Real-time tuning soft task priorities with quadratic programming

Yiyao Zhu[†1,2], Jian Li[†1,2], Yuquan Wang[1,2], Yinyin Su[3], Yongquan Chen[*1,2]

***Abstract*— When robot simultaneously executes multiple tasks with potential incompatibilities, in order to realize objectives and satisfy constraints, how to assign appropriate priorities to each task is an open research problem. Most of the existing methods are only aimed at tuning task weights in static scenario. In this paper, for the scenario with randomly dynamic factors, we propose a real-time task priorities tuning method based on global objectives and constraints to deal with conflicting tasks for adapting environmental changes. The soft task priorities are real-time computed by a constrained optimization problem at each time point. For fast computation and smooth transition to task weights, we transform the problem to the formation of quadratic programming. We benchmark our method on a simulated 6 DOF UR5 arm comparing with real-time Bayesian Optimization tuning and no-tuning. And we also validate the effectiveness of our method in experiments with randomly dynamic obstacle or target.**

I. INTRODUCTION

Multiple tasks can be executed simultaneously by a highly redundant robot, such as humanoids and complex manipulators. But undesired motion will be generated when there are incompatibilities between different tasks. Especially for the problem that is about to be discussed and solved in this paper, aimed at the dynamic scenario with random factors, soft task priorities (will be explicitly explained in the next paragraph) should be real-time fast computed and be in smooth transition [1] for automatically adapting the dynamic scenario.

Task priorities are introduced to scalarize the procedure by assigning a weight to each task satisfying multiple objectives and constraints in a sequential way. Typically task priorities are employed to assure the completion of critical tasks, such as avoiding joint limits. There are two main concepts that can be found in the literature: strict task priorities [2], [3] and soft task priorities [1], [4]. Strict task priorities is a hierarchical scheme designed to ensure directly executing high-priority tasks that are considered more important than the others and low-priority tasks are in the null-space in terms of higher priority tasks [5], [6]. Tasks with high-priority are guaranteed to be executed while the tasks with low-priority may not be achieved. Soft task priorities, which are generally learned offline [7] from demonstration [8], is a weighting strategy that assigns a weight to each task representing its relative importance with respect to other tasks. This method can lead to continuous change of weights so that smooth transitions will be generated.

† The authors are contributed equally to this work.
1 Shenzhen Institute of Artificial Intelligence and Robotics for Society.
2 Institute of Robotics and Intelligent Manufacturing, The Chinese University of Hong Kong, Shenzhen, China.
3 Department of Mechanical and Energy Engineering, Southern University of Science and Technology, Shenzhen, China.
* Corresponding author. email: yqchen@cuhk.edu.cn

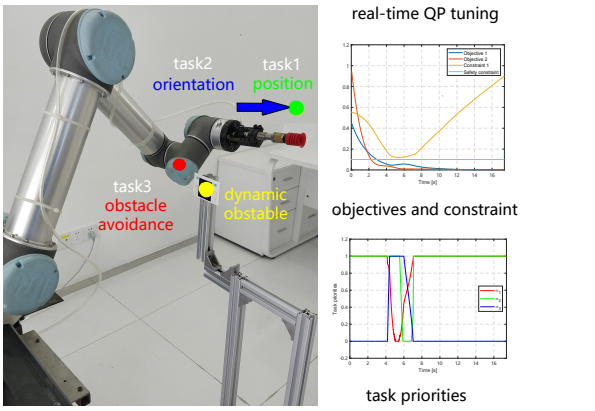

Fig. 1 The UR5 arm reaches a desired position while maintaining certain orientation with end-effector and avoiding dynamic obstacle. Our method real-time tunes task priorities depending on global objectives and constraints.

Dealing with incompatibilities between tasks, the undesired robot movements typically happen if the task priorities are not tuned precisely [9]. For priorities tuning, some articles propose to automatically offline learn task priorities using parameters auto-tuning algorithm such as Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [10], Bayesian Optimization (BO) [11], etc. The task priorities are learned by optimizing a specific fitness function. By this mean, the issue of incompatibilities between tasks can be addressed and also the global motion can be optimized. But this is only applicable in static scenario. When it comes to dynamic scenario with random factors such as randomly dynamic target or obstacle, the task priorities learned offline are not corresponding to the real application. Therefore we resort to an online tuning method.

There are some primary explorations for real-time tuning soft task priorities. We can online tune task weights depending on their variance, and the method could solve difficult control problems but fails with incompatibility, see [9]. By mixture of torque controllers, we can react dynamic scenario by selecting task weight policy learned offline which means that too much training has to be done in advance, see [12]. By adding dynamic-consistency to Generalized Hierarchical Control (GHC), we can smoothly rearrange task priorities dealing with environmental changes, but the method is only verified with limited number of tasks, see [13].

