Value Iteration for Learning Concurrently Executable Robotic Control Tasks

Sheikh A. Tahmid
University of Waterloo
Waterloo, Canada
sheikh.abrar.tahmid@uwaterloo.ca

Gennaro Notomista
University of Waterloo
Waterloo, Canada
gennaro.notomista@uwaterloo.ca

ABSTRACT

Many modern robotic systems such as multi-robot systems and manipulators exhibit redundancy, a property owing to which they are capable of executing multiple tasks. This work proposes a novel method, based on the Reinforcement Learning (RL) paradigm, to train redundant robots to be able to execute multiple tasks concurrently. Our approach differs from typical multi-objective RL methods insofar as the learned tasks can be combined and executed in possibly time-varying prioritized stacks. We do so by first defining a notion of task independence between learned value functions. We then use our definition of task independence to propose a cost functional that encourages a policy, based on an approximated value function, to accomplish its control objective while minimally interfering with the execution of higher priority tasks. This allows us to train a set of control policies that can be executed simultaneously. We also introduce a version of fitted value iteration to learn to approximate our proposed cost functional efficiently. We demonstrate our approach on several scenarios and robotic systems.

KEYWORDS

multi-objective RL; redundant robots; reinforcement learning; optimization-based control

ACM Reference Format:
Sheikh A. Tahmid and Gennaro Notomista. 2025. Value Iteration for Learning Concurrently Executable Robotic Control Tasks. In *Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Detroit, Michigan, USA, May 19 – 23, 2025*, IFAAMAS, 9 pages.

1 INTRODUCTION

Training a robot to be able to concurrently execute multiple control objectives is an active area of research in robotics and Reinforcement Learning (RL). This is especially useful for robotic systems that have the capability of executing multiple tasks concurrently — also known as redundancy — such as multi-robot systems [18], [27], [13], manipulators [18], [20], [30] and humanoids [36], [8]. Multi-objective RL [12], [28], which aims to train an agent to solve multiple possibly competing objectives, is a possible approach that has been applied to robotic control problems [35], [14]. While multi-objective RL methods may work, they generally result in polices that cannot execute subsets of the objectives or reprioritize tasks during online execution.

In this paper, we propose an alternative approach where we instead train multiple tasks using RL to be compatible with each other. This allows us to combine possibly time-varying subsets of tasks which may be useful in many systems such as those operating over long durations where objectives are likely to change over time [6]. Furthermore, several works that study concurrently executing multiple tasks using redundant robots focus on executing tasks in a prioritized fashion [2], [20], [24], [22]. This is useful in robotics applications where one should, for instance, prioritize safety-critical tasks over others. Our approach in this paper extends this idea as our method encourages each task's learned policy to not interfere with the execution of higher priority tasks. Several works propose methods for combining multiple deep RL policies together [33], [11], [25], [34], [21]. We opt to make use of the work in [21] which combines multiple value functions implemented as neural networks—which may be learned through various methods within RL [10], [16]—into a single min-norm controller capable of concurrently executing each task in a prioritized fashion. The controller in [21] allows us to execute possibly time-varying subsets of tasks with possibly time-varying priorities. The work in [21], however, does not enforce nor encourage any constraints on the learned functions themselves, causing there to be no guarantees on how trained tasks execute together. In our work, we propose a particular cost functional that can be used to encourage learned value functions to satisfy a notion of independence, defined based on definitions of independence in [1] and [24], which allows the tasks to be executed concurrently.

We believe our proposed approach can be especially useful in applications that involve the coordinated control of multi-robot systems. In persistent environmental monitoring with multiple robots [17], [23], for example, a team of robots must efficiently explore and monitor the given environment while maintaining battery levels. Being able to switch priorities or the set of tasks being executed would also be useful in such an application in order to respond to emergencies or monitor particular areas of interest. Multi-robot systems are, in general, not differentially flat [4], resulting in it being difficult to plan their behaviour using classical methods. As a result, RL has shown to be a useful tool in controlling multi-robot systems to perform complex tasks [23].

The main contributions of this work are as follows:

- We define notions of task *independence* and *orthogonality* for learned cost-to-go functions that characterize the ability to concurrently execute certain tasks together.
- We propose a form of a cost functional and show with a proof that a set of tasks can be made *independent*, and therefore

Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Y. Vorobeychik, S. Das, A. NowÃĺ (eds.), May 19 – 23, 2025, Detroit, Michigan, USA. © 2025 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

possible to execute together concurrently, by solving our proposed cost functional.
- We propose a version of fitted value iteration [3] for the continuous control setting, similar to [16], to directly learn our proposed cost functional within the deep RL paradigm.
- We demonstrate our approach by training control policies for robotic systems in such a way that the policies' respective tasks, which may have otherwise had conflicting objectives, can be simultaneously executed.

2 RELATED WORKS

This paper is concerned with training robotic control tasks within the RL paradigm, with the overall aim being to execute multiple tasks concurrently, which is one shared by existing multi-objective RL methods. We define notions of independence and orthogonality between learned tasks, similar to definitions from existing work discussed in 2.3. We combine learned tasks using a min-norm controller proposed in [21], which unlike other similar works, allows us to execute learned tasks in possibly time-varying prioritized stacks.

2.1 RL and Fitted Value Iteration for Robotic Control Tasks

Existing work in deep RL for continuous control tasks in robotics predominantly uses policy iteration methods such as [29], [15]. When the system dynamics are known, one alternative is to train a neural network to approximate an optimal cost-to-go function through fitted value iteration [3], where each iteration has the following two main steps:

$$J^k_{\text{tar}}(x_0) = \min_{u_1,\ldots,u_l \in \mathcal{U}} \sum_{i=0}^{l-1} \gamma^i \cdot c(x_i, u_i) + \gamma^l J(x_l; \theta_k), \quad \forall x_0 \in \mathcal{D} \tag{1}$$

$$\theta_{k+1} = \min_{\theta} \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \left\| J(x;\theta) - J^k_{\text{tar}}(x) \right\|^2 \tag{2}$$

with state space $\mathcal{X} \subseteq \mathbb{R}^n$, input space $\mathcal{U} \subseteq \mathbb{R}^m$, network parameters $\theta \in \mathbb{R}^d$, instantaneous cost function $c(x,u) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}_{\geq 0}$, a sequence of states $x_1, \ldots, x_l \in \mathcal{X}$ reached by selecting the sequence of inputs $u_1, \ldots, u_n \in \mathcal{U}$, discount factor $0 < \gamma \leq 1$, number of lookahead steps $l \geq 1$ and dataset of states $\mathcal{D}$. To avoid biased estimates based on a particular choice of the number of look-ahead steps l, we can use the forward view of TD(λ) [32]. We assume the system to have deterministic dynamics. Note that RL is typically posed as a maximization problem where a value function may be learned, but in this paper, we pose it as a minimization problem where the value function may also be referred to as a cost-to-go function. We will use the two terms interchangeably in this paper. For robotic control tasks, the state and input space is continuous, meaning that in the general case, it is not clear how to find the inputs $u_1, \ldots, u_n \in \mathcal{U} \subseteq \mathbb{R}^m$ that minimize (1). The work in [16] proposes the Continuous Fitted Value Iteration (CFVI) algorithm. The work in [16], which extends work in [5], shows that the optimization problem in (1) has an analytical solution for the continuous control setting under certain assumptions. In our work, we propose a similar version of fitted value iteration as CFVI for a similar problem setting with different assumptions on the instantaneous cost.

