2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)
Beijing, China, November 6-9, 2018

Sample-efficient learning of soft task priorities through Bayesian optimization.

Yinyin Su[1], Yuquan Wang[1*] and Abderrahmane Kheddar [2]

Abstract— **In recent optimization task-space controller, hierarchical task prioritization can be made strict or soft within a given level. Soft hierachization is made using task weighting. Yet the latter is not automated and weights are set ad-hoc. This empirical approach could be time-consuming and even leads to an infeasible result. During a specific episode in order to approximate the evolution of the weight of a task, we assign a Radial basis function network(RBFN) to each of the tasks. We use the Bayesian Optimization procedure to regulate the RBFNs corresponding to different tasks based on performances indexes that are extracted for a fixed episode. We benchmark the proposed solution with a dual-arm manipulation simulation where multiple potentially conflicting tasks are involved. First of all We can find that the proposed approach outperforms a hand-tuned controller in terms of tracking errors. In comparison with tuning the weights using another stochastic optimization technique, i.e. CMA-ES, we can find that the proposed approach requires much less samples to evaluate.**

I. INTRODUCTION

Using optimization-based formulations, we can develop reactive robot controllers that could simultaneously satisfy multiple tasks and objectives in a modular way. Incorporating multiple objectives into a unified minimization or maximization index normally requires a scalarization procedure, where we assign a weight to different tasks and objectives. Ideally the weight should be selected according to an underlying *value function*. However the value function is typically unknown and difficult to estimate [1]. This leads to the fact that robot task programming usually requires an experienced human decision maker to select weights from different trials case-by-case.

When there are successful implementation examples to learn from, we can apply the different learning techniques. For instance, we can try to extract task constraints from demonstrations, see [2]. We can learn the mapping between the observed motor primitive and the joint torque [3]. When there are multiple prioritized motor primitives, we can use the redundancy of a robot arm to simultaneously fulfill the prioritized motor primitives. Basically we can modify the motor primitive with a lower priority to ensure the motor primitive with a higher priority. Or we can directly learn the null space projection matrices from human demonstrations, see [4]. However in case of reactive control, we are in lack of successful examples, therefore we need to resort to other techniques.

In the context of learning, a *hyperparameter optimization* algorithm chooses a proper set of parameters for a specific learning algorithm. As reported by[5], [6] and [7], among different tuning algorithms: grid search, Bayesian optimization, random search and gradient-based optimization, Bayesian optimization is a sample-efficient and computationally affordable approach. Bayesian Optimization builds a surrogate model for the cost function being optimized using data points sampled so far.Based on this surrogate, it predicts the expected cost of points that have not been sampled yet. Then it does an auxiliary maximization of the acquisition function to find the next point to be sampled. It is an effective framework for optimizing objective functions without knowing its gradient using very few iterations. Reported by [8], we can apply constrained Bayesian optimization to tune the controller gains of individual tasks that are specified in a hybrid controller. As presented in [9], parameters of a Linear Quadratic Regulator (LQR) problem are obtained in an data-efficient way by global Bayesian optimization through the Entropy Search algorithm. The safe Bayesian optimization introduced in [10] restricts the cost function to a safe set such that it optimizes the controller parameters of a quadrotor while keeping the safety constraints.

Alternative to Bayesian Optimization, we can apply genetic algorithms to parameter tuning problems. The Covariance Matrix Adaption Evolution Strategy (CMA-ES), which is a black-box learning algorithm, is proposed to solve for the task priorities in [11]. In another modified version [12] (1+1)-CMA-ES, the constraint satisfaction is explicitly considered. However as a generic algorithm typically requires a considerably big population size to search for, both (1+1)-CMA-ES and CMA-ES require way more iterations than Bayesian Optimization. More details can be found later in the benchmark results in Section IV.

Recently, the deep reinforcement learning algorithm, i.e. Deep Deterministic Policy Gradient (DDPG) [13] is also introduced to optimizing policy parameters over a continuous state-action space. Due to the use of replay butter, batch normalization and two target networks, DDPG can converge stably and quickly. However, DDPG itself also relies on a good amount of hyperparameters, e.g. the layers of networks, the numbers of neurons in layers, the size of batch size, the reply buffer data management. While optimizing the controller weights, the newly introduced hyperparameters are

This paper is partially supported by the National Key Research and Development Program of China(Project No.2017YFB1303702)the National Natural Science Foundation of China (grant No: U1613216), the State Joint Engineering Lab on Robotics and Intelligent Manufacturing, and Shenzhen Engineering Lab on Robotics and Intelligent Manufacturing, from Shenzhen Gov, China.

