MEWA: A Benchmark For Meta-Learning in Collaborative Working Agents

Radu Stoican, Angelo Cangelosi, Thomas Weisswange

2023

Preprint:

This is an accepted article published in IEEE Symposium Series on Computational Intelligence (SSCI 2023). The final authenticated version is available online at: https://doi.org/10.1109/SSCI52147.2023.10371913

MEWA: A Benchmark For Meta-Learning in Collaborative Working Agents

Radu Stoican
Manchester Centre for Robotics and AI
University of Manchester
Manchester, United Kingdom

Angelo Cangelosi
Manchester Centre for Robotics and AI
University of Manchester
Manchester, United Kingdom

Thomas H. Weisswange
Honda Research Institute Europe GmbH
Offenbach, Germany

***Abstract*—Meta-reinforcement learning aims to overcome important limitations in reinforcement learning, like low sample efficiency and poor generalization, by creating agents that adapt to new tasks. The development of intelligent robots would benefit from such agents. Long-standing issues like data collection and generalization to real-world dynamic environments could be mitigated by sample-efficient adaptable algorithms. However, most such algorithms have only been proven to work in low-complexity environments. These provide no guarantee that a near-optimal global policy does not exist, which makes it difficult to evaluate adaptable policies. This hinders the in-depth analysis of an agent's potential to adapt, while also introducing a gap between controlled experiments and real-world applications. We propose MEWA, a collection of task distributions used as a benchmark for adaptable agents. Our tasks contain a shared structure that an agent can leverage to learn the task-specific structure of new tasks. To ensure our environment is adaptive, we select some of the task parameters using the solution to a constrained optimization problem. Other parameters are randomized, allowing the creation of arbitrary task distributions. We evaluate three state-of-the-art meta-reinforcement learning algorithms on our benchmark, that were previously shown to adapt to new tasks with a simpler structure. Results show that the algorithms can reach meaningful performance on the task, but cannot yet fully adapt to the task-specific structure. We believe this benchmark will help identify some of the issues that hinder adaptability, ultimately aiding in the design of new algorithms, more suitable for real-world human-robot applications.**

I. Introduction

While reinforcement learning (RL) has achieved impressive results in controlled environments and simulations, its usefulness in real-world applications is still limited. Current RL algorithms suffer from poor generalization to new tasks and low sample efficiency [1]–[3], making them too unreliable and expensive for most applications [4]. This is especially true in the field of human-robot interaction (HRI). Human behavior is often complex and difficult to predict, so adaptable agents would have more success in interacting with humans. Additionally, improvements in sample efficiency would reduce the cost of collecting data from robots interacting with humans.

Recently, many researchers have attempted to use meta-reinforcement learning (meta-RL) to mitigate these issues. A meta-RL agent is expected to learn how to reinforcement learn [2], [5]. In the literature, meta-RL is most commonly used for few-shot adaptation [2]. During the meta-training phase, the agent learns an inductive bias by training on multiple tasks drawn from a task distribution. The agent is then expected to use this inductive bias during meta-testing, to quickly adapt to any new task from the same distribution. In meta-RL, each task is defined as a Markov decision process (MDP). An important assumption is that all MDPs in the distribution have the same state and action spaces, offering structure shared across all states. The transition probabilities and reward functions, on the other hand, are task-specific. For an agent with an inductive bias learned from the shared structure, the objective of learning a new task is reduced to adapting to the task-specific structure. Therefore, a meta-trained agent should be more sample efficient when adapting to new tasks than a standard RL agent learning from scratch. Besides standard RL, several other methods, more similar to meta-RL, have been used in the literature. The most relevant of these are multi-task RL [3], life-long RL [6], transfer learning [7], [8], or domain adaptation/generalization [9]. However, these methods are either limited by the assumptions they make or are not designed for few-shot adaptation [1].

We believe few-shot meta-RL is promising for HRI. For example, it could enable robots to adapt to the specific behavior of each human. Still, despite its potential, the current applications of meta-RL in HRI are limited. Meta-RL has been previously used for evaluating bi-directional trust as a robot adapts to a human [10]. Additionally, [11] presents a method of learning socially accepted behavior by adapting to new reward functions. These applications are a good start toward using meta-RL for adaptable robots.

In this work, we propose **Me**ta-learning for **W**orker **A**dapting (MEWA)[1], a new benchmark for few-shot meta-RL. MEWA is designed in the context of human-robot collaboration (HRC) [12]. We build MEWA as an extension of a previously proposed HRC task [13]. Fig. 1 shows the simulated version of our benchmark. We chose to focus specifically on adaptation to new tasks. Our design aims to guarantee that only algorithms that can truly adapt perform well on MEWA. These algorithms could later be used in agents that interact with real humans.