We hereby propose our method derived from the algorithm about task priorities automatically learned offline [10], [11], [14] where the task weights are learned by optimizing global objectives while satisfying constraints. Inspired by it, in this paper, based on global objectives and constraints, the real-time tuning method is proposed to automatically deal

with dynamic scenario about random factors (eg. randomly dynamic obstacle or target in the robot workspace). We employ a simple controller solved as a Least-Squares problem, then superimpose the joint commands by computing a weighted sum of each controller output as used in [15], [16], [17]. By employing mixture of controllers above, the real-time optimization problem is transformed to a quadratic programming problem. The contribution of this paper is proposing a real-time tuning method for the scenario with randomly dynamic factors based on global objectives and constraints. The method could fast output task weights with smooth transition, then real-time deal with conflicting tasks to realize objectives and satisfy constraints.

We benchmark our method comparing with no-tuning or real-time BO tuning method in the simulation where 6 DOF UR5 manipulator bypasses obstacle to reach the target in the static scenario. Moreover, we validate our method in the experiment with dynamic obstacle or target.

The remainder is organized as the following. Section II states the proposed real-time tuning method for task priorities, including single task controller, multi-task controller, real-time tuning procedure and robot structure constraints. Section III presents the simulation and experimental results. Section IV draws conclusion and discusses the possible future work.

II. PROPOSED METHOD

In Section II-A, for each sub-task, we introduce Least-Squares to fast obtain joint accelerations. In Section II-B, we define the formulation of multi-task controller with soft task priorities $\boldsymbol{\alpha}$. In Section II-C, we propose to real-time tune the task priorities based on global objectives and constraints. For simplifying computation, we transform the problem to the formation of quadratic programming. In Section II-D, we add robot structure constraints (eg. joint angle, velocity and acceleration bounds) by limiting task weight $\boldsymbol{\alpha}$.

A. Controller for a single sub-task

For the i-th single sub-task controller, we compute joint accelerations using Least-Squares. Given an n DOF robot, we define e, $\dot{e}$ and $\ddot{e} \in \mathbb{R}^n$ as the task error, its derivative and second derivative for the i-th single sub-task:

$$e = f_i - f_i^* \tag{1}$$

$$\dot{e} = \dot{f}_i - \dot{f}_i^* = \frac{\partial f_i}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}}_i - \dot{f}_i^* = J_{f_i}\dot{\boldsymbol{q}}_i - \dot{f}_i^* \tag{2}$$

$$\ddot{e} = \dot{J}_{f_i}\dot{\boldsymbol{q}}_i + J_{f_i}\ddot{\boldsymbol{q}}_i - \ddot{f}_i^{\,*} \tag{3}$$

where $\boldsymbol{q}_i$, $\dot{\boldsymbol{q}}_i$ and $\ddot{\boldsymbol{q}}_i \in \mathbb{R}^n$ represent joint positions, velocities and accelerations respectively. J_{f_i} and $\dot{J}_{f_i}$ are task Jacobian and its derivative. f_i and $\dot{f}_i$ are the task item and its derivative. $f_i{}^*$ and $\dot{f}_i^*$ are the task objective and its derivative. The expected error dynamics is:

$$\ddot{e} = -k_d\dot{e} - k_p e \tag{4}$$

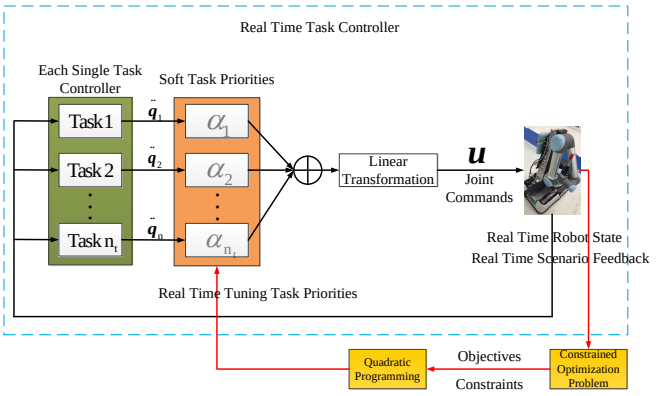

Fig. 2 In the inner loop, the joint commands $\boldsymbol{u}$ could be joint velocities or joint torques which consist of the output from single task controller weighted by soft task priorities. In the out loop, the task weight $\boldsymbol{\alpha}$ is learned by quadratic programming depending on objectives and constraints. The real time task weight tuning is on the basis of real time robot state and scenario feedback.