2.2 Multi-Objective RL

Multi-objective RL has been used to solve decision and control problems that involve optimizing for multiple objectives [12], [28]. It is possible to treat multi-objective RL as regular RL by using a linear combination of instantaneous rewards. However, this has problems discussed in [12] such as the increased complexity of reward-shaping and the lack of explainability in resulting policies. The work in [35] and [14], which both find Pareto optimal policies, show that multi-objective RL is viable in robotic control problems. However, multi-objective RL methods generally result in policies that are not separable nor capable of complying with time-varying task priorities, unlike our proposed approach of training multiple tasks to be compatible with each other.

2.3 Definitions of Control Tasks and Task Independence

There exist previous work that mathematically define robotic control tasks, along with notions of *independence* and *orthogonality* which affect the ability to concurrently execute multiple tasks. Several works such as [31], [2] have studied executing multiple Jacobian-based tasks concurrently, often in a prioritized fashion, for redundant kinematic robots. The work in [1] analyzes the stability of such algorithms, also defining notions of independence and orthogonality between Jacobian-based tasks. Set-based tasks were introduced in [7] and the concurrent execution of multiple of them is analyzed in [19]. The work in [24] uses Extended-Set Based tasks [22] to generalize Jacobian-based tasks, as well as the definitions of independence and orthogonality introduced in [1]. The definitions in [24] can be used to describe the cost-to-go functions that we train in this paper. However, these definitions do not take into account the system dynamics, unlike Definition 1 in Section III of this work which does.

2.4 Composition of Learned Control Tasks

Section 2.1 in [21] summarizes some of the previous literature on composing learned deep RL policies [33], [11], [25], [34]. In our work, we opt to use the min-norm controller proposed in [21] as our method of combining multiple learned tasks as it allows us to execute possibly time-varying subsets of tasks with possibly time-varying priorities. We explain the controller in [21] further in the next section. While [21] proposes a method of combining multiple learned tasks, it does not offer a way to ensure that the learned tasks are compatible with each other. Our proposed method allows us to train tasks in such a way that it is possible to combine and execute them together using the method in [21].

3 BACKGROUND AND PROBLEM FORMULATION

We consider the problem of using RL to learn a set of tasks for a redundant robotic system in such a way that the tasks can be executed concurrently.

Firstly, we assume that the system is deterministic and control-affine. These assumptions may not be applicable to all applications. However, in robotics, it is common to assume that the dynamics are approximately known and many robotic systems, including most mobile robots and manipulators, can be modelled as control-affine systems. The dynamics can be represented with:

$$\dot{x}(t) = f(x) + g(x)u \tag{3}$$

where $x \in \mathcal{X} \subseteq \mathbb{R}^n$ and $u \in \mathcal{U} \subseteq \mathbb{R}^m$ denote the state and input, respectively, and $f : \mathbb{R}^n \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ are continuous vector fields.

We assume that each task $1, \ldots, N$ is learned sequentially in the order of importance with task 1 being the most important. This is a reasonable assumption in robotics as it is often desirable to prioritize more safety-critical tasks over others. We assume that each task i has an associated state cost, $q_i(x)$, that is positive semi-definite. To learn each task i, we consider solving the following infinite horizon problem:

$$J_i^\star(x(t)) = \min_{u(\cdot)} \int_t^\infty e^{-\beta\tau} \left(q_i(x) + u^\top R_i(x)u\right) d\tau \tag{4}$$

where $\beta \geq 0$ and $R_i : \mathbb{R}^n \to \mathbb{S}^n_{++}$ maps each state x to a positive definite matrix. We assume that each task i has a reachable terminal goal state $x_T \in \mathcal{X}$ where $q_i(x_T) = 0$. As this problem cannot be solved analytically in the general case, we consider using deep RL to learn parametric approximations which we will denote as $\tilde{J}_i$ for each task i. Note that the continuous dynamics of the system described in (3) can be discretized which would allow us to use an actor-critic [10] method or version of fitted value iteration [3], described in (1) and (2), to find some $\tilde{J}_i$.

If we learn an approximation $\tilde{J}_i$ for task i, one way to execute the optimal policy is to use as input the $u^\star(.)$ that solves the optimization problem in (4). However, to simply make progress on task i, we can pick an input that drives $\tilde{J}_i$ toward 0 or, in other words, makes the time derivative negative at a particular state. To make progress on task i at every state $x \in \mathcal{X}$ as fast as if we were executing the optimal policy, we can pick an input $u \in \mathcal{U}$ that makes the time derivative negative by the same amount as if we were explicitly using the optimal input $u^\star(x)$ at state x. This motivates the use of the pointwise min-norm controller proposed in [21].

Given a set of learned cost-to-go functions $\tilde{J}_1, \ldots, \tilde{J}_N$, each implemented with a neural network, we can select inputs that execute each task concurrently using the min-norm quadratic program proposed in [21]:

$$\begin{aligned} \min_{u\in\mathcal{U},\,\delta\in\mathbb{R}^N} \quad & \|u\|^2 + \kappa\|\delta\|^2 \\ \text{s.t.} \quad & L_f\tilde{J}_1(x) + L_g\tilde{J}_1(x)u \leq -\sigma_1(x) + \delta_1 \\ & \vdots \\ & L_f\tilde{J}_N(x) + L_g\tilde{J}_N(x)u \leq -\sigma_N(x) + \delta_N \\ & K\delta \geq 0 \end{aligned} \tag{5}$$

Note that $L_f\tilde{J}_i(x)$ and $L_g\tilde{J}_i(x)$ are Lie derivatives where $L_f\tilde{J}_i(x) = \frac{\partial\tilde{J}_i}{\partial x}(x)f(x)$ and $L_g\tilde{J}_i(x) = \frac{\partial\tilde{J}_i}{\partial x}(x)g(x)$, meaning that the lefthand side of each constraint is simply the time derivative of the cost-to-go function for a task. Since each $J_1^\star, \ldots, J_N^\star$ is positive semi-definite and smaller values represent being closer to a goal state, each corresponding $\tilde{J}_1, \ldots, \tilde{J}_N$ can be treated as a candidate Control Lyapunov Function (CLF). Note that since $\tilde{J}_1, \ldots, \tilde{J}_N$ are approximated by neural networks, $L_f\tilde{J}_i(x)$ and $L_g\tilde{J}_i(x)$ can be calculated using back-propagation. As mentioned previously, picking an input u that makes the time derivative of a particular $\tilde{J}_i$ negative represents making progress on that task.

$\sigma_1(x), \ldots, \sigma_N(x)$ are positive semi-definite scalar functions, parameterized by the current state x, designed to recover inputs equivalent to the optimal input with respect to each approximate cost-to-go function.

Remark 1. Note that the expressions $\sigma_1(x), \ldots, \sigma_N(x)$ in [21] are derived from [9], [26] assuming that the approximated cost functionals do not use a discount factor. However, it is still reasonable to use the controller in (5) with tasks trained with a slight discount factor, meaning γ close to 1 in the discrete case in (1) and β close to 0 in the equivalent continuous formulation in (4), as the recovered policies would still be close enough to optimal.

