# Dynamic Multiobjective Control for Continuous-Time Systems Using Reinforcement Learning

Victor G. Lopez, *Student Member, IEEE*, and Frank L. Lewis, *Fellow, IEEE*

***Abstract*—This paper presents an extension of the reinforcement learning algorithms to design suboptimal control sequences for multiple performance functions in continuous-time systems. The first part of the paper provides the theoretical development and studies the required conditions to obtain a state-feedback control policy that achieves Pareto optimal results for the multiobjective performance vector. Then, a policy iteration algorithm is proposed that takes into account practical considerations to allow its implementation in real-time applications for systems with partially unknown models. Finally, the multiobjective linear quadratic regulator problem is solved using the proposed control scheme and employing a multiobjective optimization software to solve the static optimization problem at each iteration.**

***Index Terms*—Multiobjective optimization, nonlinear systems, Pareto optimality, reinforcement learning.**

## I. Introduction

Reinforcement learning is a set of artificial intelligence methods that has had an increasing success in the last decade for providing a system with the ability to improve its performance as it gains experience while attempting to achieve its goals [1]–[4]. In the last few years, reinforcement learning approaches have been adopted in control theory where the performance of a dynamical system is measured by means of a scalar function that represents the cost spent by the system along time. Reinforcement learning techniques, properly defined for control of dynamical systems, are described in [5].

Many engineering problems require describing the goals of a system by means of two or more performance indices, rather than the single cost function employed in classical optimal control. Using several performance indices provides more flexibility to represent the expected behavior of the system in ways that are difficult to express otherwise. Examples of these applications can be found in [6]. The study of multiobjective optimization control is therefore a natural extension of the usual analysis in the current literature [7].

Manuscript received October 17, 2017; revised June 22, 2018; accepted August 24, 2018. Date of publication September 10, 2018; date of current version June 26, 2019. This work was supported in part by the U.S. National Science Foundation under Grant ECCS-1405173, in part by the ONR Grant N00014-17-1-2239, and in part by the National Natural Science Foundation of China under Grant 61633007. The work of V. G. Lopez was supported by the Mexican Council of Science and Technology (Conacyt). Recommended by Associate Editor Y. Song. *(Corresponding author: Victor Gabriel Lopez Mejia.)*

V. G. Lopez is with the UTA Research Institute, University of Texas at Arlington, Fort Worth, TX 76118 USA (e-mail: victor.lopezmejia@mavs.uta.edu).

F. L. Lewis is with the UTA Research Institute, University of Texas at Arlington, Fort Worth, TX 76118 USA, and he is a Qian Ren Consulting Professor at the State Key Laboratory of Synthetical Automation for Process Industries, Northeastern University, Shenyang 110819, China (e-mail: lewis@uta.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TAC.2018.2869462

Multiobjective optimal control has been studied in [7]–[9] where the concept of Pareto domination is employed to compare the desirability of two vector functions. Using this notion, the control input is designed such that improving an objective function unilaterally implies making another worse [10], [11]. Most of the papers in the literature deal with static multiobjective optimization. To the best of our knowledge, no other paper presents a practical method of policy iteration to solve the multiobjective optimization problem for nonlinear dynamical continuous-time systems.

Several methods exist to compute a Pareto optimum. These include numerical methods embedded in software packages, as well as analytic procedures such as the weighted sum, or scalarization, technique [11]. The weighted sum method consists of combining the different cost indices into a single scalar function by computing their convex sum. This is a practical method in many applications, but it presents many technical drawbacks. These include the presence of unreachable Pareto optimal results, a strong, nonintuitive dependence on the selected sum weights, and a large computational burden as the number of desired solutions increases [11]. For these reasons, our paper presents a general approach for multiobjective control that can be applied with any of the existing multiobjective optimization methods.

The main motivation of our work is to design a control strategy that allows to solve optimization problems that cannot be expressed by a single cost function. When the objectives of the system are conflicting with each other, a tradeoff must be achieved. The controller is based on reinforcement learning methods to avoid the difficult task of solving the Hamilton–Jacobi–Bellman equation [2] and to relax the need of full knowledge of the dynamic model of the system.

As main contribution of the paper, a reinforcement learning algorithm, based on policy iteration, is proposed to achieve an online solution of the multiobjective optimization problem. It is rigorously proven that, under the provided conditions, this algorithm yields a single Pareto optimal solution, in a nontrivial extension from the single-objective optimization problem. Furthermore, only partial knowledge of the system dynamics is required to achieve optimal control. This algorithm is finally formulated and analyzed for the specific case of linear systems.

The paper is organized as follows. Basic definitions for multiobjective optimization and for the multiobjective optimal control problem are described in Section II. Section III shows the basic transformations employed to obtain an iterative suboptimal control sequence. In Section IV, a policy iteration algorithm to solve the multiobjective optimization problem is designed, with considerations to allow its implementation in practical applications. Section V studies the linear systems case. Finally, Section VI concludes with a numerical example.

