2016 IEEE International Conference on Robotics and Automation (ICRA)
Stockholm, Sweden, May 16-21, 2016

Learning soft task priorities for control of redundant robots

Valerio Modugno[1,4], Gerard Neumann[2], Elmar Rueckert[2], Giuseppe Oriolo[1], Jan Peters[2,3], Serena Ivaldi[2,4]

***Abstract*— One of the key problems in planning and control of redundant robots is the fast generation of controls when multiple tasks and constraints need to be satisfied. In the literature, this problem is classically solved by multi-task prioritized approaches, where the priority of each task is determined by a weight function, describing the task strict/soft priority. In this paper, we propose to leverage machine learning techniques to learn the temporal profiles of the task priorities, represented as parametrized weight functions: we automatically determine their parameters through a stochastic optimization procedure. We show the effectiveness of the proposed method on a simulated 7 DOF Kuka LWR and both a simulated and a real Kinova Jaco arm. We compare the performance of our approach to a state-of-the-art method based on soft task prioritization, where the task weights are typically hand-tuned.**

I. Introduction

Exploiting the redundancy in robotic systems to simultaneously fulfil a set of tasks is a classical problem for complex manipulators and humanoid robots [1], [2]. Several controllers have been proposed in the literature, where the tasks combination is determined by the relative importance of the tasks, expressed by the task priorities. There are two main approaches for prioritized multi-task controllers. The first is based on *strict task priorities*, where a hierarchical ordering of the tasks is defined: critical tasks (or tasks that are considered as more important) are fulfilled with higher priorities, and the low-priority tasks are solved in the null-space of the higher priority tasks [3], [4]. The second is based on *soft task priorities*, where the solution is typically given by a combination of weighted tasks [5]. The importance or "soft priority" of each individual task is represented by a scalar weight function, which evolves in time depending on the sequencing of the robot actions. By tuning the time-dependent vector of scalar weights, the global robot motion can be optimized. In simulation studies, it was shown that adapting these weights may result in a seamless transition between tasks (*i.e.*, reaching for an object, staying close to a resting posture and avoiding an obstacle), as well as in continuous task sequencing [6].

When complex robots, such as humanoids, need to perform manipulations while fulfilling many tasks and constraints (*e.g.*, tracking a desired trajectory, avoiding obstacles, controlling the interaction forces), the strict task priorities approaches typically require a priori a definition of the task hierarchy. For instance, Sentis and Khatib [7] defined three levels of hard priorities *i.e.*, constraints of utter importance (such as contacts, near-body objects, joint-limits and self-collisions), operational tasks demands (*i.e.*, manipulation and locomotion) and adaptable postures (*i.e.*, the residual motion). However, in many contexts, it is difficult to organize the tasks in a stack and pre-define their relative importance in forms of priorities. When priorities are strict, a higher-priority task can completely block lower-priority tasks, which can result in movements that are not satisfactory for the robot mission (*i.e.*, its "global" task). Another issue concerns the occurrence of discontinuities in the control law due to sudden changes in the prioritization [8].

Soft task priorities provide an appealing alternative solution. However, the simultaneous execution of different elementary tasks with variable soft priorities can lead to incompatibilities that might generate undesired movements or prevent the execution of some tasks. These issues are well explained in [9], where the authors modulate the task weights based on the movement variance to handle incompatibilities during online execution. Finally, when the number of tasks increases, for example in whole-body control of humanoid robots, and some tasks related to safety (*e.g.*, balance) are given high priority, it is generally difficult to define suitable task activations. In this case the priorities and their transitions are manually tuned by expert users [10] or defined before-hand [11]. Among the methods based on

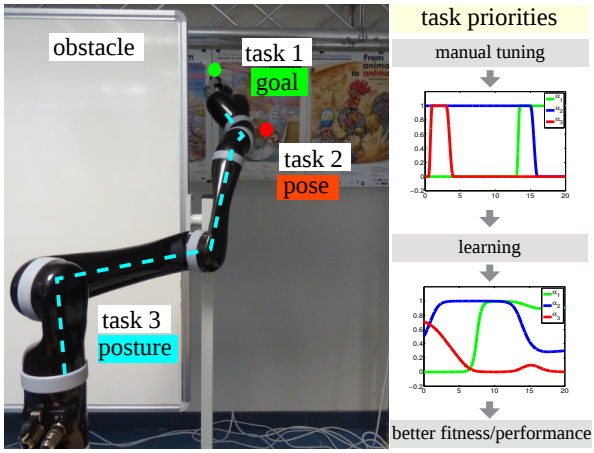


Fig. 1. The Jaco arm must reach a goal behind a wall (obstacle) while fulfilling a pose task on joint 4 and a full posture task. The initial sequencing of task priorities is not efficient. Our method allows the automatic learning of the temporal profiles of the task priorities from scratch.

*This paper was supported by the FP7 EU projects CoDyCo (No. 600716 ICT 2011.2.1 Cognitive Systems and Robotics).