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

[1*]Corresponding author. email: yuquan@kth.se

[2] Interactive Digital Humans (IDH) CNRS-UM LIRMM at Montpellier, France

not easy to define. Sometimes it even leads to divergence. More importantly, training a deep neural network typically requires even more samples. This is critical for evaluating an expensive process.

If we want to apply Bayesian Optimization to tune the weights of an reactive robot control problem, we need to reformulate the reactive control problem such that a stationary process is available for us to evaluate. For hyperparameter tuning problems, or even optimization problems in general, it is normally assumed that the process, so as the underlying value function, is stationary, otherwise we need to track a dynamic optimality point. The stationary process assumption makes it difficult to directly apply stochastic optimization techniques to a reactive controller online. Rather, people try to define a certain time interval during which the robot performance is measured by a cumulative measure, see [11] and [12]. Or in other words, the performance is measured on a trajectory level instead of on a reactive level, see [14]. Following the examples from [11] and [12], we use RBFN to approximate the evolution of the weights of different tasks and extract performance indexes with respect to a certain episode. The relation between the episodic performance indexes and the RBFN parameters is nonlinear and in lack of an analytic gradient. We use Gaussian Process as a surrogate model to replace it and use the expected improvement acquisition function to sample the next point.

The rest of paper is organized as follows: We formulate our problem in Section II, where the reactive controller and the approximation of the task weights are presented. Then in Section III, we introduce the proposed Bayesian optimization procedure in more details. We present the benchmark results finally in Section IV and draw conclusions and state the future work in Section V.

II. Problem Formulation

As shown in recent publications [15], [16], we can solve multiple subtasks for a redundant robot in an unified quadratic programming(QP) formulation. In Section II-A, we define the soft task priorities as the weights of the subtasks when we scalarize the objective function. Then in Section II-B, following the successful examples in [11] and [12] we model the soft task priorities for a fixed episode with a normalized RBFN. With this formulation, learning of the soft task priorities is equivalent to the learning of the hyperparameters of the corresponding RBFN.

A. Soft task priorities in a QP controller

We can use linearly constrained quadratic programs to describe complex robot motions, see [17]. Different constraints and objectives could be used to describe different aspect of a certain task and each of them could be stabilized with an individual control Lyapunov function [18]. Normally the equalities and inequalities are considered as hard constraints which have higher priorities than the objectives.

Considering an n degrees of freedom robot, we denote the joint command as $\boldsymbol{u} \in \mathbb{R}^n$, where we use the bold symbol to denote a vector. Suppose I_o, I_e and I_{ie} represent the set

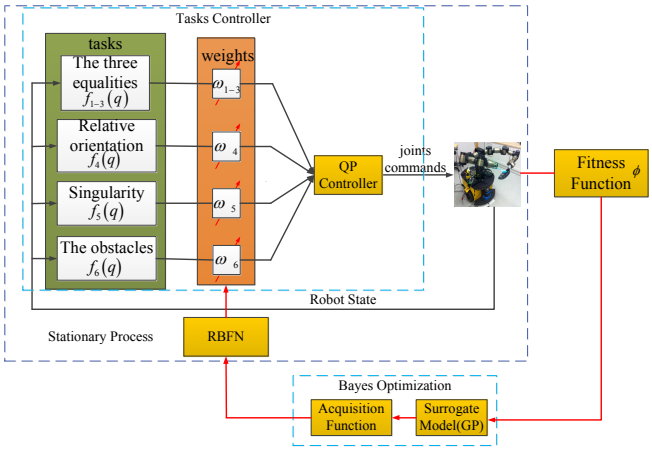


Fig. 1 Outline of the proposed frame which holds two loops. The inner loop is QP controller circle and outer loop enables the optimization of weights using learning algorithm Bayesian optimization. The RBFN is used to convert reactive controller to the stationary process.

of objectives, equality constraints and inequality constraints respectively and the corresponding gradients are available, we can construct the following QP:

$$\begin{aligned} \min_{\boldsymbol{u}} \quad & \boldsymbol{u}^T Q \boldsymbol{u} + \sum_{i \in I_o} \omega_i \frac{\partial f_i}{\partial \boldsymbol{u}}^T \boldsymbol{u} + \sum_{i \in I_{ie}, I_e} w_i \|\mu_i\|_2, \\ s.t. \quad & \frac{\partial f_i}{\partial \boldsymbol{u}}^T \boldsymbol{u} + \mu_i \leq -k_i(f_i - b_i) - \frac{\partial f_i}{\partial t}, \quad \forall i \in I_{ie}, \\ & \frac{\partial f_i}{\partial \boldsymbol{u}}^T \boldsymbol{u} + \mu_i = -k_i(f_i - b_i) - \frac{\partial f_i}{\partial t}, \quad \forall i \in I_e \end{aligned} \tag{1}$$

where Q is a diagonal positive definite matrix. It is used to prioritize each joint and minimize the output tasks of controller, namely, decrease the use of energy as much as possible on the premise of better performance of robot. k_i, b_i are the positive gains and bounds related to every constraint.