## II. Basic Definitions

In this section, various definitions to develop multiobjective optimization algorithms for dynamical systems are reviewed.

### A. Pareto Optimality

Multiobjective optimization deals with the problem of minimizing two or more objective functions simultaneously [10]. In mathematical terms, this problem is expressed as follows:

$$\min_{x\in X} V(x) \tag{1}$$

where $x \in \mathbb{R}^n$ is selected inside a feasible set $X$ and $V : \mathbb{R}^n \to \mathbb{R}^M$ is a vector function with $M$ elements, $V(x) = [V_1(x), \ldots, V_M(x)]^T$, with $V_i(x), i = 1, \ldots, M$, the functions to be minimized. In the general case, there does not exist a solution $x$ that achieves the minimization of all functions $V_i(x)$ simultaneously, and the concepts of Pareto domination and Pareto optimality must be introduced.

*Definition 1:* A vector $W \in \mathbb{R}^M$ is said to Pareto dominate vector $V \in \mathbb{R}^M$ if $W_j \leq V_j$ for all $j = 1, \ldots, M$, and $W_j < V_j$ for at least one $j$, where $V_j$ and $W_j$ are the $j$th entries of vectors $V$ and $W$, respectively.

The following definition states a specific notation that we use throughout this paper.

*Definition 2:* Notation $W \leq V$ for vectors $W \in \mathbb{R}^M$ and $V \in \mathbb{R}^M$, indicate that $W$ is not Pareto dominated by $V$, i.e., either $V = W$ or there is at least one entry $j$ such that $V_j > W_j$. Notation $W \preceq V$ means that $W_j \leq V_j$ for all $j = 1, \ldots, M$.

Employing these definitions, the concept of Pareto optimality can be stated as follows.

*Definition 3:* A solution $x^*$ of problem (1) is said to be Pareto optimal if $V(x^*) \leq V(x)$ for all $x \in X$.

The outcome $V(x^*)$ of a Pareto optimal solution $x^*$ is also said to be Pareto optimal. In general, a multiobjective optimization problem has multiple Pareto optimal outcomes, and the set of all Pareto optimal outcomes for a given problem is regarded as the Pareto front. Here, we represent the Pareto front as $\mathcal{V}^\pi$, such that $V(x^*) \in \mathcal{V}^\pi$.

### B. Multiobjective Performance of a Dynamical System

Consider a general nonlinear system with dynamics

$$\dot{x} = f(x, u) \tag{2}$$

where $x \in \mathbb{R}^n$ and $u \in \mathbb{R}^m$ are the state vector and the control input of the system, respectively, and $f$ is a continuously differentiable function.

In the multiobjective optimization problem, the performance of system (2) is evaluated with respect to $M$ different performance indices as follows:

$$J_j(x(0), u) = \int_0^\infty L_j(x(0), u)dt \tag{3}$$

where each $L_j$ is a continuously differentiable function and $j = 1, \ldots, M$. The feedback control function $u(x)$ is said to be *admissible* if it is continuous, stabilizes the dynamics (2), and makes $J_j(x, u(x))$ finite for all $j = 1, \ldots, M$. The class of functions satisfying these properties is denoted as $U^0$. Define the vector $J$ as $J = [J_1, \ldots, J_M]^T$. It is our interest to find a function $u(x) \in U^0$ such that vector $J$ is minimized in the Pareto sense.

For a fixed control policy $u(x)$, define the value functions

$$V_j(x(t)) = \int_t^\infty L_j(x(\tau), u)d\tau, \qquad j = 1, \ldots, M. \tag{4}$$

Let $V = [V_1, \ldots, V_M]^T$. A differential equivalent to the value function (4) is given by the Bellman equation

$$0 = L_j(x, u) + \nabla V_j^T f(x, u) \triangleq H_j(x, \nabla V_j, u) \tag{5}$$

where $\nabla V_j$ is the gradient of $V_j$, and $H_j(x, \nabla V_j, u)$ is the $j$th Hamiltonian function of the system. Note that the orbital derivative of $V_j(x)$ is given by

$$\dot{V}_j(x) = \nabla V_j \dot{x} = -L_j(x, u). \tag{6}$$

Define also

$$V^*(x(0)) \triangleq \inf_{u\in U^0} J(x(0), u) \tag{7}$$

where, in general, $V^*$ is not unique, and $V^* \in \mathcal{V}_j^\pi$ with $\mathcal{V}_j^\pi$ the Pareto front of vector $J$.

Define the Pareto optimal vector $H^*$ as follows:

$$H^*(x, \nabla V) = \min_{u\in U^0} H(x, \nabla V, u) \tag{8}$$

where $H = [H_1, \ldots, H_M]^T$ and $\nabla V = [\nabla V_1, \ldots, \nabla V_M]^T$. $H^*$ is Pareto optimal in the sense that, for each state vector $x$ and vector $V$, $H^*(x, \nabla V) \leq H(x, \nabla V, u)$ for every control policy $u \in U^0$.

