**manuscript No.**
(will be inserted by the editor)

# A novel optimization perspective to the problem of designing sequences of tasks in a reinforcement learning framework

**Ruggiero Seccia · Francesco Foglino · Matteo Leonetti · Simone Sagratella***

Received: date / Accepted: date

**Abstract** Training agents over sequences of tasks is often employed in deep reinforcement learning to let the agents progress more quickly towards better behaviours. This problem, known as *curriculum learning*, has been mainly tackled in the literature by numerical methods based on enumeration strategies, which, however, can handle only small size problems. In this work, we define a new optimization perspective to the curriculum learning problem with the aim of developing efficient solution methods for solving complex reinforcement learning tasks. Specifically, we show how the curriculum learning problem can be viewed as an optimization problem with a nonsmooth and nonconvex objective function and with an integer feasible region. We reformulate it by defining a grey-box function that includes a suitable scheduling problem. Numerical results on a benchmark environment in the reinforcement learning community show the effectiveness of the proposed approaches in reaching better performance also on large problems.

**Keywords** Curriculum learning · (Deep) Reinforcement learning · Black-box optimization · Sequential Model-Based Optimization.

* Corresponding author

Simone Sagratella, Ruggiero Seccia
Department of Computer, Control and Management Engineering Antonio Ruberti, Sapienza University of Rome, Roma, Italy
E-mail: {sagratella,seccia}@diag.uniroma1.it

Matteo Leonetti
Department of Informatics, King's College London, London, UK
E-mail: matteo.leonetti@kcl.ac.uk

Francesco Foglino
School of Computing, University of Leeds, Leeds, UK
E-mail: scff@leeds.ac.uk

## 1 Introduction

Deep Reinforcement Learning is characterized by such a large and complex environment that classical exploration methods designed for Markov Decision Processes (MDPs) cannot be adequately implemented [14]. In this setting, an increasingly popular approach is represented by *curriculum learning*, where the knowledge gained in simple tasks is exploited to learn a similar but more complex task. In particular, in curriculum learning, we are interested in defining sequences of simple tasks so that training the agent on these problems allows us to reduce the overall training time for the final task and/or leads to improvements of the agent's performance over the final challenging task. Curriculum learning, by exploiting knowledge transfer and generalization from less complex tasks, represents an effective exploration strategy, which allows the agent to advance towards better policies with better performance over the final complex task. Curriculum learning is can be seen as two main challenges: *task generation*, that is, the problem of defining the set of tasks on which to train the agent before the final task; *task scheduling*, that is the problem of choosing the schedule of tasks on which to train the agent [14]. Automatically generating the set of potential tasks in a domain-independent way is currently an open problem [14], and tasks are commonly created manually by domain experts. Hereafter we focus on the sequencing problem and consider the set of potential tasks as already available.

Curricula are often present in many deep reinforcement learning applications, in which either designers [15] or users [17] create subtasks that are intuitively expected to facilitate learning. However, given the set of potential tasks, the curricula definition process can be improved by introducing more rigorous techniques from both the fields of computer science and optimization. In [22] an automated method for creating a curriculum graph is proposed where a heuristic metric to estimate the relatedness between tasks is used to build the graph. This method does not take into consideration the agent's unique abilities and learning progress during training, and does not explore the curricula space, returning a solution with no quality guarantee. In [16] the design of a curriculum is formulated as a Markov Decision Process, which directly models the accumulation of knowledge as the agent interacts with tasks but does not gain knowledge on relations between tasks. More recently, [8] proposed a set of greedy strategies to solve the task scheduling problem. This approach is mainly based on enumeration strategies to find the best task schedule and cannot be extended to cases where the number of potential curricula becomes too large. We aim at developing a new optimization-based approach to efficiently and effectively tackle the challenging problem of task scheduling in curriculum learning.

We address the curriculum learning problem by modeling it as an optimization program. We show its difficulty in being solved through standard optimization techniques. Specifically, this problem is a highly nonconvex, nonsmooth optimization program with an integer feasible region whose dimensions can easily blow up with the number of available tasks. To tackle this prob-

lem numerically, we explicitly show how the applied curriculum influences the final task performance. In particular, we apply several numerical methods to determine a set of designing parameters that define the relationships between tasks and then solve a linear scheduling problem to return tentative optimal curricula. Numerical results suggest the effectiveness of all the proposed methods in improving the performances on the final task with model-based algorithms outperforming the other approaches mentioned above. This study strongly extends the work done in the conference proceeding [9]. Specifically, in this paper, we provide all theoretical foundations and guarantees of the proposed approach. Moreover, we discuss new extensive numerical experiments over large problems where curricula enumeration is not computationally feasible, and we benchmark the numerical performance of the curriculum learning reformulation against a standard reinforcement learning framework.

In Section 2, after a brief introduction to reinforcement learning, see e.g. [12, 13], we define the curriculum learning problem and discuss the difficulty in solving this optimization problem through standard methods for nonlinear programming or through Derivative-Free (DF) algorithms. In Section 3, we reformulate the curriculum learning problem as a new grey-box optimization problem, including a suitable scheduling problem. In Section 4, several approaches for tackling this grey-box reformulation are discussed. Both heuristic methods to determine approximate solutions of the grey-box problem, and DF optimization methods that estimate the best parameters with a Bayesian approach are investigated. Lastly, in Section 5, numerical results on a benchmark environment in the curriculum learning literature are shown, and in Section 6 conclusions on the proposed methods are drawn.