We believe MEWA will be a step forward toward solving one of the current problems faced by the meta-RL research community: state-of-the-art (SOTA) meta-RL algorithms have

[1]The code for MEWA is available at: https://github.com/RStoican/MEWA

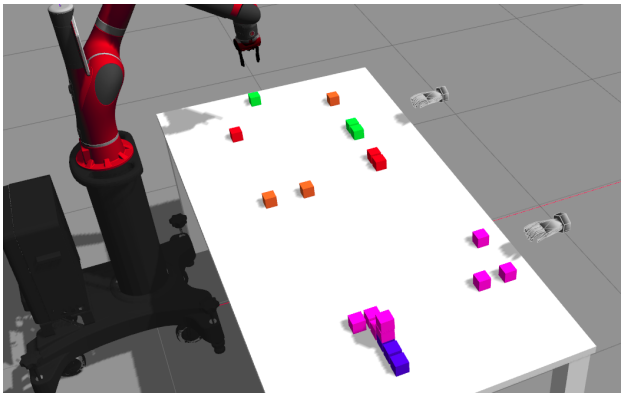

Fig. 1: A simulated version of our benchmark. The environment contains a Sawyer robot, a human, and the blocks required to complete a task.

only been shown to solve simple task distributions, with superficial differences between tasks [14]–[17]. These are usually referred to as narrow distributions. Recently, more effort has been put into designing benchmarks for wider task distributions [17], [18]. MEWA takes inspiration from two such benchmarks. We follow the main idea of Meta-World [17], i.e. aiming to design wider task distributions. We also make MEWA transparent and easy to analyze, in a similar manner to Alchemy [18]. At the same time, we aim to tackle two significant limitations in these existing benchmarks. First, there is no formal process for defining adaptive tasks, i.e. tasks that force agents to be adaptable. Second, there is no guarantee that a close-to-optimal non-adaptable policy does not exist. Such a policy could perform well on most of the tasks in the distribution. MEWA, on the other hand, proposes an environment built on such guarantees. We ensure that the set of policies available for use in a task distribution fulfills two conditions,

1) there is no globally optimal non-adaptable policy, and
2) there is a large gap between the expected return of any non-adaptable policy and the optimal (adaptable) policy.

The former condition ensures only an adaptable policy is optimal. The latter ensures it is easy to identify agents that do not adapt their policy to each task. In other words, there is a positive correlation between an agent's return and its ability to adapt. Based on these conditions, we propose a method for generating adaptive tasks. These tasks are parameterized by the solution to an optimization problem. The proposed benchmark aims to, together with existing methods, help in developing more powerful meta-RL approaches in the future.

II. Related Work

The number of existing benchmarks for meta-RL is limited. One of the most relevant for our work is Meta-World [17], which introduces several distributions of robotic manipulation tasks. It focuses on promoting wider task distributions for evaluating meta-RL algorithms. To the best of our knowledge, there is no formal definition for the width of a task distribution. In Meta-World, a narrow distribution is defined as a single task with many parametric variations. A wide distribution is then defined as a collection of such tasks (i.e. a collection of different narrow distributions). Meta-World proposes three task distributions: a narrow distribution (ML1) and two wide distributions (ML10, ML45). However, it is unclear how much shared structure there is for meta-RL agents to leverage when learning different manipulation tasks. In MEWA, we provide transparent tasks, with clearly defined shared and task-specific structures. An additional consequence of this limitation is the large gap between Meta-World's narrow and wide distributions. While SOTA meta-RL algorithms can achieve high performance on the narrow distribution [16], [17], it is unclear how much more powerful these algorithms have to be to solve the wide distribution.

Another relevant benchmark is Alchemy [18]. Its environment is a procedurally generated video game. Alchemy's main goals are to be interesting and accessible. One of its limitations is the low task complexity, already brought to attention by the authors [18]. This, together with the tasks' transparency, allowed Bayes-optimal and brute-force policies to be computed as baselines. While Alchemy's environment contains a fixed number of objects, MEWA's tasks can be defined as MDPs of varying complexities. Despite this, we are still able to provide the expected return of an optimal adaptable agent as a baseline, even if computing Bayes-optimal agents is infeasible. This leads to complex, but still transparent tasks. Finally, the only agents evaluated on Alchemy are trained using black-box recurrent-based meta-RL algorithms. To the best of our knowledge, there are currently no other works testing different meta-RL approaches on Alchemy. In our work, we instead provide results for SOTA algorithms from two main meta-RL approaches.

Recently, there has been a high number of RL benchmarks focused on generalization [19]–[23]. While these have similar goals to us, they are either limited in some aspects, focused on specific applications, or not appropriate for evaluating few-shot adaptation. MetaDrive [19] is an autonomous driving benchmark for generalization, safe RL, and multi-agent RL. CARL [20] extends existing RL environments to task distributions, but focuses on zero-shot generalization. Similarly to our work, URLB [21] is designed for adaptable agents, but in the context of pre-training through unsupervised RL. CausalWorld [22] presents a distribution of manipulation tasks for causal structure and transfer learning. Another large collection of manipulation tasks is given by RLBench [23], which focuses on many different topics, including few-shot meta-RL. These benchmarks do not share some of the properties MEWA has, which are required for evaluating few-shot meta-RL algorithms: wide task distributions, transparency, or an adaptive environment.

III. The MEWA Benchmark

A. Basic Rules

The learning environment of the benchmark is given by a collaborative game between the RL agent and another agent. The second agent will be called a *worker*, while *RL agent* and

agent will be used interchangeably throughout the paper. The worker has no learning capabilities and simply reacts to the agent's actions. At the start of a game, the agent is presented with several sets of differently colored blocks, such that each set has the same number of blocks. The agent and the worker have the common goal of building a specific structure using these blocks. At each step, the agent selects a color and hands a block of that color to the worker. The worker uses this block to build a sub-structure, where each sub-structure contains only blocks of the same color. After the worker receives enough blocks of the same color, the finished sub-structure is added to the main structure. The environment is episodic, with an episode ending when the structure is completed.