[1] Dipartimento di Ingegneria Informatica, Automatica e Gestionale, Sapienza Universita di Roma, via Ariosto 25, 00185 Roma, Italy. modugno@diag.uniroma1.it

[2] Intelligent Autonomous Systems Lab., TU Darmstadt, Germany.

[3] Max Planck Institute for Intelligent Systems.

[4] Inria, Villers-lès-Nancy, F-54600, France; CNRS, Loria, UMR n.7503 and Université de Lorraine, Loria, UMR n.7503, Vandoeuvre-lès-Nancy, F-54500, France. serena.ivaldi@inria.fr

soft priorities, recently Liu et al. [6] proposed a generalized projector (GHC) that handles strict and non-strict priorities with smooth transitions when tasks priorities are swapped. Despite the elegant framework, their controller needs again a lot of manual tuning. The evolution of the tasks priorities in time, the timing and the tasks transitions need to be designed by hand. While this approach could still be easy for few tasks and simple robotic arms, it quickly becomes unfeasible for complex robots such as humanoids performing whole-body movements that usually require a dozen of tasks and constraints (*e.g.*, control balance, posture, end-effectors, stabilize head gaze, prevent slipping, control the interaction forces *etc.*). With the increasing abilities of humanoid robots, the number of tasks increases, together with their weights or priorities: their manual tuning through a sequence of complex manipulations becomes a major bottleneck.

In this paper, we propose a framework that addresses the issue of automatically optimising the task priorities by means of a learning algorithm. The proposed concept of learning the soft priorities can be applied to existing multi-task frameworks, such as the GHC [10]. However, we use here a simpler controller based on a regularized version of the Unified Framework for Robot Control (UF) [13] proposed by Peters *et al*. In our framework, the task weight functions are parametrized functional approximators that can be automatically learned by state-of-the-art stochastic optimization algorithms. The *temporal* profiles of the task weights can be learned by optimizing a given fitness function, used to evaluate the performance of candidate task priorities. In contrast to many cost functions used in whole-body optimisation frameworks, here we do not require the fitness to be a linear or quadratic function.

We show the effectiveness of our approach on both a simulated and a real 6 degrees of freedom (DOF) Kinova Jaco arm, on a goal reaching problem with several elementary tasks. Furthermore, we compare the performance of our controller with the state-of-the-art method GHC proposed by Liu *et al.* [10] on a simulated 7 DOF Kuka LWR arm.

The paper is organized as follows. Section II presents the proposed approach, describing the structure of the controllers, the task weight functions and the learning procedure. We present the experimental results in Section III, draw conclusions and discuss future work in Section IV.

II. Methods

Let us consider a "global task" or a "mission" for a redundant robot: for example, *to reach a goal point behind a wall while avoiding an obstacle*. The overall movement can be entirely planned by exploring all the possible joint configurations, or it can be generated by a combination of a set of controllers solving simpler elementary tasks (for example: control the end-effector, control the pose of a particular link, *etc.*). We assume that the set of elementary tasks is known, and that each task can be executed by a given torque controller. The global movement can be evaluated by a fitness function ϕ that can be used as a measure of the ability of the robot to fulfil its mission. Our method aims

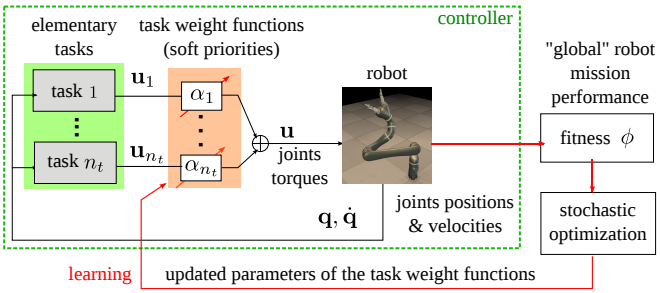


Fig. 2. Overview of the proposed method. The controller consists of a weighted combination of elementary tasks, where the weight functions represent the soft task priorities. An outer learning loop enables the optimization of the task weights parameters.

at automatically learning the task priorities (or task weight functions) to maximize the robot performance.

An overview of the proposed approach is illustrated in Fig. 2. In Section II-A we describe the controller $\mathbf{u}_i$ for each elementary task: a regularized version of the Unified Framework [13]. In Section II-B we describe the multi-task controller with learned task priorities. In Section II-C we describe the parametrized task weight functions α_i used by our multi-task controller, and discuss the parameters optimization. As a learning algorithm, we propose the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [14], a derivative-free stochastic optimization algorithm, in view of its good exploration properties and ease of use.