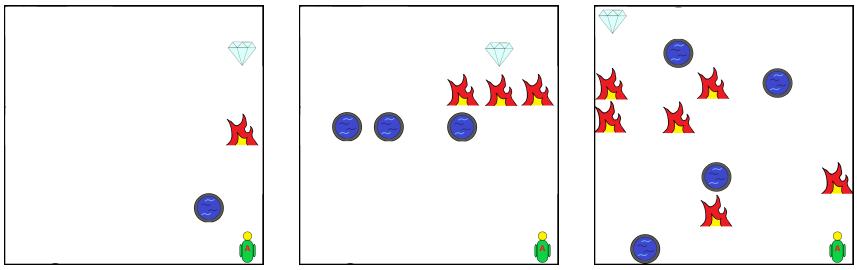
## 2 The curriculum learning problem

We start by describing the problem of learning one task through reinforcement learning, and then consider the case of learning a sequence of tasks forming a curriculum.

### 2.1 Reinforcement Learning

In order to describe the decision problem we are focusing on, we make use of a classic GridWorld as a running example. In the GridWorld domain, shown in Figure 1, we consider an agent that starts from a specific cell and has to find a treasure while moving over free cells. The agent aims to reach the treasure in the minimum number of steps, avoiding some obstacles represented by fires and pits. In Figure 1 (a) one fire, one pit, and the treasure are depicted in the $8\times8$ cells board.

The agent can observe, at any given time, the position of the treasure and the content of every cell that is at most 2 cells away from the agent (i.e., it sees everything in the $5\times5$ square centred at the agent). All possible values



(a) (b) (c)

**Fig. 1** Example of a GridWorld interface: (a) $task_1$, (b) $task_2$, (c) $task_F$.

for the position of the agent and the content of the cells surrounding it form the state space $S$. During an episode, the agent repeatedly chooses an action from the action set $A$. In GridWorld, $A$ contains four actions corresponding to moving in one of the four cardinal directions. An episode starts in an initial state $s_0 \in S$, in which the agent chooses an action $a_1 \in A$, resulting in the environment transitioning into the next state $s_1 \in S$. The agent receives a reward $r_1$ from a reward function $r : S \times A \times S \to \mathbb{R}$ and the cycle repeats from $s_1$ until a maximum number of steps $T$ is reached. Some states can be *absorbing*, that is, they cannot be left with any action, and from them the agent receives a reward of 0 until the end of the episode. In GridWorld, the treasure and the pits are absorbing states and they result in the termination of the episode, since no more reward can be collected from them. The behaviour of the agent is encoded in a function $\pi : S \to A$, returning an action for each state, called the *policy*. The policy is learned over a number of training episodes so as to maximize its expected *return* $E[G_\pi]$, that is, the expected cumulative discounted reward obtained by the agent while following the policy $\pi$: $G_\pi = \sum_{i=0}^{T} \gamma^{i-1} r_i$. The discount factor $0 \leq \gamma \leq 1$ encodes a preference for earlier rewards over later ones in the episode. In GridWorld, the reward the agent receives is $-2500$, $-500$ and $-250$ respectively for reaching a pit, a fire or for entering the cell next to a fire. Furthermore, the agent receives a reward of $+200$ for reaching the treasure, and -1 in all the other cases. Episodes end after 50 actions.

Reinforcement Learning (RL) has a number of algorithms widely used to improve the policy. A formal description of this training process can be found, e.g., in [21]. Here we just mention that a family of algorithms build a parametrized approximation $\hat{q}_\pi(s, a|\theta)$ of the *value function* $q_\pi(s, a) = E|_{s'}[r(s, a, s') + \gamma q(s', \pi(s'))]$, which represents the expected return for taking action $a$ in state $s$ and following the policy $\pi$ thereafter. The expectation is taken with respect to the next state $s'$. The value function is used to compute an improved policy, which in turn results in a higher value, in an optimization process called *policy iteration*, culminating in the optimal policy $\pi^*$ of value $q^*(s, a) = E|_{s'}[r(s, a, s') + \gamma max_{a'} q(s', a')$. A policy can be obtained from a value function by choosing greedily the action with the highest expected return in every state: $\pi_\theta(s) = argmax_a \ \hat{q}(s, a|\theta)$. The function representation

(for instance, a neural network) of $\hat{q}(s, a|\theta)$, and initial value of the parameter vector $\theta$ influence the speed and the point of convergence of the policy.

In the GridWorld domain we are considering, we represented the function $\hat{q}_\pi(s, a|\theta)$ as $|A|$ linear function approximators $\hat{q}_{\pi,a}(s|\theta) = \theta^T\phi(s)$. Different choices are possible for the vector $\phi$ of basis functions. In GridWorld, we used the Tile Coding basis [21], computed over the state vector $s \in S$, whose variables are described a the beginning of this section.

## 2.2 Curriculum Learning

Let us assume that we want the agent to learn the GridWord task shown in Figure 1 (c). We denote this environment as the final task, $task_F$. Figure 1 also contains two other easier GridWord environments, $task_1$ and $task_2$, which we can use to build up to $task_F$. We denote $\mathcal{T}$ the set of the $n$ available tasks except the final task. We assume that we have an episode budget to train the agent, by using an RL algorithm, given by $N$ (number episodes per task) times $L$ (number of tasks that can be used to train the agent before the final task).

By way of example, consider the three tasks in Figure 1, and assume $L = 2$. We have 5 different sequences for the RL training (including the execution of the final task at the end)

1. $task_F \rightarrow task_F \rightarrow task_F$
2. $task_1 \rightarrow task_F \rightarrow task_F$
3. $task_2 \rightarrow task_F \rightarrow task_F$
4. $task_1 \rightarrow task_2 \rightarrow task_F$
5. $task_2 \rightarrow task_1 \rightarrow task_F$.

