Sample-efficient learning of soft priorities for safe control with constrained Bayesian optimization

Jian Li[1,2], Yiyao Zhu[1,2], Langcheng Huo[1,2], Yongquan Chen[1,2*]

***Abstract*—Exploiting the redundancy of the robot, a complex motion can be achieved by executing multiple tasks simultaneously, where the key is tuning the task priorities. Generally, the task priorities are predefined manually. In order to generate the task priorities automatically, different frameworks are proposed to address this problem. In this paper, we employ a black-box optimization method to learn the soft tasks priorities, guaranteeing that the robot motion is optimized with high efficiency and no constraints violations occur. We compare the performance of several variants of Bayesian optimization with respect to fitness value and constraints violations, and finally we retain the one with Gaussian Process with multiple outputs, which is sample-efficient and leads to no safety constraints violations, as the optimal solution to address the problem.**

I. INTRODUCTION

A redundant manipulator or humanoid is able to execute multiple tasks simultaneously to achieve a complex motion, while keeping the safety constraints satisfied. This is usually accomplished by a prioritized multi-task controller [1], where task priorities are introduced to represent the relative importance between different tasks [2].

A set of tasks can be organized hierarchically in the light of strict priorities [3]. However, in many cases, it is difficult to define the strict priorities a priori. Additionally, when incompatibilities occur between different tasks, a task with higher priority might totally block the one with lower priority [4], which usually generates undesired robot movement. Moreover, the control output might become discontinue at some point due to the change of task priorities [5]. Another way to organize the tasks is to assign each task with a weighting function [6], i.e. soft priority which changes the task's relative importance with respect to time in a continuous way. In some cases, conflict tasks also can cause incompatibilities. Both manners can be formulated as a quadratic programming problem, where safety constraints are considered [7] [8].

Strict priorities are usually predefined empirically by experts. Though there are some proposals employ new frameworks to achieve smooth transitions [2], [9], e.g. Generalized Hierarchical control (GHC), it still needs plenty of manual tuning. Likewise, soft task priorities require manual tuning as well.

This paper is partially supported by Shenzhen Fundamental Research grant (JCYJ20180508162406177) and the National Natural Science Foundation of China (U1613216) from The Chinese University of Hong Kong, Shenzhen. This paper is also partially supported by funding from Shenzhen Institute of Artificial Intelligence and Robotics for Society.

[1] Shenzhen Institute of Artificial Intelligence and Robotics for Society, China.

[2] Institute of Robotics and Intelligent Manufacturing, The Chinese University of Hong Kong, Shenzhen, China.

[*] Corresponding author. email: yqchen@cuhk.edu.cn

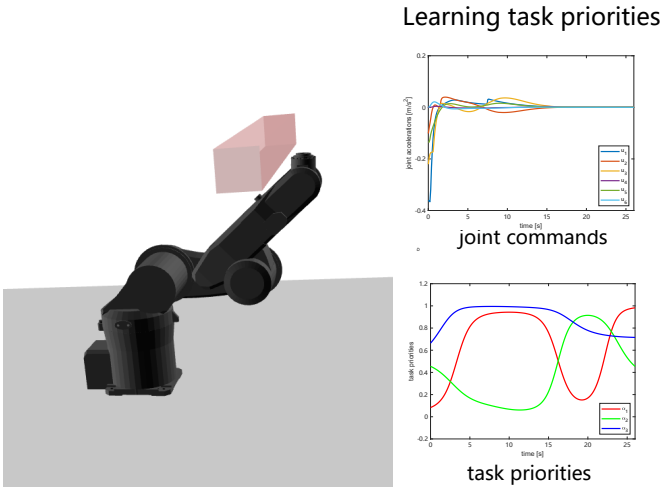


Fig. 1. A KUKA KR5 robot arm reached a desired point behind a cuboid with the its end-effector by automatically learning the profiles of the task priorities using a constrained Bayesian optimization method, without violating any safety constraints.

Therefore, the focus of research shifts to tuning the soft task priorities automatically [10], [11], [12]. However, existing solutions are either inefficient or performing poorly in constraints satisfaction. For instance, in [12], an evolutionary strategy, i.e. Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) is used to optimize the profile of task priorities, where constraints are formulated as a penalty item added to the fitness function. This is essentially a relaxation to constraints, not guaranteeing the satisfaction of constraints during the learning process. In [13], constraints are taken into account explicitly by (1+1)-CMA-ES, a modified version of [12], which addresses the problem of constraints satisfaction during learning process well. However, both CMA-ES and (1+1)-CMA-ES are inefficient and time consuming, requiring hundreds of iterations, since an evolutionary algorithm needs a substantially large population size to search for. In [14], a sample-efficient algorithm, Bayesian optimization (BO), was employed to solve the task priorities optimization problem, requiring much fewer iterations than evolutionary strategy, but constraints are not taken into account in this optimization algorithm.

Guaranteeing that the optimization process is efficient and constraints relative to safety are always satisfied during the whole learning process, is indispensable to apply these optimization methods to real robots.

To address the issues stated above, we research on constrained Bayesian optimization, typically we mainly focus

on three variants of Bayesian optimization, one with fitness function with penalty item, one with constrained acquisition function [15], and one with Gaussian Process (GP) with multiple outputs [16]. We aim to find a variant of Bayesian optimization that retains the satisfying sample-efficient feature of the normal one and guarantees no constraints violations during the optimization process.

The contribution of this paper lies in that we found a desirable method to automatically optimize the time profiles of the task priorities, which is sample-efficient and always ensures constraints satisfaction. We demonstrate the validity of the selected method by running experiments on a simulated KUKA KR5 robot arm.

The rest of this paper is organized as follows: in Section II we demonstrate the optimization framework. Then in Section III we introduce the variants of Bayesian optimization. The comparison results and effectiveness validation are shown in Section IV and finally we draw conclusions in Section V.

