelas et al. (2020) and transfer mechanisms (e.g. shaping the reward and the value function) can be explored accordingly for each type of task factorization.

To conclude this work, we highlight the immediate gains from leveraging this new perspective on target task factorization. First, identifying the factors modified in a curriculum can provide researchers with a deeper understanding of *why* a curriculum works or not, as well as provide alternatives to the chosen source tasks. This understanding can also inform researchers on how to choose the most appropriate learning and teaching algorithms that can utilize the modified factors (as our use of progressive neural networks for hierarchical mission factorization in `ReachNinja`). Lastly, decoupling some factors from the task representation can assist in accomplishing specific objectives. For example, when the objective of the learning is robustness, the modified factor should be the environment of the task.

References

Andrew G Barto and Sridhar Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete event dynamic systems*, 13(1):41–77, 2003.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48, 2009.

Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. Minimalistic gridworld environment for openai gym. https://github.com/maximecb/gym-minigrid, 2018.

Zih-Yun Chiu, Yi-Lin Tuan, Hung-yi Lee, and Li-Chen Fu. Parallelized reverse curriculum generation. *CoRR*, abs/2108.02128, 2021.

Leah Chrestien, Tomas Pevny, Antonin Komenda, and Stefan Edelkamp. Heuristic search planning with deep neural networks using imitation, attention and curriculum learning. *arXiv preprint arXiv:2112.01918*, 2021.

Scott Emmons, Ajay Jain, Michael Laskin, Thanard Kurutach, Pieter Abbeel, and Deepak Pathak. Sparse graphical memory for robust planning. In *NeurIPS*, 2020.

Ben Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. Search on the replay buffer: Bridging planning and reinforcement learning. In *NeurIPS*, pp. 15220–15231, 2019.

Kuan Fang, Yuke Zhu, Silvio Savarese, and Li Fei-Fei. Adaptive procedural task generation for hard-exploration problems. *arXiv preprint arXiv:2007.00350*, 2020.

Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. In *CoRL*, volume 78 of *Proceedings of Machine Learning Research*, pp. 482–495. PMLR, 2017a.

Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. In *Conference on robot learning*, pp. 482–495. PMLR, 2017b.

Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1514–1523. PMLR, 2018.

Pierre Fournier, Olivier Sigaud, Mohamed Chetouani, and Pierre-Yves Oudeyer. Accuracy-based curriculum learning in deep reinforcement learning. *CoRR*, abs/1806.09614, 2018.

Ann M Gentile. Skill acquisition: Action, movement, and the neuromotor processes. *Movement science: Foundations for physical therapy in rehabilitation*, 1987.

Keya Ghonasgi, Reuth Mirsky, Sanmit Narvekar, Bharath Masetty, Adrian M Haith, Peter Stone, and Ashish D Deshpande. Capturing skill state in curriculum learning for human skill acquisition. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 771–776. IEEE, 2021.

Alex Graves, Marc G Bellemare, Jacob Menick, Remi Munos, and Koray Kavukcuoglu. Automated curriculum learning for neural networks. In *international conference on machine learning*, pp. 1311–1320. PMLR, 2017.

Anderson Nascimento Guimarães, Herbert Ugrinowitsch, Juliana Bayeux Dascal, Alessandra Beggiato Porto, and Victor Hugo Alves Okazaki. Freezing degrees of freedom during motor learning: A systematic review. *Motor control*, 24(3):457–471, 2020.

Halfbrick. Fruit ninja, 2010. URL https://www.halfbrick.com/.

Nicola J Hodges, Spencer Hayes, Robert R Horn, and A Mark Williams. Changes in coordination, control and outcome as a result of extended practice on a novel motor skill. *Ergonomics*, 48(11-14):1672–1685, 2005.

Jörg Hoffmann, Julie Porteous, and Laura Sebastia. Ordered landmarks in planning. *J. Artif. Intell. Res.*, 22:215–278, 2004.

Zhiao Huang, Fangchen Liu, and Hao Su. Mapping state space using landmarks for universal goal reaching. In *NeurIPS*, pp. 1940–1950, 2019.

Auke Ijspeert, Jun Nakanishi, and Stefan Schaal. Learning attractor landscapes for learning motor primitives. *Advances in neural information processing systems*, 15, 2002.

Boris Ivanovic, James Harrison, Apoorva Sharma, Mo Chen, and Marco Pavone. Barc: Backward reachability curriculum for robotic reinforcement learning. In *ICRA*, pp. 15–21. IEEE, 2019.

Lu Jiang, Deyu Meng, Qian Zhao, Shiguang Shan, and Alexander G Hauptmann. Self-paced curriculum learning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

Jonas Karlsson. Task decomposition in reinforcement learning. In *Proceedings of the AAAI Spring Symposium on Goal-Driven Learning, Stanford, CA*, 1994.

Junsu Kim, Younggyo Seo, and Jinwoo Shin. Landmark-guided subgoal generation in hierarchical reinforcement learning. *CoRR*, abs/2110.13625, 2021.

Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, and Eiichi Osawa. Robocup: The robot world cup initiative. In *Agents*, pp. 340–347. ACM, 1997.

Jens Kober and Jan Peters. Learning motor primitives for robotics. In *2009 IEEE International Conference on Robotics and Automation*, pp. 2112–2118, 2009. doi: 10.1109/ROBOT.2009.5152577.

George Konidaris, Leslie Pack Kaelbling, and Tomas Lozano-Perez. From skills to symbols: Learning symbolic representations for abstract high-level planning. *Journal of Artificial Intelligence Research*, 61:215–289, 2018.

M Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. *Advances in neural information processing systems*, 23, 2010.

Kara Liu, Thanard Kurutach, Christine Tung, Pieter Abbeel, and Aviv Tamar. Hallucinative topological memory for zero-shot visual planning. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pp. 6259–6270. PMLR, 2020.

Patrick MacAlpine and Peter Stone. Overlapping layered learning. *Artif. Intell.*, 254:21–43, 2018.

Tambet Matiisen, Avital Oliver, Taco Cohen, and John Schulman. Teacher–student curriculum learning. *IEEE transactions on neural networks and learning systems*, 31 (9):3732–3740, 2019.

Harold James Ruthven Murray. *A history of chess*. Clarendon Press, 1913.

Sanmit Narvekar, Jivko Sinapov, Matteo Leonetti, and Peter Stone. Source task creation for curriculum learning. In *Proceedings of the 2016 international conference on autonomous agents & multiagent systems*, pp. 566–574, 2016.

Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E. Taylor, and Peter Stone. Curriculum learning for reinforcement learning domains: A framework and survey. *J. Mach. Learn. Res.*, 21:181:1–181:50, 2020.

OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. Solving rubik's cube with a robot hand. *CoRR*, abs/1910.07113, 2019.

Vitchyr Pong, Murtaza Dalal, Steven Lin, Ashvin Nair, Shikhar Bahl, and Sergey Levine. Skew-fit: State-covering self-supervised reinforcement learning. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pp. 7783–7792. PMLR, 2020.

Rémy Portelas, Cédric Colas, Lilian Weng, Katja Hofmann, and Pierre-Yves Oudeyer. Automatic curriculum learning for deep rl: A short survey. *arXiv preprint arXiv:2003.04664*, 2020.

Sébastien Racanière, Andrew K. Lampinen, Adam Santoro, David P. Reichert, Vlad Firoiu, and Timothy P. Lillicrap. Automated curricula through setter-solver interactions. *CoRR*, abs/1909.12892, 2019.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL http://jmlr.org/papers/v22/20-1364.html.

Clément Romac, Rémy Portelas, Katja Hofmann, and Pierre-Yves Oudeyer. Teachmyagent: a benchmark for automatic curriculum learning in deep RL. In *ICML*, volume 139 of *Proceedings of Machine Learning Research*, pp. 9052–9063. PMLR, 2021a.

Clément Romac, Rémy Portelas, Katja Hofmann, and Pierre-Yves Oudeyer. Teachmyagent: a benchmark for automatic curriculum learning in deep rl. In *International Conference on Machine Learning*, pp. 9052–9063. PMLR, 2021b.

Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv e-prints*, pp. arXiv–1606, 2016.

Andrei A Rusu, Matej Večerík, Thomas Rothörl, Nicolas Heess, Razvan Pascanu, and Raia Hadsell. Sim-to-real robot learning from pixels with progressive nets. In *Conference on Robot Learning*, pp. 262–270. PMLR, 2017.

Tim Salimans and Richard Chen. Learning montezuma's revenge from a single demonstration. *CoRR*, abs/1812.03381, 2018.

Nikolay Savinov, Alexey Dosovitskiy, and Vladlen Koltun. Semi-parametric topological memory for navigation. In *ICLR (Poster)*. OpenReview.net, 2018.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Naman Shah and Siddharth Srivastava. Using deep learning to bootstrap abstractions for hierarchical robot planning. *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2022.

Wenling Shang, Alexander Trott, Stephan Zheng, Caiming Xiong, and Richard Socher. Learning world graphs to accelerate hierarchical reinforcement learning. *CoRR*, abs/1907.00664, 2019.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy P. Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nat.*, 550(7676):354–359, 2017.

Petru Soviany, Radu Tudor Ionescu, Paolo Rota, and Nicu Sebe. Curriculum learning: A survey. *arXiv preprint arXiv:2101.10382*, 2021.

Sainbayar Sukhbaatar, Zeming Lin, Ilya Kostrikov, Gabriel Synnaeve, Arthur Szlam, and Rob Fergus. Intrinsic motivation and automatic curricula via asymmetric self-play. In *ICLR (Poster)*. OpenReview.net, 2018.

Richard S Sutton. The quest for a common model of the intelligent decision maker. *arXiv preprint arXiv:2202.13252*, 2022.

Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

Matthew E Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(7), 2009.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IROS*, pp. 23–30. IEEE, 2017.

Lisa Torrey and Matthew E Taylor. Help an agent out: Student/teacher learning in sequential decision tasks. In *Proceedings of the Adaptive and Learning Agents workshop (at AAMAS-12)*, pp. 41–48, 2012.

Yiru Wang, Weihao Gan, Jie Yang, Wei Wu, and Junjie Yan. Dynamic curriculum learning for imbalanced data classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5017–5026, 2019.

Lucas Willems, Salem Lahlou, and Yoshua Bengio. Mastering rate based curriculum learning. *arXiv preprint arXiv:2008.06456*, 2020.

Lunjun Zhang, Ge Yang, and Bradly C. Stadie. World model as a graph: Learning latent landmarks for planning. In *ICML*, volume 139 of *Proceedings of Machine Learning Research*, pp. 12611–12620. PMLR, 2021.

Mohammad Zohaib. Dynamic difficulty adjustment (dda) in computer games: A review. *Advances in Human-Computer Interaction*, 2018, 2018.