Slack variables $\delta_1, \ldots, \delta_N \in \mathbb{R}$, which form $\delta \in \mathbb{R}^N$, are added to (5) to make the quadratic program feasible for every state $x \in \mathcal{X}$ as well as prioritize the tasks. $K \in \mathbb{R}^{N\times N}$ is used to specify the possibly time-varying priorities between tasks by specifying the relationship between different values of δ_i for $i \in \{1, \ldots, N\}$. Higher priority tasks are allowed less slack than lower priority ones, meaning that the controller will prioritize picking inputs close to optimal for higher priority tasks than lower priority ones. $\kappa \geq 0$ is a hyperparamter used in the objective function in (5) to control the total amount of slack allowed.

If the quadratic program in (5) is used to execute our learned tasks, being able to execute N tasks concurrently then depends on the relationships between $L_g\tilde{J}_1(x), \ldots, L_g\tilde{J}_N(x)$. Specifically, if $(L_g\tilde{J}_1(x))^\top, \ldots, (L_g\tilde{J}_N(x))^\top$ are linearly independent at some state $x \in \mathcal{X}$, then it is possible to pick an input $u \in \mathcal{U}$ at that state that drives each $\tilde{J}_1, \ldots, \tilde{J}_N$ toward 0 and therefore make progress on each task, assuming that there are no additional constraints on the input.

Definition 1 (Independent and Orthogonal Tasks)**.** Consider a set of differentiable, positive semi-definite functions encoding tasks for a control affine system (1), of the form $V_i : \mathcal{X} \to \mathbb{R}_{\geq 0}$. We assume that at least one of $L_gV_1, \ldots, L_gV_N$ is non-zero $\forall x \in \mathcal{X}$. We also assume that $\forall i \in \{1, \ldots, N\}$, $L_gV_i(x) = 0$ only if $V_i(x) = 0$. $V_1, \ldots, V_N$ are *independent* to each other at state x if the nonzero vectors within $(L_gV_1(x))^\top, \ldots, (L_gV_N(x))^\top$ are linearly independent. $V_1, \ldots, V_N$ are *orthogonal* to each other at state x if $\langle (L_gV_i(x))^\top, (L_gV_j(x))^\top \rangle = 0\ \forall i, j \in \{1, \ldots, N\}$. Note that if tasks are orthogonal at x, then they are also independent at x.

Remark 2. Assuming that at least one of $L_gV_1, \ldots, L_gV_N$ is non-zero is a reasonable assumption in our application as it is unlikely for the gradients of each of the trained neural networks to be perfectly zero.

If a set of tasks are independent at every state $x \in \mathcal{X}$, then there is always a feasible solution $u \in \mathcal{U}$ to the problem in (5) that makes the time derivative of each $\tilde{J}_1, \ldots, \tilde{J}_N$ negative when the respective task has not already been completed, thus driving each $\tilde{J}_1, \ldots, \tilde{J}_N$

toward 0, and therefore concurrently executing each task. In our chosen setting, the problem of training a robotic system to be able to execute multiple objectives deduces to training a set of learned tasks $\tilde{J}_1, \ldots, \tilde{J}_N$ such that $\tilde{J}_1, \ldots, \tilde{J}_N$ are independent at every state $x \in \mathcal{X}$. In this paper, we solve this problem by designing a cost functional, that when approximated using an RL algorithm, results in learned tasks that satisfy our definition of independence. This, along with a variant of the fitted value iteration algorithm to fit this proposed cost functional, are two of the main contributions of the paper and are the subject of the next section.

4 MAIN RESULTS

In 4.1, we propose a version of the fitted value iteration algorithm for approximating cost functionals of the form in (4) for the continuous control setting. We can use this algorithm to approximate a cost functional that we propose in 4.2. In 4.2, we prove, among other things, that our proposed cost functional can be used to define learned tasks to be independent to each other and thus possible to execute simultaneously.

4.1 Continuous Fitted Value Iteration

We propose a form of fitted value iteration for learning cost-to-go functions of the form in (4). The main idea of the CFVI [16] algorithm is to extend the fitted value iteration algorithm by solving the optimization problem in (1) analytically for the continuous control setting. We extend this approach in Proposition 1 for cost functionals of the form in (4).

Proposition 1. *Given a control-affine system and the k-th iteration of an estimate of the cost-to-go function described in (4), J^k, the optimal policy based on the fitted value iteration update step in (1) in the continuous setting is:*

$$u^\star(t) = -\frac{1}{2} R(x)^{-1} \left(L_g J^k(x)\right)^\top$$

Proof. This proof is very similar to Theorem 1 in [16]. We first define discretized dynamics for the system by using a Δt that is small enough to appproximate the dynamics in (3). We define $\bar{f}(x,u)$ such that:

$$\bar{f}(x,u) = x + (f(x) + g(x)u)\Delta t$$

where $\bar{f}(x,u)$ calculates the new state after sending input $u \in \mathcal{U}$ at state $x \in \mathcal{X}$ for a duration of Δt. We then substitute $\bar{f}(x,u)$ and the instantaneous cost into the fitted value iteration update step in (1) with $l = 1$:

$$J^k_{\text{tar}}(x) = \min_{u \in \mathcal{U}} \left\{ \left(q(x) + u^\top R(x) u\right) \cdot \Delta t + \gamma J^k(\bar{f}(x,u)) \right\}.$$

We can then rewrite this using the Taylor approximation of J^k around x:

$$J^k_{\text{tar}}(x) = \min_{u \in \mathcal{U}} \Big\{ \left(q(x) + u^\top R(x) u\right) \Delta t + \gamma \left(J^k(x) + \left(\frac{\partial J^k}{\partial x}\right)^\top \bar{f}(x,u)\Delta t + O(\Delta t, x, u) \right) \Big\}$$

where $O(\Delta t, x, u)$ represents the higher order terms. In the continuous setting, Δt approaches 0, which means that the higher order terms, $O(\Delta t, x, u)$, also approach 0. By the same logic in [16], γ also approaches 1 at the continuous time limit as Δt approaches 0. To find the optimal input, we can then simplify, removing terms that do not rely on u, removing $O(\Delta t, x, u)$ and substituting $\gamma = 1$ to get the following unconstrained optimization problem:

$$\min \left\{ u^\top R(x) u + L_g J^k(x) u \right\}$$

for which the solution is $u^\star = -\frac{1}{2} R(x)^{-1} \left(L_g J^k(x)\right)^\top$. Note that $R(x)^{-1}$ is defined since $R(x)$ is positive definite. □

We can then use this analytical solution for the optimal policy with respect to J^k to perform the fitted value iteration steps shown in (1) and (2). To avoid biased estimates for a particular choice of l, we use the forward view of TD(λ) [32].

This version of fitted value iteration is very similar to CFVI [16]. However, instead of assuming that the input cost is separable from the state and strictly convex, we instead assume that it is quadratic with respect to some function that maps the current state to a positive definite matrix. This differing assumption allows us to propose cost functionals, of a form different than those that the CFVI algorithm is able to learn, for training tasks that satisfy our definition of independence.