II. PROBLEM FORMULATION

In this section, we first introduce a single task controller, then superimpose their weighted sum to derive a multi-task controller. In order to make this problem tractable, the soft priorities were modeled by a Radial Basis Function Network (RBFN). Finally we formulate the problem of learning time profiles of task priorities, which minimizes the given fitness while the safety constraints are always satisfied during the whole learning process.

A. Controller for a single elementary task

We use a task space controller [17] for the i-th elementary task with an analytical solution. The desired error dynamics is:

$$\ddot{e} = -k_d \dot{e} - k_p e \tag{1}$$

where $\boldsymbol{e}$, $\dot{\boldsymbol{e}}$, $\ddot{\boldsymbol{e}} \in \mathbb{R}^n$ represent the task error, its derivative and second derivative respectively; and k_d and k_p are, respectively, derivative and proportional gains. Definition of e are given as follows:

$$e = f_i - f_i^* \tag{2}$$

$$\dot{e} = \dot{f}_i - \dot{f}_i^* = \frac{\partial f_i}{\partial \boldsymbol{q_i}} \dot{\boldsymbol{q}}_{\boldsymbol{i}} - \dot{f}_i^* = J_{f_i} \dot{\boldsymbol{q}}_{\boldsymbol{i}} - \dot{f}_i^* \tag{3}$$

$$\ddot{\boldsymbol{e}} = \dot{J}_{f_i} \dot{\boldsymbol{q}}_{\boldsymbol{i}} + J_{f_i} \ddot{\boldsymbol{q}}_{\boldsymbol{i}} - \ddot{f}_i^* \tag{4}$$

where $\boldsymbol{q}_i$, $\dot{\boldsymbol{q}}_i$ and $\ddot{\boldsymbol{q}}_i \in \mathbb{R}^n$ denote joint positions, velocities and accelerations respectively; J_{f_i} and $\dot{J}_{f_i}$ are Jacobian and its derivative respectively; f_i and $\dot{f}_i$ are the task and its derivative respectively; ${f_i}^*$ and $\dot{f}_i^*$ are the task objective and its derivative respectively. Here we suppose $\ddot{f}_i^* = 0$. For accomplishing the desired error dynamics (1), we minimize

$$\|\boldsymbol{e}_{f_i}\|_2 = \|\ddot{\boldsymbol{e}} + K_d \dot{\boldsymbol{e}} + K_p \boldsymbol{e}\|_2 = \left\| J_{f_i} \ddot{\boldsymbol{q}} + C_{e_{f_i}} \right\|_2 \tag{5}$$

with

$$C_{e_{f_i}} = \left(\dot{J}_{f_i} + K_d J_{f_i} \right) \dot{\boldsymbol{q}} - K_d \dot{f}_i^* + K_p e - \ddot{f}_i^*$$

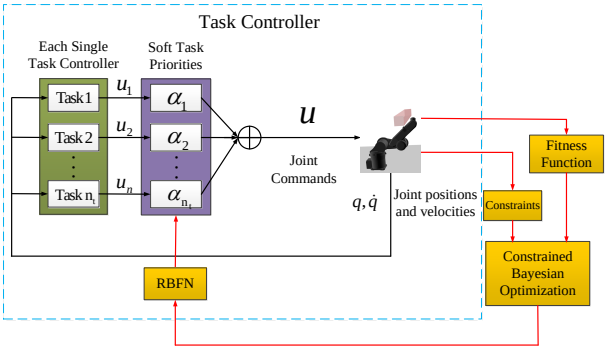


Fig. 2. The framework employed to learning the task priorities automatically. A multi-task controller is formed by superimposing the output of each single task controller assigned with a soft task priority. The time profiles of the task priorities is modeled by a RBFN. A black-box optimization method is used to optimized the parameters of the RBFN, considering the constraints satisfaction.

Then the controller essentially is an optimization problem:

$$\arg\min_{\ddot{\boldsymbol{q}}_i} \|e_{f_i}\|_2^2 = \arg\min_{\ddot{\boldsymbol{q}}_i} \left\| J_{f_i} \ddot{\boldsymbol{q}}_i + C_{e_{f_i}} \right\|_2^2 \tag{6}$$

By employing least-squares, we can derive an analytical solution of the controller:

$$\boldsymbol{u}_i = \ddot{\boldsymbol{q}}_i = -({J_{f_i}}^T J_{f_i})^{-1} {J_{f_i}}^T C_{e_{f_i}} \tag{7}$$

where u_i is joint command.

B. Controller for multiple elementary tasks with soft priorities

In order to achieve a global task through multiple elementary tasks, we assign a time-dependent task weight or priority $\alpha_i(t)$ to each elementary task. Then they can be superimposed to get the final joint command as in [18]. The i-th controller is described as u_i. Therefore the multi-task controller is formulated as:

$$\boldsymbol{u} = \sum_{i=1}^{n_t} \alpha_i(t)\, \boldsymbol{u}_i, \tag{8}$$

where t is time, and n_t is the number of tasks. Due to the infeasibility of finding the optimal functions $\hat{\alpha}_i^*(t)$, we transform this functional optimization problem into a numerical optimization problem, then the task priorities can be represented as parameterized functional approximators, $\alpha_i(t) \to \alpha_i(\boldsymbol{\pi}_i, t)$, where $\boldsymbol{\pi}_i$ is a set of parameters which determine the time profile of the i-th task priorities or weight function. Then the multi-task controller is reformulated as:

$$\boldsymbol{u} = \sum_{i=1}^{n_t} \alpha_i(\boldsymbol{\pi}_i, t)\, \boldsymbol{u}_i. \tag{9}$$