where k_d and k_p are the derivative and proportional gains respectively. We minimize the following equation to achieve the expected error dynamics in equation (4) when supposing $\ddot{f}_i^* = 0$.

$$e_{f_i} = \ddot{e} + k_d\dot{e} + k_p e = J_{f_i}\ddot{\boldsymbol{q}}_i + \underbrace{\left(\dot{J}_{f_i} + k_d J_{f_i}\right)\dot{\boldsymbol{q}}_i - k_d\dot{f}_i^* + k_p e - \dot{f}_i^*}_{C_{e_{f_i}}} \tag{5}$$

Then the Least-Squares is given by:

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

For the joint accelerations $\ddot{\boldsymbol{q}}_i$ to the i-th single sub-task, employing analytical expression of Least-Squares to solve (6), we obtain the final result from sub-task controller:

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

where ${J_{f_i}}^\dagger$ is the Moore-Penrose pseudoinverse to the J_{f_i}.

B. Controller for multiple sub-tasks with soft priorities

Following fig. 2, we combine n_t sub-tasks to accomplish global mission. And each single sub-task is weighted by task priority α at the time t as:

$$\ddot{\boldsymbol{q}}_t = \sum_{i=1}^{n_t} \alpha_{i,t}\ddot{\boldsymbol{q}}_{i,t} \tag{8}$$

where $\alpha_{i,t}$ is the task priority for the i-th task at the time t computed by real-time tuning method, $\ddot{\boldsymbol{q}}_{i,t} \in \mathbb{R}^n$ are the joint accelerations output from i-th single sub-task controller in (7), $\ddot{\boldsymbol{q}}_t \in \mathbb{R}^n$ are the final joint accelerations by mixture of controllers.

In order to conveniently using (8) in the following formulation, we apply $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_{n_t}]^T$ and $[\ddot{\boldsymbol{q}}] = [\ddot{\boldsymbol{q}}_1, \ddot{\boldsymbol{q}}_2, \ldots, \ddot{\boldsymbol{q}}_{n_t}]$ to simplify the expression of it, the equal equation is given by:

$$\ddot{\boldsymbol{q}}^* = [\ddot{\boldsymbol{q}}]\boldsymbol{\alpha} \tag{9}$$

C. Real-time tuning soft task priorities

The ultimate purpose for tuning soft task weight is leading robot to realize global objectives while satisfy constraints. Hence, we design a real-time tuning method based on objectives and constraints to deal with conflicting tasks.

For the next predicted time point $t+1$, the objectives and constraints are formulated as:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \sum_{i \in I_o} \varepsilon_i g_{o,i}(\boldsymbol{q_{t+1}}, t+1) \\ s.t. \quad & g_{c,i}(\boldsymbol{q_{t+1}}, t+1) \leq b_i, \quad \forall i \in I_{ie}, \\ & g_{c,i}(\boldsymbol{q_{t+1}}, t+1) = b_i, \quad \forall i \in I_e \end{aligned} \tag{10}$$

where I_o is the objectives, which should finally be realized when robot stops execution. I_{ie} and I_e represent inequality constraints and equality constraints respectively, and each constraint must be satisfied during the whole execution procedure. ε_i is the static scale factor to balance every value of objective. Up to now, we formulate the objectives and constraints based tuning problem as a real-time constrained optimization problem.

As the mapping between task weights $\boldsymbol{\alpha}$ and g_o or g_c is complex, for fast computation and smooth transition to the task weights, we resort to derivative formation of g_o and g_c proposed in [18] to generate task weights at each time step. The real-time constrained optimization problem is reformulated as:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \sum_{i \in I_o} \varepsilon_i \frac{\partial {g_{o,i}}^T}{\partial \boldsymbol{q}} \dot{\boldsymbol{q}}_{t+1} \\ s.t. \quad & \frac{\partial {g_{c,i}}^T}{\partial \boldsymbol{q}} \dot{\boldsymbol{q}}_{t+1} \leq -k_i(g_{c,i} - b_i) - \frac{\partial g_{c,i}}{\partial t}, \quad \forall i \in I_{ie}, \\ & \frac{\partial {g_{c,i}}^T}{\partial \boldsymbol{q}} \dot{\boldsymbol{q}}_{t+1} = -k_i(g_{c,i} - b_i) - \frac{\partial g_{c,i}}{\partial t}, \quad \forall i \in I_e \end{aligned} \tag{11}$$

Where k_i and b_i are scaling gain and bound respectively for the i-th constraint, $\dot{\boldsymbol{q}}_{t+1} \in \mathbb{R}^n$ is the predicted joint velocities at the next time point.