4.2 Cost Functionals for Independence

Consider the following cost functional:

$$\min_{u(\cdot)} \int_t^\infty e^{-\beta \tau} \left(q_{N+1}(x) + \|u\|^2 + \sum_{i=1}^N (L_g \tilde{J}_i(x) u)^2 \lambda_i \right) d\tau \qquad (6)$$

where $\lambda_1, \ldots, \lambda_N > 0$, $\beta \geq 0$ and $\tilde{J}_1, \ldots, \tilde{J}_N$ are a set of N learned tasks that we wish to train a new task to be independent to. It is clear that (6) is an instance of (4), as $\|u\|^2 + \sum_{i=1}^N (L_g V_i(x) u)^2 \lambda_i$ can be rearranged as

$$u^\top \left(I + \sum_{i=1}^N (L_g \tilde{J}_i(x))^\top (L_g \tilde{J}_i(x)) \lambda_i \right) u$$

and $I + \sum_{i=1}^N (L_g \tilde{J}_i(x))^\top (L_g \tilde{J}_i(x)) \lambda_i$ is positive definite. This means that the method of value iteration from the previous subsection can be used to approximate the cost functional proposed in (6). $\lambda_1, \ldots, \lambda_N$ are hyperparameters that determine how much to penalize inputs which interfere with previously trained, higher priority tasks. Through Proposition 2, we show that choosing a large enough $\lambda_1, \ldots, \lambda_N$ would result in a newly trained task, that fits our proposed cost functional, to be independent to $\tilde{J}_1, \ldots \tilde{J}_N$.

Proposition 2. *There exist scalars $\lambda_1, \ldots, \lambda_N > 0$, such that if $\tilde{J}_{N+1}$ fits a cost functional of the form in (6) that uses $\lambda_1, \ldots, \lambda_N$, then for every state x where $\tilde{J}_1, \ldots, \tilde{J}_N$ are independent, $\tilde{J}_1, \ldots, \tilde{J}_N, \tilde{J}_{N+1}$ are also independent at x.*

Proof. If our deep RL algorithm for training the function $\tilde{J}_{N+1}$ converges such that it solves the HJB equation for the cost functional in (6), then the optimal input, based on Proposition 1 as well as the work in [5], minimizes the following unconstrained optimization problem:

$$\min_u \left\{ \|u\|^2 + \sum_{i=1}^N (L_g \tilde{J}_i(x) u)^2 \lambda_i + L_g \tilde{J}_{N+1}(x) u \right\}$$

The solution to this is $u^\star = -\frac{1}{2}R(x)^{-1}(L_g\tilde{J}_{N+1}(x))^\top$ where $R(x) = I + \sum_{i=1}^{N}(L_g\tilde{J}_i(x))^\top L_g\tilde{J}_i(x)$. If $\lambda_1, \dots, \lambda_N$ are large enough, $u^\star$ approximates a solution to the following constrained optimization problem:

$$\min_u \left\{\|u\|^2 + L_g\tilde{J}_{N+1}(x)u\right\}$$
$$\text{s.t.} \sum_{i=1}^{N}(L_g\tilde{J}_i(x)u)^2 = 0$$

which means that $\langle L_g\tilde{J}_i(x), -\frac{1}{2}R(x)^{-1}(L_g\tilde{J}_{N+1}(x))^\top\rangle = 0$ for all $i \in \{1, \dots, N\}$. For the sake of a contradiction, assume that for some $x \in \mathcal{X}$, $\tilde{J}_1, \dots, \tilde{J}_N$ are independent at x and $\tilde{J}_1, \dots, \tilde{J}_N, \tilde{J}_{N+1}$ are not independent at x meaning that $L_g\tilde{J}_{N+1}(x)$ can be written as a linear combination of $L_g\tilde{J}_1(x), \dots, L_g\tilde{J}_N(x)$. Let $c_1, \dots, c_N \in \mathbb{R}$ be the multipliers for this linear combination. We can see that

$$\langle c_i(L_g\tilde{J}_i(x))^\top, -\frac{1}{2}R(x)^{-1}(L_g\tilde{J}_{N+1}(x))^\top\rangle = 0\,, 1 \le i \le N.$$

By summing each inner product together, we can then see that $\langle (L_g\tilde{J}_{N+1}(x))^\top, R(x)^{-1}(L_g\tilde{J}_{N+1}(x))^\top\rangle = 0$. $R(x)$ is positive definite, so its inverse must also be positive definite. This means that $(L_g\tilde{J}_{N+1}(x))^\top$ must be 0. However, we assumed $\tilde{J}_1, \dots, \tilde{J}_N$ to be independent, so a linear combination of $(L_g\tilde{J}_1(x))^\top, \dots, (L_g\tilde{J}_N(x))^\top$ cannot be 0, so this is a contradiction. Therefore, if $\tilde{J}_1, \dots, \tilde{J}_N$ are independent at x, then $\tilde{J}_1, \dots, \tilde{J}_N, \tilde{J}_{N+1}$ are also independent at x. □

Proposition 2 suggests that we can train a set of N tasks to be independent by training them in order using the cost functional in (6) with large enough choices of $\lambda_1, \dots, \lambda_N$.

In some cases, it is possible for a resulting learned task $\tilde{J}_{N+1}$ to not only be independent to previously trained tasks $\tilde{J}_1, \dots, \tilde{J}_N$, but to also be orthogonal. We show in Proposition 3 that if $\tilde{J}_1, \dots, \tilde{J}_N$ are independent, then the optimal policy to execute task $\tilde{J}_{N+1}$ is $-\frac{1}{2}(L_g\tilde{J}_{N+1}(x))^\top$ if and only if $\tilde{J}_{N+1}$ is orthogonal to each of $\tilde{J}_1, \dots, \tilde{J}_N$.

Proposition 3. *Assume that a function $\tilde{J}_{N+1}$ solves the HJB equation and is defined based on the cost functional in (6). Without loss of generality, let $\lambda_1 = \cdots = \lambda_N = 1$. We assume as before that at least one of $L_g\tilde{J}_1(x), \dots, L_g\tilde{J}_N(x), L_g\tilde{J}_{N+1}(x)$ is non-zero $\forall x \in \mathcal{X}$. At a state x, if $J_1, \dots, J_N$ in the cost functional of $\tilde{J}_{N+1}$ are independent, then $\tilde{J}_{N+1}$ is orthogonal to each of $J_i \in \{J_1, \dots, J_N\}$ at x if and only if the optimal input at x with respect to $\tilde{J}_{N+1}$ is equal to $-\frac{1}{2}(L_g\tilde{J}_{N+1}(x))^\top$.*

Proof. (⇒) Assume that $\tilde{J}_{N+1}$ is orthogonal to each of $\tilde{J}_1, \dots, \tilde{J}_N$ at x. As shown in the work in [5] and used similarly in Proposition 2, the optimal input with respect to $\tilde{J}_{N+1}$ is an input that minimizes the following optimization problem:

$$\min_u \left\{\|u\|^2 + \sum_{i=1}^{N}(L_g\tilde{J}_i(x)u)^2 + L_g\tilde{J}_{N+1}(x)u\right\}$$

We can see that if $\tilde{J}_{N+1}$ is orthogonal to each of $\tilde{J}_1, \dots, \tilde{J}_N$ at x, then

$$\sum_{i=1}^{N}\left(L_g\tilde{J}_i(x)\left(-\frac{1}{2}(L_g\tilde{J}_{N+1})^\top(x)\right)\right)^2 = 0$$

We also know that $-\frac{1}{2}(L_g\tilde{J}_{N+1}(x))^\top$ minimizes the function $\|u\|^2 + L_g\tilde{J}_{N+1}(x)u$. Therefore, if $\tilde{J}_{N+1}$ is orthogonal to each of $\tilde{J}_1, \dots, \tilde{J}_N$ at x, then the optimal input at x is $-\frac{1}{2}(L_g\tilde{J}_{N+1}(x))^\top$.