C. Normalized RBFN

For purpose of turning the functional optimization problem into a numerical optimization problem, we model the i-th task priority by a radial basis function network (RBFN):

$$\alpha_i(\boldsymbol{\pi}_i, t) = \text{Sigmoid}\left(\frac{\sum_{k=1}^{n_r} \pi_{ik}\Psi_k(\mu_k, \sigma_k, t)}{\sum_{k=1}^{n_r}\Psi_k(\mu_k, \sigma_k, t)}\right), \quad (10)$$

where n_r is the number of radial basis functions we used and $\boldsymbol{\pi_i} = (\pi_{i1}, ..., \pi_{in_r})$ is the set of parameters of the i-th task priority. And the sigmoid function is used to squash the output to range $[0, 1]$. 0 means that the corresponding task is deactivated and 1 means that it's fully activated. Here we define the radial basis function as:

$$\Psi_k(\mu_k, \sigma_k, t) = \exp\left(-\frac{1}{2}\left(\frac{t-\mu_k}{\sigma_k}\right)^2\right), \quad (11)$$

where μ_k and σ_k are mean and variance respectively. Means and variances are set to be fixed for reducing the learning parameters and improving the convergence rate.

D. Learning the soft task priorities

During the whole learning process, in order to find the optimal parameters $\boldsymbol{\pi}$* while guarantee safety, fitness function and constraints are introduced.

Fitness function $\boldsymbol{\phi} = \boldsymbol{\phi}(\boldsymbol{q}_{t=1,...,T}, \boldsymbol{u}_{t=1,...,T})$ evaluates the performance of the global motion with the present parameters $\boldsymbol{\pi}$ over T time steps. It can compromise energy consumption term or total error of the desired trajectory.

In most situation constraints are concerned with the safety of the robot, $e.g.$ joint angle, velocity, acceleration limits, collision with obstacle, which are inequalities. Thus we only consider inequality constraints here.

Then, the numerical optimization problem can be written as

$$\begin{aligned} \boldsymbol{\pi}^* &= \arg\min_{\boldsymbol{\pi}} \ \boldsymbol{\phi}(\boldsymbol{q}_t, \boldsymbol{u}_t) \\ s.t \ \ & g_i(\boldsymbol{q}_t, \boldsymbol{u}_t) \leq 0, \quad \forall i \in I_{ie} \end{aligned}, \quad (12)$$

where I_{ie} represents the set of inequality constraints.

To solve the problem stated above, we need to find a suitable learning method. Generally, there no explicit mapping between $\boldsymbol{\pi}$ and $\boldsymbol{\phi}$, so a derivative-free learning method is more general and desired. Additionally, fast computation is possible in trial-and-error learning according to recent research.

In next section, several sample-efficient methods will be proposed to minimize the fitness function $\boldsymbol{\phi}$ while guarantee that the constraints will be always satisfied during the whole learning procedure.

III. PROPOSED METHOD

In this section, we first review Bayesian optimization, which employs Gaussian Process (GP) as the surrogate model and *Expected improvement* (EI) as the acquisition function. Then we introduce three variants of Bayesian optimization: one with fitness function with penalty item, one with constrained acquisition function, and one with Gaussian Process (GP) with multiple outputs.

A. BO with no constraints

Surrogate model aims to model the performance function $f(\boldsymbol{\pi})$ in a probabilistic manner over its domain $\boldsymbol{\Pi}$, e.g. the objective function $\boldsymbol{\phi}$ in (12), making use of previous evaluation. Here GP is chosen as the surrogate model and defined by a mean function $m(\boldsymbol{\pi})$ and a covariance function $k(\boldsymbol{\pi}_i, \boldsymbol{\pi}_j)$, which represents the covariance between any two function values $f(\boldsymbol{\pi}_i)$ and $f(\boldsymbol{\pi}_j)$, where $i, j \in \mathbb{N}$. The covariance function is also known as kernel. The selection of the kernel function will be discussed in the experiment section since it is problem-dependent and makes assumptions about rate of change and smoothness of the objective function.

The observation set is defined by n past observations, i.e. $\mathcal{D}_n = \{\boldsymbol{\pi}_{1:n}, \hat{f}_{1:n}\}$. Since there is noise in a real system, the true function value should be $\hat{f}(\boldsymbol{\pi}) = f(\boldsymbol{\pi}) + \omega$ with $\omega \sim \mathcal{N}(0, \sigma_\omega^2)$. Based on observation set, the covariance and mean of desired point are given by

$$\begin{aligned} \sigma_n^2(\boldsymbol{\pi}^*) &= k(\boldsymbol{\pi}^*, \boldsymbol{\pi}^*) - \mathbf{k}_n(\boldsymbol{\pi}^*)\left(\mathbf{K}_n + \mathbf{I}_n\sigma_\omega^2\right)^{-1}\mathbf{k}_n^{\mathrm{T}}(\boldsymbol{\pi}^*) \\ \mu_n(\boldsymbol{\pi}^*) &= \mathbf{k}_n(\boldsymbol{\pi}^*)\left(\mathbf{K}_n + \mathbf{I}_n\sigma_\omega^2\right)^{-1}\hat{\mathbf{f}}_n \end{aligned} \quad (13)$$

where $\hat{\mathbf{f}}_n = [\hat{f}(\boldsymbol{\pi}_1), \ldots, \hat{f}(\boldsymbol{\pi}_n)]^{\mathrm{T}}$ is the observed function value vector, the covariance matrix $\mathbf{K}_n \in \mathbb{R}^{n\times n}$ has elements $[\mathbf{K}_n]_{(i,j)} = k(\boldsymbol{\pi}_i, \boldsymbol{\pi}_j), i, j \in \{1, \ldots, n\}$, and the vector $\mathbf{k}_n(\boldsymbol{\pi}^*) = [k(\boldsymbol{\pi}^*, \boldsymbol{\pi}_1), \ldots, k(\boldsymbol{\pi}^*, \boldsymbol{\pi}_n)]$ includes the covariance between the function value of desired point and the previous observations. We denote idntity matrix as $\mathbf{I}_n \in \mathbb{R}^{n\times n}$.