Noted: The controller proposed in [18] is typically designed for multiple tasks. In other words, the controller should include all tasks defined for humanoid robot. Apparently different from the controller, our objectives and constraints based method only include global objectives and constraints which are generally not identical with the tasks included by the controller.

Using (9), we replace $\dot{\boldsymbol{q}}_{t+1}$ with task priorities $\boldsymbol{\alpha}$ employing the mapping between joint accelerations and velocities as:

$$\dot{\boldsymbol{q}}_{t+1} = \dot{\boldsymbol{q}}_t + \ddot{\boldsymbol{q}}^* \Delta t = \dot{\boldsymbol{q}}_t + [\ddot{\boldsymbol{q}}] \boldsymbol{\alpha} \Delta t \tag{12}$$

Where Δt is the task weights tuning cycle, $\dot{\boldsymbol{q}}_t \in \mathbb{R}^n$ is the joint velocities at the time t. Then we apply equation(12) to replace $\dot{\boldsymbol{q}}_{t+1}$, where $\ddot{\boldsymbol{q}}_i$ is directly computed by the i-th single sub-task controller. The equation now is:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \boldsymbol{\alpha}^T Q_\alpha \boldsymbol{\alpha} + \sum_{i \in I_o} \varepsilon_i \frac{\partial {g_{o,i}}^T}{\partial \boldsymbol{q}} [\ddot{\boldsymbol{q}}] \Delta t \boldsymbol{\alpha} \\ s.t. \quad & \frac{\partial {g_{c,i}}^T}{\partial \boldsymbol{q}} [\ddot{\boldsymbol{q}}] \Delta t \boldsymbol{\alpha} \leq -k_i(g_{c,i} - b_i) - \frac{\partial g_{c,i}}{\partial t} - \frac{\partial {g_{c,i}}^T}{\partial \boldsymbol{q}} \dot{\boldsymbol{q}}_t, \\ & \frac{\partial {g_{c,i}}^T}{\partial \boldsymbol{q}} [\ddot{\boldsymbol{q}}] \Delta t \boldsymbol{\alpha} = -k_i(g_{c,i} - b_i) - \frac{\partial g_{c,i}}{\partial t} - \frac{\partial {g_{c,i}}^T}{\partial \boldsymbol{q}} \dot{\boldsymbol{q}}_t \end{aligned} \tag{13}$$

where the definition of $g_{c,i}$ in inequality or equality constraints is $\forall i \in I_{ie}$ or $\forall i \in I_e$ respectively. We smooth the task weights $\boldsymbol{\alpha}$ transition by adding a quadratic regularization term $\boldsymbol{\alpha}^T Q_\alpha \boldsymbol{\alpha}$. And Q_α is a diagonal positive-definite matrix. In order to balance the value between quadratic and primary terms, the parameters in the Q_α is adaptive with the value of primary item.

D. Add robot structure constraints

The robot structure constraints (eg. joint angle, velocity and acceleration bounds) should be satisfied for safety during tuning task weight. And they are bounded as:

$$\begin{aligned} \boldsymbol{q}_{min} \leq \boldsymbol{q}_{t+1} \leq \boldsymbol{q}_{max} \\ \dot{\boldsymbol{q}}_{min} \leq \dot{\boldsymbol{q}}_{t+1} \leq \dot{\boldsymbol{q}}_{max} \\ \ddot{\boldsymbol{q}}_{min} \leq \ddot{\boldsymbol{q}}^* \leq \ddot{\boldsymbol{q}}_{max} \end{aligned} \tag{14}$$

where $\boldsymbol{q}_{t+1}$, $\dot{\boldsymbol{q}}_{t+1}$ and $\ddot{\boldsymbol{q}}^* \in \mathbb{R}^n$ are the joint angles, velocities and accelerations respectively. With equation $\boldsymbol{q}_{t+1} = \boldsymbol{q}_t + \frac{1}{2}\ddot{\boldsymbol{q}}^* \Delta t^2 = \boldsymbol{q}_t + \frac{1}{2}[\ddot{\boldsymbol{q}}] \boldsymbol{\alpha} \Delta t^2$, equation (9) and equation (12), we describe the constraints of joint angles, velocities and accelerations in terms of task priorities $\boldsymbol{\alpha}$ respectively in order to limit robot structure constraints by limiting task weights $\boldsymbol{\alpha}$. Then the robot structure constraints are transformed as:

$$\begin{aligned} \boldsymbol{q}_{min} \leq \boldsymbol{q}_t + \frac{1}{2}[\ddot{\boldsymbol{q}}] \boldsymbol{\alpha} \Delta t^2 \leq \boldsymbol{q}_{max} \\ \dot{\boldsymbol{q}}_{min} \leq [\ddot{\boldsymbol{q}}] \Delta t \boldsymbol{\alpha} + \dot{\boldsymbol{q}}_t \leq \dot{\boldsymbol{q}}_{max} \\ \ddot{\boldsymbol{q}}_{min} \leq [\ddot{\boldsymbol{q}}] \boldsymbol{\alpha} \leq \ddot{\boldsymbol{q}}_{max} \end{aligned} \tag{15}$$

We add constraints in equation (15) to equation (13). For the task weight output stability reasons, we limit task weights $\boldsymbol{\alpha}$ to range $[0, 1]$. Then the real-time tuning method based on objectives and constraints is now completely proposed and we hereby name the method as real-time Quadratic Programming (QP) tuning.