As highlighted by the blue font in (1), we used a slack variable μ_i in each constraint $f_i, \forall i \in I_e, I_{ie}$ and penalize the weighted sum of the normed slack variables $\|\mu\|_2$, the weighted sum of joint commands and the gradient of $f_i, \forall i \in I_o$. In this way, we convert the hard constraints into soft ones, the soft task weights are $w_i, \forall i \in I_e, I_{ie}$.

B. Normalized RBFN

The QP given in (1) is a reactive controller in the sense that in each time step we solve for the locally optimal joint command $\boldsymbol{u}$. The robot joint configuration is dynamic, so does the constraint $f_i, \forall i \in I_e, I_{ie}$. Therefore instead of using static soft task priorities $w_i, \forall i \in I_e, I_{ie}, I_o$, we define them as a function of time: $w_i(t), \forall i \in I_e, I_{ie}, I_o$.

For a given time interval, we model $w_i(t)$ using a radial basis function network(RBFN). As shown in Fig. 2, the i-th task weight $w_i(t)$ can be modeled as:

$$\omega_i = Sigmoid\left(\frac{\sum_k \pi_{ik} \Psi_k(\mu_k, \sigma_k, t)}{\sum_k \Psi_k(\mu_k, \sigma_k, t)}\right) \tag{2}$$

where we used n_k radial basis functions and $\pi_i = \{(\pi_{ik}, \sigma_k, \mu_k) \ | k = 1, \ldots, n_k\}$. Each radial basis function

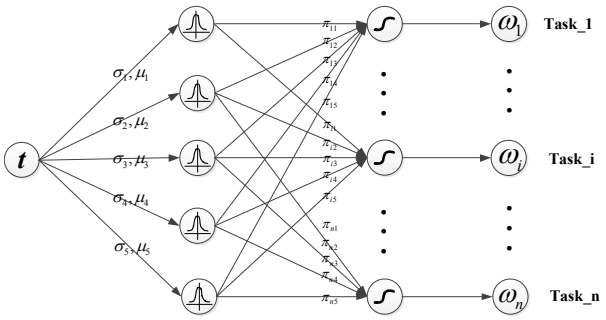


Fig. 2 RBF Network . t denotes the time step of accomplishing the tasks modeled by RNFN, ω_i is the i-th task priority at each time step and π_{ik}, σ_k and μ_k for $k = 1, \ldots, n_k$ (e.g. $n_k = 5$) are the parameters of RBFN.

has a weight π_{ik}, the mean μ_k and the variance σ_k as:

$$\Psi_k(\mu_k, \sigma_k, t) = \exp\left(-\frac{1}{2}\left(\frac{t-\mu_k}{\sigma_k}\right)^2\right) \tag{3}$$

For numerical stability reasons, we use the function $Sigmoid\,(x) = 1/(1+\exp(-x))$ to squash the outputs of RBFN to range $[0, 1]$. When the task priority equal to 1, this task dominate importantly. We use fixed σ_k and μ_k for all the RBFs $k = 1, \ldots, n_k$, in every task due to the fact that this choice would reduce the unknown learning parameters and improve the rate of convergence, see [11].

Connecting the QP defined by (1) and the RBFN defined by (2), we can tell that learning a soft task priority $w_i(t)$ and diagonal elements of $Q \in \mathbb{R}^n$ in (1) is equivalent to learning the corresponding RBF parameters $\boldsymbol{\pi}$ in (2). With the fixed σ_k and μ_k for all the RBFs, the total number of parameters $\boldsymbol{\pi}$ of an RBFN is: $(n+2+dim(I_o)+dim(I_e)+dim(I_{ie}))n_k$.

Suppose in a robotic task, the fitness function $\phi(\boldsymbol{q}_t, \boldsymbol{u}_t)$ measures the performance for a given time interval t : $0 \to T$, where $\boldsymbol{q}_t \in \mathbb{R}^n$ and $\boldsymbol{u}_t \in \mathbb{R}^n$ denote the joint configurations and joint commands at time t. Then the soft task priority learning is the following minimization problem:

$$\boldsymbol{\pi}^* = \arg\min_{\boldsymbol{\pi}} \phi(\boldsymbol{q}_t, \boldsymbol{u}_t) \tag{4}$$

where $\boldsymbol{\pi}^*$ denotes the optimal parameters of RBFN when the robot reach to the final state. In different tasks, the fitness function $\phi(\boldsymbol{q}_t, \boldsymbol{u}_t)$ could cover different performance indexes, e.g. energy consumption measures, distance to the obstacle and so on. We can find a good summary in the literature, see [19] and [20].