Acquisition function trades off between exploitation, sampling where the surrogate model predicts a high function value, and exploration, sampling at locations where the uncertainty of the function value is high. The next sampling point is determined by

$$\boldsymbol{\pi}_n = \arg\max_{\boldsymbol{\pi}\in\boldsymbol{\Pi}} \mathcal{A}(\boldsymbol{\pi}), \quad (14)$$

where $\mathcal{A}(\boldsymbol{\pi})$ represents the acquisition function. Let $\boldsymbol{\pi}^+$ be the best point in $\mathcal{D}_n$, namely

$$f(\boldsymbol{\pi}^+) = \min_{\boldsymbol{\pi}\in\mathcal{D}_n} f(\boldsymbol{\pi}).$$

Let $\tilde{f}(\boldsymbol{\pi})$ be the Gaussian process posterior distribution for $f(\boldsymbol{\pi})$. We define the improvement function as

$$I(\boldsymbol{\pi}) = \max\left\{0, f(\boldsymbol{\pi}^+) - \tilde{f}(\boldsymbol{\pi})\right\}.$$

Then the EI acquisition function is

$$\mathcal{A}(\boldsymbol{\pi}) = \mathbb{E}[I(\boldsymbol{\pi})]. \quad (15)$$

Theoretical results show that under certain conditions, applying this acquisition function will lead to convergence to the global minimum after a finite number of iterations.

Although (14) is also solved as an optimization problem, no evaluations on the real robot are required but only computation with GP, based on the assumption of cheap computational resources.

Algorithm 1 Bayesian optimization

Inputs: Domain $\mathbf{\Pi}$ and Initial parameters $\boldsymbol{\pi}_0$
Output: The optimal parameters
1: **for** $n = 1, 2, 3, \cdots$ **do**
2: Update the Gaussian process
3: $\boldsymbol{\pi}_n \leftarrow \arg\max_{\boldsymbol{\pi} \in \mathbf{\Pi}} \mathbb{E}[I(\boldsymbol{\pi})]$
4: $y_n \leftarrow \hat{f}(\boldsymbol{\pi}_n)$
5: $\mathcal{D}_n = \mathcal{D}_{n-1} \bigcup \{\boldsymbol{\pi}_n, y_n\}$
6: **end for**
7: **return** $\boldsymbol{\pi}^*$

Algorithm 2 BO with fitness function with penalty item

Inputs: Domain $\mathbf{\Pi}$ and Initial parameters $\boldsymbol{\pi}_0$
Output: The optimal parameters
1: **for** $n = 1, 2, 3, \cdots$ **do**
2: Update the Gaussian process
3: $\boldsymbol{\pi}_n \leftarrow \arg\max_{\boldsymbol{\pi} \in \mathbf{\Pi}} \mathbb{E}[I(\boldsymbol{\pi})]$
4: $y_n \leftarrow \hat{f}(\boldsymbol{\pi}_n) + p(\boldsymbol{\pi}_n)$
5: $\mathcal{D}_n = \mathcal{D}_{n-1} \bigcup \{\boldsymbol{\pi}_n, y_n\}$
6: **end for**
7: **return** $\boldsymbol{\pi}^*$

Algorithm 3 BO with constrained acquisition function

Inputs: Domain $\mathbf{\Pi}$ and Initial parameters $\boldsymbol{\pi}_0$
Output: The optimal parameters
1: **for** $n = 1, 2, 3, \cdots$ **do**
2: Update the Gaussian process
3: $\boldsymbol{\pi}_n \leftarrow \arg\max_{\boldsymbol{\pi} \in \mathbf{\Pi}} \mathbb{E}[I_C(\boldsymbol{\pi})]$
4: $y_n \leftarrow \hat{f}(\boldsymbol{\pi}_n)$
5: $\mathcal{D}_n = \mathcal{D}_{n-1} \bigcup \{\boldsymbol{\pi}_n, y_n\}$
6: **end for**
7: **return** $\boldsymbol{\pi}^*$

Algorithm 4 BO with Gaussian Process with multiple outputs

Inputs: Domain $\mathbf{\Pi}$
GP prior $k((\boldsymbol{\pi}, i), (\boldsymbol{\pi}', j))$
Lipschitz constant L
Initial safe set $S_0 \subseteq \mathbf{\Pi}$
Output: The optimal parameters
1: **for** $n = 1, 2, 3, \cdots$ **do**
2: $S_n \leftarrow \bigcup_{i \in I_{ie}} \bigcap_{\boldsymbol{\pi} \in S_{n-1}} \left\{\boldsymbol{\pi}' \in \mathbf{\Pi} | l_n^i(\boldsymbol{\pi}) - L\|\boldsymbol{\pi} - \boldsymbol{\pi}'\| \geq 0\right\}$
3: $M_n \leftarrow \left\{\boldsymbol{\pi} \in S_n | l_n^f(\boldsymbol{\pi}) \leq \min_{\boldsymbol{\pi}' \in S_n} u_n^f(\boldsymbol{\pi}')\right\}$
4: $G_n \leftarrow \{\boldsymbol{\pi} \in S_n | e_n(\boldsymbol{\pi}) \geq 0\}$
5: $\boldsymbol{\pi}_n \leftarrow \arg\max_{\boldsymbol{\pi} \in G_n \cup M_n} \max_{i \in I} w_n(\boldsymbol{\pi}, i)$
6: Measurements: $\hat{f}(\boldsymbol{\pi}_n), \hat{g}_i(\boldsymbol{\pi}_n) \forall i = 0, \ldots, n_{ie}$
7: Update GP with new data
8: **end for**
9: **return** $\boldsymbol{\pi}^*$

B. BO with fitness function with penalty item

This variant improves the optimization performance by appending a penalty item to the original objective function when the constraints violations happen. It uses a new objective function $f_p(\boldsymbol{\pi}) = f(\boldsymbol{\pi}) + p(\boldsymbol{\pi})$ with $p(\boldsymbol{\pi}) = \sum_{t=1}^{T} p_t$, where t is the time step. We defined p_t as:

$$p_t = \sum_{i=1}^{n_{ie}} max\left(0, g_i\left(\boldsymbol{q}_t, \boldsymbol{u}_t\right)\right)^2, \tag{16}$$

where n_{ie} denotes the number of inequality constraints.

