Concurrent Skill Composition using Ensemble of Primitive Skills

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCDS.2022.3177691, IEEE Transactions on Cognitive and Developmental Systems

Concurrent Skill Composition using Ensemble of Primitive Skills

Paresh Dhakan, Kathryn Kasmarik, Philip Vance, Iñaki Rañó and Nazmul Siddique

***Abstract*—One of the key characteristics of an open-ended cumulative learning agent is that it should use the knowledge gained from prior learning to solve future tasks. That characteristic is especially essential in robotics, as learning every perception-action skill from scratch is not only time-consuming but may not always be feasible. In the case of reinforcement learning, this learned knowledge is called a policy. The lifelong learning agent should treat the policies of learned tasks as building blocks to solve those future tasks. One of the categorizations of tasks is based on its composition, ranging from primitive tasks to compound tasks that are either a sequential or concurrent combination of primitive tasks. Thus, the agent needs to be able to combine the policies of the primitive tasks to solve compound tasks, which are then added to its knowledge base. Inspired by modular neural networks, we propose an approach to compose policies for compound tasks that are concurrent combinations of disjoint tasks. Further, we hypothesize that learning in a specialized environment leads to more efficient learning; hence, we create scaffolded environments for the robot to learn primitive skills for our mobile robot-based experiments. We then show how the agent can combine those primitive skills to learn solutions for compound tasks. That reduces the overall training time of multiple skills and creates a versatile agent that can mix and match the skills.**

***Index Terms*—Compositionality, Open-Ended Learning, Curriculum Learning, Lifelong Learning, Self-Generation of Tasks.**

I. Introduction

A key characteristic of an open-ended lifelong learning agent is that it can learn to perform multiple tasks of increasing difficulty. For that, it should be able to determine what tasks it should learn [1] [2], when it should learn them [2], and autonomously learn the skills with minimal external intervention. That has been a focus area of intrinsically motivated open-ended learning [3] research where the tasks to learn are identified using the concept of novelty [4] or curiosity [5], the skills are acquired cumulatively using a hierarchical structure for complex skills [6], and the learning is bootstrapped by measuring the current competency of the agent [7]. In addition, the agent should also be able to exploit the learned knowledge to improve the performance of the task at hand [8]. Since such an autonomous open-ended learning agent does not know in advance what skills it will require to learn, typically, the solution is to learn those skills from scratch. However, in many robotics applications, this is not only time-consuming but also impractical. When an agent is considered a monolithic entity, any change to its sensors or actuators requires relearning skills. Since such monolithic entities tend to have a larger internal representation of the world, the learning is much slower. Thus, for autonomous, open-ended, and continuous learning, the agent must be able to reuse the learned knowledge and do so in a modular fashion and mix and match skills as required.

Reinforcement Learning (RL), where an agent learns by interacting with its environment, is most suitable for open-ended learning. However, in many cases, the amount of interaction needed to learn a task is relatively large [9], since the agent has to find itself in a similar situation to be able to explore other available actions, making it impractical for many robotics applications. To overcome this, sample efficient algorithms such as [10] are continuously being developed. Also, different approaches, such as imitation learning [11] [12], and transfer learning [13] [14], are used. However, further benefits can be gained by reusing the previously learned knowledge to create a solution for future tasks. In RL, a solution to achieve the task is called a policy, which is a mapping from states to action. Two or more policies can then be combined to learn solutions for compound tasks. Existing literature reveals two common ways of combining RL policies: (1) Sequentially [15] [16] – where the policies for the subtasks that may or may not be organized in a hierarchy are invoked in a sequence to solve a more complex task. This technique broadly falls under hierarchical reinforcement learning. (2) Concurrently [8] [17] – where the policies of the subtasks are merged to form a combined policy that is used to solve the complex task. This technique is termed compositionality. Although compositionality exists in literature, its potential as an alternative approach for sample efficient RL has not been adequately investigated.

When tasks are concurrently combined, their solution space can be complementary, contradictory, or disjoint. This article focuses on a concurrent combination of disjoint tasks with some initial comments on the other task categories mentioned in the future work section. Inspired by the modular neural networks [18], we propose an ensemble method of composing RL policies represented by neural networks. Once the tasks are identified either by novelty, technique detailed in [19], or other task generation techniques and the corresponding policies learned, those policies can be combined to form solutions for future tasks. Such reuse of knowledge and its integration with the continuous learning cycle is an essential alternative to sample efficient RL and a logical extension to lifelong learning agent architectures such as those proposed in [20] [21]. That learned knowledge used as a building block is typically learned in the same non-scaffolded, i.e., non-specialized environment as the other complex skills resulting in inefficiently learned primitive skills. Hence, we hypothesize that skill composition is more effective when the constituent primitive skills are learned in a scaffolded environment. The scaffolded environment is a special environment suited to

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCDS.2022.3177691, IEEE Transactions on Cognitive and Developmental Systems

learning a specific primitive skill and can be constructed autonomously as shown in [22]. Our hypothesis is based on the premise that the scaffolded environment provides the agent with a better learning opportunity. Thus, we create scaffolded environments for the robot to learn primitive skills. Using mobile robot-based experiments, we demonstrate how the composition of primitive skills learned in a scaffolded environment can be used as a skill for a more complex task. We compare the results of primitive skills learned in the scaffolded versus non-scaffolded environment to demonstrate learning effectiveness in a scaffolded environment. The primitive skills are then combined to generate the skills for the compound tasks. These composed skills are compared with the skills learned from scratch to demonstrate the effectiveness of the proposed skill combination method.

Thus, this article's contributions are: i) two variants of a method to compose policies for compound tasks that are concurrent combinations of disjoint tasks, ii) a comparison of the performance of skills learned in scaffolded and non-scaffolded environments, and also a comparison of skills learned from scratch with the composed skills, and iii) experimental evaluation of the proposed skill composition method on a mobile robot. The rest of the article is organized as follows: Section 2 reviews the literature and details the related work. In Section 3, we propose how primitive skills can be concurrently combined. Section 4 then describes the setup of our experiments, the scaffolded and the non-scaffolded environments, and details the results of the experiments with the composed skills. Finally, in Section 5, we provide our concluding remarks and highlight potential future directions for this research.

II. Methods of Combining Skills

The RL-specific literature review shows several approaches to reusing previously learned knowledge to create a solution for future tasks. Since we aim to apply this to the lifelong learning agent, we limit this review's scope to approaches where the RL policies for simpler tasks are combined to form a policy for a more complex task. In particular, we consider the sequential and concurrent combination of skills. Consider that a mobile robot has to pick up an object from destination A and deliver it to destination B. Both these tasks have to be carried out one after the other in order. To form a compound skill, the skill to solve the first task is sequentially combined with the skill to solve the second task. That is an example of a sequential combination of skills. Consider that a mobile robot is following a track on the floor and comes across an obstacle. It navigates around the obstacle and again starts to follow the track. That is an example of a modular combination where the robot stops using the first skill when the second skill is triggered. Finally, consider that the mobile robot has to follow a moving target while avoiding obstacles along the way. That is an example of a concurrent combination of skills where both the skills are active simultaneously.

A. Sequential Combination of Policies

In this approach, the policies are combined in sequential order, i.e., the policies are executed one after the other. That is akin to a planning problem where the previously learned policies are sequenced to accomplish a complex task. The tasks may or may not be hierarchically structured; however, the same concept of sequentially combining the policies can be applied to both. Hierarchical RL [23] aims to decompose the task into subtasks, learn the policies for each of the subtasks, and then treat that policy as a macro action. The solution to the complex task is a policy that sequentially invokes these macro actions. An 'option' [24] is a well-defined macro action that is denoted by an initiation set of states I, termination condition β, and the closed-loop policy π. The option is like a subroutine that gets called when the agent is in one of the specific set of states. When invoked, it follows the policy and ends when the termination condition is satisfied. There has been extensive research in this area, ranging from auto-generation of options [25] to integrating this with the core RL algorithm to form algorithms such as option-critic [26]. MAXQ [27] introduces mechanisms for abstraction and sharing for RL to solve tasks that have complex hierarchical structures. The concept exploits the regularities found when a complex task is decomposed. Modular RL is similar to the sequential combination of RL policies in that it switches from one policy to another; however, the decision to sequence the policy is made at a run time based on the initiation trigger or termination state. Modular RL [28][29] decomposes a task, and each module solves a portion of the task. For the final solution, a selector then selects the policy of the subtask.