In general, it is possible to select different control inputs, $u^{*1}$ and $u^{*2}$, such that $H(x, \nabla V, u^{*1})$ and $H(x, \nabla V, u^{*2})$ are both Pareto optimal. For this reason, we make the following assumption, useful for the analysis of Section III.

*Assumption 1:* If there exists $u^*$ such that $H^*(x, \nabla V) = 0$ for all $x$ and a given function $V(x)$, then select $u = u^*$.

Assumption 1 is a restriction on the procedure employed to find a Pareto optimal vector in (8), and states that if the vector of zeros is one of the possible Pareto optimal results for $H^*$, then control policy $u$ must be selected accordingly. The consequences of Assumption 1 are studied in Lemmas 4 and 5 in Section III.

## III. Multiobjective Suboptimal Control Sequences

This section defines and analyzes transformations to design suboptimal control policies in an iterative manner. This is an extension for multiobjective optimization of the results in [3].

Considering the multiobjective optimal control problem described in Section II, define $\mathcal{V}$ as the set of all continuously differentiable functions $V : \mathbb{R}^n \to \mathbb{R}^M$ such that $V(0) = 0$. Define also $\mathcal{V}^0$ as the subset of $\mathcal{V}$ such that $u(x, \nabla V) \in U^0$, i.e., the feedback control policies based on the vector function $V$ are admissible. Define the transformations $T_1$, $T_2$, and $T$ as follows.

*Definition 4:*

1) Define the function $T_1 : \mathcal{V}^0 \to U^0$ as $T_1(V) = T_1(V_1, \ldots, V_M) = u$ where $V = [V_1, \ldots, V_M]^T$ and $u = u(x, \nabla V)$.
2) Define the function $T_2 : U^0 \to \mathcal{V}$ as $T_2(u) = V$, where $V \in \mathbb{R}^M$ and $V_j(x) = J_j(x, u)$, $j = 1, \ldots, M$.
3) Define the composite mapping $T : \mathcal{V}^0 \to \mathcal{V}$ as $T(V_1, \ldots, V_M) = T_2(T_1(V_1, \ldots, V_M)) = J(x, u)$, where $u = u(x, \nabla V)$.

Our objective now is to use these transformations to design a control sequence that converges to an optimal policy $u^*(x) \in U^0$. We begin our analysis by studying some of the properties of the vector functions defined in Section II, as well as those of transformations $T_1$, $T_2$, and $T$.

Lemma 1 allows to compare two vector functions, $V$ and $W$, with entries $V_j$ and $W_j$ as in (4), when the respective Hamiltonian functions are known.

*Lemma 1:* If $H(x, \nabla V, u) \leq H(x, \nabla W, u)$ for given vector functions $V$, $W$, and control $u$, then $W \leq V$.

*Proof:* By Definition 2, we have that either $H(x, \nabla V, u) = H(x, \nabla W, u)$ or there exists an entry $j$ such that $H_j(x, \nabla V, u) < H_j(x, \nabla W, u)$. Assume the latter case and consider this same entry $j$. By definition of $H_j$ in (5), we have $L_j(x, u) + \nabla V_j^T f(x, u) < L_j(x, u) + \nabla W_j^T f(x, u)$, which implies $\nabla V_j^T f(x, u) < \nabla W_j^T f$

$(x,u)$; that is, $\dot{V}_j < \dot{W}_j$. Integrating the inequality along the same motions yields $W_j < V_j$ and, therefore, $W \preceq V$. □

Lemma 2 and Theorem 1 relate the Pareto optimality of vector $V$ with the Pareto optimality of vector $H$ in (8).

*Lemma 2:* Consider a control policy $u^*$ such that (8) holds. If $V_j^*(x)$, $j = 1,\dots,M$, solves the Bellman equation (5) for $u^*$, then $V^*(x)$ is Pareto optimal.

*Proof:* Consider a control policy $\bar{u}$ such that $H(x,\nabla V,\bar{u}) \neq H(x,\nabla V,u^*)$. By Pareto optimality of $H_j(x,\nabla V_j,u^*)$, there exists an entry $j$ such that

$$H_j(x,\nabla V_j,\bar{u}) > H_j(x,\nabla V_j,u^*). \tag{9}$$

For this entry $j$, let $V_j^*$ solve the Bellman equation for $u^*$ and $\bar{V}_j$ solve the Bellman equation for $\bar{u}$. Then, $H_j(x,\nabla V_j^*,\bar{u}) > H_j(x,\nabla V_j^*,u^*) = H_j(x,\nabla \bar{V}_j,\bar{u}) = 0$. Now, by Lemma 1, $H_j(x,\nabla V_j^*,\bar{u}) > H_j(x,\nabla \bar{V}_j,\bar{u})$ implies $\bar{V}_j > V_j^*$. □

*Theorem 1:* Let the control policy $u^*$ be such that (8) holds, and $V^*$ such that $V_j^*$ solves the $j$th Bellman equation (5) for $u^*$, for every entry $j = 1,\dots,M$. Then, $V^* \in \mathcal{V}_J^\pi$ with $\mathcal{V}_J^\pi$ the Pareto front of $J$ as defined in (7).