C. BO with constrained acquisition function

This method is proposed in [15]. Several modifications need to be done to add constraints to Bayesian optimization via EI acquisition function. Here we augment the definition of $\boldsymbol{\pi}^+$ to be the best feasible point in $\mathcal{D}_n$. To perform Bayesian optimization with multiple constraints, each constraint is modeled with an independent Gaussian process. This method evaluates g_i at the next sampling point $\boldsymbol{\pi}_n$ and adds $(\boldsymbol{\pi}_n, g_i)$ to $\mathcal{C}_i$ which is employed to calculate the Gaussian process posterior. Then the constrained improvement function can be formulated as:

$$I_C(\boldsymbol{\pi}) = \tilde{\Delta}(\boldsymbol{\pi}) \max\left\{0, f\left(\boldsymbol{\pi}^+\right) - \tilde{f}(\boldsymbol{\pi})\right)\right\} = \tilde{\Delta}(\boldsymbol{\pi}) I(\boldsymbol{\pi}),$$

where $\Delta(\boldsymbol{\pi}) \in \{0, 1\}$ is a constraint satisfaction indicator function and $\tilde{\Delta}(\boldsymbol{\pi})$ stands for its Gaussian process posterior distribution. The satisfaction indicator function are set to 1 if all the constraints are satisfied and 0 otherwise. Therefore the constrained acquisition function is:

$$\mathcal{A}(\boldsymbol{\pi}) = \mathbb{E}\left[I_C(\boldsymbol{\pi})\right] = \mathbb{E}[\tilde{\Delta}(\boldsymbol{\pi})]\mathbb{E}[I(\boldsymbol{\pi})], \tag{17}$$

where the second equality derives from the independence of the constraint g_i and fitness function f.

D. BO with Gaussian Process with multiple outputs

This method is proposed in [16]. In order to model the correlation between different functions, i.e. performance and constraints in this paper, a new surrogate function is considered:

$$h(\boldsymbol{\pi}, i) = \begin{cases} f(\boldsymbol{\pi}) & \text{if } i = 0 \\ g_i(\boldsymbol{\pi}) & \text{if } i \in I_{ie} \end{cases}, \tag{18}$$

where $i \in I$ with $I = \{0, I_{ie}\}$.

For instance, the kernel for performance and one constraint is:

$$k\left((\boldsymbol{\pi}, i), (\boldsymbol{\pi}', j)\right) = \begin{cases} \delta_{ij} k_f(\boldsymbol{\pi}, \boldsymbol{\pi}') + k_{fg}(\boldsymbol{\pi}, \boldsymbol{\pi}') & \text{if } i = 0 \\ \delta_{ij} k_g(\boldsymbol{\pi}, \boldsymbol{\pi}') + k_{fg}(\boldsymbol{\pi}, \boldsymbol{\pi}') & \text{if } i = 1 \end{cases},$$

where δ_{ij} is the Kronecker delta.

In order to guarantee safety, confidence internals that include both performance f and constraint g_i with high probability are constructed. By employing the posterior GP estimate

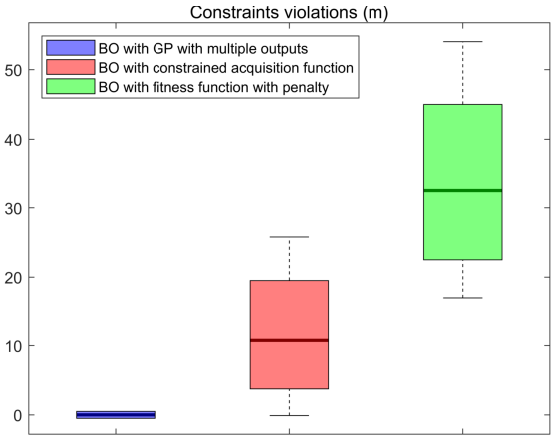


Fig. 3. Constraints violations comparison of three variants of BO. BO with GP with multiple outputs guarantees the constraints satisfaction during the learning process while the other two sometimes failed.

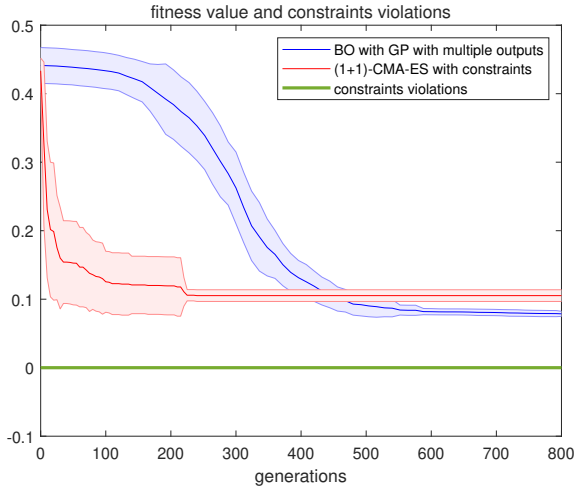


Fig. 4. Fitness and constraints violations comparison between BO with GP with multiple outputs and (1+1)-CMA-ES with covariance constrained adaptation. Both algorithm ensure constraints satisfaction. It's noteworthy that this variant of BO converges a lot faster and is more computationally affordable.