III. Proposed Method

Using the equivalence between soft task priority learning and RBFN weights learning, solving for the RBFN parameters $\boldsymbol{\pi}$ from (4) is equivalent to learning the task weights $w_i, \forall i \in I_o, I_e, I_{ie}$ and Q in (1).

In view of the fact that robotic systems are computationally expensive and time consuming to evaluate, we propose to use Bayesian Optimization to (4). We can find an overview of the steps in Fig. 1. Approximating the gradient

$$\frac{\partial \phi(\boldsymbol{q}_t, \boldsymbol{u}_t)}{\partial \boldsymbol{\pi}_{i, \forall i \in I_o, I_e, I_{ie}}} \tag{5}$$

with genetic algorithm is an expensive procedure. Bayesian optimization offers an alternative way to evaluate it using a probabilistic approach. The objective function (4) is approximated with a surrogate model, and then we use an acquisition function to predict where to explore or exploit instead of using the gradient (5). In the following we introduce our particular choice of the surrogate model and the acquisition function. The acquisition function tells us which is the next sample $\boldsymbol{x}_{t+1}$ to explore and the surrogate model would predict its performance f_{t+1}. After description of the termination condition, we close this section by summarizing the proposed method in algorithm 1.

Surrogate model is a probabilistic representation of the actual target function, i.e. equation (4), using previous or existing evaluations. It resembles a mapping between the hyperparameters, e.g.soft task priorities in our case, and the score, so it could be viewed as a hyper surface. There are different common options include random forests regression and Gaussian Process($\mathcal{GP}$). Grid and random search create a piecewise constant function which do not fit our continuous soft task priority tuning problem, therefore we use $\mathcal{GP}$ as our surrogate model.

$\mathcal{GP}$ is well-suited to tasks where the objective function is continuous. It is completely defined by its mean m and covariance $\mathcal{C}$ at a particular configuration $\boldsymbol{x}$: $f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), \mathcal{C}(\mathbf{x}, \mathbf{x}'))$. It means that instead of returning the a deterministic scalar value $f(\mathbf{x})$ for input $\mathbf{x}$, the $\mathcal{GP}$ returns the mean and variance of Gaussian distribution over the target function value at $\mathbf{x}$.

In Bayes optimization, the posterior distribution can be produced by the Bayes formula of prior $P(f)$ and the likelihood function $P(\mathcal{D}|f)$. So the posterior distribution is $P(f|\mathcal{D}) \propto P(\mathcal{D}|f)P(f)$. The observation data set $\mathcal{D} = \{\mathbf{x}_{1:t}, f_{1:t}\}$ is initialized by the random $\mathbf{x}_{1:t}$ and its corresponding $f_{1:t}$ obtained by the objective function f coming from the specified case, e.g. the fitness function in our problem:

$$f(\mathbf{x}) = \phi(\mathbf{x}) = \phi(\boldsymbol{q}_t, \boldsymbol{u}_t) \tag{6}$$

where $\mathbf{x}$ is the $\boldsymbol{\pi}$ in equation (4). $\mathcal{D}$ could be modeled by $\mathcal{GP}$ following the multivariate normal distribution $\mathcal{N}(0, \boldsymbol{C})$, where $\boldsymbol{C}$ is the kernel matrix. we choose the element of kernel matrix $\mathcal{C}(\mathbf{x}_i, \mathbf{x}_j)$ as a Matern 5/2 kernel: $k(r^2) = \left(1 + \sqrt{5\,r^2} + \frac{5\,r^2}{3}\right)\exp\left(-\sqrt{5\,r^2}\right)$, where $r^2 = \|\mathbf{x}_i - \mathbf{x}_j\|^2$.It equals to 1 as $(\mathbf{x}_i, \mathbf{x}_j)$ get close together and 0 as they get further apart. This condition highlights the influence of points in the vicinity, which is crucial for convergence.

We can derive the posterior predictive distribution using the Sherman-Morrison-Woodbury formula:

$$P(f_{t+1}|\mathcal{D}_{1:t}, \mathbf{x}_{t+1}) \sim \mathcal{N}(\mu_t(\mathbf{x}_{t+1}), \sigma_t(\mathbf{x}_{t+1}))$$

In a real system, the noise should be considered. So we introduce the Gaussian noise ϵ following $\epsilon \sim \mathcal{N}(0, \sigma^2_{noise})$

to transfer the target function. The mean and covariance is:

$$\sigma^2_{t,noise}(\mathbf{x}_{t+1}) = \mathcal{C}(\mathbf{x}_{t+1}, \mathbf{x}_{t+1}) - \boldsymbol{\mathcal{C}}^T\left(\boldsymbol{C} + \sigma^2_{noise} I\right)^{-1}\boldsymbol{\mathcal{C}}$$
$$\mu_{t,noise}(\mathbf{x}_{t+1}) = \boldsymbol{\mathcal{C}}^T\left(\boldsymbol{C} + \sigma^2_{noise} I\right)^{-1} f_{1:t} \tag{7}$$