The sequence of the tasks in $\mathcal{T}$ used to train the agent before the final task is called a *curriculum*. The first sequence consists in using the entire budget on the final task, and corresponds to learning without a curriculum. The other sequences are all possible curricula of length at most 2.

In a curriculum, we train the agent sequentially over the first $L$ tasks by using an RL algorithm. The agent knowledge is represented by the value function $\hat{q}_\pi(s, a|\theta)$, whose parameters are transferred from one task to the following one in the sequence, so that the final parameters at the end of one task initialize the value function at the beginning of the next one.

After the learning procedure over a curriculum $c$ of $L$ tasks, we obtain the value function parameter $\theta^0(c)$. The quality of curriculum $c$ can be measured by considering the total return on the $N$ episodes in the final task. Let $G^i_\theta$ be the return obtained by the agent starting from a value function parameter $\theta$ in episode $i$. The total return after $N$ episode is:

$$U(c) \triangleq \sum_{i=1}^{N} G^i_{\theta^i(c)}, \tag{1}$$

where $\theta^i(c)$ is the policy parameter obtained by the RL algorithm at the end of the $i$th episode of the final task.

Our objective is to find a curriculum $c^*$ that maximize this cumulative reward, i.e. an optimal solution of

$$\underset{c \in \mathcal{C}}{\text{maximize}} \quad U(c), \tag{2}$$

where $\mathcal{C}$ represents the set of all feasible curricula obtained from $\mathcal{T}$.

Problem (2) presents two main drawbacks: (i) it is in general nonsmooth, nonconvex, and even discontinuous with a black-box nature; (ii) it is a combinatorial optimization problem whose dimension sharply increases with respect to the number $n$ of the available tasks and the maximum length $L$ of the curricula. In this respect, note that the dimension of the problem grows according to the following formula

$$|\mathcal{C}| = \sum_{k=n-L}^{n} \frac{n!}{k!}. \tag{3}$$

Thus, the lack of a closed analytical form of the objective function does not allow us to resort to general methods for Mixed-Integer NonLinear Programs (MINLP), see e.g. [2]. Furthermore, the combinatorial nature of the problem and the large number of possible combinations make the application of standard Derivative Free methods unfeasible, see e.g. [6,7]. Some algorithms to find good solutions of problem (2) are discussed in [8]. However, those methods cannot be efficiently applied to large problems (where enumeration is not feasible) and constitute only a preliminary step to efficiently tackle the curriculum learning problem.

In the next section we introduce a new grey-box reformulation for problem (2) by modeling how learning each task, and the order with which these tasks are learnt, influence the final performance, and then by defining an optimization scheduling problem.

## 3 The grey-box reformulation

A curriculum $c = (m_0, \ldots, m_{L-1})$ is composed by at most $L \leq n$ tasks in the set $\mathcal{T}$ organized in a specific order and without repetitions. It represents the sequence of tasks on which the policy of the agent is optimized before addressing the final task $m_L$.

In order to define a feasible curriculum, let us introduce the binary variables $\delta \in \{0,1\}^n$ and $\gamma \in \{0,1\}^{n \times (n-1)}$ for any curriculum $c \in \mathcal{C}$. Each $\delta_i$ indicates whether the $i$th task of $\mathcal{T}$ is in the curriculum $c$ ($\delta_i = 1$) or not ($\delta_i = 0$). On the other hand, $\gamma_{ij}$, with $i \neq j$, models the order of the tasks in $c$: $\gamma_{ij} = 1$ if and only if both the $i$th and $j$th tasks of $\mathcal{T}$ are in the curriculum $c$ and the $i$th task is scheduled after the $j$th task. Moreover, we introduce integer variables $x \in [0, L-1]^n \cap \mathbb{N}^n$ indicating the order of the tasks in the curriculum $c$; in this way, let $\widehat{c} = (\widehat{m}_0, \ldots, \widehat{m}_{L-1})$ be the curriculum under consideration, then $\widehat{m}_{j-1} = task_i \in \mathcal{T}$ where $x_i$ is the $j$th smallest value among the $x_k$ with $k : \delta_k = 1$. Note that $(x, \delta)$ uniquely determines a curriculum $c$, while $\gamma$ can be derived by $(x, \delta)$. Therefore also $(\delta, \gamma)$ can be used to identify any curriculum.

Below we provide the reader with an example that clarifies how the definition of a tuple $(x, \delta)$ uniquely determines a curriculum $c$.

*Example 1* Consider the set of available tasks $\mathcal{T} = \{task_1, task_2, task_3, task_4, task_5\}$, the pair $(x, \delta)$ with $x = (0, 0, 1, 3, 2)$ and $\delta = (0, 0, 1, 1, 1)$ defines the curriculum $c = (m_0, m_1, m_2) = (task_3, task_5, task_4)$ while $\gamma$ could be derived by $(x, \delta)$. □

We are ready to define a feasible curriculum by introducing the following definition.

**Definition 1 (Feasible curriculum)** A curriculum $c$ defined by the variables $(x, \delta)$ is feasible, i.e. $c \in \mathcal{C}$, if $(x, \delta)$ satisfies:

$$x_i \leq (L-1)\delta_i \quad i = 1, \ldots, n \tag{4}$$

$$x_i \in [0, L-1] \cap \mathbb{N} \quad \delta_i \in \{0, 1\} \quad i = 1, ..., n \tag{5}$$

$$m(x, \delta) \geq 1 \tag{6}$$

$$M(x, \delta) \leq |\mathcal{A}_\delta| - 1 \tag{7}$$

where

$$\mathcal{A}_\delta \triangleq \{i \in \{1, ..., L\} : \delta_i = 1\}$$

$$m(x, \delta) \triangleq \min_{i,j \in \mathcal{A}_\delta} \quad |x_i - x_j|$$

$$M(x, \delta) \triangleq \max_{i,j \in \mathcal{A}_\delta} \quad |x_i - x_j|.$$

□

Namely, a curriculum $c$ is feasible if $(x, \delta)$ defines an ordered set of tasks, (4) and (5), which does not have two overlapping tasks, (6), and does not have gaps in the defined order, (7).