To add more task-specific structure to the environment, we enforce a correct order of completing the sub-structures. At each time step, a sub-goal is defined as (the color of) the next sub-structure that must be completed. Completing sub-goals in the wrong order will still finish the task, but will lead to a sub-optimal return. Finally, the worker has its own set of colored blocks. Whenever a sub-goal is completed, the worker lets the agent know by adding some of these blocks to the structure.

We make the environment stochastic by allowing the worker to make mistakes at specific points in the game. Whenever the agent chooses to play the second block of a structure, there is a chance the worker will make a mistake. A mistake is simply defined as building the wrong type of structure. When a wrong structure is completed, the blocks are returned to the agent instead of being used to complete the goal. This creates a delay in the game. Section III-C explains how these mistakes are used to design an environment that forces an optimal agent to be adaptable.

B. Observation Space, Action Space, and Reward Function

The observation space of the environment is given by the set of vectors that encode the agent's blocks, the worker's sub-structures and possible mistakes, and the progress towards the goal. We also provide a version of our benchmark in which the observations are images collected by a simulated Sawyer robot inside the Gazebo environment, as shown in Fig. 1. Images add complexity at the perception layer, but the underlying mechanics of the low-dimensional and high-dimensional versions of the benchmark are the same. Therefore, since the main goal of this benchmark is to test an agent's ability to adapt to changes in the environment, all the results presented in this work are from agents using vector observations. The action space is discrete and depends on the number of unique colors in the agent's set of blocks. We define one action per color in both environments using low- or high-dimensional observations.

The reward function is designed to highlight two of the core features of the environment: sub-goals must be completed in a specific order and each sub-goal must be completed as quickly as possible. We use dense rewards. Assume the goal is reached after completing N sub-goals, each of a different color. When working on the i-th sub-goal, $i \in \{1,2,...,n-1\}$, the agent receives a reward r_i at each step, such that $r_i < 0$ and $r_i < r_{i+1}$, with $r_n = 0$ as the final reward. Therefore, whenever the agent completes the next sub-goal, the rewards it receives in the future improve. Section III-D describes how we choose a reward function that leads to an adaptive environment. Finally, we choose to normalize rewards to $[0,1]$ both during meta-training and meta-testing. This helps mitigate the distraction dilemma [3], [24], caused by agents focusing more on learning tasks with higher reward magnitudes.

C. Task Distribution

We follow a hierarchical approach to defining the task distribution, similar to the one used in Meta-World [17]. At the upper level of the hierarchy, each task has N sub-goals and different colors, an order in which sub-goals must be completed, K blocks per color available to the agent, and a requirement of M blocks for completing each sub-goal. We call this a wide task distribution.

At the lower level, each task has variations, which are given by the mistake probabilities introduced in Section III-A. We call this a narrow task distribution. This means that every task from the previously defined wide distribution is in itself a (narrower) distribution. The probability that an environment transition will lead to a mistake is given both by the type of mistake and the specific worker used in that task variation. At each timestep, a mistake's type x is equal to the number of sub-structures that are on the worker's side (i.e. have at least one block and have not yet been completed). We refer to these as active sub-structures.

To give more shared structure between task variations, we design the worker's probability $w_x \in [0,1]$ of making a mistake of type x to always decrease as the number of active sub-structures increases, i.e. $w_x > w_{x+1}$. This might seem counter-intuitive in a scenario with real humans. However, thanks to these additional dynamics, delaying a sub-structure's completion might be an optimal policy. This forces the agent to adapt its policy to latent human behavior. For a skilled worker, an optimal agent will greedily complete sub-structures. For a worker that is likely to make mistakes, the agent acts optimally by building multiple sub-structures in parallel. Therefore, there is no globally optimal policy for all workers.

D. Adaptive Tasks

The goal of MEWA is to only use task variations that have the properties required to ensure only adaptable agents are optimal. Therefore, we present our method of optimizing the parameters of task variations for an adaptive environment. We call a state s critical if there is at least one action a the agent can take that could lead to a mistake happening. In a critical state in which there is at least one 1-block structure, the agent can take one of the two actions:

- A *risky* action adds the second block to a structure. This leads to a stochastic environment transition, dependent on the current worker's behavior;
- A *safe* action starts a new structure, if possible. This is a deterministic transition that leads to another critical state but with a reduced chance of a mistake happening.

In a critical state in which there is at least one 1-block structure and one other structure that can be completed in one step, one of the following two actions is optimal:

- A *complete* action finishes a structure. This is a deterministic transition that increases the chance of a mistake happening in the future (because the structure is removed from the game). If this leads to a sub-goal being completed, it also increases future rewards, as explained in Section III-B;
- A *postpone* action adds the second block to a 1-block structure. Similarly to a *risky* action, this could lead to a mistake happening. Taking the *postpone* action first, and only then completing the other structure is a viable strategy, as it keeps the mistake probability low.

We wish to optimize two aspects of the task: transition probabilities and the reward function. For every task variation, we refer to its optimal policy as a task-specific optimal (t-optimal) policy. We note that a t-optimal policy must have two properties,

- for every non-critical state, it takes the optimal action, and
- for every critical state, it takes a fixed number of
  - *safe* actions, then enough *risky* actions to reach the next state without a mistake occurring, or
  - *postpone* actions, then a *complete* action.