Acquisition function decides how to explore the uncertainty, so we can balance between exploitation and exploration. We denote an acquisition function as $\mathcal{A}(\cdot)$,the next sampling point $\mathbf{x}_{t+1}$ to be sampled:

$$\mathbf{x}_{t+1} = \arg\max_{\mathbf{x}} \mathcal{A}(\mathbf{x}/\mathcal{D}) \tag{8}$$

Once $\mathbf{x}_{t+1}$ is defined, we can feed it to (7) to calculate the posterior predictive distribution.

Among different common acquisition functions, we choose the *Expected Improvement*. It is explicit for Gaussian posteriors and it is greedy which agrees with our goal to minimize the amount of samples. In particular, we have: $\mathcal{A}(\mathbf{x}) = \max\left(0, \mu\left(\arg\min_{\mathbf{x}}(f(\mathbf{x}))\right) - f(\mathbf{x})\right)$, where $\mu(\cdot)$ is the posterior mean calculated in (7). To escape a local objective function minimum, the EI with additive noise modify their behavior when they estimate that they are overexploiting an area. Define t_r to be the value of the exploration ratio, if the next predictive point satisfy $\sigma^2_{t+1}(\mathbf{x_{t+1}}) < t_r \sigma^2_{noise}$, the algorithm declares that x is overexploiting.

Termination condition We introduce two termination conditions: the maximum episode and limited learning rate. The algorithm terminates depending on which one would be triggered first. We terminate if it reaches maximum iteration Or alternatively the average rate of change $\Delta_f = \frac{1}{\omega}\sum_{j=1}^{w}(f_{t+w+j} - f_{t+j})/w$ is small enough. w is time window(several episodes). We terminate when $|\Delta_f| \leq \varepsilon$.

Algorithm 1 Bayesian optimization algorithm

1: **Input:** initial variable bounds of RBFN parameters $\boldsymbol{x_0}$
2: **Output:** the optimal weights of RBFN :$\boldsymbol{x}^*$
3: Random the points $\boldsymbol{x_{1:t}}$ within given bounds and generate its evaluations $y_{1:t} = f(\mathbf{x}_{1:t}) + \varepsilon_{1:t}$, build up data set $\mathcal{D}$.
4: **for** generation $g = 1, 2, 3, \cdots$ **do**
5: Update the $\mathcal{GP}$ with all data available, $P(f|\mathcal{D}) \propto \int P(\mathcal{D}|f,\theta)P(f)P(\theta)d\theta$
6: select the point $\boldsymbol{x}$ which maximizes the criterion: $\boldsymbol{x}_{t+1} = arg\max \mathcal{A}(\boldsymbol{x}|P(f|\mathcal{D}))$.
7: compute the next predictive distribution $P(f_{t+1}|\mathcal{D}, \mathbf{x}_{t+1})$
8: Reach the fixed iterations or $|\Delta_f| \leq \varepsilon$.
9: **end for**
10: **return** $\boldsymbol{x}^*$

IV. Simulation Verification

In this section, we use a pan cleaning task that is described in Section IV-A as an example to examine the proposed Algorithm 1. We can tell that the learned soft task priorities outperform the manually selected ones. More importantly, if the task changes, e.g. change the relative motion between the two arms, the learned soft task priorities still give us satisfactory results. In the comparison with learning soft task priorities using the CMA-ES, we can tell that the proposed method uses much less samples.

A. Dual-arm task

We choose two simulated Puma 560 robots, each has 6 degrees of freedom. We use the QP controller formulated in equation (1) to integrate the subtasks, which are described by equality or inequality constraints: $f_i(q) = b_i$ or $f_i(q) \leq b_i$. In particular, we on purpose choose the following subtasks, which are potentially conflicting between each other:

a) Relative position task: We use an equality constraint to define the relative position between the frying pan and the cleaning utensil:

$$f_{1-3}(\boldsymbol{q}) = p_1(\boldsymbol{q}) - p_2(\boldsymbol{q}) - d(t, x_1, y_1, z_1, x_2) = b_{1-3} \tag{9}$$

where f_{1-3} is the position tasks along three different axis x_i, y_i, z_i which are columns of $R_i \in SO(3)$. p_1 is the center of fry pan and p_2 is the tip position of the cleaning utensil, $d(t, x_1, y_1, z_1, x_2)$ is relative distance between p_1 and p_2, which is used to decide the cleaning area of the fry pan.