*Example 2* The pair $(\widetilde{x}, \widetilde{\delta})$ with $n = 5$, $L = 4$, $\widetilde{x} = [0, 0, 1, 2, 4]$ and $\widetilde{\delta} = [0, 0, 1, 1, 1]$ cannot be a feasible curriculum because $M(\widetilde{x}, \widetilde{\delta}) = 3 > |\mathcal{A}_{\widetilde{\delta}}| - 1 = 2$. Indeed there is a gap between 2 and 4. □

Now we introduce the following approximate linear function $\widehat{U}$ of the utility $U$ with the aim of reducing the complexity of Problem (2). The approximate model is more mathematically tractable, while providing a reasonable representation of the environment we are dealing with:

$$\widehat{U}(\delta, \gamma; u, p) \triangleq \sum_{i=1}^{n} u_i \delta_i - \sum_{i=1}^{n} \sum_{i \neq j = 1}^{n} p_{ij} \gamma_{ij}. \tag{8}$$

The approximate linear function $\widehat{U}$ is defined in order to describe how the curriculum $c$, defined by $(\delta, \gamma)$ (or equivalently by $(x, \delta)$), influences the performance over the final task. Specifically, with $\widehat{U}$ we consider the simpler case in which every task $m_i \in c$ contributes to the value of $U$ with a fixed individual

utility $u_i \geq 0$. Moreover, considering all pairs $(i,j) \in \{1,\dots,n\} \times \{1,\dots,n\}$ with $i \neq j$, if the $i$th task of $\mathcal{T}$ is in the curriculum $c$ and it is scheduled after the $j$th task of $\mathcal{T}$, then there is a penalty in $U$ equal to $p_{ij} > 0$. This concept of utilities and penalties is exploited to model how a task $m_i$ can be preparatory for another task $m_j$. If the policy is not optimized in the preparatory task $m_i$ before it is optimized in task $m_j$, then the utility given by task $m_j$ has to be reduced by the corresponding penalty $p_{ij}$. We observe that the function $\widehat{U}$ is linear with respect to $(\delta, \gamma)$ and parametric with respect to the utilities $u$ and penalties $p$. Note that the values of $u$ and $p$ are not known *a-priori* but need to be estimated through numerical methods as discussed in Section 4.

Given these utilities $u$ and penalties $p$, we maximize the utility $\widehat{U}$ from scheduling the tasks in a specific order (i.e. defining a curriculum) by modifying the indicator variables $\delta$ and $\gamma$ corresponding to feasible curricula in $\mathcal{C}$.

After having introduced the approximate function $\widehat{U}$ and defined the feasible curriculum set, we can define the Integer Linear Program (ILP) scheduling problem for curriculum learning as follows:

$$\begin{aligned}
\underset{x,\delta,\gamma}{\text{maximize}} \quad & \widehat{U}(\delta,\gamma;u,p) - \lambda \sum_{i=1}^{n} x_i && (9)\\
\text{subject to} \quad & x_i \leq (L-1)\delta_i, \quad i = 1,\dots,n && (9a)\\
& x_i - x_j + \delta_j \leq L\gamma_{ij} + (1-\delta_j)L, \quad i,j = 1,\dots,n \neq j && (9b)\\
& \gamma_{ij} + \gamma_{ji} \leq 1, \quad i,j = 1,\dots,n,\ i \neq j && (9c)\\
& x \in [0, L-1]^n \cap \mathbb{N}^n, \quad \delta \in \{0,1\}^n, \quad \gamma \in \{0,1\}^{n\times(n-1)}, && (9d)
\end{aligned}$$

where $\lambda$ is a small positive real number. The term $-\lambda \sum_{i=1}^{n} x_i$ is added in the objective function only with the aim of setting variables $x$ to correct values.

As we will show in the following propositions, Problem (9) owns some interesting properties which allow us to get feasible curricula when computing one of its solutions (see **Proposition 1**). Moreover, we have the guarantee that every optimal curriculum $\widehat{c}$ for Problem (2) can be obtained for some specific values of $(u,p)$ (see **Proposition 2**).

First of all, we prove that any optimal solution to Problem (9) defines a feasible curriculum $c \in \mathcal{C}$.

**Proposition 1** *Let $(\widehat{x}, \widehat{\delta}, \widehat{\gamma})$ be an optimal point of Problem (9) with $(u,p) \in \mathbb{R}^n \times \mathbb{R}^{n\times(n-1)}$. Then $(\widehat{x}, \widehat{\delta})$ satisfies conditions (4) - (7).*

*Proof* Conditions (4) and (5) are true because $(\widehat{x}, \widehat{\delta})$ satisfies (9a) and (9d) respectively.

Concerning condition (6), we cannot have $m(\widehat{x}, \widehat{\delta}) = 0$ because this would imply $\widehat{x}_i = \widehat{x}_j$ and $\widehat{\delta}_i = \widehat{\delta}_j = 1$, then by (9b) we would have $\gamma_{ij} = \gamma_{ji} = 1$ which violates condition (9c).