B. Concurrent Combination of Policies

In this approach, the RL policies are combined concurrently, i.e., all the policies are combined simultaneously to form a single policy that solves all individual policies' tasks. This concept is termed compositionality [17] [30] [31]. The RL policy, which is a mapping that enables the selection of action when in a specific state, can be represented as a Q-table or a neural network. When the policies are represented as a Q table, the compound task's Q function is generated by averaging the constituent Q functions. When the policies are represented as a neural network, the literature review shows that the combined policy is generated using voting, a Mixture of Experts [32] [33] [34], policy distillation [35] [36], and action selection using a Gaussian Mixture Model [37]. Using a robotic manipulator arm, Haarnoja et al. [38] demonstrate that a policy to move an object along a vertical axis and a policy to move the object along a horizontal axis can be combined to form a policy for the robot to be able to move the object to the intersection position of the two axes. Todorov [39] [34] developed a theory of compositionality applicable to a general class of stochastic optimal control problems. Simpkins et al. [40] propose a composable modular RL that combines the concept of compositionality with modular RL. In their case, the final policy is generated by combining multiple policies concurrently. Niekerk et al. [8] apply the concept of compositionality to the lifelong learning agent. Using a high-dimension video game use case, they demonstrate how an agent can combine skills from its library of already learned skills to solve a new task. Niekerk et al. conclude that learning

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCDS.2022.3177691, IEEE Transactions on Cognitive and Developmental Systems

a multimodal policy for a composite task can be difficult because of the tendency to collapse into a single mode without exploring the alternatives. Hence, it is better to learn the unimodal policies first and then combine them to produce optimal multimodal solutions.

Unlike the case of the sequential combination of policies where the technique of 'options' is commonly used, there is no common method for the concurrent combination of skills. Also, 'options' can be integrated with multi-skills/lifelong learning frameworks [41][20][21]. However, the methods found in the literature for the concurrent combination are either not task-independent or not abstract enough to be easily integrated with such frameworks. The method proposed in this article offers the simplicity of combining Q-table based policies while maintaining the scalability provided by the neural network based policies. Also, since it does not require any specific neural network architecture, it can be easily integrated with the frameworks mentioned above.

III. Concurrent Combination of Skills Represented by Neural Networks

The standard practice in RL is to train an agent from scratch for each new task. That is time-consuming and may not always be possible since the agent may need to be reset to an initial state, or the agent has to find itself in the same situation to try different actions. RL's very nature relies on several similar learning opportunities so that agents can explore other available actions. That is a significant challenge in RL. In the case of a lifelong learning agent, it continuously learns and enhances its knowledge base with additional skills. So, a logical next enhancement for lifelong RL agents is to combine the learned primitive skills to solve future tasks. At the very least, such a combined policy can be used as a starting policy by the agent and then further refined.

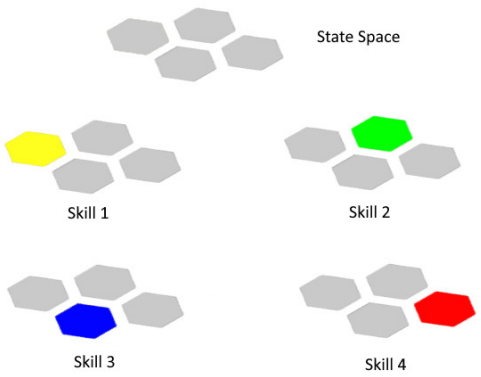


Fig. 1. The agent's state space is represented by the collection of four grey-colored hexagons, and each of the four skills for the four disjoint tasks is represented using a different colored hexagon.

A compound task can be considered as a sequential or concurrent combination of the constituent primitive tasks. Further, a concurrent combination can be an 'AND' or an 'OR' combination. In an 'AND' combination, for successful execution, all the constituent tasks are executed simultaneously. Whereas, in an 'OR' combination, the execution is considered successful if one of the constituent tasks is executed successfully. That is, the combined policy can solve either of the constituent tasks, but not all simultaneously.

In this article, we limit the scope to a concurrent combination of tasks using the 'AND' combination of disjoint tasks. When learning multiple tasks, the two tasks can be said to be 'competing' if the actions required to accomplish one task are opposite to the actions necessary to accomplish another task. The tasks are said to be 'complementary' if the actions required to accomplish one task are the same as the actions necessary to accomplish another task and 'disjoint' if they are neither competing nor complementary. The skill (i.e., the learned RL policy for a task) for the disjoint tasks can be represented as shown in Fig. 1. Like any other multi-task learning method, the limitation of the proposed approach is that the tasks should not be contradictory. Since, for such contradictory tasks, the actions may be competing in nature.

For example, consider a set of disjoint primitive tasks for a vacuum cleaning robot: i) detect the dirt, ii) clean the dirt, iii) avoid obstacles, and iv) detect an edge on the floor to keep the robot from falling off the stairs. A compound task with an 'AND' combination would be the combination of all the tasks, i.e., the robot detects the dirt, cleans the floor, avoids obstacles, and avoids falling off the stairs.

Consider that a lifelong RL agent must learn several tasks, both primitive and compound ones. Consider the primitive tasks $p_1, p_2, \cdots, p_n$ and a compound task C which is a combination of those primitive tasks. This compound task can be represented in terms of primitive tasks as $C = p_1 \wedge p_2 \wedge \cdots \wedge p_n$ where $\wedge$ represents an 'AND' combination that means to carry out all the constituent tasks simultaneously. Further, consider that π_1 is the policy for the task p_1, π_2 is the policy for the task p_2 and so on, and the policy for the compound task is π_C. Since the compound task is a combination of the constituent tasks, the policy π_C can be used as a starting policy for the compound task and refined as and when the agent gets an opportunity to learn more regarding a particular region of its state space.

We assume that the reward for each of the primitive tasks is within the same range and $r_1(s, a, s')$ is the reward for the task p_1, $r_2(s, a, s')$ is the reward for the task p_2 and so on, where s, a and s' are the agent's current state, the agent's action in the current state and the resulting state of the agent when that action is taken. Thus, the reward for the compound task can be represented as a summation of the rewards [42], as $r_C(s, a, s') = 1/n \sum_{i=1}^{n} r_i(s, a, s')$. The policy for task C can be represented as a sum of all constituent policies [17], as $\pi_C = 1/n \sum_{i=1}^{n} \pi_i$. When the policies are represented as a Q table, the Q function for the compound task can be generated by averaging the constituent Q functions [38], as $Q_C(s, a) = 1/n \sum_{i=1}^{n} Q_i(s, a)$.

Another way to represent an RL policy is by using neural networks, such as RL's DQN [43] and A2C [10] algorithms. In the actor-critic RL algorithm, the actor is implemented using a neural network. This actor, represented by $\pi(s, a, \theta)$ outputs

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCDS.2022.3177691, IEEE Transactions on Cognitive and Developmental Systems

an action a_t in state s_t, receives reward $r_{(t+1)}$ and lands in a new state $s_{(t+1)}$. Based on the reward $r_{(t+1)}$, the critic, which is also implemented as a neural network, represented by $q(s, a, \omega)$ calculates the value of how good it was to take that action, adjusts the weights ω of the critic network. This is then provided as feedback to the actor, which updates the weights θ of the actor-network.

We propose an ensemble technique similar to the average model weight ensemble [44] [45]. While the technique proposed in this article will work for single neural network based algorithms such as DQN, we use an actor-critic algorithm to show that the proposed ensemble technique differs from the existing techniques of averaging the Q values. We propose that a compound task's RL policy can be formed by averaging the learnable parameters of the constituent task's policies. Consider that the actor and critic networks for the task p_1 are $actor_1$, $critic_1$ respectively and the actor and critic networks for the task p_2 are $actor_2$, $critic_2$ respectively and so on. To create the actor and critic networks for the compound task C, we average the learnable parameters of the constituent actor networks and the constituent critic networks, as shown in Eq (1) and Eq (2).

$$actor_C = 1/n \sum_{i=1}^{n} actor_i \tag{1}$$

and

$$critic_C = 1/n \sum_{i=1}^{n} critic_i \tag{2}$$

The $actor_C$ and $critic_C$ networks can then be used to construct an actor-critic RL agent for the compound task C. That construction of the RL agent for the compound task can be done in one of the following two ways:

Method #1: For disjoint tasks, in effect, the states where the actions matter for each of the skills do not overlap. Hence, when such tasks are represented using the same state vector (representation #1), a compound skill can be constructed by effectively stacking all the constituent RL policies. A graphical representation of this is shown in Fig. 2. It shows four skills, with its state space represented by a colored and a grey area and the composed skill at the bottom of the diagram. In the diagram, the neural network based RL policy for the skill is a combination of actor and critic neural networks. Both networks take the agent's state vector as the input. The actor network's output is the probability for each of the actions in the agent's action space, and the output of the critic network is the feedback to the actor indicating how good it was to take a particular action in that state.