b) Relative orientation task: We use another inequality constraint to restrict the relative orientation within a cone:

$$f_4(\boldsymbol{q}) = x_1^T x_2 \leq b_4 \tag{10}$$

Where x_1 is the surface of normal of the fry pan and x_2 is the orientation of tip of cleaning utensil.

c) Avoiding singular configuration task: To guarantee a healthy transmission ratio between the joint space and the Cartesian space, namely, avoid the singular joint configuration, we apply the following inequality constraints:

$$f_5(\boldsymbol{q}) = \frac{-1}{2}\det\left(J_i^T J_i\right) \leq b_5, \quad i \in \{1, 2\} \tag{11}$$

where J_i is the manipulator Jacobian which is related to translational and rotational motion of the end-effector.

d) Avoiding obstacle task: Avoiding the obstacles can be formulated in the following inequalities:

$$f_6(\boldsymbol{q}) = \|\boldsymbol{p} - x_0\| \leq b_6 \tag{12}$$

where the x_0 denotes the obstacle position and $\boldsymbol{p}$ denotes a point of intersects that belongs to the two manipulators arm p_1 and p_2.

e) Joint command minimization task: For a fixed discrete time interval, $t_k \in [0, N_T]$, where N_T is number of control steps (the duration is 25s and control step is 0.2s). We use the following realization of the fitness function aftermentioned in (4):

$$\phi(\boldsymbol{q}_t, \boldsymbol{u}_t) = -\Big(\sum_{i=1}^{6} \frac{\sum_{t_k=0}^{N_T} \|\boldsymbol{b}_i(t_k) - \boldsymbol{f}_i(t_k)\|_2}{\max_{t_k \in [0, N_T]} \|\boldsymbol{b}_i(t_k) - \boldsymbol{f}_i(t_k)\|_2} + \frac{\sum_{t_k=0}^{N_T} \boldsymbol{u}(t_k)^T \boldsymbol{u}(t_k)}{\max_{t_k \in [0, N_T]} \boldsymbol{u}(t_k)^T \boldsymbol{u}(t_k)}\Big) \tag{13}$$

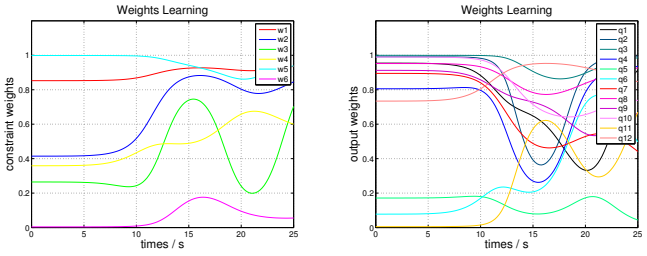
where $\|\cdot\|_2$ is the $\mathcal{L}_2$ norm (Euclidean norm). The first term of (13) penalizes the cumulated distance to the individual goal of each subtask. The second term of (13) penalizes the cumulative sum of the joint command. Both of these two terms are normalized by its own maximum.

As an example, we choose the parameters in QP controller (1) as follows: the constraint gain k_i are set to 1.0 and b_i for $i = 1, \ldots, 6$. The bound for the relative position equality constraint is: $b_{1-3} = 0$. The bound for the inequality that constrains the orientation is $b_4 = -\cos(\pi/12)$. The bound constrain the singular configuration value is $b_5 = 0.005$ and the bound on the distance to obstacle is set to $b_6 = -0.1$. We set the number of RBF in the RBFN as $n_k = 5$, the dimensionality of robot is 12, the total of objective and constraints are 6, so the total parameters in RBFN to be solved are 100.

B. comparison to hand tuning

We compare the proposed solution to static sof priorities that are predefined in an empirical way: w_i for $i = 1, \cdots, 6$. For the joint command minimizatic choose $Q_i = 0.15$, for $i = 1, \cdots, 6$ for one arm and 0.90 for $i = 7, \cdots, 12$ for another arm. On the other we obtained the soft task priorities that are shown in I In Fig. 4, we can tell that the proposed Bayesian optimi returns optimal fitness value. After about 100 episod proposed algorithm converges to the optimal fitness $\phi(\boldsymbol{q}_t, \boldsymbol{u}_t) = -0.2760$. On the contrary, through n tuning we obtained a fitness value $\phi(\boldsymbol{q}_t, \boldsymbol{u}_t) = -0.39$

We can check the performance of each task f_i in I Using the soft task priorities learned by Bayesian algo the subtask perform smoother.



(a) Soft task priorities (b) Weights of the joint command

Fig. 3 The soft task priorities w_i and the elements of the matrix Q_i are plotted in (a) and (b), respectively.