Finally, to prove condition (7), we assume by contradiction that

$$M(\widehat{x}, \widehat{\delta}) > |\mathcal{A}_{\widehat{\delta}}| - 1 \tag{10}$$

Thanks to the second term in the objective function, another solution $(\widetilde{x}, \widetilde{\delta}, \widetilde{\gamma})$ with $(\widetilde{\delta}, \widetilde{\gamma}) = (\widehat{\delta}, \widehat{\gamma})$ and $M(\widetilde{x}, \widetilde{\delta}) \leq |\mathcal{A}_{\widetilde{\delta}}| - 1$ could be found with a higher value of the objective function (since $\sum_{i=1}^{n} \widetilde{x}_i < \sum_{i=1}^{n} \widehat{x}_i$), proving that $(\widehat{x}, \widehat{\delta})$ was not optimal. □

Note that, given the tuple $(\widehat{x}, \widehat{\delta})$ satisfying conditions (4) - (7), there exists a unique $\widehat{\gamma}$ such that $(\widehat{x}, \widehat{\delta}, \widehat{\gamma})$ is feasible for Problem (9). That is, given the tuple $(\widehat{x}, \widehat{\delta})$ defining the curriculum $\widehat{c}$, then the corresponding $\widehat{\gamma}$ is always univocally determined.

Finally, we prove that any feasible curriculum can be computed by solving Problem (9) for some $(u, p)$.

**Proposition 2** *Let $(\widehat{x}, \widehat{\delta})$ satisfy conditions (4) - (7). Then, there exist $(\widehat{u}, \widehat{p})$ and $\widehat{\gamma}$ such that $(\widehat{x}, \widehat{\delta}, \widehat{\gamma})$ is an optimal solution to Problem (9) with $(u, p) = (\widehat{u}, \widehat{p})$.*

*Proof* As already discussed, there exists a unique $\widehat{\gamma}$ such that $(\widehat{x}, \widehat{\delta}, \widehat{\gamma})$ is feasible for Problem (9). We observe that $(\widehat{x}, \widehat{\delta}, \widehat{\gamma})$ is a solution to Problem (9) with the following $(\widehat{u}, \widehat{p})$:

$$\widehat{u}_i = \begin{cases} 0 & \text{if } \widehat{\delta}_i = 0 \\ 2\epsilon n + \lambda L & \text{otherwise} \end{cases}, \qquad \widehat{p}_{ij} = \begin{cases} \epsilon & \text{if } \widehat{\gamma}_{ij} = 1 \\ 2\epsilon & \text{otherwise} \end{cases},$$

with $\epsilon > 0$. In this way, tasks not in $\mathcal{A}_{\widehat{\delta}}$ cannot be chosen at the optimum, because their utility is zero while their penalty is strictly positive, so that choosing them would imply a reduction in the objective function. On the other hand, tasks in $\mathcal{A}_{\widehat{\delta}}$ are chosen because their utility is always greater than the maximum reduction in the objective function achieved when performing that task and their order is determined by $\widehat{p}_{ij}$. □

*Example 3* Consider the case $\widehat{c} = \emptyset$, namely $\mathcal{A}_{\widehat{\delta}} = \emptyset$. In this case, $\widehat{u} = 0$ and $\widehat{p} = \epsilon$, so an optimal solution to Problem (9) is $(\widehat{x}, \widehat{\delta}, \widehat{\gamma}) = (0, 0, 0)$ which corresponds to the empty curriculum we were looking for. □

Thanks to **Proposition 1** and **Proposition 2** we are able to reformulate Problem (2) as a grey-box function $\Psi : \mathbb{R}^{n \times n} \to \mathbb{R}$, which takes the parameters $(u, p)$, computes a curriculum $c$ by solving Problem (9) and returns the cumulative reward $U(c)$. By using the grey-box function $\Psi$, Problem (2) can be equivalently reformulated as

$$\underset{(u,p) \in \mathbb{R}_+^n \times \mathbb{R}_+^{n \times (n-1)}}{\text{maximize}} \quad U(u, p). \tag{11}$$

Note that **Proposition 2** ensures that a tuple $(\widehat{u}, \widehat{p})$ exists such that the optimal solution to Problem (9) defines the curriculum $\widehat{c}$, which in turn is an optimal solution to Problem (2). In the next section we discuss the problem of computing a good pair $(\widehat{u}, \widehat{p})$ in order to get a curriculum $c$ that maximizes the overall performance $U(c)$.

We conclude this section with a technical result. We can characterize the optimal solutions by observing that if, at the optimum, some tasks are not chosen, then all chosen tasks have a value of $x_i$ not less than 1. In particular, the following proposition holds:

**Proposition 3** *Let $(\widehat{x}, \widehat{\delta}, \widehat{\gamma})$ be an optimal point of Problem (9). If $|\mathcal{A}_{\widehat{\delta}}| \neq L$ then $\widehat{x}_i \geq 1$ for all $i \in \mathcal{A}_{\widehat{\delta}}$.*

*Proof* $|\mathcal{A}_\delta| \neq L$ implies that exists $j$ such that $\delta_j = 0$. Let $(\widehat{x}, \widehat{\delta}, \widehat{\gamma})$ be a feasible solution with two indices $i, j$ such that $\widehat{x}_j = 0$, $\widehat{\delta}_j = 0$ and $\widehat{x}_i = 0$, $\widehat{\delta}_i = 1$, $i \neq j$. By contradiction we can prove that $(\widehat{x}, \widehat{\delta}, \widehat{\gamma})$ is not optimal. If $(\widehat{x}, \widehat{\delta}, \widehat{\gamma})$ was optimal then by constraint (9b) we would have $\widehat{\gamma}_{ji} = 1$. However, we can find another solution $(\widetilde{x}, \widetilde{\delta}, \widetilde{\gamma}) = (\widehat{x}, \widehat{\delta}, \widehat{\gamma})$ except for $\widetilde{x}_i \geq 1$ with $\widetilde{\gamma}_{ji} = 0$, that is feasible but with a greater objective function. Therefore, $(\widehat{x}, \widehat{\delta}, \widehat{\gamma})$ cannot be optimal. □

## 4 Numerical methods for estimating $(u, p)$

Different approaches can be applied in order to get good estimates of $(u, p)$ so that the grey-box function $\Psi$ is maximized and the curriculum learning problem is solved efficiently. We report three of them below.