A task is considered adaptive when different variations have different t-optimal policies and the reward function punishes the usage of any t-optimal policy for variations in which it is not optimal. We define an *average* worker as a worker for which all possible t-optimal policies are optimal. Finding an average worker for a task would allow us to sample new workers from the space around it. Therefore, if there is enough distance between a sampled worker and the average worker, we can expect the former to have a smaller number of optimal policies. Moreover, if the sampling is done from different regions around the average worker, these optimal policies will be different across workers. Besides the optimal worker, it is also important to find a reward function that makes the environment adaptive. If the rewards received during early sub-goals are too low, the agent will be motivated to complete those sub-goals as quickly as possible. This leads to *complete* actions always being optimal over *postpone* actions, irrespective of the worker's behavior. Rewards that are too high have the opposite effect, i.e. the agent is not pressured to complete sub-goals quickly. Therefore, we need to find a reward function that ensures all actions are relevant.

For a task with N sub-goals, we find the average worker $\mathbf{w} = [w_1, w_2, ..., w_N]$ and the optimal reward function $\mathbf{r} = [r_1, r_2, ..., r_N]$ by solving the constrained optimisation problem

$$\begin{aligned} &\text{minimize } f_1(\mathbf{w},\mathbf{r}) + f_2(\mathbf{w},\mathbf{r}) \\ &\text{subject to } g_k(\mathbf{w}) \leq 0, \quad k = 0,\ldots,N-1 \\ &\qquad\qquad h_l(\mathbf{r}) \leq 0, \quad l = 0,\ldots,N-1, \end{aligned} \tag{1}$$

where f_1 and f_2 are the functions we are trying to optimise, while g_k and h_l are the constraint functions. The constraints ensure that all elements in $\mathbf{w}$ are valid probabilities and that both $\mathbf{r}$ and $\mathbf{w}$ follow the constraints mentioned in Sections III-B and III-C, respectively. When defining $f_1(\mathbf{w},\mathbf{r})$ and $f_2(\mathbf{w},\mathbf{r})$, we only need to consider critical states, since transitions from other states do not depend on the worker.

The first part of the objective function, $f_1(\mathbf{w},\mathbf{r})$, ensures that, in a critical state s in which *risky* and *safe* actions are possible, both actions are optimal for the average worker. Given the critical state s in which the i-th sub-goal contains only one block, the agent must transition (in one or more steps) to a state s'. In s', the i-th sub-goal has two blocks and no mistake has been made. We define the return of transitioning from s to s' as

$$c_i(\mathbf{w},\mathbf{r},x,y) = (d(w_x,x)+y)r_i - \sum_j r_j, \tag{2}$$

where there are x active sub-structures at state s, the agent adds y more by taking *safe* actions, then keeps taking the *risky* action until s' is reached. The right-most sum is taken over the number of new sub-structures added by the *safe* actions. The expected number of steps required to reach s' by taking the risky action is given by $d(w_x,x)$. As before, w_x is the probability of making a mistake given x sub-structures and worker $\mathbf{w}$, leading to

$$d(w_x,x) = \frac{1+w_x(M-1)}{1-w_x}, \tag{3}$$

where M is the number of blocks required to complete a sub-structure in the current task. Finally, for $\mathbf{w}$ to be average, the number of *safe* actions y and the final number of extra sub-structures x should be irrelevant, i.e. they all lead to the same cost. Moreover, this must hold for all sub-goals. This leads to the first part of the objective function,

$$f_1(\mathbf{w},\mathbf{r}) = \sum_{i=0}^{N-1}\sum_{x=0}^{N-1} f_i'(\mathbf{w},\mathbf{r},x) + \sum_{y=0}^{x-1} f_i''(\mathbf{w},\mathbf{r},x,y),$$

where

$$\begin{aligned} f_i'(\mathbf{w},\mathbf{r},x) &= (c_i(\mathbf{w},\mathbf{r},x,x) - c_i(\mathbf{w},\mathbf{r},x+1,0))^2 \\ f_i''(\mathbf{w},\mathbf{r},x,y) &= (c_i(\mathbf{w},\mathbf{r},x,y) - c_i(\mathbf{w},\mathbf{r},x,y+1))^2. \end{aligned}$$

The second part of the objective function, $f_2(\mathbf{w},\mathbf{r})$, deals with critical states in which *complete* or *postpone* actions can be taken. Let i be the index of the sub-goal that must be completed next. In this type of state, the i-th structure already contains two blocks. Then, let j be the index of one of the structures that have exactly one block. We define the cost of adding the second block to structure j before completing the i-th sub-goal as

$$e(\mathbf{w},\mathbf{r},x,i) = d(w_x,x)r_i, \tag{4}$$

where is the number of extra structures, besides the i-th structure, and $d(w_x,x)$ is defined in Eq. 3. If $\mathbf{w}$ is an average worker, then it should not matter if the agent adds the second block to the j-th structure when working on the j-th sub-goal

(i.e. Eq. 2) or on the i-th sub-goal (i.e. Eq. 4). This leads to defining the second part of the objective as

$$f_2(\mathbf{w},\mathbf{r}) = \sum_{j=1}^{N-1}\sum_{i=0}^{j-1}\sum_{x=j-i}^{N-1}(c_j(\mathbf{w},\mathbf{r},x,x) - e(\mathbf{w},\mathbf{r},x,i))^2.$$

Finally, we convert the constrained optimization problem from Eq. 1 into a loss function [25]. Then, we train a deep neural network to find the optimal average worker $\mathbf{w}$ and rewards $\mathbf{r}$. This can be done to each task in a wide distribution to define the optimal $\mathbf{w}$ and $\mathbf{r}$ of its narrow distribution. We found that a naive approach of overfitting a network to each task is sufficient to solve Eq. 1, but this could be extended to a more general model that solves any task without the need for retraining [26].