A. Controller for a single elementary task

We hereby describe the torque controller for the *i*-th elementary task. To simplify the controller design, we decided to adopt the Unified Framework (UF) [13]. We consider the well-known rigid-body dynamics of a robot with *n* DOF, *i.e.*,

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{f}(\mathbf{q},\dot{\mathbf{q}}) = \mathbf{u}_i(\mathbf{q},\dot{\mathbf{q}}), \tag{1}$$

where $\mathbf{q}$, $\dot{\mathbf{q}}$, $\ddot{\mathbf{q}} \in \mathbb{R}^n$ are, respectively, the joints positions, velocities and accelerations, $\mathbf{M}(\mathbf{q}) \in \mathbb{R}^{n\times n}$ is the generalized inertia matrix, $\mathbf{f}(\mathbf{q},\dot{\mathbf{q}}) \in \mathbb{R}^n$ accounts for Coriolis, centrifugal and gravitational forces and $\mathbf{u}_i(\mathbf{q},\dot{\mathbf{q}}) \in \mathbb{R}^n$ is the vector of the commanded torques of the *i*-th task. Using the same notation as in [13], we describe the task as a constraint, given by $\mathbf{h}_i(\mathbf{q},\dot{\mathbf{q}},t) = \mathbf{0}$, where $\mathbf{h}_i \in \mathbb{R}^m$ is at least a twice differentiable function, where *m* is the task dimension. By differentiating the constraint with respect to time, we obtain:

$$\mathbf{A}_i(\mathbf{q},\dot{\mathbf{q}},t)\ddot{\mathbf{q}} = \mathbf{b}_i(\mathbf{q},\dot{\mathbf{q}},t), \tag{2}$$

where $\mathbf{A}_i(\mathbf{q},\dot{\mathbf{q}},t)$ is a known $m\times n$ matrix and $\mathbf{b}_i(\mathbf{q},\dot{\mathbf{q}},t)$ is a $m\times 1$ vector. For example, given a simple tracking control task with $\mathbf{h}_i(\mathbf{q},\dot{\mathbf{q}},t) = \mathbf{q} - \mathbf{q}^{des}$, where $\mathbf{q}^{des}$ corresponds to a desired trajectory. By computing the second order derivative in *t* we obtain $\ddot{\mathbf{q}} = \ddot{\mathbf{q}}^{des}$, where $\mathbf{b} = \ddot{\mathbf{q}}^{des}$ and $\mathbf{A}_i = \mathbf{I}$ (with $\mathbf{I}$ the identity matrix). Applying Gauss's principle, it is possible to derive a controller that fulfils the constraints by minimizing the cost function $\zeta_i(t) = \mathbf{u}_i^\top \mathbf{N}_i(t)\mathbf{u}_i$, where $\mathbf{N}_i(t)$ is a positive semidefinite matrix. The optimization problem is defined by

$$\mathbf{u}_i^* = \arg\min_{\mathbf{u}_i} \zeta_i(t) = \arg\min_{\mathbf{u}_i} \left[\mathbf{u}_i^\top \mathbf{N}_i(t)\mathbf{u}_i\right], \tag{3}$$

subject to Eq. 1 and 2. The solution to this optimization problem is given by

$$\mathbf{u}_i = \mathbf{N}_i^{-\frac{1}{2}}(\mathbf{A}_i\mathbf{M}^{-1}\mathbf{N}_i^{-\frac{1}{2}})^{\#}(\mathbf{b}_i + \mathbf{A}_i\mathbf{M}^{-1}\mathbf{f}), \tag{4}$$

where $(\cdot)^{\#}$ is the Moore-Penrose pseudoinverse and the upper script in $\mathbf{N}_i^{-\frac{1}{2}}$ denotes the inverse of the matrix square root. Controllers derived from UF are sensitive to kinematic singularities, due to the matrix inversion [15]. To overcome this problem, we reformulate the UF controller in a *regularized* fashion, as classically done at the kinematic level, for instance in [16]. The objective function of UF can be reformulated in such a way that the solutions of the optimization problem naturally exhibit a damped least squares structure (at the price of a loss of precision in the execution of the elementary task). Given the dynamical model of the robot (Eq. 1) and the constraint that describes the task (Eq. 2), we define the regularized optimal control problem:

$$\arg\min_{\mathbf{u}_i} \zeta_i(t) = \arg\min_{\mathbf{u}_i} \left[(\mathbf{A}_i\ddot{\mathbf{q}} - \mathbf{b}_i)^2 + \mathbf{u}_i^{\top}\frac{\mathbf{N}_i(t)}{\lambda_i}\mathbf{u}_i \right], \tag{5}$$

where λ_i is the regularizing factor with a l^2-weighted norm for the regularization term. In the simplest case, λ_i can be a manually-tuned constant value, or automatically determined by more sophisticated methods, as done in [17], based on the smallest singular value of the matrix to invert. To derive the closed form solution of the optimization problem, we substitute $\ddot{\mathbf{q}}$ in Eq. 5 with the expression obtained by solving the dynamical constraint for $\ddot{\mathbf{q}}$. The resulting closed form solution of the controller for a single elementary task is then:

$$\mathbf{u}_i = \mathbf{N}_i^{-1}\tilde{\mathbf{M}}_i^{\top}(\mathbf{I}\lambda_i^{-1} + \tilde{\mathbf{M}}_i\mathbf{N}_i^{-1}\tilde{\mathbf{M}}_i^{\top})^{-1}(\mathbf{b}_i + \tilde{\mathbf{M}}_i\mathbf{f}), \tag{6}$$

with $\tilde{\mathbf{M}}_i = \mathbf{A}_i\mathbf{M}^{-1}$.