- Since Problem (11) can be regarded as a black-box optimization problem, we can resort to many DF algorithms in order to compute (approximate) optimal points $(\widehat{u}, \widehat{p})$ of (11). We can rely on Sequential Model-Based Optimization (SMBO) methods, which define a probabilistic model of the objective function and use the information gained during the iterations to update the probabilistic model following a Bayesian approach. At each iteration, the algorithm picks a new promising point by maximizing an acquisition function, and uses the information gained with this new sample to update the surrogate model. See [10,19,20] and references therein for further details.
- An heuristic procedure can be implemented to compute a good estimate $(\overline{u}, \overline{p})$ and then it can be used to get the value of the objective function, $\Psi(\overline{u}, \overline{p})$.
- We can leverage the heuristic procedure described above to find a promising region of confidence centered in $(\overline{u}, \overline{p})$. Thus we can further constraint the black-box optimization problem to be within this confidence region and resort to DF algorithms which allows us to define a probability distribution on the parameters $(u, p)$ to optimize and solve the problem as a sequential model-based algorithm. This approach can be seen as a local-search method to improve the estimate obtained by the heuristic method. In the rest of this article we will consider Tree-structured Parzen Estimators (TPEs) [5] as an example of a DF method implementing this approach.

Computing a good estimate for $(\overline{u}, \overline{p})$ can be critical for obtaining a good numerical performance. Here we propose a method that is justified by the way the approximate function $\widehat{U}$ is defined in (8). In this regard, in the simpler case in which the real utility function $U$ is equivalent to the approximate function $\widehat{U}$ (discussed in the previous section), we would have for any $(i, j)$ with $i \neq j$:

$$U(m_i, m_j) = \overline{u}_i + \overline{u}_j - \sum_{k=1,\, k\neq i}^{n} \overline{p}_{ik} - \sum_{k=1,\, j\neq k\neq i}^{n} \overline{p}_{jk} + \overline{U},$$

$$U(m_i) = \overline{u}_i - \sum_{k=1,\, k\neq i}^{n} \overline{p}_{ik} + \overline{U}, \qquad U(m_j) = \overline{u}_j - \sum_{k=1,\, k\neq j}^{n} \overline{p}_{jk} + \overline{U},$$

where $\overline{U}$ is an unknown constant. This implies

$$\overline{p}_{ji} = U(m_i, m_j) - U(m_i) - U(m_j) + \overline{U}, \tag{12}$$

$$\begin{aligned} \overline{u}_i &= U(m_i) + \sum_{k=1,\, k\neq i}^{n} \overline{p}_{ik} - \overline{U} \\ &= U(m_i) + \sum_{k=1,\, k\neq i}^{n} (U(m_k, m_i) - U(m_k) - U(m_i)) + (n-2)\overline{U}. \end{aligned} \tag{13}$$

We observe that computing this estimate requires $n(n+1)$ evaluations of $U$. However, these evaluations can be run in parallel over several processors and each of them considers at most two tasks so that the computation time per evaluation is usually quite small.

In the following section we test the effectiveness of the proposed methodologies on a benchmark environment from the curriculum learning community.

## 5 Experimental evaluation

We base our numerical experiments over the GridWorld domain, reinforcement learning setting commonly used in the evaluation of curriculum learning methods, see e.g. [22]. The dimension of each task is similar to the one shown in Figure 1 but with a variable number of fires and pits so to easily control the complexity of the environment.

We analyze different optimization techniques to solve the curriculum learning problem. In particular, we compare five different methods:

- **$C_0$:** the agent is trained with the empty curriculum. This solution is considered as a benchmark to understand to what extent the application of the curriculum learning procedure can improve the performance over the final task.
- **Heuristic:** formulas (12) and (13) are implemented to obtain a good estimate for $(\overline{u}, \overline{p})$, with $\overline{U}$ computed so that $\min_{(i,j)} \overline{p}_{ij} \geq 0$ and $\min_i \overline{u}_i \geq 10 \max_{(i,j)} \overline{p}_{ij}$. Then the scheduling problem (9) is solved to determine the tentative curriculum $c$.

- **GREEDY Par:** the curriculum is incrementally defined by the application of a greedy heuristic as proposed in [8]. In particular, at each iteration the $n$ tasks which mostly improve the final performance are considered and the one which leads to the best improvement in performance is taken. Note that this method does not leverage the grey-box reformulation but it aims at computing a good feasible point of problem (2). This method can be considered as the state of the art and is used as a benchmark to understand if the proposed methodology leads to any improvement.
- **GP:** a SMBO approach is implemented to tackle Problem (11). In particular, the optimization problem is modeled as a Gaussian Process where new points are drawn by maximizing the Expected Improvement as the acquisition function. The method searches for the best values $(\overline{u}, \overline{p})$ on the box $[0, 1000]^n \times [0, 100]^{n \times (n-1)}$ since no *a priori* knowledge is incorporated, but 10 starting points are sampled to build the a priori distribution.
- **TPE:** a Tree-structured Parzen Estimator is used as surrogate model of Problem (11). This method is employed as a local-search technique to improve the estimates $(\overline{u}, \overline{p})$ obtained with the heuristic approach. In particular, a Gaussian distribution of $(u, p)$ is defined that is centered in the values $(\overline{u}, \overline{p})$ and with variance proportional to the variance within the values $(\overline{u}, \overline{p})$.