III. Experiments

In III-A, we benchmark the real-time QP tuning method on a simulated UR5 arm. For ideally comparing with other methods, we hereby choose a static simulation scenario. In III-B and III-C, we present the drawback of no-tuning and real-time Bayesian Optimization (BO) tuning respectively comparing with the proposed real-time QP tuning in the same simulated scenario. The benchmark result shows that real-time QP tuning outperforms the other two methods. In III-D, we validate our method in the experiment with randomly dynamic factors, the result shows that our method could efficiently realize dynamic objectives and satisfy constraints by QP real-time tuning task weights.

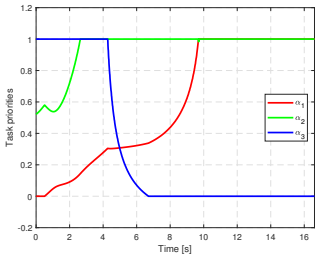

Fig. 3 QP tuning: The task priorities.

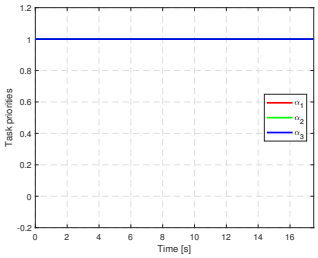

Fig. 4 No-tuning: The task priorities.

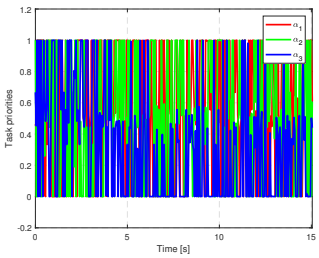

Fig. 5 BO tuning: The task priorities.

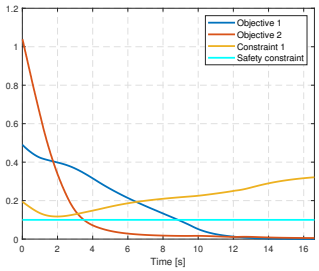

Fig. 6 QP tuning: Objectives and constraint.

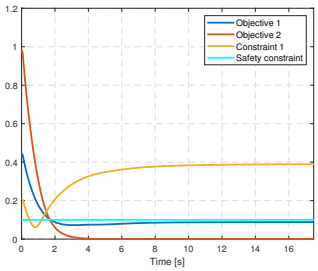

Fig. 7 No-tuning: Objectives and constraint.

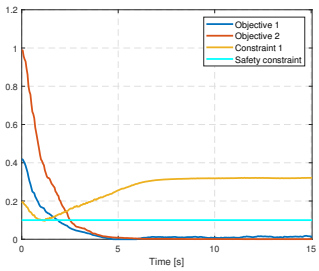

Fig. 8 BO tuning: Objectives and constraint.

Benchmark the proposed method. The values of objective1 and objective2 are g_{o1} and $g_{o2}+1$ respectively ($+1$ is to make the value of objective2 as 0 that is identical with objective1 when it is realized). Constraint1 is the distance between end-effector and obstacle. Safety constraint is the safe distance. Depending on our method, the task weight tuning principle is: the constraint is given priority to be satisfied when approximating the bound while the global objectives are realized with the full ability when the constraint value keeps away from the bound. For the ability of satisfying constraint and accomplishing objectives, the sequence from the good to the bad is QP tuning (constraint1 does not break bound; objectives go to zero), BO tuning (constraint1 breaks bound a little; objectives nearly go to zero), no-tuning (constraint1 breaks bound a lot; objectives do not go to zero).

A. Real-time QP tuning in simulation

The simulation scenario is designed as following: The UR5 arm (6 DOF) bypasses a static obstacle to reach the desired position with the end-effector while maintaining the desired orientation.

There are three sub-tasks given as following to satisfy the requirements proposed in the scenario. Task1, the UR5 end-effector reaches the desired position $\boldsymbol{p}^* = [0.6, -0.5, 0.11]$ in Cartesian space. Task2, keep the end-effector at a certain orientation. Task3, keep the desired joint configuration $[-2.8, -2.0, -1.15, 0, 0, 0](radian)$ for the arm bypassing a static obstacle.