C. Constraints modification

We compare the relative trajectory between the two arms in Fig. 6a, where the two arms are desired to follow a circular motion. The trajectories obtained using the learned weights are closer to the ground truth. From Fig. 6b-6d, we changed the desired motion from a circle to a rectangle, rose and cardioid, we can find that using the learned priorities,a finer trajectories can be obtained.

D. Comparison with CMA-ES

Using the same dual-arm task, we compare the performance of the proposed method to that of the state-of-the-art

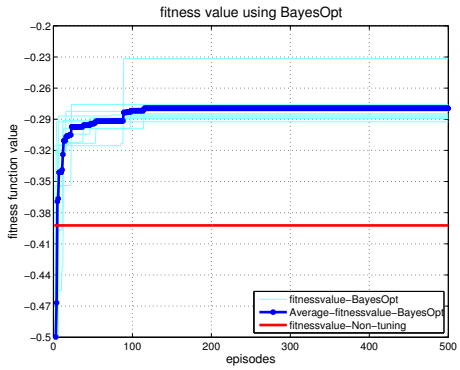


Fig. 4 This figure shows the comparison between proposed method with respect to manual tunning. x axis represents the amount of episodes, y axis represents the fitness value ϕ. The cyan lines correspond to the fitness function values by running 10 trials of Algorithm 1 and the blue line is the average lines.

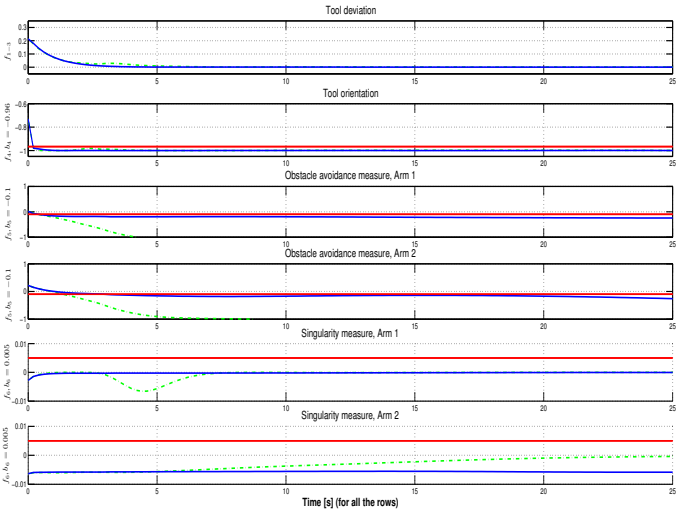


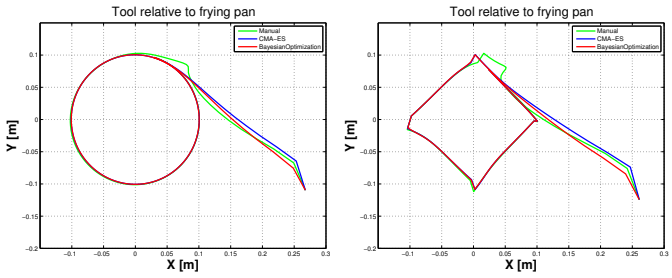
Fig. 5 Time evolution of $f_i(q)$ using different weights. The blue line indicate the f_i that are prioritized by the learned weights and the green one is prioritized by the predefined weights. The red lines denote the bound b_i for different f_i.

soft task priority learning approach CMA-ES [12]. In both cases, we use the same random initial RBFN parameters.

We run 10 trials for both cases and plot the average fitness function convergence curve in Fig. 7. First of all, we can find that both of them outperform the pre-defined weights that are given by experts manually.

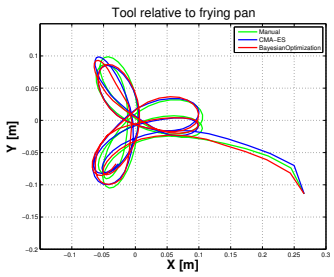
At the end, i.e. 1000 episodes, while CMA-ES can obtain a slightly smaller fitness function value ($\phi = -0.2629$) than the proposed solution ($\phi = -0.2760$), the proposed solution converges a lot faster thanks to the Bayesian optimization approach. It reaches ($\phi = -0.3$) in about 30 episodes and ($\phi = -0.28$) in about 100 episodes. As a comparison, CMA-ES needs about 300 and 500 episodes to reach the same levels respectively.

The reason is that in every episode, CMA-ES must sample $\lambda = 17$ candidates, where λ is determined by the amount of unknown parameters N as: $\lambda = 4 + \lfloor 3 \log N \rfloor$. It means that in each episode, the CMA-ES method essentially needs to evaluate the fitness function (4) for 17 times while the Bayesian optimization only evaluate it once.