B. Controller for multiple elementary tasks with soft task priorities

With reference to the scheme of Fig. 2, we consider a number n_t of elementary tasks, that can be combined by the robot to accomplish a given "global" mission. The solution of the i-th task is provided by the torque controller $\mathbf{u}_i$ described in the previous section. Each task is modulated by a task priority or task weight function $\alpha_i(t)$. The ensemble $\{\alpha_i(t)\}_{i=1,\ldots,n_t}$ constitutes the *activation policy* that determines the overall robot movement. The robot controller is therefore given by

$$\mathbf{u}(\mathbf{q},\dot{\mathbf{q}},t) = \sum_{i=1}^{n_t} \alpha_i(t)\,\mathbf{u}_i(\mathbf{q},\dot{\mathbf{q}}) \ , \tag{7}$$

where t is the time, and $\mathbf{q}$ and $\dot{\mathbf{q}}$ are the robot joint positions and velocities. The task priorities $\alpha_i(t)$ are scalar functions and their time profile can be optimized. We automatically tune the task priorities with a learning algorithm. We seek the best task weight functions that maximize a defined performance measure, or fitness, evaluating the execution of the global task. As finding the optimal functions $\alpha_i^*(t)$ is an intractable problem, we turn the functional optimization problem into a numerical optimization problem by representing the task priorities with parametrized functional approximators, $\alpha_i(t) \rightarrow \hat{\alpha}_i(\boldsymbol{\pi}_i,t)$, where $\boldsymbol{\pi}_i$ is the set of parameters that shape the temporal profile of the i-th task weight function. The controller then becomes:

$$\mathbf{u}(\mathbf{q},\dot{\mathbf{q}},t) = \sum_{i=1}^{n_t} \hat{\alpha}_i(\boldsymbol{\pi}_i,t)\,\mathbf{u}_i(\mathbf{q},\dot{\mathbf{q}}) \tag{8}$$

Finding the optimal task priorities consists therefore in finding the optimal parameters $\boldsymbol{\pi}_i^*$, which can be done applying a learning method to maximize the fitness ϕ.

C. Learning the task priorities

We model the task priorities by a weighted sum of normalized Radial Basis Functions (RBF):

$$\hat{\alpha}_i(\boldsymbol{\pi}_i,t) = S\left(\frac{\sum_{k=1}^{n_r}\pi_{ik}\psi_k(\mu_k,\sigma_k,t)}{\sum_{k=1}^{n_r}\psi_k(\mu_k,\sigma_k,t)}\right), \tag{9}$$

where $\psi_k(\mu_k,\sigma_k,t) = \exp\left(-1/2[(t-\mu_k)/\sigma_k]^2\right)$, with (μ_k,σ_k) being mean and variance of the basis functions, n_r is the number of RBFs and $\boldsymbol{\pi}_i = (\pi_{i1},\ldots,\pi_{in_r})$ is the set of parameters of each task priority. $S(\cdot)$ is a sigmoid function that squashes the output to the range $[0,1]$. When the task priority is 1, the task is fully activated; when its value is 0, the task is not active.

In our method, learning the task priorities is implemented by learning the free parameters $\boldsymbol{\pi}_i$ of the weight functions (Eq. 9). We optimize the parameters with respect to a known fitness function $\phi = \phi(\mathbf{q}_{t=1,\ldots,T},\mathbf{u}_{t=1,\ldots,T},t)$, given T time steps. ϕ describes the performance of the controller in fulfilling its global mission. The fitness function could be a simple measure of success in case of goal reaching, the time duration of a movement, the energy consumption *etc.* More criteria for optimizing robot motions in optimal control frameworks are reported in [18]. If the fitness function ϕ is differentiable with respect to the controls and the parameters (which requires the function approximators to be differentiable as well with respect to the controls [17]), then gradient methods can be used. If the fitness is not differentiable with respect to the parameters, then a derivative-free method can be used. Thus, derivative-free methods are appealing, since they do not constrain the design of the fitness function. Furthermore, recent results showed that it is possible to achieve very fast performances in trial-and-error learning with derivative-free methods [19].

