Proceedings of Machine Learning Research 120:1–10, 2020 2nd Annual Conference on Learning for Dynamics and Control

Counterfactual Programming for Optimal Control

Luiz F. O. Chamon LUIZF@SEAS.UPENN.EDU
Santiago Paternain SPATER@SEAS.UPENN.EDU
Alejandro Ribeiro ARIBEIRO@SEAS.UPENN.EDU
Electrical and Systems Engineering, University of Pennsylvania.

Editors: A. Bayen, A. Jadbabaie, G. J. Pappas, P. Parrilo, B. Recht, C. Tomlin, M. Zellinger

Abstract

In recent years, considerable work has been done to tackle the issue of designing control laws based on observations to allow unknown dynamical systems to perform pre-specified tasks. At least as important for autonomy, however, is the issue of learning which tasks can be performed in the first place. This is particularly critical in situations where multiple (possibly conflicting) tasks and requirements are demanded from the agent, resulting in infeasible specifications. Such situations arise due to over-specification or dynamic operating conditions and are only aggravated when the dynamical system model is learned through simulations. Often, these issues are tackled using regularization and penalties tuned based on application-specific expert knowledge. Nevertheless, this solution becomes impractical for large-scale systems, unknown operating conditions, and/or in online settings where expert input would be needed during the system operation. Instead, this work enables agents to autonomously pose, tune, and solve optimal control problems by compromising between performance and specification costs. Leveraging duality theory, it puts forward a counterfactual optimization algorithm that directly determines the specification trade-off while solving the optimal control problem.

Keywords: Autonomous systems, optimal control, constrained optimization, feasibility, constraint learning.

1. INTRODUCTION

Autonomous systems are machines that can modify their behavior in response to unforeseen events and/or operating conditions. They are (or are set to become) key tools in robotics (Kober et al., 2013) and smart (grid, home, city) applications (Gharaibeh et al., 2017). One important aspect of autonomy that has attracted considerable attention in recent years is that of learning to perform tasks in uncertain or unknown environments. Here, the agent has limited (or no) knowledge of the underlying dynamical system and operational conditions and must design control laws to perform a pre-specified task based only on observations. System identification (Johansson, 1993), adaptive control (Kokotović, 1991), and reinforcement learning (Sutton and Barto, 2018), to name a few, have been used to address these issues.

Though learning *how* to perform tasks is paramount to achieve autonomous behavior, at least as important is the issue of learning *which* tasks can be performed in the first place. This decision-making aspect of autonomy is both fundamental and challenging, especially when agents must make decisions that *violate* their specifications. This is critical when multiple tasks and constraints are simultaneously required from the agent, resulting in infeasible settings. These situations arise due to over-specification, scenario uncertainty, or changing operating conditions, and are only aggravated when dynamical system models are learned through simulations.

When faced with the problem of dealing with (possibly conflicting) goals and requirements, agents are often designed using domain expert knowledge and/or prior information to weight and combine multiple objectives into a single cost, leading to so-called *regularized problems*. Although solutions obtained using this approach are often Pareto optimal (Ehrgott, 2005; Boyd and Vandenberghe, 2004), i.e., none is better than the others in every aspect, their performance in practice can vary widely. Thus the importance of properly tuning the regularization parameters, a task that is both application- and scenario-specific. This well-known issue has lead to several alternative formulations and heuristics (Miettinen, 1998; Das and Dennis, 1998; Messac et al., 2003; Mueller-Gritschneder et al., 2009). Alternatively, chance constraints can be used to relax the hard requirements by imposing a minimum probability of satisfying them, an approach often used in controlled Markov decision processes (Howard and Matheson, 1972; Geibel and Wysotzki, 2005; Paternain et al., 2019a,b), optimal control (Caillau et al., 2018; Ono et al., 2010), and model predictive control (Schwarm and Nikolaou, 1999; Li et al., 2000).

The main issue with these approaches is that they heavily rely on expert knowledge to inform the solution of multi-objective problems. Hence, adapting to the varying trade-offs of non-stationary environments would require additional input from experts during the system operation, renouncing on a key component of autonomy. What is more, designing regularization parameters or chance constraints that encode trade-off preferences is a challenging task in and of itself. Since properly encoding this information may be difficult, it may lead to solutions with poor practical performance despite their theoretical optimality.

This work provides a systematic approach to tune the agent specifications according to the inherent trade-offs of the underlying dynamical system and operating environment. This approach leverages counterfactuals, conditional statements that describe alternative versions of the world, to interrogate optimal control problems as to what would happen if the specifications had been different. This counterfactual evidence can then be used to automatically tune compromises and update requirements in a principled way without having to repeatedly solve different versions of the control problem. To do so, it first puts forward a mathematical formulation of compromise (Section 2) and shows that Lagrange multipliers can be used to counterfactually determine specifications that balance performance gains and costs (Section 3). This result leads to a low complexity saddle point algorithm that simultaneously solves the original optimization problem while tuning its compromises (Section 3.1). This method enables, for instance, an agent to autonomously pose control problems adapted to its operating conditions (Section 4).