As the three sub-tasks given above are potentially conflicting, the robot can not achieve the goal by simultaneously executing all tasks. Therefore, we apply our method real-time QP tuning to deal with the incompatibilities by tuning task weights. We design the objectives: reach the desired position in g_{o1} and keep the certain orientation in g_{o2}. Moreover, we define the constraint: avoid obstacle in g_{c1}.

$$g_{o1}(\boldsymbol{q}) = \|\boldsymbol{p} - \boldsymbol{p}^*\|_2 \tag{16}$$

where $\boldsymbol{p} \in \mathbb{R}^3$ is the position of the end-effector, $\boldsymbol{p}^* \in \mathbb{R}^3$ is the desired position and $\boldsymbol{q} \in \mathbb{R}^n$ is the joint configuration.

$$g_{o2}(\boldsymbol{q}) = {}^O\boldsymbol{x}^T \mathbf{n_d} \tag{17}$$

where ${}^O\boldsymbol{x}$ is the orientation axis of the end-effector and $\mathbf{n_d}$ is the desired orientation.

$$g_{c1}(\boldsymbol{q}) = -\|\boldsymbol{p} - \boldsymbol{p}_{so}\|_2 \leq b_{c1} < 0 \tag{18}$$

where $\boldsymbol{p}_{so} \in \mathbb{R}^3$ is the static obstacle position, b_{c1} is the safety distance to the obstacle.

Based on the objectives g_{o1}, g_{o2} and the constraint g_{c1}, with our real-time QP tuning method in equation (13), the optimization problem is:

$$\begin{aligned} &\min_{\boldsymbol{\alpha}} \quad \boldsymbol{\alpha}^T Q_\alpha \boldsymbol{\alpha} + (\varepsilon_1 \frac{\partial g_{o1}}{\partial \boldsymbol{q}}^T + \varepsilon_2 \frac{\partial g_{o2}}{\partial \boldsymbol{q}}^T)[\ddot{\boldsymbol{q}}]\Delta t \boldsymbol{\alpha} \\ &s.t. \quad \frac{\partial g_{c1}}{\partial \boldsymbol{q}}^T [\ddot{\boldsymbol{q}}]\Delta t \boldsymbol{\alpha} \leq -k(g_{c1} - b_{c1}) - \frac{\partial g_{c1}}{\partial \boldsymbol{q}}^T \dot{\boldsymbol{q}}_t \end{aligned} \tag{19}$$

where the objectives are realized by g_{o1} and g_{o2}, the constraint is satisfied by g_{c1}. Other parameters in the function have been explained in Section II.

As showed in 6, the safety constraint is satisfied during the whole process which means that the robot arm successfully bypasses the obstacle. Additionally, we can also see that objectives are accomplished as their values finally go to zero. In conclusion, the robot satisfies global requirements.

B. Comparison with no-tuning in simulation

In no-tuning case, we directly apply the multi-task controller proposed in Section II-B, and each task priority $\boldsymbol{\alpha}$

are set to 1 shown in fig 4. The method can not deal with conflicting tasks. Hence in fig. 7 we can see that the constraint is not satisfied as the constraint1 is below the safety constraint and the objectives are not realized as the values do not go to zero.

C. Comparison with real-time BO tuning in simulation

We hereby compare our method with Bayesian Optimization (BO) for the reasons that the task weight learned offline by BO or CMA-ES [10], [11] is also based on global objective and constraint. The global fitness function is employed to accomplish objective and the penalty item is used to satisfy constraints. For the task priorities real-time tuned by Bayesian Optimization (BO), the fitness function is designed as:

$$\phi_{t+1}(\boldsymbol{q}_{t+1}) = \varepsilon_1 g_{o1}(t+1) + \varepsilon_2 g_{o2}(t+1) \tag{20}$$

where ε_1 and ε_2 are scaling factors, $g_{o1}(t+1)$ and $g_{o2}(t+1)$ are the predicted objective function, ϕ_{t+1} is the predicted fitness value, $\boldsymbol{q}_{t+1} \in \mathbb{R}^n$ is the predicted joint angles at the next time point.

Based on the multi-task controller in Section II-B, we map task weights $\boldsymbol{\alpha}$ with joint angles $\boldsymbol{q}_{t+1}$. Then we real-time obtain task weights by minimizing ϕ_{t+1}:

$$\boldsymbol{\alpha}^* = \arg\min_{\boldsymbol{\alpha}} \ \phi_{t+1}(\boldsymbol{q}_{t+1}) \tag{21}$$

We satisfy constraints by adding penalty item to the fitness value. Concretely speaking, we set fitness function to $1000 >> \phi_{t+1}$ whenever the predicted robot performance violates constraints eg. collision with obstacle, breaking joint angle, velocity or acceleration bounds, etc.