As optimization algorithm, we use CMA-ES [14], which is a stochastic derivative-free optimization strategy that is suitable for non-linear and non-convex objective functions. This method belongs to the class of evolutionary algorithms. At each iteration of the algorithm, new candidate solutions are generated from a population of candidates through stochastic sampling. The fitness function is then used to evaluate the candidates. In our case, each candidate is a possible set of parameters for the task priorities $\mathbf{x} = \{\boldsymbol{\pi}_1,\ldots,\boldsymbol{\pi}_{n_t}\}$ (Eq. 9). At each iteration of the algorithm (called *generation*), a new population of candidates is generated by sampling a multivariate normal distribution $\mathcal{N}(\mathbf{m},\boldsymbol{\Omega})$, where $\mathbf{m}$ and

$\mathbf{\Omega}$ represent respectively mean and covariance of the distribution. A fitness value is computed for each candidate in the current population and, according to the fitness, only the most promising candidates are kept to update the search distribution. Given the n_c candidates $\{\mathbf{x}_1,\ldots,\mathbf{x}_{n_c}\}$, the algorithm selects the $n_b < n_c$ best ones according to their ordered fitness values $\{\hat{\mathbf{x}}_1,\ldots,\hat{\mathbf{x}}_{n_b}\}$. It uses the selected candidates to compute the mean of the sampling distribution at the next iteration: $\mathbf{m}^{(new)} = \sum_{i=1}^{n_b} \omega_i \hat{\mathbf{x}}_i$, with $\sum_{i=1}^{n_b} \omega_i = 1$. Then the covariance matrix is updated as:

$\mathbf{\Omega}^{(new)} = (1-c_1-c_2)\mathbf{\Omega} + c_1 p_\Omega p_\Omega^\top + c_2 \sum_{i=1}^{n_b} \omega_i \mathbf{y}_i (\mathbf{y}_i)^\top$

with $\mathbf{y}_i = (\hat{\mathbf{x}}_i - \mathbf{m})$, c_1 and c_2 two predefined parameter (see [14] for more details). The symbol p_Ω is a term measuring the correlation among successive generations. The covariance is related to the *exploration rate* of the algorithm, a scalar value between [0,1] and the only parameter of the algorithm that needs to be tuned. This version of CMA-ES does not support constrained optimization, which means that the optimized solutions that are not physically feasible on the real robot must be dropped and the learning algorithm restarted. In the follow-up of this work, we will use a version that supports constraints [20].

III. Experiments

In this section we discuss our experiments on learning the task priorities. We start showing on a simulated Kinova Jaco arm that the our learning method improves the performance of the movement in terms of fitness values, over existing task priorities that have been manually tuned. We also show that the optimized trajectories are robust with respect to the initialization of the learning process. We compare on a real Jaco arm some typical learned policies with the manually tuned one, showing that our method improves the real robot motion. Finally, we compare on a simulated Kuka LWR the performance of our method with the state-of-art GHC controller [6] . We show that our method is not only better in terms of performance, but also computationally 10 times faster.

A. Learning the task priorities for the Kinova Jaco arm

The setting for the first experiment is shown in Figure 1. The Kinova Jaco arm (6 DOF), starting from its zero configuration, must reach a desired position behind a wall with its end-effector. The goal position is difficult to reach, and the robot kinematics is such that it is not straightforward to manually design a trajectory that does not collide with the obstacle and brings the hand to the goal.

There are 3 given elementary tasks. The first is about reaching the Cartesian position $\mathbf{p}^*$ =[0, -0.63, 0.7] with the end-effector (*goal*). The second is about reaching the Cartesian pose [-0.31, -0.47, 0.58] with the 4th link. The third is about keeping the joint configuration [120, 116, 90, 0, 0, 0] (degrees).We design the following fitness function $\phi \in [-1,0]$:

$$\phi(\mathbf{q}_{1,\ldots,T}, \mathbf{u}_{1,\ldots,T}) = -\frac{1}{2}\left(\frac{\sum_i^T \|\mathbf{p}_i - \mathbf{p}^*\|}{\varepsilon_{\max}} + \frac{\sum_i^T \|\mathbf{u}_i\|_2^2}{u_{\max}}\right)$$

where T is the number of control steps (the task duration is 20 seconds, and we control at 10ms), $\mathbf{p}_i$ describes the end-effector position at time i and $\mathbf{p}^*$ is the goal position, $\|\cdot\|_2^2$ is the square of the ℓ^2 norm and $\varepsilon_{\max}^{-1}$ and $u_{\max}^{-1}$ are two scaling factors. The first term of ϕ penalizes the cumulated distance from the goal that enforces a minimum time transfer trajectory for the robot arm, while the second term penalizes the global control effort. To ensure that the generated controls are feasible for the real Jaco robot, we set the fitness to -1 whenever the generated policy violates one of the robot constraints: a collision with the environment, joints position ranges and maximum joint torques. This ad-hoc solution is also a consequence of the learning algorithm.

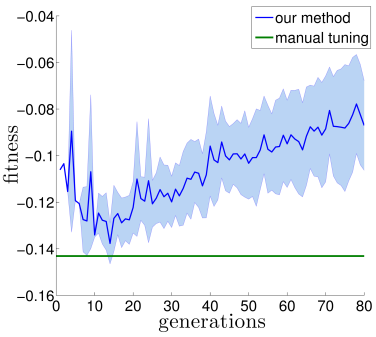


Fig. 3. Average fitness value for the task priorities learned with our method, for the 3-tasks experiment with the simulated Jaco arm. The horizontal line indicates the fitness of the manually tuned solution. The mean and standard deviation of the fitness for the learned policies is computed over 100 restarts of CMA-ES, each with 80 generations and random initialization of the parameters. We only retained the fitness for the experiments that provided solutions satisfying the real robot constraints.