*Proof:* As $u^*$ makes $H^*$ Pareto optimal, then, by Lemma 2, $V^*$ is also Pareto optimal. Now, for all entries of vector $J$, we have

$$\begin{aligned} J_j &= \int_0^\infty L_j(x,u)dt \\ &= \int_0^\infty H_j(x,\nabla V_j^*,u)dt - V_j^*(x(\infty)) + V_j^*(x(0)). \end{aligned}$$

As $u^* \in U^0$, then $V(x(\infty)) = 0$. Therefore, $J_j = V_j^*(x(0))$ for all entries $j$, and Pareto optimality of $V^*$ implies Pareto optimality of $J$. This is $V^* \in \mathcal{V}_J^\pi$ as in (7). □

The proof of Theorem 1 shows that $V^* = J$ when $V^*$ solves the Bellman equation for $u^*$. Lemma 3 and Theorem 2 show that solving the Bellman equation, regardless of the control function $u$, is a sufficient and necessary condition for a vector $V$ to satisfy the equality $V = T_2(u) = J$.

*Lemma 3:* $V = T_2(u)$ if and only if $V_j$ satisfies $H_j(x,\nabla V,u) = 0$, for $j = 1,\dots,M$.

*Proof:* If $H_j(x,\nabla V,u) = 0$, then $\dot{V}_j = \nabla V^T f(x,u) = H_j(x,\nabla V,u) - L_j(x,u) = -L_j(x,u) = \dot{J}_j$, and integrating both sides of the equality along the same motions for all entries of the vector, yields $V = J = T_2(u)$. Conversely, if $V = J$, then $\dot{V}_j = \dot{J}_j = -L_j(x,u)$, which implies $H_j = 0$. □

*Theorem 2:* Let $V \in \mathcal{V}^0$ and $W \in \mathcal{V}$. Now, $W = T(V)$ if and only if $H(x,\nabla W,u(x,\nabla V)) = 0$.

*Proof:* The proof follows directly from Lemma 3 and Definition 4. □

Clearly, if $V$ is such that $H(x,\nabla V,u) = 0$, then $H^*(x,\nabla V) \leq 0$, which means that the vector of zeros does not Pareto dominates $H^*$. However, this does not necessarily imply that all the elements of $H^*$ are nonpositive. Lemma 4 solves this inconvenience.

*Lemma 4:* Let Assumption 1 hold. Then, $H_j^*(x,\nabla V) \leq 0$ for all entries of $H^*$.

*Proof:* By Assumption 1, if the vector of zeros is Pareto optimal, then $H_j^* = 0$ for all entries of $j$. If $H = 0$ is not Pareto optimal, then by definition of Pareto optimality we have $H_j^*(x,\nabla V) \leq 0$ for all entries of $j$ with at least one strict inequality. □

As studied below, Lemma 4 allows guaranteeing that all the entries of a vector are at least as small as the entries of another ($V \preceq W$) when an iterative algorithm is employed.

The following theorem shows the recursion required later in this section to design a suboptimal control sequence.

*Theorem 3:* Let $V \in \mathcal{V}^0$ and $\bar{V} = T(V)$, and let Assumption 1 hold. Then, $H^*(x,\nabla V) \preceq 0$ implies $V^* \preceq \bar{V} \preceq V$, with $V^*$ Pareto optimal.

*Proof:* Take $u = u^*(x,\nabla V)$. By Assumption 1 and Lemma 4, $H_j^*(x,\nabla V) \leq 0$ for every $j = 1,\dots,M$. Then, we can express $\dot{V}_j = H_j^*(x,\nabla V) - L_j(x,u) \leq -L_j(x,u) = \dot{J}_j$.

As $\bar{V}_j = T(V) = J_j$ implies $\dot{\bar{V}}_j = \dot{J}_j$, then $\dot{V}_j \leq \dot{\bar{V}}_j$. Integrating the inequality we get $\bar{V}_j \leq V_j$ for all entries $j$. □

In the single objective optimization problem, it is clear that an iterative repetition of the operation in Theorem 3 leads the function vector $\bar{V}$ to the unique optimal value function $V^*$. In the multiobjective optimization case, Assumption 1 is required to prevent leaping among different Pareto optima at each iteration, as proven in Lemma 5 and Theorem 4.

*Lemma 5:* Let $V^*$ be Pareto optimal and let Assumption 1 hold. If $W^*$ is any other Pareto optimal value function such that $V^* \neq W^*$, then $W^* \neq T(V^*)$.

*Proof:* Assume $W = T(V^*)$. If Assumption 1 holds, by Lemma 4 we have $H_j^*(x,\nabla V) \leq 0$ for all entries $j$. By Theorem 3, we have $W_j \leq V_j^*$ for all $j$. As $W_j^* > V_j^*$ for some $j$, for any other Pareto optimal vector $W^*$, then $W^*$ cannot be reached. □

*Theorem 4:* If a Pareto optimal solution $V^* \in \mathcal{V}^0$ exists, then $V^* = T(V^*)$. Conversely, $V = T(V)$ implies $V = V^*$.