The proposed framework is implemented in Python 3.6 on a Intel(R) Core(TM) i7-3630QM CPU 2.4GHz. by means of the following libraries:

docplex (v 2.8.125): the CPLEX's API [11] used for solving the ILP (9) with the version 12.9 of CPLEX. We set the running time to 60 seconds per iteration and the mipgap to $10^{-2}$.

GPyOpt (v 1.2.5): used for the implementation of the GP method described above [1,18].

hyperopt (v 0.2): used for the implementation of the aforementioned TPE approach [3–5].

Burlap: used to model the GridWord domain along with the Sarsa($\lambda$) code as learning algorithm to update the policy and Tile Coding as the function approximator (*http://burlap.cs.brown.edu*).

We consider three different experiments on the GridWorld domain. In the first two experiments, the total number of feasible curricula is such that solving the curriculum learning problem by enumeration is still feasible even if computationally expensive. In the third problem, $|\mathcal{C}|$ increases so that enumeration strategies are no longer feasible. While the first two experiments are discussed together, the third one is treated separately since its analysis is done in a different manner.

In the first example, $n = 12$ potential tasks are defined and the maximum length of the curriculum is set to $L = 4$, obtaining 13345 feasible curricula according to Formula (3). In the second case, $n = 7$ tasks are defined and all of them can be considered together in a single curriculum, i.e. $L = 7$, for a total of 13700 possible combinations of tasks. For the former example we set $N = 300$ while for the latter we set $N = 400$ number of episodes per task.

See [8] for further details about the problem settings. Algorithm $C_0$ and the Heuristic require respectively 1 and $n(n+1)$ curriculum evaluation, while the number of curriculum evaluations granted to the other algorithms is set to 300.

In Table 1 for each method we report:

**Gain**: the gain on the cumulative reward with respect to the benchmark performance obtained with $C_0$;

**Rank**: the ranking of the returned solution with respect to all the possible curricula;

**#Curr***: the number of curriculum evaluations needed before the best performance, computed by the algorithm, was achieved without considering the number of curricula evaluated to get the starting point.

**Table 1** Results obtained on the GridWorld domain problem ($c^*$ indicates any curriculum that solves Problem (2)).

| | $n=12, L=4, \lvert\mathcal{C}\rvert=13345$ | | | $n=7, L=7, \lvert\mathcal{C}\rvert=13700$ | | |
|---|---|---|---|---|---|---|
| **Method** | **Gain** | **Rank** | **#Curr*** | **Gain** | **Rank** | **#Curr*** |
| $C_0$ | 1.000 | 11499 | 1 | 1.000 | 4535 | 1 |
| GREEDY Par | 1.215 | 144 | 373 | 1.210 | 260 | 155 |
| Heuristic | 1.217 | 121 | 144 | 1.181 | 417 | 49 |
| GP | 1.234 | 32 | 17 | 1.289 | 38 | 19 |
| TPE | 1.256 | 4 | 84 | 1.326 | 14 | 162 |
| $c^*$ | 1.275 | 1 | - | 1.430 | 1 | - |

Our numerical results show how the grey-box reformulation and the numerical methods we introduced can improve the performance over the final task if compared to the performance obtained when training the agent directly on the final task (algorithm $C_0$). As a proof of the effectiveness of the proposed methods, we highlight how even the simple heuristic procedure is always able to find better solutions than $C_0$ with improvements ranging from 18.1% to 43.0% and similar solutions to those returned by GREEDY Par. Moreover, the application of a Bayesian optimizer to estimate the utilities $u$ and penalties $p$ seems to be a successful choice in order to further improve the final performance. Finally, the local search performed by TPE around the tentative point $(\overline{u}, \overline{p})$ leads to a remarkable improvement of the final performance by finding, in both scenarios, one of the 15th best solutions out of the more than 13000 possible curricula.

In order to further assess the effectiveness of the introduced grey-box approach, in the third experiment we consider a far bigger setting, also based on the GridWorld domain, where the number of potential tasks, $n$, and the maximum number of tasks used for building each curriculum, $L$, are both equal to 15. In this setting the overall number of potential curricula is bigger than 3.5 trillion so that an enumeration strategy in order to compute an curriculum $c^*$ solution to Problem (2) is not feasible. In this setting we compare the solution returned by GREEDY Par, GP and TPE. The three algorithms are left running for 24 hours once the starting point has been found. In Figure 2 we plot