Method #2: When skills are represented using a task-specific state vector (representation #2), the combined skill can be generated by aligning the constituent state spaces and using the AND combination of the constituent skills. Fig. 3 shows a graphical representation of this. Same as in Method #1, the neural network based RL policy for the skill shown is a combination of actor and critic neural networks. However, the state-space of each task (shown using different colored pieces) depends on the skill being learned. This mechanism is particularly beneficial in multi-agent RL, where each agent is responsible for learning a particular skill.

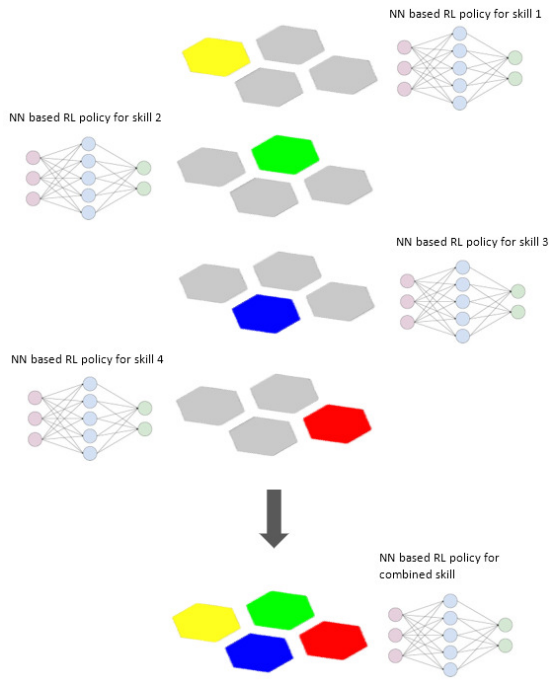


Fig. 2. Skill composition method #1 – same state vector for all tasks (representation #1)

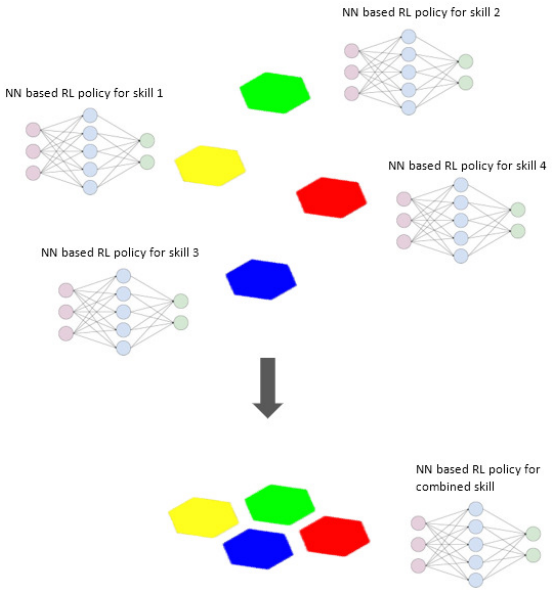


Fig. 3. Skill composition method #2 – task-specific state vector (representation #2)

While each composition method has its own merits and demerits, both lead to reduced compound skills training time. That is not just because the agent is not required to learn the skill from scratch but also because different agents can learn the constituent policies in parallel. Algorithm 1 details the pseudo-code for skill composition. The constituent networks' learnable parameters are averaged to construct a new combined actor-critic RL agent to solve the compound task.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCDS.2022.3177691, IEEE Transactions on Cognitive and Developmental Systems

Algorithm 1 Skill Composition

/*__Assumption__*/
Input: We have an array of learned policies for the primitive skills in $policy[]$

/*__Initialize variables__*/
Initialize $combinedActorParams$, $combinedCriticParams$

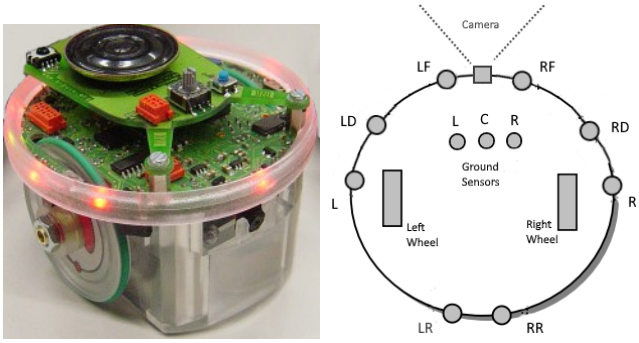
/*__Ensemble of actor and critic networks__*/
for $i = 1$ **to** $numberSkills$ **do**
$actor\_i\_params = getLearnableParameters(actor\_i\_policy)$
$combinedActorParams = combinedActorParams + actor\_i\_params$
$critic\_i\_params = getLearnableParameters(critic\_i\_policy)$
$combinedCriticParams = combinedCriticParams + critic\_i\_params$
end for

/*__Calculate average of learnable parameters__*/
$combinedActorParams = combinedActorParams/numberSkills$
$combinedCriticParams = combinedCriticParams/numberSkills$

/*__Create combined agent__*/
$combinedAgent = Create\_RL\_Agent(combinedActorParams, combinedCriticParams)$

IV. Experiments, Results, and Discussion

For our experiments, we use Webots to simulate an e-puck mobile robot. We use MATLAB RL Toolbox's A2C algorithm to implement the RL agent as a control program for the e-puck. Typically, experiments in the area of developmental robotics use the iCub robot [7]. The usage of mobile robots for such experiments is still a novel application. Further, as detailed in the following subsection, we create scaffolded environments for the e-puck to learn primitive skills. Those environments are designed to provide maximum opportunity for the robot to learn those primitive skills. Also, we create non-scaffolded environments to test and compare the skills for compound tasks learned from scratch with the skill generated by combining previously learned primitive skills.



(a) (b)

Fig. 4. (a) e-puck mobile robot, (b) A plan view of e-puck with labeled state attributes

A. Setup of the Experiments

Mobile Robot: An e-puck, shown in Fig. 4(a), is a small differential wheeled mobile robot. It has eight infrared distance sensors with a range of 6 centimeters that we label in a clockwise direction as *Right-Front, Right-Diagonal, Right, Right-Back, Left-Back, Left, Left-Diagonal, Left-Front*. It has three ground sensors that we label as $Left, Center, Right$. E-puck also has a 52x39 pixels resolution color camera. Thus, the mobile robot state in our experiments is a vector represented by $[p^{FR}, p^{RD}, p^{R}, p^{RR}, p^{RL}, p^{L}, p^{LD}, p^{FL}, g^{L}, g^{C}, g^{R}, c]$, as shown in the labeled e-puck diagram Fig. 4(b). The action space of the RL agent (detailed below) consists of the following: turn left, step forward and turn right.

In Webots, the proximity sensor value ranges from 0 to 2000, with a high value indicating that an object is nearby. The ground sensor value ranges from 0 to 1000, with a high reading indicating that the sensor is detecting dark area / black color on the ground. The proximity sensor values, ground sensor values, and camera output were discretized to binary values in the experiments. For the proximity sensor, if there was no object in the vicinity, the sensor reading was considered 0, and when there was an object in the vicinity, it was considered 1. The proximity sensor's binary value was considered 1 if the actual sensor reading was greater than 500, otherwise considered 0. For ground sensors, the value 1 indicated that the sensor is detecting black color, and 0 indicated that it is detecting white. The ground sensor's binary value was considered 1 if the actual sensor value was greater than 300, otherwise considered 0. Regarding the camera identifying the randomly moving blue colored robot, we use Webots API for the recognition. API returns the number of blue-colored objects recognized in the frame. When the blue-colored robot was in view and recognized, the camera output was considered 1 and 0 otherwise.