E. Evaluation Method

Our target is to show a lack of adaptation to changes in the environment, given by a narrow task distribution. Empirically, we found that a task with $N = 4$ different colors is enough to do that. Therefore, we design a narrow distribution over this single task for evaluating meta-RL agents. A higher N would increase the performance gap between an adaptive and a non-adaptive agent, but it would also increase the complexity of the environment, making it computationally expensive to meta-train and meta-test agents, or verify that the environment is adaptive.

For $N = 4$, each variation of the task can be described by a 4-component vector of probabilities $\mathbf{w} \in [0,1]^4$. Given the average worker $\mathbf{w}_{avg}$ and a parameter σ for this task, we split the distribution of $\mathbf{w}$ into three regions. During meta-training, we sample workers from any of these regions. However, when constructing the meta-testing set of workers, we only sample from regions that are at least $2 * \sigma$ away from $\mathbf{w}_{avg}$. Avoiding the regions around $\mathbf{w}_{avg}$ makes it simpler to build a set of tasks that can only be solved optimally by adaptable agents.

For meta-testing, we create a curated set of workers that are distinct enough to have different optimal policies. We start by discretizing the worker space $[0,1]^4$ in steps of 0.05, starting from the 0-vector (i.e. the worker that never makes any mistake). For each worker, we then run all possible distinct *t-optimal* policies and compute their return. These policies are identical to the ones used to compute the maximum expected baseline (see Section III-F). The curated set is selected from these workers, based on two criteria. First, for each worker, the returns of sub-optimal policies must be significantly lower than the returns of optimal policies. This makes it easier to identify agents that use a sub-optimal policy during evaluation. We use the mean absolute deviation (MAD) to compute the variability of the policies' returns. Second, we only select workers that have a distinct return distribution from the workers that are already in the curated set. This ensures there exists no globally optimal policy that can solve all task variations, so non-adaptive agents will never be optimal. The Mann-Whitney U test is used to compute the difference between distributions.

We create the curated set of task variations for the narrow distribution by adding the worker with the highest MAD. We then perform the Mann-Whitney U test between this worker and all the others. We consider all workers that have a p-value ≤ 0.05 and add only the one with the highest MAD to the curated set. We repeat this process until there are no more workers that have statistically significant differences from the ones in the curated set. The last step of the process is to compute the average normalized return of each policy over the current set of workers, as shown in Fig. 2. Given the policy π^* with the highest average return on all workers, we aim to minimize its global performance by iteratively adding workers for which π^* is sub-optimal. Whenever we reach a point in which adding a new worker leads to an increase in the average return, we stop and consider the curated set complete. An alternative would be that, instead of stopping, we select a new π^* and repeat the process. However, we found, empirically, that stopping early leads to the most diverse set of workers.

For experiments on wide distributions, we create a set of curated tasks. We sample the parameters of each task as the number of different colors $N \in \{3,4\}$, the order of sub-goals, the number of blocks available to the agent $K \in \{3,4,5\}$ and the number of blocks required to complete each sub-goal $M \in \{3,4,5\}$, with $M \leq K$. For each task, we then create a narrow distribution of workers, using the approach described in the previous paragraph.

F. Baselines

To assess the performance of meta-RL agents in our environment, we follow the idea of designing accessible benchmarks [18], by providing a set of three baselines. The simplest of these is given by a policy that takes actions at random. A second baseline is given by a *random task-specific* policy. Whenever this policy reaches a critical state, it will randomly select one valid type of action: *risky*, *safe*, *complete*, or *postpone*. In non-critical states, this policy takes the optimal action. Overall, this baseline simulates a policy that has perfect knowledge of the shared structure of a task

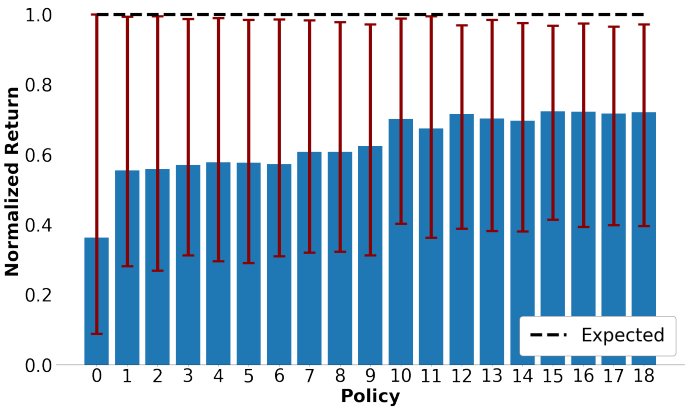

Fig. 2: The normalized returns of all distinct t-optimal policies, averaged over a set of different workers, for a single task with $N = 4$ different colors. A policy that reaches a return of 1 is globally optimal for all workers. The error bars give the minimum and maximum returns.

(e.g. the correct sub-goal order) but completely ignores the task-specific structure (i.e. the worker's behavior).