*Proof:* Consider two Pareto optimal vectors $V^*$ and $W^*$. By Theorem 3, if $\bar{V} = T(V^*)$, then $W^* \preceq \bar{V} \preceq V^*$; by Lemma 5 and definition of Pareto optimality, $\bar{V} \preceq V$ implies $\bar{V} = V^*$. Conversely, if $V = T(V)$, by Theorem 2 we have $H^*(x,\nabla V) = 0$ and $V$ solves the Bellman equation (5); by Lemma 2, $V = V^*$.

We finally formalize the idea of using the result in Theorem 3 to build a sequence of successive approximations that converge to a Pareto optimal solution $V^*$.

*Theorem 5:* Take $V^* \in \mathcal{V}^0$ and $V^{k+1} = T(V^k)$. Then $V^* \leq V^{k+1} \preceq \cdots \preceq V^0$ for a Pareto optimal solution $V^*$.

*Proof:* The proof follows inductively from Theorem 3, noting that the current estimate of the optimal value function at step $k$ is $V^k = T_2(u^k)$ and taking the control policy at step $k+1$ based on $V^k$, i.e., $u^{k+1} = u(x,\nabla V^k) = T_1(V^k)$. Convergence to a single Pareto optimal result is provided by Theorem 4. □

## IV. Integral Reinforcement Learning Algorithm for Implementation of Multiobjective Suboptimal Control

In this section, we use the analysis of Section III to design an integral reinforcement learning (IRL) algorithm using the structure of policy iteration [1], [5], [12], that is shown to converge to a Pareto optimal solution of vector $V$, then used to generate the optimal policy $u(x,\nabla V^*)$. Here, it is assumed that the state values of system (2) are known, even if part of its mathematical model is uncertain.

In the work presented in [12], an IRL algorithm that converges to the solution $V^*$ of the Bellman equation for a single performance index was developed. This section presents the IRL in multiobjective optimization form.

Notice that the $j$th value function (4) can be expressed as follows:

$$V_j(x(t)) = \int_t^{t+T} L_j(x(\tau),u)d\tau + V_j(x(t+T)) \tag{10}$$

where for any time interval $T > 0$. Given the functions $V_j(x)$ and $L_j(x,u)$, (10) does not require knowledge about the system dynamics (2). Lemma 6 shows that the solution $V_j(x)$ of (10) is the value function (4) that solves (5).

**Algorithm 1:** Integral Multiobjective Policy Iteration.

1. Select an admissible control policy $u^0$.
2. Solve for $V^k$ from the set of equations

$$V_j^k(x(t)) = \int_t^{t+T} L_j(x(\tau), u)d\tau + V_j^k(x(t+T)). \quad (11)$$

3. Update the control policy as

$$u^{k+1} = \arg\min_u H(x, \nabla V^k, u), \quad (12)$$

Go to step 2. On convergence, stop. □

*Lemma 6:* Assume the control policy $u(x)$ stabilizes the system dynamics (2). Then, the solution $V_j(x)$ of (10) is equivalent to the solution of the Bellman equation (5).

*Proof:* If (5) holds for $V_j$, then $\dot{V}_j = \nabla V_j^T f(x,u) = -L_j(x,u)$. Integrating both sides of the equation, we get

$$\begin{aligned}\int_t^{t+T} L_j(x,u)d\tau &= -\int_t^{t+T} \dot{V}_j(x(\tau))d\tau \\ &= -V_j(x(t+T)) + V_j(x(t))\end{aligned}$$

which is the same equation as (10). □

The following algorithm presents the multiobjective optimal controller by reinforcement learning. The policy evaluation step consists of solving (10). This corresponds to the transformation $T_2$ in Definition 4. The policy improvement step is based on (8), and corresponds to the transformation $T_1$. Convergence of Algorithm 1 is proven in Theorem 6.

*Theorem 6:* Assume there exists an admissible control input $u$ for system (2). Perform Algorithm 1 such that Assumption 1 holds in step 3. Then, Algorithm 1 converges to a Pareto optimal solution $V^*$. Moreover, the control policy $u(x, \nabla V^*)$ optimizes the performance index vector $J$.

*Proof:* From (12) and Assumption 1, we have that $H_j(x, \nabla V_j^k, u^{k+1}) \leq 0$ for $j = 1, \ldots, M$. As function $V_j^{k+1}$ solves (11), then by Lemma 6 $H_j(x, \nabla V_j^{k+1}, u^{k+1}) = 0$. From both results we get $H_j(x, \nabla V_j^k, u^{k+1}) \leq H_j(x, \nabla V_j^{k+1}, u^{k+1})$. Lemma 1 implies that $V_j^{k+1} \leq V_j^k$ and vector $V^{k+1}$ is not Pareto dominated by $V^k$. By Theorem 5, these properties hold for every iteration until a Pareto optimal vector $V^*$ is obtained.