Arenas: Several arenas were created in Webots simulation

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCDS.2022.3177691, IEEE Transactions on Cognitive and Developmental Systems

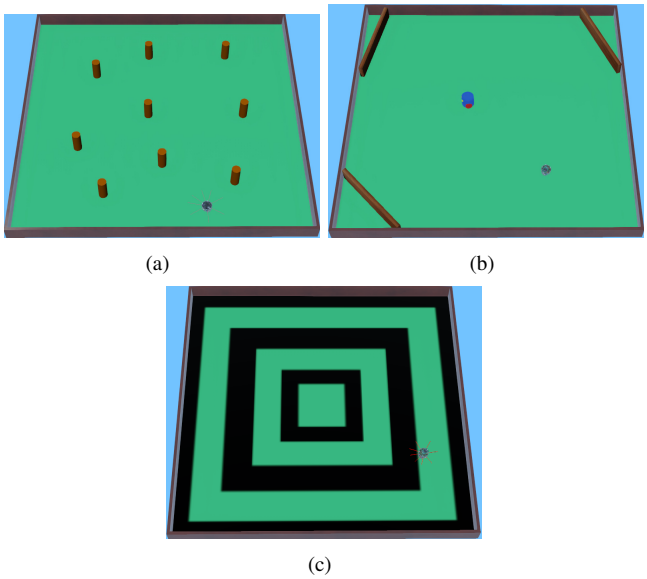


Fig. 5. (a) Training arena with obstacles (Training_Arena_1), (b) Training arena with randomly moving blue robot (Training_Arena_2), (c) Training arena with a colored pattern on the floor (Training_Arena_3).

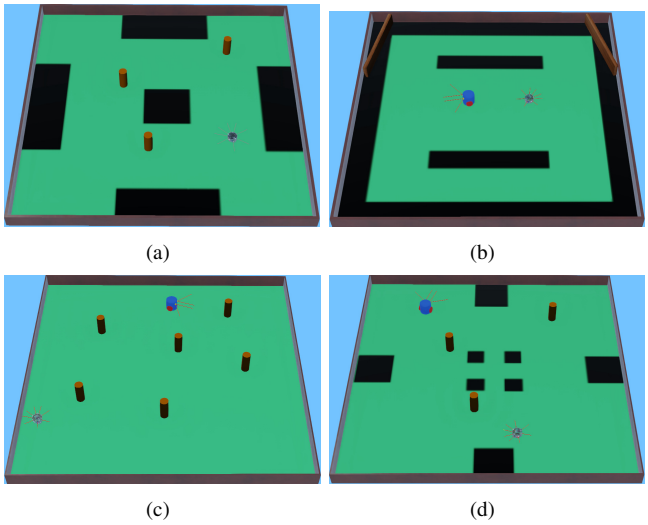


Fig. 6. (a) Test arena with black regions on the floor and obstacles (Test_Arena_1). (b) Test arena with black regions on the floor and randomly moving blue robot (Test_Arena_2), (c) Test arena with obstacles and randomly moving blue robot (Test_Arena_3), (d) Walled arena with black regions on the floor, randomly moving blue robot, and scattered obstacles (Test_Arena_4).

software. Fig. 5(a), 5(b), and 5(c) are the scaffolded training arenas used to train the e-puck to learn the specific primitive skill. Fig. 5(a) is a 2m x 2m walled arena with scattered obstacles. In this arena, e-puck can learn primitive skill such as avoiding obstacles/walls. Fig. 5(b) is a 2m x 2m walled arena with a blue-colored robot. This blue robot moves in a straight line, and when it detects an obstacle, it changes its direction and continues moving in a straight line. In this arena, the e-puck can learn primitive skills such as following the blue robot. Fig. 5(c) is a 2m x 2m arena in which the robot can learn primitive skills such as following a track on the floor or avoiding certain regions on the floor. Such a specialized set of environments allows us to investigate the performance of our algorithm under controlled conditions.

Fig. 6(a), 6(b), 6(c), and 6(d) show the 2m x 2m test/non-scaffolded arenas. Fig. 6(a) is a walled arena with several black regions on the floor and a few scattered obstacles. Fig. 6(b) is a walled arena with black regions on the floor and has a randomly moving blue robot. Fig. 6(c) is a walled arena with a few scattered objects and a randomly moving blue robot. Fig. 6(d) shows a walled arena with black regions on the floor, a randomly moving blue robot, and a few scattered obstacles. It is an arena that contains all the aspects that we want to demonstrate with our experiments.

RL algorithm: RL consists of algorithms that are either value-based methods where the agent learns the value function that determines how good it is to take a particular action in a specific state and policy-based methods where the agent directly optimizes the policy by sampling several rollouts of the episode. The actor-critic family of algorithms is a model-free and on-policy algorithm. They are a hybrid approach where the critic is trained to estimate the value function and provide feedback to the actor to optimize the policy. The A2C uses the 'Advantage' instead of the 'Value' function, which leads to learning stability. In our experiments, the learning rate parameter was set to 0.01. The EntropyLossWeight, the parameter that promotes exploration, was set to 0.03. The actor and the critic networks were created using the same architecture. The networks consisted of an input layer (with the number of nodes depending on the state vector), a fully connected layer, and a *'leaky RELU'* layer followed by a fully connected layer.

Primitive and Compound Tasks: For the experiments, we manually select the primitive tasks listed in Table 1. The compound tasks are then formed by combining those primitive tasks, as shown in Table 1. The primitive tasks are the elemental tasks, whereas the compound tasks show the constituent tasks' composition. The column 'Task Id' shows the notation used to represent the task. Prefix 'p' is used to represent a primitive task, and 'C' is used to represent a compound task. Column 'Task Composition' details the compound task composition. The 'Task Description' column describes the task. 'Arena where Trained' column details the arena in which the training of the task took place.

The compound task C_1 is an 'AND' composition of p_1 (avoid obstacles) and p_3 (avoid black region). The compound task C_2 is an 'AND' composition of p_2 (follow blue robot) and p_3 (avoid black region). The compound task C_3 is an 'AND' composition of p_1 (avoid obstacles) and p_4 (follow blue robot). The compound task C_4 is an 'AND' composition of p_1, p_2, and p_3. Shown below are the state vectors for the primitive and the compound tasks for the two ways of skill composition proposed in this article.

B. Results

Results for Primitive Tasks Table 2 shows the results of the training for primitive tasks. The training consisted of running 50 episodes of 20,000 steps each. The tasks are considered

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCDS.2022.3177691, IEEE Transactions on Cognitive and Developmental Systems

TABLE I
HANDCRAFTED PRIMITIVE AND COMPOUND TASKS

Task Id	Task Composition	Task Description	Arena where Trained
p_1	N/A	Avoid obstacles	Scaffolded environment – Fig. 5(a). Non-scaffolded environment – Fig. 6(d).
p_2	N/A	Follow the randomly moving blue robot	Scaffolded environment – Fig. 5(b). Non-scaffolded environment – Fig. 6(d).
p_3	N/A	Avoid the black regions on the floor	Scaffolded environment – Fig. 5(c). Non-scaffolded environment – Fig. 6(d).
C_1	p_1 AND p_3	Avoid obstacles AND avoid the black regions on the floor.	Fig 6(a).
C_2	p_2 AND p_3	Follow the blue robot AND avoid the black regions on the floor.	Fig. 6(b).
C_3	p_1 AND p_2	Avoid obstacles AND follow the randomly moving blue robot.	Fig. 6(c).
C_4	p_1 AND p_2 AND p_3	Avoid obstacles AND follow the blue robot AND avoid the black regions on the floor.	Fig. 6(d).

State Vector for p_1, p_2, and p_3:
[p^{FR} p^{RD} p^{R} p^{RR} p^{RL} p^{L} p^{LD} p^{FL} g^{L} g^{C} g^{R} c]

State Vector for C_1, C_2, C_3, and C_4:
[p^{FR} p^{RD} p^{R} p^{RR} p^{RL} p^{L} p^{LD} p^{FL} g^{L} g^{C} g^{R} c]

Fig. 7. State vector details for skill composition method #1 (skills represented using representation #1, i.e. same state vector for all the tasks).