The last baseline computes the expected return of an optimal adaptable agent. Computing an oracle policy by brute force or a Bayes-optimal policy as in [18] is intractable for all but the simplest tasks in our environment. Instead, we leverage our knowledge of the environment's mechanics to reduce the space of possible optimal policies. We know that only *t-optimal* policies can be optimal in a task, so we can safely ignore all other policies (e.g. policies that complete sub-goals in random order). We create a set of policies $\pi(a|s,\rho_1,\rho_2,q)$, where π is a *t-optimal* policy. Given a critical state in a task with N sub-goals, $\rho_1 \in \{0,1,\ldots,N\}$ is the number of *safe* actions to take before taking a *risky* action and $\rho_2 \in \{0,1,\ldots,\rho_1\}$ is the number of *postpone* actions to take before taking a *complete* action. In some tasks, given the number of blocks per color K and the number of blocks required to complete each sub-structure M, we might have $K < M$. The optimality of such actions depends on each specific task and worker. For $q \in \{0,1\}$, the *t-optimal* policy will only use leftover blocks if $q = 1$ and there are no other *safe* actions. We iterate over ρ_1, ρ_2, and q to create the set of all possible *t-optimal* policies for a specific task. The maximum expected return over a narrow distribution is computed by running each *t-optimal* policy on each worker, selecting the maximum return, and taking their average. Similarly, for a wide distribution, the same process is repeated for each task, and the average is taken as the baseline.

IV. Experiments

A. Task Adaptiveness

We provide empirical results showing that our worker distribution, for a task with $N = 4$ colors, is adaptive. First, Fig. 2 shows that, for a curated set of task variations, no *t-optimal* policy can reach the expected maximum return for all workers. Therefore, a non-adaptive agent that chooses any of these policies will never be optimal. Moreover, the maximum return given by the top error bars shows that each policy is (almost) optimal for at least one worker.

Second, we show that, for a task using the average worker, any *t-optimal* policy is optimal. We use the same task with $N = 4$ and the same discretized worker space as in Section III-E. For each worker, we run each *t-optimal* policy and measure the variability of their returns, using MAD. The average worker has a normalized MAD of only 0.013. This indicates that any *t-optimal* policy is optimal for the average worker. At the same time, the rest of the workers have higher variability. So, for each of these, there is a subset of *t-optimal* policies that are sub-optimal. Fig. 3 presents a visual explanation of these results. To improve visualization, we only present the results on a subset of the worker space

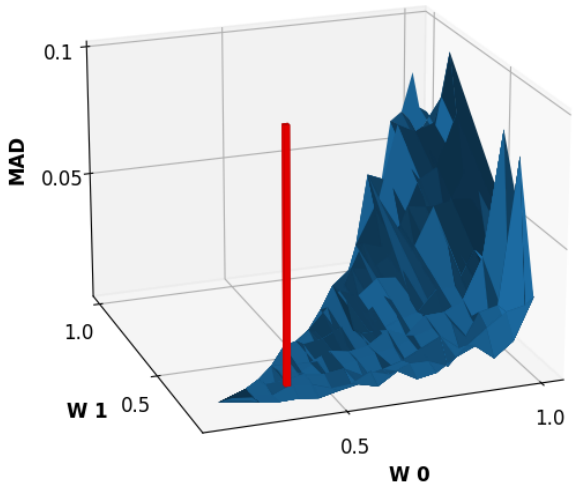

Fig. 3: The MAD of a subset of the discretized worker space $[0,1]^4$. We choose the subset of vectors where the last two dimensions are fixed to the average worker. The red bar represents the average worker.

B. Performance Results

We present results for three meta-RL algorithms: MAML [14], PEARL [15] and VariBAD [16]. The literature has shown these algorithms can achieve high performance in other adaptive environments, like Meta-World's ML1 [17]. For PEARL, two changes have to be made before it can be used in our environment. We modify the soft actor-critic (SAC) algorithm [27] used by PEARL to work on discrete action spaces [28]. Moreover, we use the version of SAC that automatically tunes the temperature hyperparameter [29], since manually choosing the right temperature can be difficult.

We first meta-train the agents on a narrow distribution given by variations over a single task. We then meta-test these agents on the curated set of workers. On each worker, the agents are run for 10 episodes. An adaptable agent is any agent that improves its performance across episodes. The results are shown in Fig. 4 (top). We present the average normalized return per episode. Overall, all algorithms can learn the shared structure among task variations, e.g. the optimal order of completing tasks. PEARL shows no signs of adapting over 10 episodes. MAML's performance decreases as the meta-testing progresses, reaching a point slightly below the random policy. VariBAD is the only algorithm that shows signs of adapting to specific workers, during the first 3 episodes. However, this is still not enough to reach the performance of the random task-specific policy, and it is overall far from the maximum expected return. We measure the adaptability of each algorithm as the average difference between the last and first episodes. PEARL has a value of 0.001, which indicates stagnation, MAML reaches -0.1, while VariBAD reaches -0.06, but shows signs of adapting during the first 3 episodes, with an adaptability value of 0.03. Overall, none of the algorithms manage to fully solve the task, showing that even simple variations given by the workers' probability distributions can be challenging.