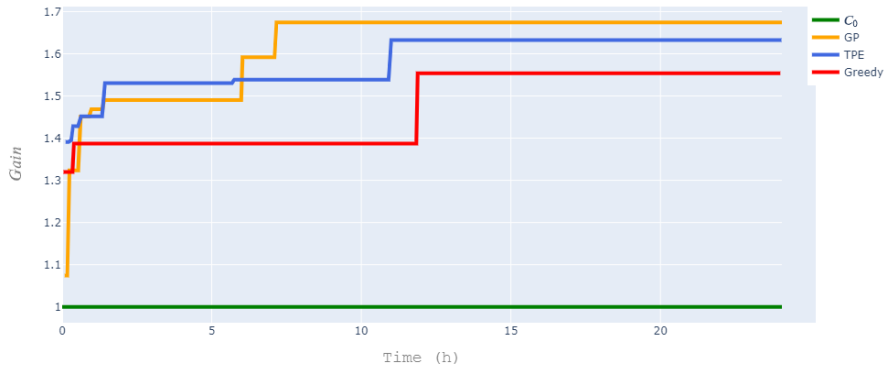



**Fig. 2** Performance curve: gain in performance obtained by GREEDY par, GP and TPE with respect to $C_0$.

the best values found by the three methods over the iterations and reported as the gain with respect to the solution returned when directly training over the empty curriculum $C_0$. All the methods outperform the solution returned by $C_0$. Both GP and TPE performs better than GREEDY par. Running GP let us achieve a gain near of 70% in around 7 hours, while TPE a gain bigger than 60% in 11 hours of computations. We finally remark how the long computational time needed to achieve these results is neglectable if compared to the gain we were able to achieve from a performance perspective.

## 6 Conclusions

We considered the problem of curriculum learning and propose a grey-box approach to solve the difficult combinatorial decision problem behind it. A parametric model that describes the relation among the preparatory tasks and the final task to be learnt was introduced, and an ILP scheduling problem was formulated to obtain tentative optimal curricula. We show how the introduced scheduling problem is well-defined: it is always able to return a feasible curriculum, and with the property that any feasible curriculum can be obtained by setting in the scheduling formulation the values defining tasks' relations. Then, numerical methods were defined to determine the parameters governing the relations among tasks, and to solve the underlying scheduling problem. Experimental results demonstrate the effectiveness of the proposed reformulation in modeling relations between tasks and improving the performance on the final task, with model-based methods outperforming the previous approaches suggested so far.

## References


1. Gpyopt: A bayesian optimization framework in python. http://github.com/SheffieldML/GPyOpt (2016)
2. Belotti, P., Kirches, C., Leyffer, S., Linderoth, J., Luedtke, J., Mahajan, A.: Mixed-integer nonlinear optimization. Acta Numer. **22**, 1–131 (2013)
3. Bergstra, J.: Hyperopt: Distributed asynchronous hyperparameter optimization in python (2013)
4. Bergstra, J., Yamins, D., Cox, D.D.: Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures (2013)
5. Bergstra, J.S., Bardenet, R., Bengio, Y., Kégl, B.: Algorithms for hyper-parameter optimization. In: Advances in neural information processing systems, pp. 2546–2554 (2011)
6. Custódio, A.L., Scheinberg, K., Nunes Vicente, L.: Methodologies and software for derivative-free optimization. Advances and Trends in Optimization with Engineering Applications pp. 495–506 (2017)
7. Di Pillo, G., Liuzzi, G., Lucidi, S., Piccialli, V., Rinaldi, F.: A DIRECT-type approach for derivative-free constrained global optimization. Computational Optimization and Applications **65**(2), 361–397 (2016)
8. Foglino, F., Christakou, C.C., Leonetti, M.: An optimization framework for task sequencing in curriculum learning. In: 2019 Joint IEEE 9th International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob), pp. 207–214. IEEE (2019)
9. Foglino, F., Leonetti, M., Sagratella, S., Seccia, R.: A gray-box approach for curriculum learning. In: World Congress on Global Optimization, pp. 720–729. Springer (2019)
10. Frazier, P.I.: A tutorial on bayesian optimization. arXiv preprint arXiv:1807.02811 (2018)
11. IBM: IBM Decision Optimization (2019). URL http://ibmdecisionoptimization.github.io/docplex-doc/mp/refman.html
12. Leonetti, M., Kormushev, P., Sagratella, S.: Combining local and global direct derivative-free optimization for reinforcement learning. Cybernetics and Information Technologies **12**(3), 53–65 (2012)
13. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529 (2015)
14. Narvekar, S., Peng, B., Leonetti, M., Sinapov, J., Taylor, M.E., Stone, P.: Curriculum learning for reinforcement learning domains: A framework and survey. Journal of Machine Learning Research **21**(181), 1–50 (2020). URL http://jmlr.org/papers/v21/20-212.html
15. Narvekar, S., Sinapov, J., Leonetti, M., Stone, P.: Source task creation for curriculum learning. In: Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems, pp. 566–574. International Foundation for Autonomous Agents and Multiagent Systems (2016)
16. Narvekar, S., Sinapov, J., Stone, P.: Autonomous task sequencing for customized curriculum design in reinforcement learning. In: IJCAI, pp. 2536–2542 (2017)
17. Peng, B., MacGlashan, J., Loftin, R., Littman, M.L., Roberts, D.L., Taylor, M.E.: An empirical study of non-expert curriculum design for machine learners. In: In Proceedings of the IJCAI Interactive Machine Learning Workshop (2016)
18. Rasmussen, C.E.: Gaussian processes in machine learning. In: Advanced lectures on machine learning, pp. 63–71. Springer (2004)
19. Shahriari, B., Swersky, K., Wang, Z., Adams, R.P., De Freitas, N.: Taking the human out of the loop: A review of bayesian optimization. Proceedings of the IEEE **104**(1), 148–175 (2016)
20. Snoek, J., Larochelle, H., Adams, R.P.: Practical bayesian optimization of machine learning algorithms. In: Advances in neural information processing systems, pp. 2951–2959 (2012)
21. Sutton Richard S. Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press (2018)
22. Svetlik, M., Leonetti, M., Sinapov, J., Shah, R., Walker, N., Stone, P.: Automatic curriculum graph generation for reinforcement learning agents. In: AAAI, pp. 2590–2596 (2017)