By Theorem 1, if (11) holds for $V^*$, then $V^* = J^*$. Thus, $V^*$ guarantees a Pareto optimal performance of the system. □

*Remark 1:* Step 3 in Algorithm 1 can be solved by any multiobjective optimization method. In this paper, we avoid the use of the weighted-sum method because it reduces the problem to a single-objective formulation that is often restrictive and is not suitable for general applications [11]. The technical drawbacks of the weighted sum method include its inability to reach results in nonconvex sections of the Pareto optimal set, a strong, nonintuitive dependence on the sum weights, and a large computational burden as the optimization problem grows in complexity.

*Remark 2:* Equation (11) avoids the use of the system dynamics (2) in the policy evaluation step of the algorithm, and (12) requires only partial knowledge of the mathematical model of the system [12].

In the following section it is shown how to use partial knowledge of a linear system with a particular Pareto optimization solver in Algorithm 1.

## V. Multiobjective Linear Quadratic Regulator

Consider a system with linear dynamics

$$\dot{x} = Ax + Bu. \quad (13)$$

The performance of the system is measured using $M$ different performance indices with quadratic terms, given by

$$J_j = \int_0^\infty \left(x^T Q_j x + u^T R_j u\right)dt \quad (14)$$

$j = 1, \ldots, M$, where $Q_j > 0$ and $R_j > 0$ are symmetric matrices. Express each of the $M$ value functions in quadratic form as follows:

$$V_j = x^T P_j x, \quad (15)$$

where $j = 1, \ldots, M$, with $P_j = P_j^T > 0$.

In order to apply the multiobjective IRL algorithm, express the functions (15) in the form (10); that is,

$$\begin{aligned}x^T(t)P_j x(t) = &\int_t^{t+T} \left(x^T Q_j x + u^T R_j u\right)d\tau \\ &+ x^T(t+T)P_j x(t+T). \quad (16)\end{aligned}$$

Solving this equation becomes an easier task if we employ the Kronecker product to express the term $x^T P_j x$ as $x^T P_j x = \text{vec}(P_j)^T (x \otimes x)$, where $\text{vec}(P_j)$ is the column vector obtained by stacking the columns of $P_j$. Moreover, as matrix $P_j$ is symmetric and the expression $x \otimes x$ includes all possible products of the entries of $x$, each of the vectors $\text{vec}(P_j)$ and $x \otimes x$ include repeated terms. Represent these vectors after removing all the redundant terms as $\bar{p}_j$ and $\bar{x}$, respectively, which consist of $n(n+1)/2$ components. Now, we can write

$$x^T P_j x = \bar{p}_j^T \bar{x}. \quad (17)$$

Using the expression (17), we rewrite (16) as follows:

$$\bar{p}_j^T(\bar{x}(t) - \bar{x}(t+T)) = \int_t^{t+T} \left(x^T Q_j x + u^T R_j u\right)d\tau \quad (18)$$

and the goal is to find the values of $\bar{p}_j$ that satisfy (18) given the measurements $x(t)$ and $x(t+T)$, and the employed control input $u$. This objective can be achieved using recursive least squares after collecting several samples of (18) [5].

The Hamiltonian functions for this system are as follows:

$$H_j = x^T Q_j x + u^T R_j u + 2x^T P_j (Ax + Bu). \quad (19)$$

The optimal control policy $u^*$ for system (13) is the input $u = -Kx$ that makes the vector $H = [H_1, \ldots, H_M]^T$ Pareto optimal.

Several methods can be used to determine $u^*$. Here, we propose a general procedure that allows this problem to be solved by any multiobjective optimization software package.

Substitute the policy $u = -Kx$ in each of the Hamiltonian functions (19), to obtain

$$\begin{aligned}H_j = &x^T Q_j x + x^T K^T R_j K x \\ &+ x^T P_j (A - BK)x + x^T (A - BK)^T P_j x. \quad (20)\end{aligned}$$

It is well known that the minimization of each individual $H_j$ with respect to $K$ is achieved using the optimal gain matrix $K^* = R_j^{-1} B^T P_j$. However, this optimization problem can be characterized differently to be programed in a multiobjective optimization solver. Theorem 7 shows that minimizing (20) by means of matrix $K$ is equivalent to minimize

**Algorithm 2.**

1. Select an admissible control policy $u^0 = K^0 x$.
2. Solve the set of (11) for $V^k$.
3. Solve the multiobjective optimization problem (23) and update the control policies as $u^{k+1} = K^{k+1} x$.

Go to step 2. On convergence, stop. □

the sum of the eigenvalues of the matrix $K^T R_j K - P_j BK - K^T B^T P_j$. To simplify the notation, define the variables

$$S_j = Q_j + K^T R_j K + P_j (A - BK) + (A - BK)^T P_j \tag{21}$$

and

$$S'_j = K^T R_j K - P_j BK - K^T B^T P_j. \tag{22}$$

The $i$th eigenvalue of a matrix $S$ is denoted as $\lambda_i(S)$.