State Vector for p_1:
[p^{FR} p^{RD} p^{R} p^{RR} p^{RL} p^{L} p^{LD} p^{FL}]

State Vector for p_2:
[c]

State Vector for p_3:
[g^{L} g^{C} g^{R}]

State Vector for C_1= (p_1+ p_3):
[p^{FR} p^{RD} p^{R} p^{RR} p^{RL} p^{L} p^{LD} p^{FL} g^{L} g^{C} g^{R}]

State Vector for C_2= (p_2+ p_3):
[g^{L} g^{C} g^{R} c]

State Vector for C_3= (p_1+ p_2):
[p^{FR} p^{RD} p^{R} p^{RR} p^{RL} p^{L} p^{LD} p^{FL} c]

State Vector for C_4= (p_1+ p_2+ p_3):
[p^{FR} p^{RD} p^{R} p^{RR} p^{RL} p^{L} p^{LD} p^{FL} g^{L} g^{C} g^{R} c]

Fig. 8. State vector details for skill composition method #2 (skills represented using representation #2, i.e. task-specific state vector).

'maintenance' tasks [46], i.e., the aim of the agent is to maintain the task state. Hence, unlike in the 'achievement' task type, the episode does not end once the agent reaches the desired state. So the term 'episode' merely means a collection of 20,000 steps. The task is learned in a scaffolded and non-scaffolded environment with the same vector for all the tasks (representation #1). Also scaffolded and non-scaffolded with task-specific state vector (representation #2). 'Average Reward per Episode' was used to measure agent performance. This is the average cumulative reward for an episode of 20,000 steps. The experiment for every task was run 10 times with different start positions of the e-puck mobile robot, and the standard deviation was calculated. The figure number shown in the square bracket is the training arena where the training for the primitive task took place. The table also shows results from a statistical comparison of the training in scaffolded versus non-scaffolded environments for state vector representation #1 and scaffolded versus non-scaffolded environments for state vector representation #2. Since the results data is not normally distributed, we ran a non-parametric method to compare the data points. The status quo or the Null hypothesis H_0 is that the cumulative reward received by the agent training in the non-scaffolded environment will be greater than or equal to that received by the agent training in the scaffolded environment. That is to say, a scaffolded environment does not lead to better training. We ran Mann-Whitney U-Test on the 50 data points (an average of 10 runs for each episode) to determine if this hypothesis should be rejected or not. The alpha value for this statistical test was 0.05. The Null H_0 and the alternative H_1 hypothesis can be represented as:

H_0: average cumulative reward in the non-scaffolded environment $\geq$ average cumulative reward in the scaffolded environment

H_1: average cumulative reward in the non-scaffolded environment $<$ average cumulative reward in the scaffolded environment

Table 2 shows a statistically significant difference in the agent's performance for the scaffolded versus non-scaffolded environment for all the tasks. Results in bold show the best performance for the primitive task. The Mann-Whitney U-Test results show that for tasks p_1, p_2, and p_3 tasks represented using state vector representation #1, data support the alternative

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCDS.2022.3177691, IEEE Transactions on Cognitive and Developmental Systems

TABLE II
RESULTS OF THE LEARNING PHASE FOR THE PRIMITIVE TASKS. AVERAGE REWARD PER EPISODE MEASURED FOR TEN TRIALS IN THE SCAFFOLDED AND THE NON-SCAFFOLDED ENVIRONMENT AND COMPARED USING MANN-WHITNEY U-TEST.

Task Id	**Average reward per episode for training in scaffolded environment and state vector representation #1. [Training Arena]**	**Average reward per episode for training in non-scaffolded environment and state vector representation #1. [Training Arena]**	**Mann-Whitney U-Test of the results shown in the previous two columns. Reject H_0 hypothesis?**	**Average reward per episode for training in scaffolded environment and state vector representation #2. [Training Arena]**	**Average reward per episode for training in non-scaffolded environment and state vector representation #2. [Training Arena]**	**Mann-Whitney U-Test of the results shown in the previous two columns. Reject H_0 hypothesis?**
p_1	14816 ±3450 [Fig. 5(a)]	11077 ±1594 [Fig. 6(d)]	p-value = 1.00, Reject H_0	**15976 ±471** [Fig. 5(a)]	15880 ±566 [Fig. 6(d)]	p-value = 0.79, Reject H_0
p_2	**13117 ±2445** [Fig. 5(b)]	-1369 ±4366 [Fig. 6(d)]	p-value = 1.00, Reject H_0	11796 ±2346 [Fig. 5(b)]	8255 ±688 [Fig. 6(d)]	p-value = 1.00, Reject H_0
p_3	11799 ±5337 [Fig. 5(c)]	11077 ±1594 [Fig. 6(d)]	p-value = 0.97, Reject H_0	13206 ±4764 [Fig. 5(c)]	**19885 ±51** [Fig. 6(d)]	p-value = 0.00, Fail to Reject H_0

hypothesis H_1 that training in a scaffolded environment leads to better learning. That is because the scaffolded environment minimizes triggering the non-skill specific sensors and allows the agent to focus on learning just one skill.

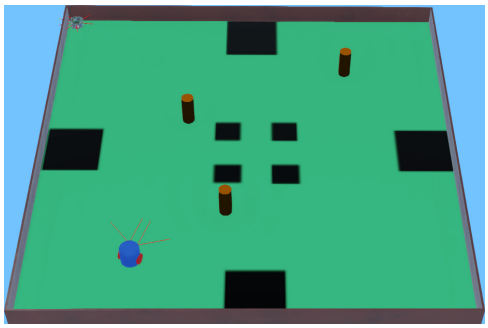


Fig. 9. Reward hacked for task p3. E-puck seen at the top left corner of the arena has stumbled upon a situation where it keeps pushing itself against the wall to gain a positive reward.

For tasks represented using state vector representation #2, Mann-Whitney U-Test results show that the data for p_1 and p_2 also supports the alternative hypothesis H_1, i.e., training in a scaffolded environment leads to better learning. Task p_3, where the agent performance in the non-scaffolded environment seems better, is an anomaly. Upon closer examination, it was seen that it was a case where the agent had come up with an unexpected way of gaining the reward. As shown in Fig. 9, the agent found a way to push itself against a wall while on the floor's non-black region. In this case, both wheels keep turning to move forward (and it is on the non-black region, i.e., task p_3); albeit, the wall does not allow forward motion. Once the agent stumbles upon such a situation, it exploits it by remaining in that situation. Even though the agent accumulates a high reward, it does not learn the skill to avoid black regions on the floor. That cannot happen in a scaffolded environment, and for this reason, we do not use the skills learned in the non-scaffolded environment for skill composition in the next set of experiments.

In the scaffolded environment, the agent gets more opportunities to learn the skills. Although the same is true for the state vector representation #2, the advantage is not as pronounced. While the environment can trigger non-skill specific sensors on the e-puck, those sensors are not part of the agent's state vector and do not interfere with the agent's learning of the skill. For the compound task experiments in the following subsection, only the state vector representation #1 is used.

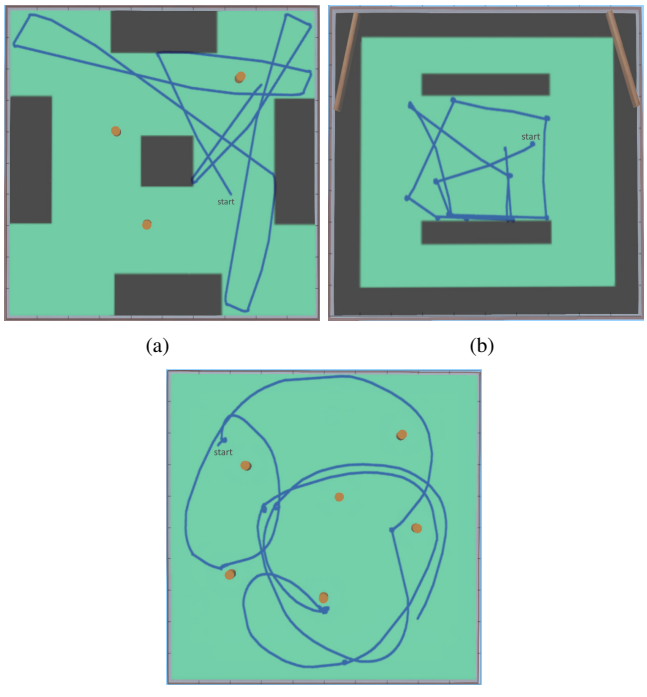
Results for Compound Tasks Firstly, to check the validity of the composed skills, the performance of the combined policies was first tested in the test arenas. Fig. 9(a), 9(b), and 9(c) show the trajectory of the e-puck robot executing the combined policies for tasks C_1, C_2, and C_3, respectively. For this qualitative visual validation, the primitive skills learned in a scaffolded environment were combined using method #1.