Fig. 3 shows the average fitness value computed over the eleven optimized trajectories satisfying the constraints, found on 100 restarts of CMA-ES.

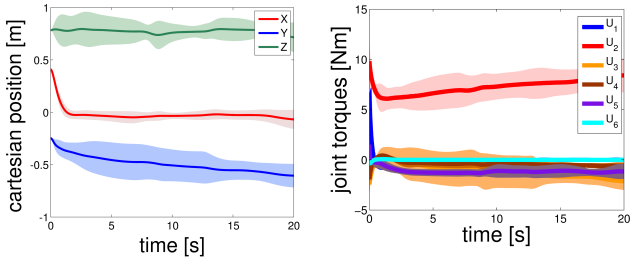
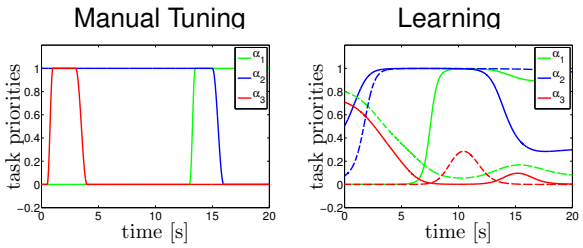


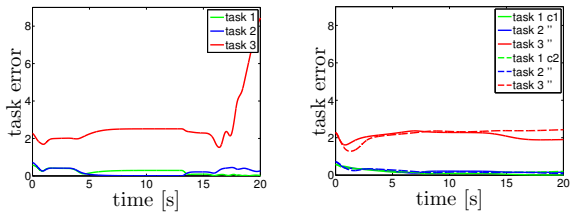
Fig. 4. Mean and variance of the Cartesian trajectory of the end-effector and the joints torques of the simulated Jaco arm, generated by learned task priorities over 100 trials of the 3-tasks experiment (see text in Section III-B). Even starting from random initialization of the task weight parameters, the learning process is eventually able to produce similar optimized motions of the robot that fulfil its 'global' task.

B. Robustness of the learning process

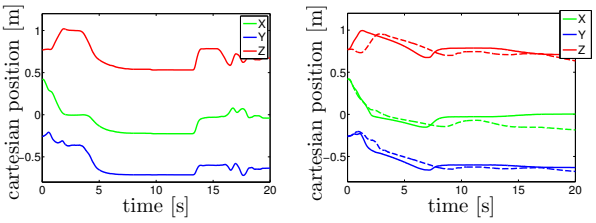
Different profiles of the task priorities can yield similar movements of the robot. It is however important to show that the learning process is able to optimize the task priorities in a robust way, that is providing similar optimal solutions. We therefore execute N=100 replicates of the experiment in Section III-A, with a simulated Jaco arm and three tasks. In each experiment, CMA-ES runs for 100 generations with an exploration rate of 0.5. The parameters are randomly



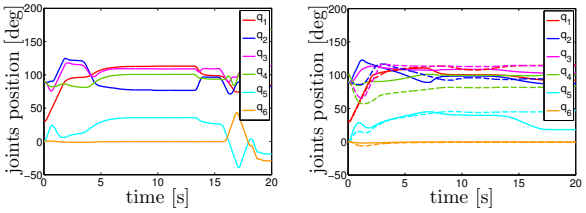
(a) The task priorities evolving in time.



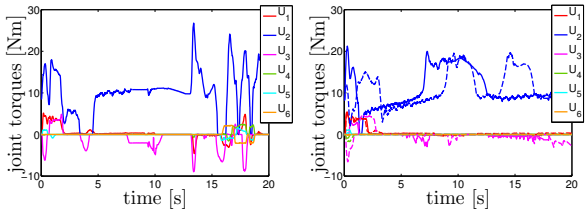
(b) The task errors.



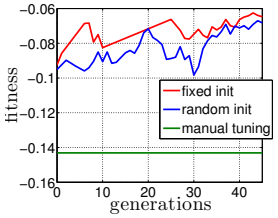
(c) The end-effector trajectory in the Cartesian space.



(d) The joints trajectories.



(e) The measured joints torques (estimated by the motor currents).



(f) The fitness values.