*Theorem 7:* Let $H_j = x^T S_j x$, where $S_j$ is the symmetric matrix (21). Then, solving the minimization problem

$$K^* = \arg\min_K H_j$$

is equivalent to solving the eigenvalue minimization problem

$$K^* = \arg\min_K \sum_{i=1}^{n} \lambda_i(S'_j)$$

with $S'_j$ as in (22).

*Proof:* Take the optimal matrix $K^*$ such that $H_j^* = x^T S_j^* x$, with $S_j^* = Q_j + K^{*T} R_j K^* + P_j(A - BK^*) + (A - BK^*)^T P_j$, is minimal; this means $x^T S_j^* x \leq x^T S_j x$ for $S_j$ in (21) using any matrix $K$. Now we can write $x^T (S_j - S_j^*) x \geq 0$ and, therefore, $S_j - S_j^*$ is a positive semidefinite matrix. Note that for the matrices (21) and (22), we have $S_j - S_j^* = S'_j - S'^*_j$. As all the eigenvalues of $S'_j - S'^*_j$ are nonnegative, and the trace of a matrix is equal to the sum of its eigenvalues, then $tr(S'_j - S'^*_j) \geq 0$, which implies $tr(S'_j) \geq tr(S'^*_j)$. We conclude that matrix $K^*$ generates the matrix $S'^*_j$ with minimal sum of its eigenvalues. □

By Lemma 7, minimization of the Hamiltonian vector $H$ can be achieved by finding the gain matrix $K^*$ such that, for given matrices $P_j, j = 1, \ldots, M$, we have

$$K^* = \arg\min_K \begin{bmatrix} \sum_{i=1}^n \lambda_i \left(K^T R_1 K - P_1 BK - K^T B^T P_1\right) \\ \vdots \\ \sum_{i=1}^n \lambda_i \left(K^T R_M K - P_M BK - K^T B^T P_M\right) \end{bmatrix}. \tag{23}$$

*Remark 3:* Problem (23) is expressed without knowledge of matrix $A$ of the system dynamics (13).

Algorithm 2 expresses the policy iteration procedure presented in Algorithm 1, modified for the linear systems case.

## VI. Simulation Results

Algorithm 2 is now employed to achieve stabilization of the linearized double inverted pendulum in a cart [19], [20], represented by the dynamic (13), where

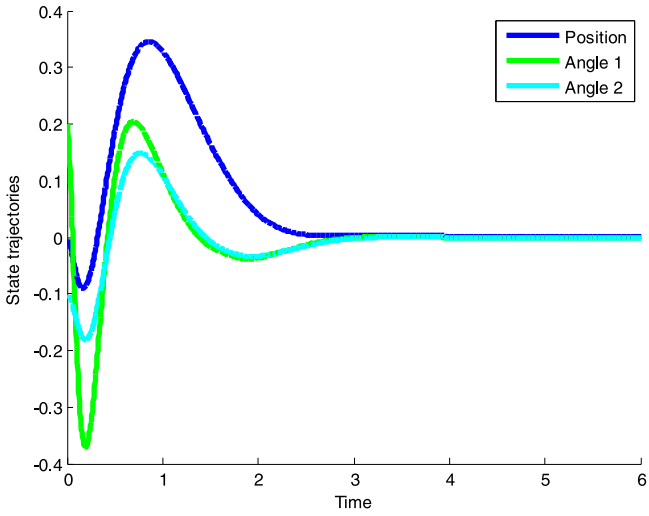


Fig. 1. State trajectories of a linear system with multiobjective optimization.

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 86.69 & -21.61 & 0 & 0 & 0 \\ 0 & -40.31 & 39.45 & 0 & 0 & 0 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 6.64 \\ 0.08 \end{bmatrix}$$

state $x_1$ is the position of the cart, $x_2$ and $x_3$ are the angles of both pendulums, and the remaining states are the velocities. Performance objectives are as follows: 1) regulation of all states is required and 2) the values of $x_2$ and $x_3$ must be as close to each other as possible. The performance indices (14) can now be defined as follows:

$$Q_1 = \begin{bmatrix} 200 & 0 & 0 & 0 & 0 & 0 \\ 0 & 200 & 0 & 0 & 0 & 0 \\ 0 & 0 & 200 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, Q_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

and $R_1 = R_2 = 1$. The sample time per iteration is $T = 0.05$.

The MATLAB function for multiobjective optimization *fgoalattain* is employed to determine the feedback control matrix $K$ at each iteration. *fgoalattain* allows to generate different points in the Pareto front of the problem. The state trajectories for $x_1, x_2$, and $x_3$ after implementation of Algorithm 2 are shown in Fig. 1. All states are shown to be stabilized by the controller. The final gain matrix $K$ is

$$K = \begin{bmatrix} 11.90 & 110.67 & -165.9 & 13.30 & 4.20 & -26.32 \end{bmatrix}.$$

## VII. Conclusion

A sequence for suboptimal control with multiple objective functions for general nonlinear systems was designed, guaranteeing its convergence to an optimal vector in the Pareto sense. The proposed policy

iteration algorithm allows solving the $M$ Bellman equations independently, using only the measurements of the system trajectories during a time interval. This control scheme can be applied in real-time without having full knowledge of the mathematical model of the system. As a case of study, the multiobjective LQR was solved.