The combined policy for C_1 shows the behavior of avoiding obstacles as well as the black region. Fig. 10(a) shows the top view of the test arena with black regions on the floor and obstacles with the trajectory of the e-puck shown in navy color, starting from the 'start' position. It shows that the e-puck is avoiding obstacles as well as black regions on the floor. The combined policy C_2 shows the behavior of the e-puck following the blue robot and avoiding the black region. Fig. 10(b) shows the top view of the test arena with randomly moving blue robot and black regions on the floor with the trajectory of the e-puck shown in navy color starting from the position marked 'start'. It shows that the e-puck is avoiding the black region and following the blue robot at the same time. The combined policy for C_3 shows the behavior of the e-puck following the blue robot while avoiding obstacles. Fig. 10(c) shows the top view of the test arena with obstacles and a randomly moving blue robot with the trajectory of the e-puck shown in navy color starting from the position marked 'start'. It shows that the e-puck is following the blue robot

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCDS.2022.3177691, IEEE Transactions on Cognitive and Developmental Systems

TABLE III
RESULTS FOR COMPOUND TASKS. AVERAGE REWARD PER EPISODE MEASURED FOR TEN TRIALS WITH STANDARD DEVIATION SHOWN. MANN-WHITNEY U-TEST BASED COMPARISON OF THE SKILL LEARNED FROM SCRATCH AND USING THE COMPOSED SKILL.

Task Id	**Task Composition**	**Average reward per episode during the learning phase for the skills learned from scratch.** **[Training Arena]**	**Average reward per episode during the test phase for the compound skill learned from scratch.** **[Tested in the arena shown in Fig. 6(d)]**	**Average reward per episode during the test phase for skills learned in a scaffolded environment and combined using method #1.** **[Tested in the arena shown in Fig. 6(d)]**	**Mann-Whitney U-Test of the results shown in the previous two columns.** **Reject H_0 hypothesis?**
C_1	p_1 AND p_3	9059 ±7675 [Fig. 6(a)]	16338 ±45	15018 ±60	p-value = 1.00, Fail to Reject H_0
C_2	p_2 AND p_3	-5023 ±3638 [Fig. 6(b)]	203 ±1537	1141 ±632	p-value = 0.00, Reject H_0
C_3	p1 AND p2	3875 ±1016 [Fig. 6(c)]	7027 ±376	4071 ±182	p-value = 1.00, Fail to Reject H_0
C_4	p_1 AND p_2 AND p_3	642 ±2990 [Fig. 6(d)]	1673 ±796	3065 ±702	p-value = 0.00, Reject H_0



(a) (b)

(c)

Fig. 10. (a) Trajectory of e-puck executing the combined policy for C_1 (avoiding obstacles and avoiding black regions on the ground), (b) Trajectory of e-puck executing the combined policy for C_2 (following blue robot and avoiding black regions on the ground), (c) Trajectory of e-puck executing the combined policy for C_3 (following blue robot and avoiding obstacles).

and avoiding obstacles.

Following the visual validation of skill composition, the RL agents' training was carried out for the compound tasks C_1, C_2, C_3, and C_4. The third column of Table 3 shows the results of the agent learning the compound task from scratch. The column 'Task Composition' shows the compound task's composition, and 'Task Description' describes the task. The training for the compound tasks constituted 50 episodes of 20,000 steps each. Ten iterations were run for each task with different start positions of the e-puck mobile robot every time. The standard deviation for 'Average Reward per Episode' was generated as shown in Table 3. The experiments were run on a Dell G3 machine with Intel 10th Gen I7 6-core CPU and 16 Gb RAM. Webots was used in the 'Fast Mode' with no graphical rendering resulting in 16x the real-time speed. The compound tasks' average learning time was approximately 35 minutes for each 50 episode run of 20000 steps, i.e., 560 minutes if the experiment was run at the real-time speed (which would be the case for a real robot).

For further validation, the learned compound skills were tested for 50 episodes, each comprising 20000 steps for compound tasks C_1, C_2, C_3, and C_4. Each test was run 10 times, and for each run, the starting position of the e-puck was different. The fourth column of Table 3 shows the results for the test phase of the compound skill learned from scratch, and the fifth column shows the results of the test phase of the composed skills learned in the scaffolded environment. The last column shows if the results for a compound skill learned from scratch and the composed skill show a statistically significant difference or not. Since the result data is not normally distributed, we ran a non-parametric method to determine if the data shows a statistically significant difference. The status quo or the Null hypothesis H_0 is that the reward received in the test phase by the agent using the policy learned from scratch is greater than the composed policy. That is to say, the agent using composed policy will not perform as well as the agent using the policy learned from scratch. We

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCDS.2022.3177691, IEEE Transactions on Cognitive and Developmental Systems

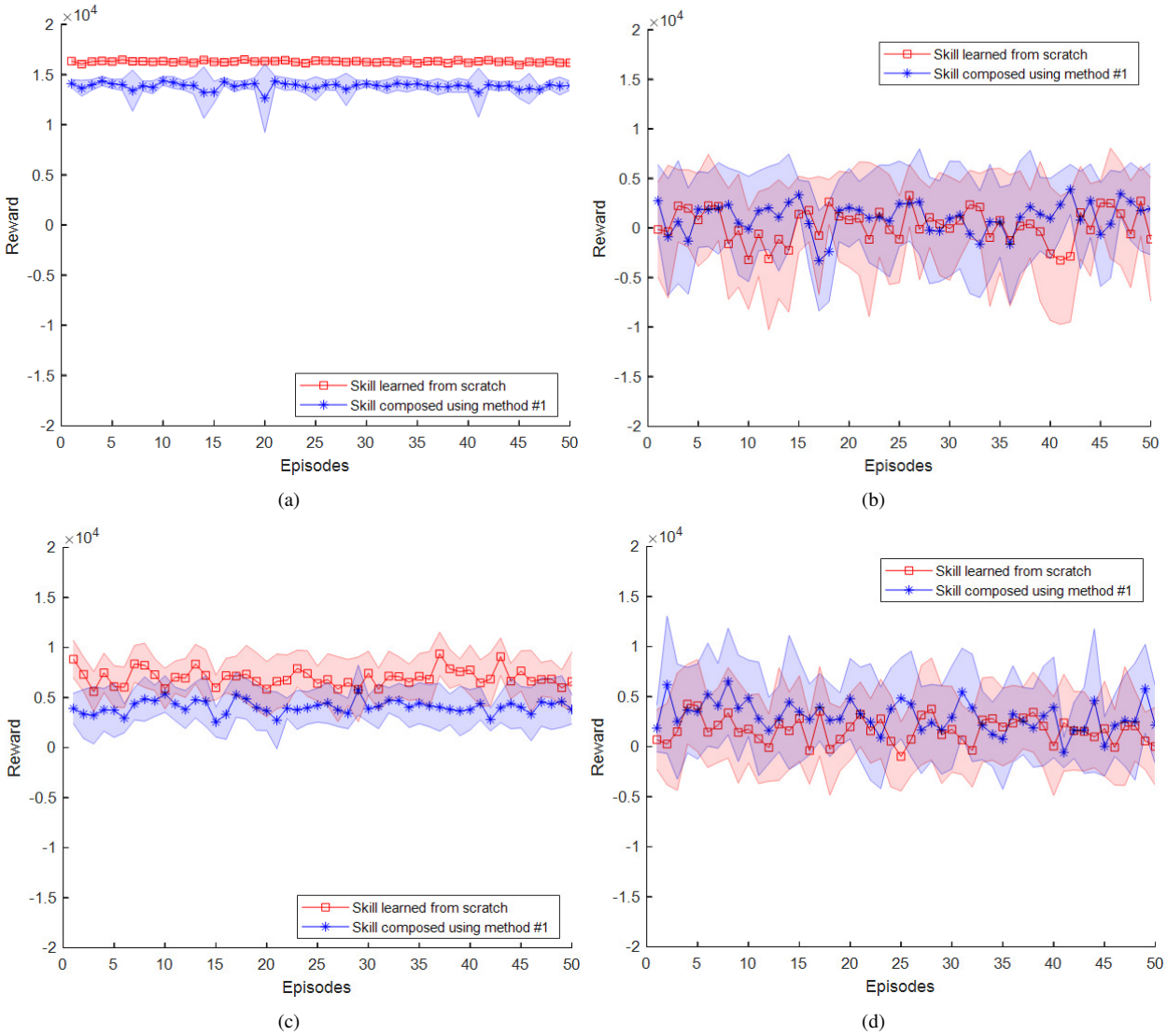


Fig. 11. (a) Episode reward plot for C_1 (avoid obstacles AND avoid the black regions on the floor), (b) Episode reward plot for C_2 (follow the blue robot AND avoid the black regions on the floor). (c) Episode reward plot for C_3 (avoid obstacles AND follow the blue robot), (d) Episode reward plot for C_4 (avoid obstacles AND follow the blue robot AND avoid the black regions on the floor)

ran Mann-Whitney U-Test on the 50 data points (an average of 10 runs for each episode) to determine if this hypothesis should be rejected or not. The alpha value for this statistical test was 0.05. The Null H_0 and the alternative H_1 hypothesis can be represented as:

H_0: average cumulative reward for the policy learned from scratch $>$ average cumulative reward for the composed policy

H_1: average cumulative reward for policy learned from scratch $\leq$ average cumulative reward for the composed policy

Fig. 11(a), 11(b), 11(c), and 11(d) show the graphical representation of Table 3, columns 4 and 5 results. The shaded plot in red is the episode reward during the test phase for the policy learned from scratch, and the shaded plot in blue is the episode reward during the test phase for the agent using the combined policy (primitive skills learned in the scaffolded environment and combined using method #1).

Mann-Whitney U-Test results show that for C_1 and C_3, the data supports the H_0 hypothesis, i.e., the policy learned from scratch is better than the composed policy. However, for C_2 and C_4, the test shows that the data suggests the rejection of the H_0 hypothesis, i.e., statistically, there is no difference between the policy learned from scratch and the composed policy and that the alternative hypothesis H_1 should be accepted. Tasks vary in complexity, resulting in a difference in skill acquisition. Generally, the hyperparameters such as the learning rate and EntropyLossWeight are tuned to ensure optimal results or policy convergence. In our experiments, however, the hyperparameters were the same for all the tasks. That may be why, in some cases, the skills of the task learned from scratch are better, and for other tasks, the composed skills are as good as those learned from scratch. Regardless, the average episode reward value for the composed skills is significant for all compound tasks, indicating that the agent demonstrates the correct behavior. Thus, the composed skill is as good as the skill learned from scratch in a good case scenario. In the worst-case scenario, the composed skill can be used as a starting

policy that can then be refined further, and such a partially learned policy will lead to reduced learning time [47]. In either case, the composition of skills results in time saved in learning the compound task's skill, thus demonstrating the advantage of using the proposed skill composition. When integrated with the lifelong learning architecture, such skill composition enables the agent to mix and match the learned skills to create skills for future tasks.

V. Conclusion and Future Work

Learning every skill from scratch is time-consuming, and even with RL's ever-improving sample efficiency, it remains a problem, especially for robotics applications. The logical next step for a lifelong learning agent is to use the learned knowledge to enable faster learning of future tasks. That learned knowledge used as a building block is typically learned in the same non-scaffolded environment as the other complex skills resulting in inefficiently learned primitive skills. This article hypothesizes that skill composition is more effective when the primitive skills are learned in specialized/scaffolded environments. We compare the results of primitive skills learned in the scaffolded versus non-scaffolded environment to demonstrate the learning effectiveness in a scaffolded environment. Further, we propose two variants of a skill composition method for RL policies represented by neural networks. The proposed technique provides an alternative to the concurrent skill composition techniques found in the literature. It offers the simplicity of combining Q-table based policies while maintaining the scalability provided by the neural network based policies. We show how the RL policies for compound tasks can be generated by a concurrent combination of the policies for primitive disjoint tasks. Those primitive skills that are the constituent skills of the composed skill are learned in a scaffolded environment. Using a mobile robot-based experiment, we then show how the combination of primitive skills could be used as a solution for a compound task with little or no additional training. A statistical comparison is used to demonstrate the effectiveness of the proposed skill combination method. In the best-case scenario, the composed policy is as good as the policy learned from scratch, and in the worst-case scenario, it can be used as a starting policy for further training providing a good trade-off between optimal solution and learning time. In either case, such reuse of the previously learned knowledge reduces the overall training time of multiple skills. That also results in a versatile system that can mix and match the skills, an essential requirement for a lifelong learning agent, especially in the robotics domain, where it may not always be feasible to learn solutions to all the tasks from scratch autonomously.

However, the contribution of this article forms a relatively small sub-system of an open-ended lifelong learning system, and the following are some of the future directions of this research described in the order of specific to general. Firstly, this article combined three primitive skills. In order to determine the limitation of the proposed method, it remains to be investigated if there is a maximum number of skills that can be combined before the performance of the combined skill deteriorates significantly. We hypothesize that a limit does exist, and the deterioration would be gradual. Secondly, this article limits the experiments to disjoint tasks. It would be interesting to see the results of the proposed method applied to the complementary and contradictory tasks. We hypothesize that when the policies for the contradictory tasks are combined, they will cancel each other out in the worst-case scenario, leading to a 'neutral' combined policy that will fail to execute either of the primitive tasks. For the complementary tasks, since the two policies agree on an action, the composed policy will be closer to the optimal policy [38]. The experiments in this article were designed for tasks that have disjoint state space. To test other types of tasks would require a different experimental setup. Also, as detailed in [48] [17], additional action arbitration infrastructure would be required to construct the combined policy. Thirdly, the proposed method should be tested on different types of robots, especially those with a much larger state and action space. In theory, the proposed composition techniques should scale; however, that remains to be investigated. Finally, this article investigated skill composition in isolation and that too with predetermined primitive tasks and scaffolded environments. For an open-ended learning system, the tasks to learn are not known upfront, and hence it is not possible to determine the design of the scaffolded environments. For such a system, the proposed skill composition method provides a partial solution to 'how to learn'. That should be integrated with a task generation method that provides the solution to the question 'what to learn' [19] and when to learn which skill [49][50][51]. Another aspect is whether these tasks are generated procedurally [19] or using intrinsic motivation [4][5]; even a relatively small number of primitive tasks will result in a combinatorial explosion of compound tasks. A potential solution suggested in [21] is to use a knowledge management module responsible for maintaining the list of tasks and skills. Furthermore, when such a lifelong learning system has built up its knowledge base of primitive skills, it should autonomously be able to select which skills should be combined and how. Those are some of the future directions of this research.

References

[1] F. Dignum and R. Conte, "Intentional Agents and Goal Formation," *Agent Theories, Architectures, and Languages*, pp. 231–243, 1997.

[2] A. Hsissi, H. Allali, and A. Hajami, "Metacognitive Scaffolding Agent Based on BDI Model for Interactive Learning Environments," *International Journal of Computer and Communication Engineering*, vol. 3, no. 2, pp. 97–100, 2014.

[3] V. G. Santucci, P. Y. Oudeyer, A. Barto, and G. Baldassarre, "Editorial: Intrinsically motivated open-ended learning in autonomous robots," *Frontiers in Neurorobotics*, vol. 13, no. January, 2020.

[4] J. Schmidhuber, "What's Interesting," pp. 1–23, 1997.

[5] P.-Y. Oudeyer, "Intelligent Adaptive Curiosity: a source of Self-Development," pp. 127–130, 2004.

[6] M. Mirolli and G. Baldassarre, "Functions and Mechanisms of Intrinsic Motivations. The Knowledge Versus Competence Distinction," in *Intrinsically Motivated Learning in Natural and Artificial Systems*, 2013, pp. 49–72.

[7] V. G. Santucci, G. Baldassarre, and M. Mirolli, "GRAIL: A goal-discovering robotic architecture for intrinsically-motivated learning," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 8, no. 3, pp. 214–231, 2016.

[8] B. van Niekerk, S. James, A. Earle, and B. Rosman, "Composing Value Functions in Reinforcement Learning," in *International Conference on Machine Learning*, 2019.

[9] F. Tanaka and M. Yamamura, "An approach to lifelong reinforcement learning through multiple environments," *Proc. of the 6th European Workshop on Learning Robot (EWLR-6)*, pp. 93–99, 1997.

[10] V. Mnih, A. P. Badia, L. Mirza, A. Graves, T. Harley, T. P. Lillicrap, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning,"

in *33rd International Conference on Machine Learning, ICML 2016*, vol. 4, 2016, pp. 2850–2869.