From fig. 8, the objective values in BO tuning are finally close to zero which is better than the no-tuning case. From fig. 5, the task priorities are chattering which leads to generation of jerky trajectory and instability of robotic dynamic system. Meanwhile, from Tab. I, with 20 generations, BO tuning is practically inapplicable for the reason that the computation is extremely time-consuming especially for high dimensions and could not meet the requirement of high frequency control of online tuning.

TABLE I: The computation time comparison.

Dimentions	BO solving time (s)	QP solving time (s)
2	0.029272	0.000169
4	0.031663	0.000354
6	0.032729	0.000581
8	0.034362	0.000913
10	0.037663	0.001456

D. Experiment with randomly dynamic target or obstacle

For the scenario with randomly dynamic factors, the task weights learned offline is not corresponding to the practical scenario for lacking dynamic scenario information. Meanwhile, it is also apparently impossible to manually tune the task weights not knowing in advance how the scenario changes. Therefore, we should apply our real-time QP tuning method to dealing with randomly dynamic factors in the scenario. Then we validate our method in the experiment with randomly dynamic target or obstacle.

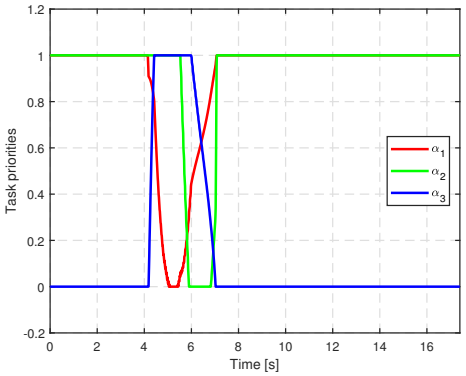

Fig. 9 QP tuning: The task priorities (dynamic obstacle).

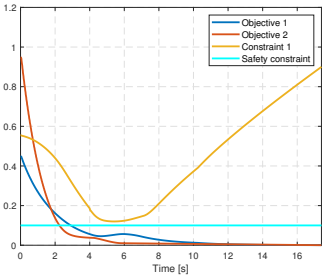

Fig. 10 QP tuning.

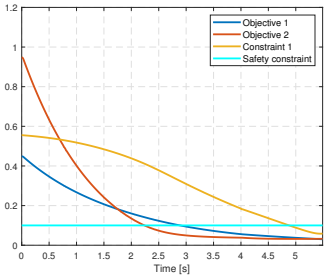

Fig. 11 No-tuning.

Objectives and constraint. With randomly dynamic obstacle in the workspace, the robot avoids a dynamic obstacle and accomplishes objectives benefit from weight tuning by our method. Comparison with no-tuning, constraint1 violates the safety bound and fails executing mission.

1) Randomly dynamic obstacle: Shown in fig. 1, the experimental scenario is almost the same as the simulation mentioned in Section III-A except for randomly dynamic obstacle. We add randomly moving AGV(automated guided vehicle) as a dynamic obstacle in the workspace of the UR5 arm. In order to obtain the position of dynamic obstacle and target object, we attach QR code on the obstacle and target object. And the camera Kinect Sensor fixed on the robot is used to acquire the position of obstacle and target object by recognizing QR code.

There are three sub-tasks. The first and the second sub-tasks are the same as the tasks in Section III-A. We change the third task to maintain the safety constraint satisfied. The third sub-task here is about avoiding dynamic task with 4th joint (For simplifying experiment, the UR5 arm avoids dynamic obstacle only by single joint).

For real-time objectives and constraints based QP tuning, the objectives remain the same as the simulation and the constraint is formulated as:

$$g_{c2}(\boldsymbol{q}) = -\|\boldsymbol{p}_4 - \boldsymbol{p}_{do}\|_2 \leq b_{c2} < 0 \tag{22}$$

where $\boldsymbol{p}_4$ is the Cartesian pose of the 4th joint, $\boldsymbol{p}_{do} \in \mathbb{R}^3$ is the dynamic obstacle position real-time acquiring by Kinect

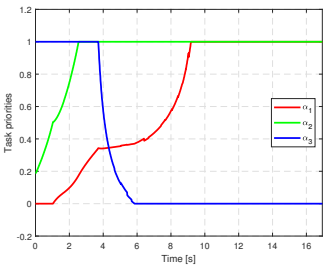

Fig. 12 The task priorities.