(⇐) Assume that $u^\star = -\frac{1}{2}(L_g\tilde{J}_{N+1}(x))^\top$ is the optimal input. We know that the optimal input must solve the following optimization problem:

$$\min_u \left\{\|u\|^2 + \sum_{i=1}^{N}(L_g\tilde{J}_i(x)u)^2 + L_g\tilde{J}_{N+1}(x)u\right\}$$

which means it must also solve the following equation when we take the gradient and set it equal to 0:

$$2u + 2g(x)^\top\left(\sum_{i=1}^{N}\left(\frac{\partial\tilde{J}_i}{\partial x}\right)^\top\frac{\partial\tilde{J}_i}{\partial x}\right)g(x)u + (L_g\tilde{J}_{N+1}(x))^\top = 0$$

Substituting in $u^\star = -\frac{1}{2}(L_g\tilde{J}_{N+1}(x))^\top$, we get:

$$2g(x)^\top\left(\sum_{i=1}^{N}\left(\frac{\partial\tilde{J}_i}{\partial x}\right)^\top\frac{\partial\tilde{J}_i}{\partial x}\right)g(x)u = 0$$
$$\sum_{i=1}^{N}(L_g\tilde{J}_i(x))^\top\langle(L_g\tilde{J}_i(x))^\top, (L_g\tilde{J}_{N+1}(x))^\top\rangle = 0$$

Since we assumed that each of $\tilde{J}_1, \dots, \tilde{J}_N$ are independent at x, we cannot find a set of non-zero constants that make this sum to 0, so it must be the case that $\langle(L_g\tilde{J}_i(x))^\top, (L_g\tilde{J}_{N+1}(x))^\top\rangle = 0$ for each $i \in \{1, \dots, N\}$.

□

If a task, $\tilde{J}_{N+1}$, can be trained such that it is orthogonal to a set of other independent tasks, $\tilde{J}_1, \dots, \tilde{J}_N$, then not only is it possible to execute each task concurrently, it is also possible to execute the optimal input for $\tilde{J}_{N+1}$ without having to calculate the gradients for tasks that $\tilde{J}_{N+1}$ was trained to be independent to. If $\tilde{J}_{N+1}$ was trained without a discount factor, then the min-norm controller in [21] would recover the exact optimal input for $\tilde{J}_{N+1}$ without interfering with the execution of higher priority tasks.

To summarize this section, we propose a cost functional in (6) that we prove in Proposition 2 can help us define tasks that are independent, and thus possible to execute together simultaneously. With Proposition 1, we propose a version of fitted value iteration [3] that extends the continuous fitted value iteration algorithm [16] to fit this cost functional efficiently. Our method of fitted value iteration, similarly to [16], can be accelerated vastly using GPUs, as calculating the optimal inputs with respect to the gradients, advancing the states to generate new value function estimates and updating the parameters of the neural network can be done in large batches using GPUs. Additionally, we show in Proposition 3 the relationship between the optimal input of a learned task and whether or not it is orthogonal to the set of tasks it was trained to be independent to.

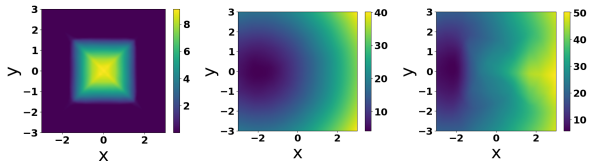

Figure 1: (a): Heatmap of function, $\tilde{J}_1$, trained for avoidance task. (b): Heatmap of function, $\tilde{J}_{\text{base}}$, trained using CFVI. (c): Heatmap of function, $\tilde{J}_{\text{ind}}$, trained using proposed method to be independent to avoidance task. Note that gradients of $\tilde{J}_{\text{ind}}$ appear to be linearly independent to those of $\tilde{J}_1$ as values in (c) appear to warp around the square region.

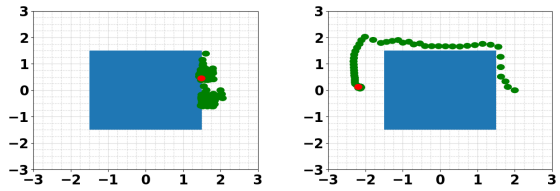

Figure 2: (a): Trajectory when combining avoidance task, $\tilde{J}_1$, with baseline go-to-point task, $\tilde{J}_{\text{base}}$. (b): Trajectory when combining $\tilde{J}_1$ with go-to-point task trained using proposed approach, $\tilde{J}_{\text{ind}}$. Green: Positions at each time step. Red: Final position for each trajectory. Blue: Region to avoid.

In the next section, we demonstrate our proposed fitted value iteration algorithm and implications of Proposition 2 in several scenarios, training robotic systems to be able to concurrently execute tasks that may have otherwise had conflicting objectives.

5 SIMULATIONS

In this section, we demonstrate our proposed method on several scenarios involving mobile robots. We show that we are able to use our approach to train cost-to-go functions, that may have had otherwise conflicting objectives, such that they are possible to combine and execute together concurrently. We compare our approach against simply learning the separate tasks using CFVI [16].

5.1 Go-to-Point and Avoid Square Region

We first demonstrate our proposed method by training a mobile robot to move to a point while avoiding a square-shaped region. The state space $\mathcal{X} = \mathbb{R}^2$ is the robot position and the input space $\mathcal{U} = \mathbb{R}^2$ is the velocity.

We first train the task of avoiding the region, using an instantaneous cost of $q_1(x) + \|u\|^2$ where $q_1(x) = 60$ when the robot is within the region and 0 otherwise. We use CFVI to train a cost-to-go function, $\tilde{J}_1$, plotted in Fig. 1a.

Next, we train two versions of the task of moving the robot to the point $\begin{bmatrix}-2 & 0\end{bmatrix}^\top$. The first version is trained using CFVI with an instantaneous cost of $q_2(x) + \|u\|^2$, while the second version is trained, using our proposed method, with an instantaneous cost of $q_2(x) + \|u\|^2 + 10^4(L_g\tilde{J}_1(x)u)^2$, where $q_2(x)$ is the distance from the current position $x \in \mathcal{X}$ to $\begin{bmatrix}-2 & 0\end{bmatrix}^\top$ multiplied by 5. We will refer to the resulting functions, both plotted in Fig. 1, as $\tilde{J}_{\text{base}}$ and $\tilde{J}_{\text{ind}}$ respectively. We can see that there is an imprint of the region in the case of $\tilde{J}_{\text{ind}}$.

Finally, we combined $\tilde{J}_1$ with $\tilde{J}_{\text{base}}$ and $\tilde{J}_{\text{ind}}$ respectively using the min-norm controller in (5). The trajectories of the resulting controllers are shown in Fig. 2. We can see that since the gradients of $\tilde{J}_1$ and $\tilde{J}_{\text{base}}$ go in opposing directions, the agent is not able to move to the goal point from certain states. Meanwhile, our proposed approach influenced the gradients of $\tilde{J}_{\text{ind}}$ and $\tilde{J}_1$ to be linearly independent, such that when they are combined with the min-norm controller in (5), the resulting controller moves the robot to the goal point while successfully avoiding the region.