[11] B. Price and C. Boutilier, "Accelerating reinforcement learning through imitation," *Journal of Artificial Intelligence Research*, vol. 19, pp. 569–629, 2003.

[12] J. Kober and J. Peters, "Imitation and reinforcement learning," *IEEE Robotics and Automation Magazine*, vol. 17, no. 2, pp. 55–62, 2010.

[13] A. Lazaric, *Transfer in Reinforcement Learning : a Framework and a Survey*. Springer Berlin Heidelberg, 2012.

[14] M. E. Taylor and P. Stone, "Transfer Learning for Reinforcement Learning Domains : A Survey," *Journal of Machine Learning Research*, vol. 10, pp. 1633–1685, 2009.

[15] S. Singh, "Transfer of Learning by Composing Solutions of Elemental Sequential Tasks," *Machine Learning*, pp. 323–339, 1992.

[16] Y. Lee, S.-h. Sun, S. Somasundaram, E. Hu, and J. J. Lim, "Composing Complex Skills by Learning Transition Policies," in *International Conference in Learning Representations*, 2019, pp. 1–19.

[17] A. H. Qureshi, J. J. Johnson, Y. Qin, B. Boots, and M. C. Yip, "Composing Ensembles of Policies with Deep Reinforcement Learning," in *International Conference in Learning Representations*, 2020, pp. 1–16.

[18] G. Auda and M. Kamel, "Modular neural networks: a survey," *International journal of neural systems*, vol. 9, no. 2, pp. 129–151, 1999.

[19] P. Dhakan, K. Kasmarik, I. Rano, and N. Siddique, "Open-Ended Continuous Learning of Compound Goals," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 13, no. 2, pp. 274 – 285, 2019.

[20] A. Stout, G. D. Konidaris, and A. G. Barto, "Intrinsically Motivated Reinforcement Learning : A Promising Framework For Developmental Robot Learning," in *In Proceedings of the AAAI Spring Symposium on Developmental Robotics, Stanford University, Stanford, CA.*, 2005, pp. 1–6.

[21] P. Dhakan, K. E. Merrick, I. Rano, and N. Siddique, "Intrinsic rewards for maintenance, approach, avoidance, and achievement goal types," *Frontiers in Neurorobotics*, vol. 12, no. October, 2018.

[22] R. Wang, J. Lehman, J. Clune, and K. O. Stanley, "Paired Open-Ended Trailblazer (POET): Endlessly Generating Increasingly Complex and Diverse Learning Environments and Their Solutions," *arXiv*, pp. 1–28, 2019.

[23] M. Hauskrecht, N. Meuleau, L. P. Kaelbling, T. Dean, and C. Boutilier, "Hierarchical Solution of Markov Decision Process using Macro-actions," in *Fourteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc, 1998, pp. 220–229.

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

[25] M. Stolle, "Automated discovery of options in reinforcement learning," Ph.D. dissertation, McGill University, 2004.

[26] P. L. Bacon, J. Harb, and D. Precup, "The option-critic architecture," *31st AAAI Conference on Artificial Intelligence, AAAI 2017*, pp. 1726–1734, 2017.

[27] T. Dietterich, "An Overview of MaxQ Hierarchical Reinforcement Learning," *International Symposium on Abstraction, Reformulation, and Approximation*, pp. 26–44, 2000.

[28] Z. Kalmár, C. Szepesvári, and A. Lorincz, "Module-Based Reinforcement Learning: Experiments with a Real Robot," *Autonomous Robots*, vol. 5, no. 3-4, pp. 273–295, 1998.

[29] E. Uchibe, M. Asada, and K. Hosoda, "Behavior coordination for a mobile robot using modular reinforcement learning," in *IEEE International Conference on Intelligent Robots and Systems*, vol. 3, 1996, pp. 1329–1336.

[30] C. Drummond, "Composing functions to speed up reinforcement learning in a changing world," in *European Conference on Machine Learning*, 1998, pp. 370–381.

[31] È. Pairet, P. Ardón, M. Mistry, and Y. Petillot, "Learning and Composing Primitive Skills for Dual-Arm Manipulation," in *20th Annual Conference of Towards Autonomous Robotic Systems*, vol. 11649 LNAI, 2019, pp. 65–77.

[32] C. Tessler, S. Givony, T. Zahavy, D. J. Mankowitz, and S. Mannor, "A Deep Hierarchical Approach to Lifelong Learning in Minecraft," *31st AAAI Conference on Artificial Intelligence, AAAI 2017*, pp. 1553–1561, 2017.

[33] B. Wu, J. K. Gupta, and M. J. Kochenderfer, "Model Primitive Hierarchical Lifelong Reinforcement Learning," in *18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019)*, 2019.

[34] P. Tommasino, D. Caligiore, M. Mirolli, and G. Baldassarre, "A Reinforcement Learning Architecture That Transfers Knowledge Between Skills When Solving Multiple Tasks," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 11, no. 2, pp. 292–317, 2019.

[35] G. Berseth, C. Xie, P. Cernek, and M. Van de Panne, "Progressive Reinforcement Learning with Distillation for Multi-Skilled Motion Control," in *International Conference on Learning Representations*, 2018, pp. 1–15.

[36] W. M. Czarnecki, S. M. Jayakumar, M. Jadcrbcrg, L. Hasenclever, Y. W. Tch, S. Osindero, N. Heess, and R. Pascanu, "Mix and match - Agent curricula for reinforcement learning," in *35th International Conference on Machine Learning, ICML 2018*, vol. 3, 2018, pp. 1761–1773.

[37] X. B. Peng, M. B. Chang, G. Zhang, P. Abbeel, and S. Levine, "MCP: Learning Composable Hierarchical Control with Multiplicative Compositional Policies," in *Advances in Neural Information Processing Systems*, 2019, pp. 3686–3697.

[38] T. Haarnoja, V. Pong, A. Zhou, M. Dalal, P. Abbeel, and S. Levine, "Composable Deep Reinforcement Learning for Robotic Manipulation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, no. 1, 2018, pp. 6244–6251.

[39] E. Todorov, "Compositionality of Optimal Control Laws," in *Advances in Neural Information Processing Systems, 2009*, vol. 3, 2009, pp. 1856–1864.

[40] C. Simpkins and C. L. Isbell, "Composable Modular Reinforcement Learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 4975–4982.

[41] K. E. Merrick and M. L. Maher, *Motivated reinforcement learning: Curious characters for multiuser games*. Springer Berlin Heidelberg, 2009.

[42] S. Russell and A. L. Zimdars, "Q-Decomposition for Reinforcement Learning Agents," in *20th International Conference on Machine Learning*, 2003, pp. 656–663.

[43] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, pp. 529–533, 2015.

[44] B. T. Polyak and A. B. Juditsky, "Acceleration of stochastic approximation by averaging," *SIAM Journal on Control and Optimization*, vol. 30, no. 4, pp. 838–855, 1992.

[45] D. M. Tax, M. Van Breukelen, R. P. Duin, and Josef Kittler, "Combining multiple classifiers by averaging or by multiplying?" *Pattern Recognition*, vol. 33, no. 9, pp. 1475–1485, 2000.

[46] L. Braubach, A. Pokahr, D. Moldt, and W. Lamersdorf, "Goal representation for BDI agent systems," in *Second International Workshop on Programming Multiagent Systems: Languages and Tools*, 2005, pp. 9–20.

[47] J. Kober and J. Peters, "Policy search for motor primitives in robotics," *Machine Learning*, vol. 84, no. 1-2, pp. 171–203, 2011.

[48] H. Sahni, S. Kumar, F. Tejani, and C. Isbell, "Learning to compose skills," in *Advances in Neural Information Processing Systems*, 2017.

[49] S. Blaes, J.-j. Zhu, and G. Martius, "Control What You Can Intrinsically Motivated Task-Planning Agent," in *Advances in Neural Information Processing Systems*, no. 32, 2019.

[50] G. Santucci, G. Baldassarre, and E. Cartoni, "Autonomous Reinforcement Learning of Multiple Interrelated Tasks," in *2019 Joint IEEE 9th international conference on development and learning and epigenetic robotics (ICDL-EpiRob)*, 2019.

[51] C. Colas, P. Fournier, O. Sigaud, M. Chetouani, and P.-Y. Oudeyer, "CURIOUS: Intrinsically motivated modular multi-goal reinforcement learning," *36th International Conference on Machine Learning, ICML 2019*, vol. 2019-June, pp. 2372–2387, 2019.