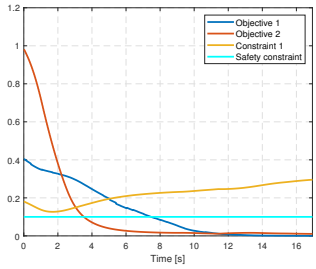

Fig. 13 Objectives and constraint.

QP tuning. In the scenario with randomly dynamic target, real-time tuned by QP, the objective1 and objective2 go to zero while constraint1 is in the safety bound. The robot successfully finishes global mission.

Sensor, b_{c2} is the safety distance to the obstacle.

Here, the real-time tuning problem can be formulated as the following:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \boldsymbol{\alpha}^T Q_\alpha \boldsymbol{\alpha} + (\varepsilon_1 \frac{\partial g_{o1}}{\partial \boldsymbol{q}}^T + \varepsilon_2 \frac{\partial g_{o2}}{\partial \boldsymbol{q}}^T)[\ddot{\boldsymbol{q}}]\Delta t \boldsymbol{\alpha} \\ s.t. \quad & \frac{\partial g_{c2}}{\partial \boldsymbol{q}}^T [\ddot{\boldsymbol{q}}]\Delta t \boldsymbol{\alpha} \leq -k(g_{c2} - b_{c2}) - \frac{\partial g_{c2}}{\partial \boldsymbol{q}}^T \dot{\boldsymbol{q}}_t \end{aligned} \tag{23}$$

From fig. 11 and fig. 10, our method successfully leads robot to avoid dynamic obstacle and realize the objectives. While for the no-tuning, constraint1 breaks safety constraint and does not realize objectives at all.

2) Randomly dynamic target: The experiment is designed as the simulation scenario in Section III-A and the only difference is the end-effector desired position that is dynamic in this case. The three given sub-tasks, objectives and constraint are identical with the formulation in Section III-A. From fig. 12 and fig. 13, by real-time tuning the task weight, the robot realizes the dynamic objectives while avoiding obstacle.

IV. Conclusion and Future Work

In this paper, we have proposed a real-time tuning method to deal with incompatibilities between conflicting tasks. The novelty of this paper lies in employing objectives and constraints to tune the task weights of conflicting tasks in the randomly dynamic scenario. Then we can turn the online tuning problem to a quadratic programming problem. Performance is verified by running simulations and experiments of a 6 DOF robot arm.

We plan to implement this online tuning method to highly redundant robot, e.g. humanoid. Additionally, we will verify the feasibility of the proposed method when the number of tasks is increasing.

Acknowledgement

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.

References

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

[2] 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.

[3] F. Flacco, A. De Luca, and O. Khatib, "Prioritized multi-task motion control of redundant robots under hard joint constraints," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 3970–3977.

[4] 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.

[5] V. Ortenzi, M. Adjigble, J. A. Kuo, R. Stolkin, and M. Mistry, "An experimental study of robot control during environmental contacts based on projected operational space dynamics," in *2014 IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2014, pp. 407–412.

[6] M. Gienger, H. Janssen, and C. Goerick, "Task-oriented whole body motion for humanoid robots," in *5th IEEE-RAS International Conference on Humanoid Robots, 2005.* IEEE, 2005, pp. 238–244.

[7] M. Liu, S. Hak, and V. Padois, "Generalized projector for task priority transitions during hierarchical control," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 768–773.

[8] J. Silvério, S. Calinon, L. Rozo, and D. G. Caldwell, "Learning task priorities from demonstrations," *IEEE Transactions on Robotics*, vol. 35, no. 1, pp. 78–94, 2018.

[9] R. Lober, V. Padois, and O. Sigaud, "Variance modulated task prioritization in whole-body control," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 3944–3949.

[10] 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.

[11] 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)*, 2018, pp. 1–6.

[12] N. Dehio, R. F. Reinhart, and J. J. Steil, "Continuous task-priority rearrangement during motion execution with a mixture of torque controllers," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2016, pp. 264–270.

[13] N. Dehio and J. J. Steil, "Dynamically-consistent Generalized Hierarchical Control," in *IEEE/RSJ Int. Conf. on Robotics and Automation*, 2019.

[14] 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.

[15] A. Dietrich, T. Wimböck, and A. Albu-Schäffer, "Dynamic whole-body mobile manipulation with a torque controlled humanoid robot via impedance control laws," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 3199–3206.

[16] A. Dietrich, T. Wimbock, A. Albu-Schaffer, and G. Hirzinger, "Reactive whole-body control: Dynamic mobile manipulation using a large number of actuated degrees of freedom," *IEEE Robotics & Automation Magazine*, vol. 19, no. 2, pp. 20–33, 2012.

[17] 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.

[18] 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, 2014.