with the past observations, the confidence intervals for the surrogate model in (18) can be defined as

$$Q_n(\boldsymbol{\pi}, i) := \left[\mu_{n-1}(\boldsymbol{\pi}, i) \pm \beta_n^{1/2} \sigma_{n-1}(\boldsymbol{\pi}, i)\right], \tag{19}$$

where β_n is a parameter that reflects the confidence interval. All possible function values are comprised in this set using Gaussian process posterior and their probability are dependent on the choice of β_n and kernel function. The following analysis requires that consecutive estimates of the lower and upper bounds are contained within each other. Hence the contained set at iteration n is defined as $C_n(\boldsymbol{\pi}, i) = C_{n-1}(\boldsymbol{\pi}, i) \cap Q_n(\boldsymbol{\pi}, i)$, where $C_0(\boldsymbol{\pi}, i)$ is $[0, \infty]$ for all $\boldsymbol{\pi} \in S_0$. Then we define the lower and upper bounds of this set as $l_n^i(\boldsymbol{\pi}) := \min C_n(\boldsymbol{\pi}, i)$ and $u_n^i(\boldsymbol{\pi}) := \max C_n(\boldsymbol{\pi}, i)$ respectively. For notational clarity, we write $l_n^f(\boldsymbol{\pi}) := l_n^0(\boldsymbol{\pi})$ and $u_n^f(\boldsymbol{\pi}) := u_n^0(\boldsymbol{\pi})$ for the performance bounds.

According to the confidence intervals stated above, the safe set can be expanded by taking advantage of the Lipschitz continuity properties:

$$S_n \leftarrow \bigcup_{i \in I} \bigcap_{\boldsymbol{\pi} \in S_{n-1}} \left\{\boldsymbol{\pi}' \in \boldsymbol{\Pi} | l_n^i(\boldsymbol{\pi}) - L \left\|\boldsymbol{\pi} - \boldsymbol{\pi}'\right\| \geq 0\right\}. \tag{20}$$

This set comprises all the points in S_{n-1}, as well as additional points that satisfy the safety constraints given the Lipschitz constant and the confidence intervals.

Having defined the safe set, we move on to trade off between exploitation and exploration. As proposed in [19], we could simply select the most uncertain point but it's not sample-efficient. Therefore, we follow the method in [16] and first define two subsets of S_n, one for improving the estimate of the minimum and another for expanding the safe set. The set of potential minimizer is defined as

$$M_n \leftarrow \left\{\boldsymbol{\pi} \in S_n | l_n^f(\boldsymbol{\pi}) \leq \min_{\boldsymbol{\pi}' \in S_n} u_n^f(\boldsymbol{\pi}')\right\}, \tag{21}$$

whose parameters are candidates for the optimum.

In the same way, a set of parameters that could expand the safe set potentially is defined as

$$G_n \leftarrow \{\boldsymbol{\pi} \in S_n | e_n(\boldsymbol{\pi}) \geq 0\}, \tag{22}$$

$$e_n(\boldsymbol{\pi}) := \{\boldsymbol{\pi}' \in \boldsymbol{\Pi} \backslash S_n | \exists i \in I_{ie} : u_n^i(\boldsymbol{\pi}) - L \left\|\boldsymbol{\pi} - \boldsymbol{\pi}'\right\| \geq 0\}. \tag{23}$$

The function e_n enumerates the number of parameters that could additionally be classified as safe if a safety function obtained a measurement which is equal to its upper confidence bound. Therefore, the set G_n could enlarge the safe set potentially.

The final step is to trade off between M_n and G_n, by selecting the most uncertain point over the performance and constraints, which means the next parameter set to be evaluated in the real robot is derived by

$$\boldsymbol{\pi}_n \leftarrow \underset{\boldsymbol{\pi} \in G_n \cup M_n}{\arg\max} \max_{i \in I} w_n(\boldsymbol{\pi}, i), \tag{24}$$

$$w_n(\boldsymbol{\pi}, i) = u_n^i(\boldsymbol{\pi}) - l_n^i(\boldsymbol{\pi}). \tag{25}$$

Pseudocode for the algorithm is found in Algorithm 4. For more details, see [16].

IV. SIMULATION VERIFICATION

Our simulation is powered by DART[1] physics engine with a 6 degrees of freedom KR5 robot arm. In this section, we first introduce the simulation setup with a KUKA KR5 robot arm, including tasks definition, kernel selection and fitness function design. Then we make a comparison between the variants introduced in Section III and select a suitable solution. To show the efficiency of the selected method, We compare this variant with (1+1)-CMA-ES with covariance

[1]https://dartsim.github.io/

constrained adaptation proposed in [20]. At last, we show the simulation result to demonstrate the effectiveness of the selected optimization method.

A. Simulation setup

We design a global task as reaching a desired point behind a cuboid with the robot arm's end-effector. Then this global task can be decomposed into three elementary tasks. The first one is to reach a Cartesian position $p^* = [0.5, 0.16, 0]$ with its end-effector. The second is also a position task, reaching a Catesian position $[0.361, -0.02, 0]$ with its 4-th link. The third is a joint angle position task, reaching a joint angle configuration [0.1, 1.54, -2.3, 0, 0, 0].

In our simulation, we employ the Matérn 5/2 kernel: $k(\boldsymbol{\pi}, \boldsymbol{\pi}') = \left(1 + \sqrt{5}r + \frac{5r^2}{3}\right) \exp\left(-\sqrt{5}r\right)$, where $r^2 = \|\boldsymbol{\pi} - \boldsymbol{\pi}'\|^2$. The profiles of the task priorities are modeled by normalized RBFN with $n_r = 5$. During the learning process, 7 inequality constraints need to be satisfied, including joint torque limits and obstacle collision avoidance, i.e. $n_{ie} = 7$.