Fig. 4 (bottom) shows the results of our experiment for the agents meta-trained on a wide distribution of several tasks

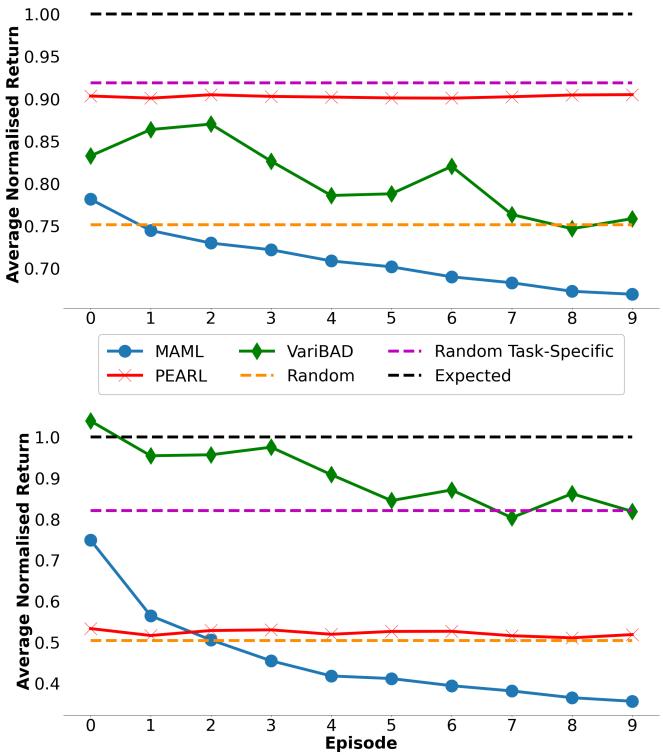

Fig. 4: The performance of three meta-RL algorithms on the curated set of tasks for a narrow distribution (top) and a wide distribution (bottom). For each algorithm in the narrow and wide distributions, we train with 5 and 3 different seeds, respectively. Performance is given by the average return, normalized to the maximum expected baseline, and measured across 10 episodes.

and meta-tested on the corresponding curated set of tasks. Agents are allowed to adapt to each task-worker pair for 10 episodes. The performance of PEARL and MAML is much lower than for the narrow distribution. PEARL shows no signs of adapting, and its performance is only slightly better than the random baseline. MAML's performance decreases with the amount of data. This shows that the agent is at least attempting to adapt but fails. This behavior is similar to the one observed in the narrow distribution, but the drop in performance is even higher. The most surprising result is given by VariBAD. While the decrease in performance shows a failure in adapting, its initial performance outperforms the maximum expected return. We have two possible explanations for this behavior. First, this might be caused by a limitation in the benchmark, since we only provide a formal method of creating task variations, while the task parameters in the wide distribution are randomly sampled. Second, VariBAD is better suited for handling this wider distribution, as it performs online adaptation [16]. This is different from the other two algorithms, which only update their meta-policy after each episode. Therefore, VariBAD can adapt within the first episode. Finally, the adaptability values are lower than in the narrow distribution experiment, with -0.01 for PEARL, -0.31 for MAML, and -0.17 for VariBAD.

V. Discussion

We have provided a benchmark for developing and evaluating adaptable agents. Our work was inspired by HRI scenarios, where unpredictable and dynamic human behavior makes adaptability essential. Our benchmark allows the random or manual generation of tasks and task variations of various complexities. We also provide a method of ensuring that these tasks are only solvable by adaptable agents. Empirical results show that current meta-RL algorithms can make progress on narrow distributions, showing early signs of sample-efficient adaptation to task variations, or at least of learning the shared structure of the task. However, even in this case, more work has to be done to completely solve our proposed distribution.

Similarly to Meta-World [17], we observe that PEARL generally outperforms MAML, but both algorithms struggle with wide task distributions. Moreover, in our case, VariBAD outperforms both PEARL and MAML. The same is true for Meta-World, with VariBAD almost completely solving the ML1 narrow task distribution [16]. This reinforces the idea that the difficulty gap between Meta-World's narrow and wide distributions is too large for meta-RL research to progress smoothly. Finally, it is important to note that VaridBAD was able to adapt and reach maximal return in two out of three ML1 tasks in only 1 episode. On the other hand, our results show that VariBAD is unable to adapt to MEWA within a single episode, but still improves its performance within multiple episodes. This offers insights into the capabilities needed to solve MEWA. An agent must learn a policy that, during meta-testing, can employ an adaptable exploration strategy, conditioned on episodes collected from the task the agent is adapting to.

A limitation of MEWA is that we do not provide a formal process for defining task parameters for adaptive wide distributions. Moreover, in this work, we decided to prioritize the complexity of the underlying task dynamics over the complexity of visual-motor elements, but MEWA could be extended to continuous state and action spaces. We also acknowledge that a more in-depth analysis of how and why meta-RL algorithms fail to solve parts of our benchmark will provide more insight into future directions for adaptable RL algorithms. Finally, we believe it will be worthwhile to investigate how the structure shared between tasks affects an agent's ability to adapt.

Although the meta-RL concept is promising in addressing challenges in HRI, current SOTA algorithms fail to deliver on this promise. Our benchmark provides an explicit means to evaluate new algorithms, focusing particularly on adaptation. We believe that this will allow future research to develop meta-RL methods that explicitly target adaptability. This is in contrast to many of the tasks used in the current literature, where algorithms that generate a good generalization policy might still achieve high performance, relative to the optimal adaptable policy.

Acknowledgment

This study was conducted as part of a research program of the Honda Research Institute Europe GmbH. In addition,

Cangelosi's work is partially supported by the US Air Force Project THRIVE++ and the UKRI Project on the TAS Trust Node.

REFERENCES

[1] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, "Meta-learning in neural networks: A survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 9, pp. 5149–5169, 2021.