## References


[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An introduction*, Cambridge, MA, USA: MIT Press, 1998.

[2] F. L. Lewis, D. Vrabie, and V. L. Syrmos, *Optimal Control*, 2nd ed. Hoboken, NJ, USA: Wiley, 2012.

[3] R. J. Leake and R. W. Liu, "Construction of suboptimal control sequences," *J. SIAM Control*, vol. 5, no. 1, pp. 54–63, 1967.

[4] D. P. Bertsekas, "Dynamic programming and suboptimal control: A survey from ADP to MPC," *Eur. J. Control*, vol. 11, pp. 310–334, 2005.

[5] K. G. Vamvoudakis, H. Modares, B. Kiumarsi, and F. L. Lewis, "Game theory-based control system algorithms with real-time reinforcement learning," *IEEE Control Syst. Mag.*, vol. 37, no. 1, pp. 33–52, Feb. 2017.

[6] G. P. Liu, J. B. Yang, and J. F. Whidborne, *Multiobjective Optimisation and Control*. Hertfordshire, U.K.: Research Studies Press, 2003.

[7] A. Gambier and E. Badreddin, "Multi-objective optimal control: An overview," in *Proc. IEEE Int. Conf. Control Appl.*, Oct. 1–3, 2007, pp. 170–175.

[8] F. Logist, S. Sager, C. Kirches, and J. F. Van Impe, "Efficient multiple objective optimal control of dynamic systems using integer controls," *J. Process Control*, vol. 20, pp. 810–822, 2010.

[9] A. Kumar and A. Vladimirsky, "An efficient method for multiobjective optimal control and optimal control subject to integral constraints," *J. Comput. Math.*, vol. 28, no. 4, pp. 517–551, 2010.

[10] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge Univ. press, 2004.

[11] M. Caramia and P. Dell'Olmo, "Multi-objective optimization," in *Multi-Objective Management in Freight Logistics. Increasing Capacity, Service Level and Safety With Optimization Algorithms*. London, U.K.: Springer, 2008.

[12] D. Vrabie, O. Pastravanu, M. Abu-Khalaf, and F. L. Lewis, "Adaptive optimal control for continuous-time linear systems based on policy iteration," *Automatica*, vol. 45, pp. 477–484, 2009.

[13] D. Vrabie, K. G. Vamvoudakis, and F. L. Lewis, *Optimal Adaptive Control and Differential Games by Reinforcement Learning Principles*. London, U.K.: Institution Eng. Technol., 2013.

[14] D. Liu, Q. Wei, D. Wang, X. Yang, and H. Li, *Adaptive Dynamic Programming With Applications in Optimal Control*, New York, NY, USA: Springer, 2017.

[15] F.-Y. Wang, H. Zhang, and D. Liu, "Adaptive dynamic programming: An introduction," *IEEE Comput. Intell. Mag.*, vol. 4, no. 2, pp. 39–47, May 2009.

[16] R. Kamalapurkar, L. Andrews, P. Walters, and W. E. Dixon, "Model-based reinforcement learning for infinite-horizon approximate optimal tracking," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 3, pp. 753–758, Mar. 2017.

[17] T. Bian, Y. Jiang, and Z.-P. Jiang, "Adaptive dynamic programming and optimal control of nonlinear nonaffine systems," *Automatica*, vol. 50, pp. 2624–2632, 2014.

[18] Q. Yang and S. Jagannathan, "Reinforcement learning controller design for affine nonlinear discrete-time systems using online approximators," *IEEE Trans. Syst., Man Cybern.-Part B, Cybern.*, vol. 42, no. 2, pp. 377–390, Apr. 2012.

[19] Q.-R. Li, W.-H. Tao, N. Sun, C.-Y. Zhang, and L.-H. Yao, "Stabilization control of double inverted pendulum system," in *Proc. 3rd Int. Conf. Innovative Comput. Inf. Control*, Jun. 18–20, 2008.

[20] J.-L. Zhang and W. Zhang, "LQR self-adjusting based control for the planar double inverted pendulum," *Physics Procedia*, vol. 24, Part C, pp. 1669–1676, 2012.

[21] Y. Song, Y. Wang, and C. Wen, "Adaptive fault-tolerant PI tracking control with guaranteed transient and steady-state performance," *IEEE Trans. Autom. Control*, vol. 62, no. 1, pp. 481–487, Jan. 2017.

[22] Y. Song, X. Huang, and C. Wen, "Tracking control for a class of unknown nonsquare MIMO nonaffine systems: A deep-rooted information based robust adaptive approach," *IEEE Trans. Autom. Control*, vol. 61, no. 10, pp. 3227–3233, Oct. 2016.