The fitness function is designed as follow:

$$\boldsymbol{\phi}(\boldsymbol{q}_{t=1,\ldots,T}, \boldsymbol{u}_{t=1,\ldots,T}) = \frac{1}{2}\left(\frac{\sum_{t=1}^{T} \|\boldsymbol{p}_t - \boldsymbol{p}^*\|_2}{e_{max}} + \frac{\sum_{t=1}^{T} \|\boldsymbol{u}_t\|_2}{u_{max}}\right)$$

where t represents the order of the control step, T is the number of total steps. The duration are set to 26s and each step is 0.01s. p_t denotes the end-effector position at control step t, and p^* is the goal position. e_{max} and u_{max} both are scaling factors. The first term of the fitness function penalizes the accumulated error of reaching the final goal while the second term penalizes the sum of joint command at each control step.

B. Comparison between variants of Bayesian optimization

We here compare the three variants described in Section III to find out which one guarantees no constraints violations during the optimization process.

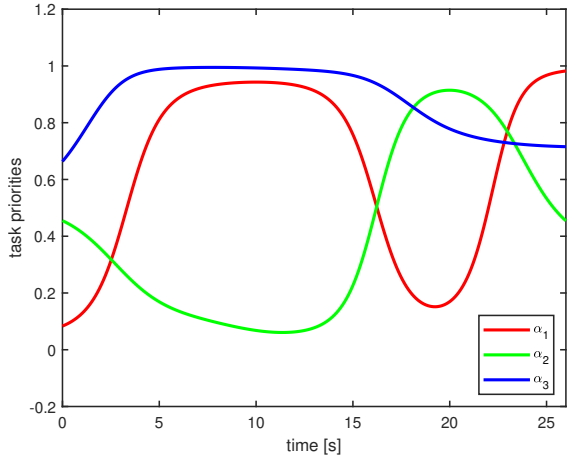
For making a fair comparison, these algorithms all start from the same initial parameters. Additionally, each method is conducted for 20 times and there are $n_{it} = 800$ iterations for each time.

The constraint violations m is defined as $m = \sum_{n=1}^{n_{it}} \sum_{i=1}^{n_{ie}} v(i,n)$ where $v(i,n) = \max\left\{0, \frac{g_i(n)}{|g_i(n)|}\right\}$ and n represents the order of the iteration. It means that number of constraints violations in every episode will be added to m.

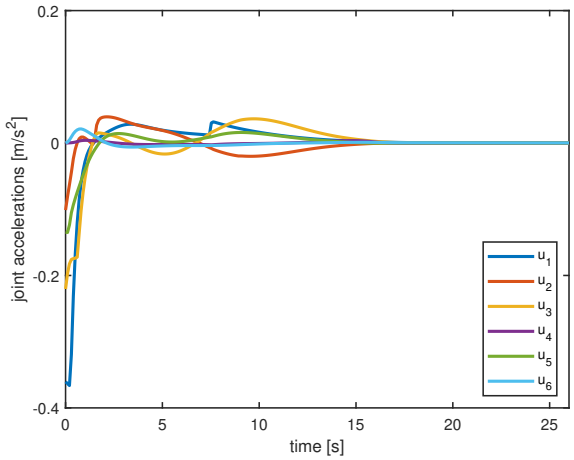
As shown in Fig. 3, BO with GP with multiple outputs causes no constraints violations, outperforming the other two variants which sometimes fail.

C. Comparison with (1+1)-CMA-ES with covariance constrained adaptation

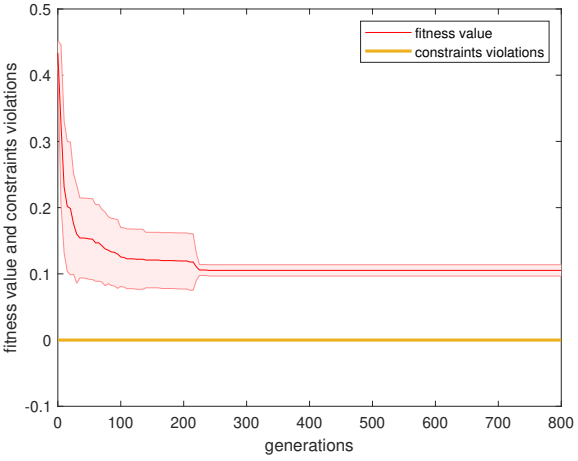
In Section IV-B we found out that BO with GP with multiple outputs is the best variant. Therefore in this experiment we compare it with the state-of-the-art priorities learning method, (1+1)-CMA-ES with covariance constrained adaptation [12], with respect to fitness value and constraints violations. The



(a) The time profiles of soft task priorities



(b) The observed joint accelerations



(c) The fitness values and constraints violations

Fig. 5. Performance of the learning result employing the best variant found in Section IV-B. Both the evolution of task priorities over time and the observed joint accelerations are smooth. This optimization approach found the solution efficiently while satisfying all the safety constraints.

simulation setup is the same as Section IV-A. Both of them are starting from a set of random and safe RBFN parameters. We conduct each simulation over 800 generations for 10 times, and the means, the standard deviation of fitness and the constraints violations are shown in Fig. 4.

As we can see, both of them never break the safety constraints during the whole learning process. When the fitness converged, the fitness mean of (1+1)-CMA-ES is only slightly better than BO with constraints.

It's noteworthy that the variant of BO is much more sample-efficient, i.e. converging a lot faster, than (1+1)-CMA-ES. Its fitness reaches 0.2 in about 20 generations and 0.1 in about 230 generations, while (1+1)-CMA-ES reaches the same level in about 330 generations and 450 generations respectively.