[2] J. Beck, R. Vuorio, E. Z. Liu, Z. Xiong, L. Zintgraf, C. Finn, and S. Whiteson, "A survey of meta-reinforcement learning," *arXiv preprint arXiv:2301.08028*, 2023.

[3] N. Vithayathil Varghese and Q. H. Mahmoud, "A survey of multi-task deep reinforcement learning," *Electronics*, vol. 9, no. 9, p. 1363, 2020.

[4] R. Kirk, A. Zhang, E. Grefenstette, and T. Rocktäschel, "A survey of generalisation in deep reinforcement learning," *arXiv preprint arXiv:2111.09794*, 2021.

[5] S. Thrun and L. Pratt, "Learning to learn: Introduction and overview," in *Learning to learn*. Springer, 1998, pp. 3–17.

[6] K. Khetarpal, M. Riemer, I. Rish, and D. Precup, "Towards continual reinforcement learning: A review and perspectives," *Journal of Artificial Intelligence Research*, vol. 75, pp. 1401–1476, 2022.

[7] H. Ju, R. Juan, R. Gomez, K. Nakamura, and G. Li, "Transferring policy of deep reinforcement learning from simulation to reality for robotics," *Nature Machine Intelligence*, pp. 1–11, 2022.

[8] F. Muratore, M. Gienger, and J. Peters, "Assessing transferability from simulation to reality for reinforcement learning," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 4, pp. 1172–1183, 2019.

[9] K. Zhou, Z. Liu, Y. Qiao, T. Xiang, and C. C. Loy, "Domain generalization: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[10] Y. Gao, E. Sibirtseva, G. Castellano, and D. Kragic, "Fast adaptation with meta-reinforcement learning for trust modelling in human-robot interaction," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 305–312.

[11] A. Ballou, C. Reinke, and X. Alameda-Pineda, "Variational meta reinforcement learning for social robotics," *arXiv preprint arXiv:2206.03211*, 2022.

[12] F. Semeraro, A. Griffiths, and A. Cangelosi, "Human–robot collaboration and machine learning: A systematic review of recent research," *Robotics and Computer-Integrated Manufacturing*, vol. 79, p. 102432, 2023.

[13] R. Stoican, A. Cangelosi, C. Goerick, and T. Weisswange, "Learning human-robot interactions to improve human-human collaboration," 2022.

[14] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International conference on machine learning*. PMLR, 2017, pp. 1126–1135.

[15] K. Rakelly, A. Zhou, C. Finn, S. Levine, and D. Quillen, "Efficient off-policy meta-reinforcement learning via probabilistic context variables," in *International conference on machine learning*. PMLR, 2019, pp. 5331–5340.

[16] L. Zintgraf, S. Schulze, C. Lu, L. Feng, M. Igl, K. Shiarlis, Y. Gal, K. Hofmann, and S. Whiteson, "Varibad: variational bayes-adaptive deep rl via meta-learning," *The Journal of Machine Learning Research*, vol. 22, no. 1, pp. 13 198–13 236, 2021.

[17] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine, "Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning," in *Conference on robot learning*. PMLR, 2020, pp. 1094–1100.

[18] J. X. Wang, M. King, N. Porcel, Z. Kurth-Nelson, T. Zhu, C. Deck, P. Choy, M. Cassin, M. Reynolds, F. Song, *et al.*, "Alchemy: A benchmark and analysis toolkit for meta-reinforcement learning agents," *arXiv preprint arXiv:2102.02926*, 2021.

[19] Q. Li, Z. Peng, L. Feng, Q. Zhang, Z. Xue, and B. Zhou, "Metadrive: Composing diverse driving scenarios for generalizable reinforcement learning," *IEEE transactions on pattern analysis and machine intelligence*, 2022.

[20] C. Benjamins, T. Eimer, F. Schubert, A. Biedenkapp, B. Rosenhahn, F. Hutter, and M. Lindauer, "Carl: A benchmark for contextual and adaptive reinforcement learning," *arXiv preprint arXiv:2110.02102*, 2021.

[21] M. Laskin, D. Yarats, H. Liu, K. Lee, A. Zhan, K. Lu, C. Cang, L. Pinto, and P. Abbeel, "Urlb: Unsupervised reinforcement learning benchmark," *arXiv preprint arXiv:2110.15191*, 2021.

[22] O. Ahmed, F. Träuble, A. Goyal, A. Neitz, Y. Bengio, B. Schölkopf, M. Wüthrich, and S. Bauer, "Causalworld: A robotic manipulation benchmark for causal structure and transfer learning," *arXiv preprint arXiv:2010.04296*, 2020.

[23] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison, "Rlbench: The robot learning benchmark & learning environment," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3019–3026, 2020.

[24] S. Kelly, T. Voegerl, W. Banzhaf, and C. Gondro, "Evolving hierarchical memory-prediction machines in multi-task reinforcement learning," *Genetic Programming and Evolvable Machines*, vol. 22, pp. 573–605, 2021.

[25] A. Aubret, "Learning increasingly complex skills through deep reinforcement learning using intrinsic motivation," Ph.D. dissertation, Université de Lyon, 2021.

[26] G. Villarrubia, J. F. De Paz, P. Chamoso, and F. De la Prieta, "Artificial neural networks used in optimization problems," *Neurocomputing*, vol. 272, pp. 10–16, 2018.

[27] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.

[28] P. Christodoulou, "Soft actor-critic for discrete action settings," *arXiv preprint arXiv:1910.07207*, 2019.

[29] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.