Fig. 5. Comparison between a manually tuned (left side) and two typical learned (right side) policies for the 3-tasks experiment performed with the real Kinova Jaco arm. On the right, a solid line corresponds to a policy optimized starting from a fixed/known initial value of the priorities (*fixed init*), in this case the priorities found by manual tuning; the dashed line corresponds to a policy optimized starting from random values (*random init*). The final fitness values are: -0.1431 (manual tuning), -0.0585 (fixed init) and -0.0644 (random init).

initialized. We compute the average and standard deviation of the solutions that satisfy the robot and task constraints. Figure 4 shows the average end-effector trajectory in the Cartesian space and the corresponding joint torques. Despite the redundancy of the robot and the one of the task priorities, the final robot movements are smooth and quite consistent with each other. Their average fitness is -0.0874 ± 0.0213. Overall, this result indicates that learning the soft task priorities starting from scratch (*i.e.*, where an initial guess for the activation of the task priorities in time is not available) is a viable and robust option for generating the motion of redundant robots.

C. Experiment on the real Kinova Jaco arm

We compare in Fig. 5 the effect of three different task prioritizations on the real Jaco arm. In the left column, we show the robot movement generated by task priorities that were manually tuned by an expert user of the Kinova arm; on the right column, we show two typical robot movements generated by learned task priorities, which were optimized with CMA-ES starting from a known initial value (the manually tuned task weight functions) and a random value. We set the exploration rate in CMA-ES to 0.5 and perform 40 generations. Learning the priorities has a beneficial effect on the smoothness of the trajectories, which becomes evident when comparing the plots of the end-effector (Fig. 5c), joints positions (Fig. 5d) and torques (Fig. 5e) and the task errors (Fig. 5b). We evaluate the fitness using the commanded joint torques $\mathbf{u}_i$ and the kinematics and dynamics model of the Jaco arm to compute $\mathbf{p}_i$. The fitness value for the manually tuned task priorities is -0.1431. The fitness values for the two optimized solutions are better: -0.0585 and -0.0644 initializing the parameters with fixed and random values respectively. Overall, this experiment illustrates that learning the task priorities improves the real robot motion with respect to an existing manually tuned solution.

D. Comparison with the state-of-the-art GHC

In this experiment we compare the performance of the task priorities learning applied to our method and to the state-of-the-art multi-task controller GHC [6].

In the GHC, each task is associated to a null space projector of the extended Jacobian that contains the analytical description of all the task objectives. Soft task prioritization is achieved because the null space projector depends on a set of manually designed weight functions ranging from 0 to 1, that control if each task is fully or partially projected in the null spaces of the other tasks with higher priority. The controller is the solution to a quadratic optimal control problem subject to the robot and task constraints – see [6]. The soft task priorities are introduced as a further constraints, formulated by $\ddot{\mathbf{q}} = \sum_{i=1}^{n_t} \mathbf{P}_i(\mathbf{\Lambda}_i)\ddot{\mathbf{q}}_i^{'}$, where $\mathbf{P}_i(\cdot)$ is the null space projector associated to the i-th task, $\mathbf{\Lambda}_i$ is a matrix that depends on the task priorities, $\ddot{\mathbf{q}}_i^{'}$ are intermediate joint accelerations associated to each task and $\ddot{\mathbf{q}}$ are the joints accelerations. To enable the comparison with our method,

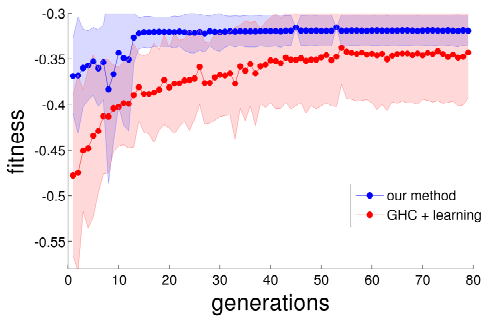


Fig. 6. Comparison between our method and the GHC modified to learn the task priorities with CMA-ES. The plot shows the mean and the standard deviation of the fitness in $R = 20$ trials of the experiment with the simulated KUKA LWR arm (see Section III-D). For both controllers, the learning is initialized with random parameters. Our method shows a faster convergence and better optimization of the fitness. The average fitness is is -0.0373 ± 0.0320 for our method and -0.0735 ± 0.0946 for the GHC+learning. The two distributions are statistically different ($p < 0.01$ with the K-S test).

we parametrized the task priority matrix $\mathbf{\Lambda}_i$ for each task i in the same way as described in Section II-B.

We compare the two methods on a reaching task with a simulated 7 DOF Kuka LWR, which was originally used in [6]. In this scenario, the robot must reach a goal point beneath a rectangular surface parallel to the ground ($z = 0.25m$), without collision. The are 2 elementary tasks. The first is about reaching the Cartesian position $\mathbf{p}^* = [0.6, 0, 0.15]$ with the end-effector (*goal*). The second is about keeping the joint configuration [0, 90, 0, -90, 0, 90, 0] (degrees). We design the following fitness function $\phi \in [-1, 0]$:

$$\phi(\mathbf{q}_{1,\dots,T}, \mathbf{u}_{1,\dots,T}) = -\frac{1}{2}\left(\frac{\sum_i^T \|\mathbf{p}_i - \mathbf{p}^*\|}{\varepsilon_{\max}} + \frac{\max(\|\mathbf{u}_{i=1,\dots,T}\|_\infty)}{u_{\max}}\right)$$

where $\|\cdot\|_\infty$ is the infinity norm. We set the fitness to -1 in case of collision. We run 20 experiments from random initial parameters for both methods, with an exploration rate of 0.1 for CMA-ES and 80 generations.