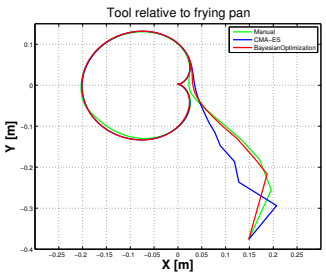


(a) Circle

(b) Rectangle



(c) Rose



(d) Cardioid

Fig. 6 From (a) to (d), we use different shapes to define the equality constraint corresponding to the relative motion between two arms.

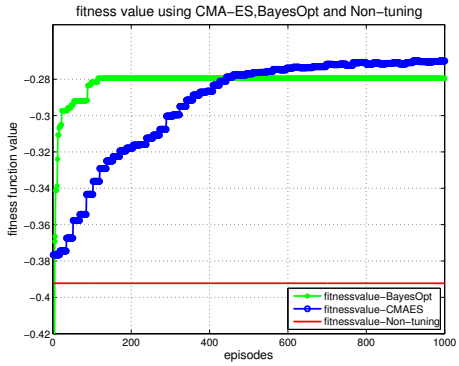


Fig. 7 In this figure we compare the proposed approach to the CMA-ES algorithm. We feed the same initial parameters and the same task to both of them. The green and blue lines are the average fitness function values using the proposed approach and CMA-ES respectively after 10 trials. Both of them performs better than the manual tunning results depicted by the red line, however the proposed approach converges a lot faster.

V. Conclusion and Future Work

In this paper we explore a sample-efficient approach to tune the soft task priorities of a QP controller. Inspired from [12], we model the soft task priorities with a normalized RBFN and then tune the hyperparameters of the RBFN to obtain the optimal soft task priorities.

The proposed solution is validated through a dual-arm simulation, where two 6DOFs arms are controlled with the QP to satisfy multiple subtasks simultaneously. The proposed solution outperforms the predefined fixed soft task priorities. When we change the relative translation equality constraint, the soft task priorities obtained from the proposed approach still works in a good way. Comparing to the evolution algorithm, i.e. CMA-ES, we can find that the proposed approach converges a lot faster and uses a lot less computational resources.

References

[1] K. Miettinen, *Nonlinear multiobjective optimization*. Springer Science & Business Media, 2012, vol. 12.

[2] A. L. P. Ureche, K. Umezawa, Y. Nakamura, and A. Billard, "Task parameterization using continuous constraints extracted from human demonstrations," *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1458–1471, 2015.

[3] J. Kober and J. Peters, "Learning prioritized control of motor primitives," in *Learning Motor Skills*. Springer, 2014, pp. 149–160.

[4] C. Towell, M. Howard, and S. Vijayakumar, "Learning nullspace policies," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, 2010, pp. 241–248.

[5] R. Martinez-Cantin, "Bayesopt: A bayesian optimization library for nonlinear optimization, experimental design and bandits," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3735–3739, 2014.

[6] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2016.

[7] 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.

[8] D. Drieß, P. Englert, and M. Toussaint, "Constrained bayesian optimization of combined interaction force/task space controllers for manipulations," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 902–907.

[9] A. Marco, P. Hennig, J. Bohg, S. Schaal, and S. Trimpe, "Automatic lqr tuning based on gaussian process global optimization," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 270–277.

[10] F. Berkenkamp, A. P. Schoellig, and A. Krause, "Safe controller optimization for quadrotors with gaussian processes," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 491–496.

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

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

[13] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[14] R. Lober, V. Padois, and O. Sigaud, "Efficient reinforcement learning for humanoid whole-body control," in *Humanoid Robots (Humanoids), 2016 IEEE-RAS 16th International Conference on*. IEEE, 2016, pp. 684–689.

[15] Y. Wang, F. Vina, Y. Karayiannidis, C. Smith, and P. Ögren, "Dual arm manipulation using constraint based programming," in *19th IFAC World Congress*, Cape Town, South Africa, August 2014.

[16] J. Vaillant, K. Bouyarmane, and A. Kheddar, "Multi-character physical and behavioral interactions controller," *IEEE transactions on visualization and computer graphics*, 2016.

[17] 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.

[18] P. Ögren, C. Smith, Y. Karayiannidis, and D. Kragic, "A multi objective control approach to online dual arm manipulation," in *International IFAC Symposium on Robotic Control*, Dubrovnik, Croatia, Sep 2012, pp. 747–752.

[19] B. Berret, E. Chiovetto, F. Nori, and T. Pozzo, "Evidence for composite cost functions in arm movement planning: an inverse optimal control approach," *PLoS computational biology*, vol. 7, no. 10, p. e1002183, 2011.

[20] S. Ivaldi, O. Sigaud, B. Berret, and F. Nori, "From humans to humanoids: the optimal control framework," *Paladyn, Journal of Behavioral Robotics*, vol. 3, no. 2, pp. 75–91, 2012.