This illustrative example demonstrates how our approach can very visibly influence the gradients of a learned cost-to-go function to be independent to other cost-to-go functions, resulting in tasks that can be executed concurrently.

5.2 Form Shape and Avoid Square Region

In this next example, we train a team of three mobile robots to avoid a square-shaped region in the centre of the environment and form a triangle simultaneously. The state space, $\mathcal{X} = \mathbb{R}^6$, is the position of all robots where $\forall x \in \mathcal{X}, x = \begin{bmatrix}p_1 & p_2 & p_3\end{bmatrix}^\top$ and $p_1, p_2, p_3 \in \mathbb{R}^2$ each represent the positions of robots 1, 2 and 3 respectively. The input space $\mathcal{U} = \mathbb{R}^6$ is the velocities of all robots where $\forall u \in \mathcal{U}, u = \begin{bmatrix}v_1 & v_2 & v_3\end{bmatrix}^\top$ and $v_1, v_2, v_3 \in \mathbb{R}^2$ each represent the velocities of robots 1, 2 and 3 respectively.

We first train the task of avoiding the region using an instantaneous cost of $q_1(x) + \|u\|^2$. $q_1(x) = a_1(x) + a_2(x) + a_3(x)$ where $a_i(x) = 35$ if robot i is inside the 1x1 region and $a_i(x) = 0$ otherwise. We train this task as a single-robot task and covert it to a multi-robot task as described in Remark 3. We will denote the approximated cost functional for this first task as $\tilde{J}_1$.

Remark 3. Consider a system comprising of N robots where each robot has a state space $\bar{\mathcal{X}} = \mathcal{R}^n$. The state space for the whole system is $\mathcal{X} = \mathcal{R}^{Nn}$. Consider a learned cost-to-go function of the form $\tilde{j}(\bar{x})$ where $\bar{x} \in \bar{\mathcal{X}}$. This can be turned into a multi-robot task by defining $\tilde{J}(x) = \sum_{i \in T} \tilde{j}(\bar{x}_i)$, for $x \in \mathcal{X}$ and $x_i \in \bar{\mathcal{X}}$, where T is the set of robots to assign the single-robot task, $\tilde{j}$, to and $\bar{x}_i$ is the subset of state variables in x used to represent the state of robot i. When calculating $L_f\tilde{J}$ and $L_g\tilde{J}$ using back-propagation, every robot not assigned task $\tilde{j}$ will simply have gradients of zero. This is similar to what is described in Remark 5 in [21].

We then train the triangle formation task. We train two trials using an instantaneous cost of $q_2(x) + \|u\|^2$ using CFVI and two trials using an instantaneous cost of $q_2(x) + \|u\|^2 + (L_g\tilde{J}_1(x)u)^2\lambda$ to test our proposed method. We set $\lambda = 50000$. $q_2(x)$ is equal to:

$$(|\|p_1 - p_2\| - 0.75| + |\|p_1 - p_3\| - 0.75| + |\|p_2 - p_3\| - 0.75|) \cdot 1.5.$$

$q_2(x) = 0$ when the robots form an equilateral triangle with each side of length 0.75.

After training all versions of the formation task, we combine each version with $\tilde{J}_1$ and test on the same 50 initial states. The initial

Table 1: Success Rates of Avoiding Region and Forming a Triangle

Training Method for Formation Task	*Success Rate*
CFVI - Trial 1	0.4
CFVI - Trial 2	0.34
Ours - Trial 1	0.74
Ours - Trial 2	0.92

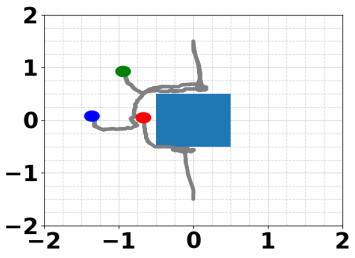

Figure 3: Robot team forming triangle while avoiding region. Coloured Dots: Robots 1-3. Lighter Blue: Region to avoid. Grey: Trajectories of each robot.

states were randomly generated from a uniform distribution of a 2x2 area encapsulating the region that $\tilde{J}_1$ was trained to avoid. We considered the team of robots to have sucessfully completed both the avoidance and formation tasks if $q_1(x) = 0$ and $q_2(x) < 0.4$.

The results of this experiment are shown in Table 1. We can see that our proposed method allowed the team of robots to successfully avoid the region and form a triangle simultaneously much more frequently than when we simply trained the formation task using CFVI.

5.3 Collaborative Transportation while Avoiding Square Regions

We extend the previous example to train a team of three mobile robots to perform the following three tasks simultaneously: avoid three square-shaped regions in the environment, send one robot to a specified point and form the shape of a triangle. Like in the previous example, the state space, $\mathcal{X} = \mathbb{R}^6$, is the position of all robots where $\forall x \in \mathcal{X}, x = \begin{bmatrix} p_1 & p_2 & p_3 \end{bmatrix}^\top$ and $p_1, p_2, p_3 \in \mathbb{R}^2$ each represent the positions of robots 1, 2 and 3 respectively. The input space $\mathcal{U} = \mathbb{R}^6$ is the velocities of all robots where $\forall u \in \mathcal{U}, u = \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix}^\top$ and $v_1, v_2, v_3 \in \mathbb{R}^2$ each represent the velocities of robots 1, 2 and 3 respectively.

We first train the task of avoiding three obstacles placed in the environment. We use an instantaneous cost function of $q_1(x) + \|u\|^2$ where $q_1(x) = b_1(x) + b_2(x) + b_3(x)$ where $b_i(x) = 25$ if robot i is inside any of the regions and $b_i(x) = 0$ otherwise. We will denote the learned cost-to-go function for this task as $\tilde{J}_1$.

We then train the triangle formation task. Similar to before, we use an instantaneous cost function of $q_2(x) + \|u\|^2$ for the baseline trials where we use CFVI and we use an instantaneous cost function of $q_2(x) + \|u\|^2 + (L_g \tilde{J}_1(x)u)^2\lambda$ to test our proposed method. We set $\lambda = 50000$. $q_2(x)$ is as defined previously in 5.3 where $q_2(x) = 0$ when the robots have formed an equilateral triangle of length 0.75. We train three baseline trials and three trials testing our proposed method.

Table 2: Success Rates of Avoiding Region, Going to Origin and Forming a Triangle

Training Method for Formation Task	*Training Method for Go-to-Point Task*	*Success Rate*
CFVI - Trial 1	CFVI	0.24
CFVI - Trial 1	Ours	0.18
CFVI - Trial 2	CFVI	0.16
CFVI - Trial 2	Ours	0.78
CFVI - Trial 3	CFVI	0.2
CFVI - Trial 3	Ours	0.7
Ours - Trial 1	CFVI	0.32
Ours - Trial 1	Ours	0.82
Ours - Trial 2	CFVI	0.18
Ours - Trial 2	Ours	0.54
Ours - Trial 3	CFVI	0.32
Ours - Trial 3	Ours	0.72

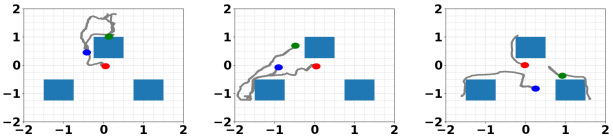

Figure 4: Robot team successfully forming triangle, sending first robot to origin and avoiding regions. Coloured Dots: Robots 1-3. Lighter Blue: Regions to avoid. Grey: Trajectories of each robot.