Our method generated solutions that satisfy the collision constraint in 90% of the cases, while the GHC succeeded only in 75%. Figure 6 shows the mean and standard deviation of the fitness: our method is faster in convergence and improves the final optimized fitness. The average fitness at the end of the learning process is -0.0373 ± 0.0320 for our method and -0.0735 ± 0.0946 for the GHC+learning. The two distributions of the fitness are statistically different ($p = 0.0073 < 0.01$, obtained with the two-sample Kolmogorov-Smirnov test). Our method is also 10 times faster in terms of computational time: on a standard i7 machine, the average time for the optimization process to find a solution with 80 generations (average over 20 trials) is $3.7 \times 10^3 \pm 2.4 \times 10^2$ seconds for our method and $3.9 \times 10^4 \pm 2.2 \times 10^3$ seconds for the GHC+learning approach.

IV. Conclusion and Future Work

In this paper we address an important issue for prioritized multi-task controllers, that is the automatic and optimal generation of task priorities through parametrized weight functions. As a first step towards an automatically tuned controller for redundant robots, we propose a novel framework with a multi-task controller where the task priorities can be learned via stochastic optimization. We show the effectiveness of our approach by comparing to GHC [10], a state-of-the-art multi-task prioritized controller. We present several results performed on a simulated 7 DOF Kuka LWR arm and both a simulated and a real 6 DOF Kinova Jaco arm. Ongoing work is focused on improving the current framework from different points of view: addressing generalization, using constraints inside the optimization, and scaling up the method to handle robots with several DOF, e.g., humanoid robots.

References

[1] Y. Nakamura, H. Hanafusa, and T. Yoshikawa, "Task-priority based redundancy control of robot manipulators," *IJRR*, vol. 6, no. 2, pp. 3–15, 1987.

[2] B. Siciliano and J.-J. Slotine, "A general framework for managing multiple tasks in highly redundant robotic systems," in *Int. Conf. Advanced Robotics*, 1991, pp. 1211–1216.

[3] L. Saab, O. Ramos, F. Keith, N. Mansard, P. Soueres, and J.-Y. Fourquet, "Dynamic whole-body motion generation under rigid contacts and other unilateral constraints," *IEEE Trans. on Robotics*, vol. 29, pp. 346–362, Jan 2013.

[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, Jan 2015.

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

[6] M. Liu, Y. Tan, and V. Padois, "Generalized hierarchical control," *Autonomous Robots*, pp. 1–15, 2015.

[7] L. Sentis and O. Khatib, "Synthesis of whole body behaviours through hierarchical control of behavioral primitives," *Int. Journal of Humanoid Robotics*, pp. 505–518, 2005.

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

[9] R. Lober, V. Padois, and O. Sigaud, "Variance modulated task prioritization in whole-body control," in *IROS*, 2015, pp. 1–6.

[10] M. Liu, S. Hak, and V. Padois, "Generalized projector for task priority transitions during hierarchical control," in *ICRA*, 2015, pp. 768–773.

[11] S.-I. An and D. Lee, "Prioritized inverse kinematics with multiple task definitions," *ICRA*, pp. 1423–1430, 2015.

[12] J. Kober, D. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *IJRR*, vol. 11, pp. 1238–1274, 2013.

[13] J. Peters, M. Mistry, F. Udwadia, J. Nakanishi, and S. Schaal, "A unifying framework for robot control with redundant dofs," *Autonomous Robots*, vol. 24, pp. 1–12, Jan 2008.

[14] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies." *Evolutionary Computation*, vol. 9, pp. 159–195, Jan 2001.

[15] S. Chiaverini, B. Siciliano, and O. Egeland, "Redundancy resolution for the human-arm-like manipulator," *Robotics and Autonomous Systems*, vol. 8(3), pp. 239–250, Jan 1991.

[16] Y. Nakamura and H. Hanafusa, "Inverse kinematic solutions with singularity robustness for robot manipulator control," *J. Dyn. Sys., Meas., Control*, vol. 108 (3), pp. 163–171, 1986.

[17] S. Ivaldi, M. Fumagalli, F. Nori, M. Baglietto, G. Metta, and G. Sandini, "Approximate optimal control for reaching and trajectory planning in a humanoid robot," in *IROS*, 2010, pp. 1290–1296.

[18] 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.

[19] A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret, "Robots that can adapt like animals," *Nature*, vol. 521, no. 7553, pp. 503–507, 2015.

[20] D. V. Arnold and N. Hansen, "A (1+ 1)-cma-es for constrained optimisation," in *GECCO*, 2012, pp. 297–304.