D. Learning soft priorities for KR5

As shown in Fig. 5(c), starting from a set of random and safe RBFN parameters, we conduct the simulation with the selected method, BO with GP with multiple outputs, over 800 generations for 10 times, and the fitness values all converge near 0.1 without any safety constraints violations, which meets the requirement of safe learning. With the decrease of the fitness, the robot arm's global motion is optimized with respect to energy and accumulated error. We show one of the solutions in Fig. 5(a) and Fig. 5(b), which represent task priorities and joint commands respectively. Furthermore, in Fig. 5(a) we can also see that the profiles of the learned task priorities, outputs of normalized RBFN, change smoothly over time, guaranteeing the smoothness of the joint commands, as shown in Fig. 5(b).

V. CONCLUSION AND FUTURE WORK

In this paper, we propose to learn the task priorities of a multi-task controller by employing a sample-efficient approach that explores the solution without violating any safety constraints during the whole training process. We made a comparison between three variants of Bayesian optimization and finally found the suitable variant of Bayesian optimization to solve this problem.

The selected method is validated by a simulation, where a robot arm reaches a goal position by executing several potentially conflicting task simultaneously. The selected approach outperforms the evolution algorithm, i.e. (1+1)-CMA-ES with active covariance adaptation, since the selected method converges faster and takes less computational resources. As shown in the simulation, our framework can be employed to generate optimized global motions that meet the safety requirement all the time.

This framework is only able to solve the optimization problem offline, due to the long computation time. Future work is focus on reducing the computation time and transfer this feature, i.e. guaranteeing safety constraints satisfaction, to online tuning task priotiites.

REFERENCES

[1] Y. Wang, F. Vina, Y. Karayiannidis, C. Smith, and P. Ogren, "Dual arm manipulation using constraint based programming," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 311–319, 2014.

[2] J. Salini, V. Padois, and P. Bidaud, "Synthesis of complex humanoid whole-body behavior: A focus on sequencing and tasks transitions," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 1283–1290.

[3] A. Escande, N. Mansard, and P.-B. Wieber, "Hierarchical quadratic programming: Fast online humanoid-robot motion generation," *The International Journal of Robotics Research*, vol. 33, no. 7, pp. 1006–1028, 2014.

[4] A. Del Prete, F. Nori, G. Metta, and L. Natale, "Prioritized motion–force control of constrained fully-actuated robots:"task space inverse dynamics"," *Robotics and Autonomous Systems*, vol. 63, pp. 150–157, 2015.

[5] C. Ott, A. Dietrich, and A. Albu-Schäffer, "Prioritized multi-task compliance control of redundant manipulators," *Automatica*, vol. 53, pp. 416–423, 2015.

[6] J. Park, Y. Choi, W. K. Chung, and Y. Youm, "Multiple tasks kinematics using weighted pseudo-inverse for kinematically redundant manipulators," in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, vol. 4. Ieee, 2001, pp. 4041–4047.

[7] L. Sentis and O. Khatib, "Synthesis of whole-body behaviors through hierarchical control of behavioral primitives," *International Journal of Humanoid Robotics*, vol. 2, no. 04, pp. 505–518, 2005.

[8] Y. Wang and A. Kheddar, "Impact-friendly robust control design with task-space quadratic optimization," in *Robotics: Science and Systems (RSS)*, 2019.

[9] M. Liu, Y. Tan, and V. Padois, "Generalized hierarchical control," *Autonomous Robots*, vol. 40, no. 1, pp. 17–31, 2016.

[10] N. Dehio, R. F. Reinhart, and J. J. Steil, "Multiple task optimization with a mixture of controllers for motion generation," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 6416–6421.

[11] S. Ha and C. K. Liu, "Evolutionary optimization for parameterized whole-body dynamic motor skills," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 1390–1397.

[12] V. Modugno, G. Neumann, E. Rueckert, G. Oriolo, J. Peters, and S. Ivaldi, "Learning soft task priorities for control of redundant robots," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 221–226.

[13] V. Modugno, U. Chervet, G. Oriolo, and S. Ivaldi, "Learning soft task priorities for safe control of humanoid robots with constrained stochastic optimization," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2016, pp. 101–108.

[14] Y. Su, Y. Wang, and A. Kheddar, "Sample-efficient learning of soft task priorities through bayesian optimization," in *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2018, pp. 1–6.

[15] J. R. Gardner, M. J. Kusner, Z. E. Xu, K. Q. Weinberger, and J. P. Cunningham, "Bayesian optimization with inequality constraints." in *ICML*, 2014, pp. 937–945.

[16] F. Berkenkamp, A. Krause, and A. P. Schoellig, "Bayesian optimization with safety constraints: safe and automatic parameter tuning in robotics," *arXiv preprint arXiv:1602.04450*, 2016.

[17] K. Bouyarmane, K. Chappellet, J. Vaillant, and A. Kheddar, "Quadratic programming for multirobot and task-space force control," *IEEE Transactions on Robotics*, vol. 35, no. 1, pp. 64–77, 2018.

[18] F. L. Moro, M. Gienger, A. Goswami, N. G. Tsagarakis, and D. G. Caldwell, "An attractor-based whole-body motion control (wbmc) system for humanoid robots," in *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. IEEE, 2013, pp. 42–49.

[19] J. Schreiter, D. Nguyen-Tuong, M. Eberts, B. Bischoff, H. Markert, and M. Toussaint, "Safe exploration for active learning with gaussian processes," in *Joint European conference on machine learning and knowledge discovery in databases*. Springer, 2015, pp. 133–149.

[20] D. V. Arnold and N. Hansen, "A (1+ 1)-cma-es for constrained optimisation," in *Proceedings of the 14th annual conference on Genetic and evolutionary computation*. ACM, 2012, pp. 297–304.