Next, we train the single-robot task of going to the origin. We use an instantaneous cost of $q_3(\bar{x}) + \|\bar{u}\|^2$ for the baseline task trained with CFVI and an instantaneous cost of $q_3(\bar{x}) + \|\bar{u}\|^2 + (L_g \tilde{J}_1(\bar{x})\bar{u})^2$ for the task trained with our proposed method where $\bar{x} \in \mathcal{R}^2$ is the state for a single robot, $\bar{u} \in \mathbb{R}^2$ is the input velocity for a single robot and $q_3(\bar{x})$ is the distance that robot is away from the origin multiplied by 12. This single-robot task is turned into a multi-robot task in the way described in Remark 3. Note that we did not train the go-to-point task to explicitly be independent to the formation task as it should already be independent by construction.

Finally, we test each combination of trained models on 50 randomly generated initial states and test the ability of each composed controller to achieve all three tasks. To randomly generate the initial states, we selected a position from a uniform distribution from space in the environment not covered by any of the three regions. We then added Gaussian noise to this position, to generate initial states where each robot was approximately close to one another. This was done to simulate the team of robots transporting objects

collaboratively. We consider the system to have successfully completed all three tasks when $q_1(x) = 0$, when no robot is inside any of the three regions, $q_2(x) < 0.75$ and when $q_3(x) < 1.8$. The avoidance task was given the highest priority and the formation task was given the lowest.

The results of this experiment are shown in Table 3. We can see that the best performing controller is composed of a formation task and go-to-point task trained using our proposed method and that the worst performing controller is one where the formation task and go-to-point task were both trained using just CFVI and not to be independent to the avoidance task. The formation task seemed to benefit less from being trained to be independent to the avoidance task, compared to the previous example in 5.3, and this may be because of how the initial states were generated and because of the regions being smaller in this scenario. However, our proposed method still appeared to have made all three tasks more likely to be able to execute together concurrently.

5.3.1 Time-Varying Task Stacks. This example also demonstrates the benefit of our overall approach to multi-objective RL. If we want to train our system to go to a different point or perform some other task while still avoiding the regions and forming a triangle, we can simply train a new third task that is independent to these two tasks.

5.4 Training Implementation Details

Note that the large values of λ used in these simulations are not necessary and practitioners should be able to recreate similar results using much smaller values of λ. Futhermore, values of λ that are too large may lead to unstable training using fitted value iteration and other deep RL methods. In some instances, practitioners may find it useful to limit the term that contains λ in the instantaneous cost. To help further guide practitioners, we have published code here: https://github.com/erablab/Value_Iteration_for_Learning_Concurrently_Executable_Robotic_Control_Tasks.

5.5 Experiment with Physical Robots

We tested the best-performing controller from section 5.3 on physical robots with a physical square-shaped obstacle, the results of which can be viewed here: https://youtu.be/8FSjqIr-pgY. We used a Vicon motion capture system to track the robots for feedback and sent velocities using the API for the DJI RoboMaster EP Core robots. The dimensions of the physical region to avoid were adjusted to account for the size of the robots.

6 CONCLUSION

In this paper, we present an approach for training multiple robotic control tasks in such a way that they can be combined and executed together in possibly time-varying prioritized stacks. We start by defining a notion of independence between learned control tasks which characterize the abililty of tasks to be concurrently executed together as part of constraints in the same min-norm optimization-based controller. We later propose a cost functional which can be fitted by RL algorithms such that the resulting learned tasks satisfy our definition of independence. We also present a variant of the fitted value iteration algorithm that can be used to fit our cost functional, extending previous work related to value iteration in the continuous control setting. Finally, we demonstrate our approach on several scenarios involving mobile robots, showing how our approach can be used to train tasks, that may have otherwise had conflicting objectives, to be concurrently executable.

REFERENCES

[1] Gianluca Antonelli. 2009. Stability Analysis for Prioritized Closed-Loop Inverse Kinematic Algorithms for Redundant Robotic Systems. *IEEE Transactions on Robotics* 25, 5 (2009), 985–994. https://doi.org/10.1109/TRO.2009.2017135

[2] P. Baerlocher and R. Boulic. 1998. Task-priority formulations for the kinematic control of highly redundant articulated structures. In *Proceedings. 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications (Cat. No.98CH36190)*, Vol. 1. 323–329 vol.1. https://doi.org/10.1109/IROS.1998.724639

[3] D. Bertsekas. 2019. *Reinforcement Learning and Optimal Control.* Athena Scientific. https://books.google.ca/books?id=ZlBIyQEACAAJ

[4] Jorge Cortes and Magnus Egerstedt. 2017. Coordinated Control of Multi-Robot Systems: A Survey. *SICE Journal of Control, Measurement, and System Integration* 10, 6 (2017), 495–503. https://doi.org/10.9746/jcmsi.10.495 arXiv:https://doi.org/10.9746/jcmsi.10.495

[5] Kenji Doya. 2000. Reinforcement Learning in Continuous Time and Space. *Neural Computation* 12, 1 (2000), 219–245. https://doi.org/10.1162/089976600300015961

[6] M. Egerstedt. 2021. *Robot Ecology: Constraint-Based Design for Long-Duration Autonomy.* Princeton University Press. https://books.google.ca/books?id=gnYvEAAAQBAJ

[7] Adrien Escande, Nicolas Mansard, and Pierre-Brice Wieber. 2014. Hierarchical quadratic programming: Fast online humanoid-robot motion generation. *The International Journal of Robotics Research* 33, 7 (2014), 1006–1028. https://doi.org/10.1177/0278364914521306 arXiv:https://doi.org/10.1177/0278364914521306

[8] Felix Faber, Maren Bennewitz, and Sven Behnke. 2008. Controlling the gaze direction of a humanoid robot with redundant joints. In *RO-MAN 2008 - The 17th IEEE International Symposium on Robot and Human Interactive Communication.* 413–418. https://doi.org/10.1109/ROMAN.2008.4600701

[9] R.A. Freeman and J.A. Primbs. 1996. Control Lyapunov functions: new ideas from an old source. In *Proceedings of 35th IEEE Conference on Decision and Control*, Vol. 4. 3926–3931 vol.4. https://doi.org/10.1109/CDC.1996.577294

[10] Ivo Grondman, Lucian Busoniu, Gabriel A. D. Lopes, and Robert Babuska. 2012. A Survey of Actor-Critic Reinforcement Learning: Standard and Natural Policy Gradients. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42, 6 (2012), 1291–1307. https://doi.org/10.1109/TSMCC.2012.2218595

[11] Tuomas Haarnoja, Vitchyr Pong, Aurick Zhou, Murtaza Dalal, Pieter Abbeel, and Sergey Levine. 2018. Composable Deep Reinforcement Learning for Robotic Manipulation. In *2018 IEEE International Conference on Robotics and Automation (ICRA).* 6244–6251. https://doi.org/10.1109/ICRA.2018.8460756

[12] Conor F. Hayes, Roxana Rădulescu, Eugenio Bargiacchi, Johan Källström, Matthew Macfarlane, Mathieu Reymond, Timothy Verstraeten, Luisa M. Zintgraf, Richard Dazeley, Fredrik Heintz, Enda Howley, Athirai A. Irissappane, Patrick Mannion, Ann Nowé, Gabriel Ramos, Marcello Restelli, Peter Vamplew, and Diederik M. Roijers. 2022. A practical guide to multi-objective reinforcement learning and planning. *Autonomous Agents and Multi-Agent Systems* 36, 1 (13 Apr 2022), 26. https://doi.org/10.1007/s10458-022-09552-y

[13] N. Hazon and G.A. Kaminka. 2005. Redundancy, Efficiency and Robustness in Multi-Robot Coverage. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation.* 735–741. https://doi.org/10.1109/ROBOT.2005.1570205

[14] Xiangkun He, Zhongxu Hu, Haohan Yang, and Chen Lv. 2024. Personalized robotic control via constrained multi-objective reinforcement learning. *Neurocomputing* 565 (2024), 126986. https://doi.org/10.1016/j.neucom.2023.126986

[15] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2019. Continuous control with deep reinforcement learning. arXiv:1509.02971 [cs.LG] https://arxiv.org/abs/1509.02971

[16] Michael Lutter, Shie Mannor, Jan Peters, Dieter Fox, and Animesh Garg. 2021. Value Iteration in Continuous Actions, States and Time. In *Proceedings of the 38th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 139)*, Marina Meila and Tong Zhang (Eds.). PMLR, 7224–7234. https://proceedings.mlr.press/v139/lutter21a.html

[17] Kai-Chieh Ma, Zhibei Ma, Lantao Liu, and Gaurav S. Sukhatme. 2018. *Multi-robot Informative and Adaptive Planning for Persistent Environmental Monitoring.* Springer International Publishing, Cham, 285–298. https://doi.org/10.1007/978-3-319-73008-0_20

[18] Dejan Milutinovic and Jacob Rosen. 2012. *Redundancy in Robot Manipulators and Multi-Robot Systems.* Springer Publishing Company, Incorporated.

[19] Signe Moe, Gianluca Antonelli, Andrew R. Teel, Kristin Y. Pettersen, and Johannes Schrimpf. 2016. Set-Based Tasks within the Singularity-Robust Multiple Task-Priority Inverse Kinematics Framework: General Formulation, Stability Analysis,

and Experimental Results. *Frontiers in Robotics and AI* 3 (2016). https://doi.org/10.3389/frobt.2016.00016

[20] Yoshihiko Nakamura, Hideo Hanafusa, and Tsuneo Yoshikawa. 1987. Task-Priority Based Redundancy Control of Robot Manipulators. *The International Journal of Robotics Research* 6, 2 (1987), 3–15. https://doi.org/10.1177/027836498700600201 arXiv:https://doi.org/10.1177/027836498700600201

[21] Gennaro Notomista. 2024. A Constrained-Optimization Approach toÂătheÂăExecution ofÂăPrioritized Stacks ofÂăLearned Multi-robot Tasks. In *Distributed Autonomous Robotic Systems*, Julien Bourgeois, Jamie Paik, Benoît Piranda, Justin Werfel, Sabine Hauert, Alyssa Pierson, Heiko Hamann, Tin Lun Lam, Fumitoshi Matsuno, Negar Mehr, and Abdallah Makhoul (Eds.). Springer Nature Switzerland, Cham, 479–493.

[22] Gennaro Notomista, Siddharth Mayya, Mario Selvaggio, MarÃŋa Santos, and Cristian Secchi. 2020. A Set-Theoretic Approach to Multi-Task Execution and Prioritization. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 9873–9879. https://doi.org/10.1109/ICRA40945.2020.9196741

[23] Gennaro Notomista, Claudio Pacchierotti, and Paolo Robuffo Giordano. 2022. Multi-Robot Persistent Environmental Monitoring Based on Constraint-Driven Execution of Learned Robot Tasks. In *2022 International Conference on Robotics and Automation (ICRA)*. 6853–6859. https://doi.org/10.1109/ICRA46639.2022.9811673

[24] Gennaro Notomista, Mario Selvaggio, MarÃŋa Santos, Siddharth Mayya, Francesca Pagano, Vincenzo Lippiello, and Cristian Secchi. 2023. Beyond Jacobian-based tasks: Extended set-based tasks for multi-task execution and prioritization. arXiv:2310.16189 [eess.SY]

[25] Xue Bin Peng, Michael Chang, Grace Zhang, Pieter Abbeel, and Sergey Levine. 2019. MCP: Learning Composable Hierarchical Control with Multiplicative Compositional Policies. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alche-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2019/file/95192c98732387165bf8e396c0f2dad2-Paper.pdf

[26] James A. Primbs, Vesna NevistiÃĎâĂą, and John C. Doyle. 1999. NONLINEAR OPTIMAL CONTROL: A CONTROL LYAPUNOV FUNCTION AND RECEDING HORIZON PERSPECTIVE. *Asian Journal of Control* 1, 1 (1999), 14–24. https://doi.org/10.1111/j.1934-6093.1999.tb00002.x arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1934-6093.1999.tb00002.x

[27] Thomas M. Roehr. 2022. Active Exploitation of Redundancies in Reconfigurable Multirobot Systems. *IEEE Transactions on Robotics* 38, 1 (2022), 180–196. https://doi.org/10.1109/TRO.2021.3118284

[28] Diederik M. Roijers, Peter Vamplew, Shimon Whiteson, and Richard Dazeley. 2013. A survey of multi-objective sequential decision-making. *J. Artif. Int. Res.* 48, 1 (oct 2013), 67âĂŞ113.

[29] John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. 2016. High-Dimensional Continuous Control Using Generalized Advantage Estimation. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1506.02438

[30] Bruno Siciliano. 1990. Kinematic control of redundant robot manipulators: A tutorial. *Journal of Intelligent and Robotic Systems* 3, 3 (01 Sep 1990), 201–212. https://doi.org/10.1007/BF00126069

[31] Bruno Siciliano and J.-J.E. Slotine. 1991. A general framework for managing multiple tasks in highly redundant robotic systems. In Fifth international conference on advanced robotics (Vol. 2. 1211 – 1216 vol.2. https://doi.org/10.1109/ICAR.1991.240390

[32] Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction* (second ed.). The MIT Press. http://incompleteideas.net/book/the-book-2nd.html

[33] Emanuel Todorov. 2009. Compositionality of optimal control laws. In *Advances in Neural Information Processing Systems*, Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta (Eds.), Vol. 22. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2009/file/3eb71f6293a2a31f3569e10af6552658-Paper.pdf

[34] Benjamin Van Niekerk, Steven James, Adam Earle, and Benjamin Rosman. 2019. Composing Value Functions in Reinforcement Learning. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 6401–6409. https://proceedings.mlr.press/v97/van-niekerk19a.html

[35] Jie Xu, Yunsheng Tian, Pingchuan Ma, Daniela Rus, Shinjiro Sueda, and Wojciech Matusik. 2020. Prediction-Guided Multi-Objective Reinforcement Learning for Continuous Robot Control. In *Proceedings of the 37th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 119)*, Hal DaumÃĺ III and Aarti Singh (Eds.). PMLR, 10607–10616. https://proceedings.mlr.press/v119/xu20h.html

[36] Chengxu Zhou, Xin Wang, Zhibin Li, Darwin Caldwell, and Nikos Tsagarakis. 2015. Exploiting the redundancy for humanoid robots to dynamically step over a large obstacle. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 1599–1604. https://doi.org/10.1109/IROS.2015.7353581