2. PROBLEM FORMULATION

Let $f_0 : \mathbb{R}^n \to \mathbb{R}$ be a performance metric we wish to optimize while abiding by a set of requirements determined by the functions $f_i : \mathbb{R}^n \to \mathbb{R}$ and the specifications $s_i \geq 0$, $i = 1, \dots, m$. These requirements may represent tasks that the agent must perform, specify system limitations and/or available resources, or describe desired behaviors (see example in Section 4). Formally, we consider the optimization program

$$\begin{aligned} p^\star(\boldsymbol{s}) \triangleq \min_{\boldsymbol{z} \in \mathbb{R}^n} \quad & f_0(\boldsymbol{z}) \\ \text{s. to} \quad & f_i(\boldsymbol{z}) \leq s_i, \quad i = 1, \dots, m, \end{aligned} \tag{PI}$$

where the vector $\boldsymbol{s} \in \mathbb{R}_+^m$ collects the specifications s_i. Problem (PI) explicitly captures the trade-off between performance and requirements. Indeed, let $\boldsymbol{s} = \boldsymbol{0}$ denote a nominal, reference specification and denote its performance by $p^\star = p^\star(\boldsymbol{0})$. Then, if $s_i > 0$, the i-th requirement becomes easier to satisfy and the performance $p^\star(\boldsymbol{s})$ of the solution improves accordingly. Naturally, there is a cost associated with modifying requirements that must be taken into account when adjusting $\boldsymbol{s}$. To do so, let $h : \mathbb{R}_+^m \to \mathbb{R}_+$ be a non-decreasing function in each argument, so that $h(\boldsymbol{s})$ describes, in units of performance (f_0), the cost associated with specification $\boldsymbol{s}$. Without loss of generality, we assume the nominal specification to be free, i.e., $h(\boldsymbol{0}) = 0$.

Our goal is to tune $\boldsymbol{s}$ so as to trade off the specification cost $h(\boldsymbol{s})$ and the performance improvement $p^\star - p^\star(\boldsymbol{s})$, i.e., to find specifications that cost at most as much as they enhance performance. Formally, we seek $\boldsymbol{s}^\dagger$ such that

$$p^\star - p^\star(\boldsymbol{s}^\dagger) \geq h(\boldsymbol{s}^\dagger). \tag{1}$$

Though useful, (1) does not consider the case in which the nominal requirements are either conflicting or so stringent that (PI) is infeasible, i.e., there is no $\boldsymbol{z}' \in \mathbb{R}^n$ such that $f_i(\boldsymbol{z}') \leq 0$ for all i. In this case, we define $p^\star(\boldsymbol{0}) = +\infty$. Under these circumstances, we wish to fall back into a laxer notion of feasibility by choosing a different reference specification in (1). Since this choice is arbitrary, we require instead that the compromise $\boldsymbol{s}^\dagger$ holds for all valid references, i.e.,

$$p^\star(\boldsymbol{s}_0) - p^\star(\boldsymbol{s}^\dagger) \geq h(\boldsymbol{s}^\dagger) - h(\boldsymbol{s}_0), \tag{2}$$

for all $\boldsymbol{s}_0 \in \mathbb{R}_+^m$ such that (PI) is feasible, i.e., such that $p^\star(\boldsymbol{s}_0) < +\infty$. We call $\boldsymbol{s}^\dagger$ a compromise because it describes a specification of (PI) that trades off performance improvement and cost in a manner that is more profitable that any other feasible specification. Observe that the compromise in (2) takes into account not only the specification cost, but also how hard the requirement is to satisfy in the first place. To see this is the case, note that the left-hand side of (1) is a measure of constraint satisfaction difficulty for the nominal specification of (PI). Indeed, for a fixed specification budget $\delta > 0$, the value $\Delta_i = p^\star - p^\star(\delta \boldsymbol{e}_i)$, where $\boldsymbol{e}_i$ is the i-th vector of the canonical basis, quantifies the performance enhancement obtained by relaxing the i-th requirement.

It is not immediate from (PI) that the equilibrium (2) exists and, if it does, whether it can be determined efficiently. Indeed, note that (2) involves the optimal value of (PI). If finding $\boldsymbol{s}^\dagger$ involves repeatedly solving (PI), the computational costs would hinder the usefulness of this approach in practice, specially in online settings [e.g., for MPC (Borrelli et al., 2017; Rawlings et al., 2017)]. In the next section, we show that under mild conditions, the compromise $\boldsymbol{s}^\dagger$ in (2) exists and can be determined counterfactually, i.e., without repeatedly solving (PI). Based on these results, we then put forward a modified Arrow-Hurwicz algorithm that directly solves (PI) for $\boldsymbol{s}^\dagger$ without checking multiple specifications (Section 3.1).

Before proceeding, we introduce a pertinent remark addressing a common alternative approach for dealing with the trade-off of multiple objectives.

Remark 1 *A typical approach for dealing with multi-objective optimization problems, such as requirement-performance trade-offs, and infeasibility is to use regularization, wherein* (PI) *is replaced by* $\min_{\boldsymbol{z} \in \mathbb{R}^n} \; f_0(\boldsymbol{z}) + \sum_{i=1}^m \gamma_i f_i(\boldsymbol{z})$ *for regularization parameters* $\gamma_i \geq 0$ *(Miettinen, 1998; Ehrgott, 2005). However, whereas* (2) *explicitly states the trade-off of interest, this compromise is implicit in the regularized version through the relation among the* γ_i *and between the* γ_i *and the* *performance objective f_0. Since there is typically no straightforward way to tune γ_i, they are typically selected by domain experts by trial-and-error or computationally intensive grid searches (e.g., cross-validation). In contrast, Algorithm 1 can be used to efficiently determine $\boldsymbol{s}^\dagger$.*

3. LAGRANGE MULTIPLIERS AS COUNTERFACTUALS

In the end of Section 2, we argued that finding $\boldsymbol{s}^\dagger$ in (2) is only feasible in practice if it does not involve repeatedly solving (PI). Our goal in this section is to extract counterfactual evidence from (PI) to tune its requirements so as to achieve (2) without testing multiple specifications. A *counterfactual* is a conditional proposition in which the premise is false and the consequent describes how the world would have been *if the premise were true* (Pearl, 2009; Woodward, 2005; Lewis, 1974). It is immediate that (PI) yields counterfactuals of the form "if the requirements had been $\boldsymbol{s}$, then the optimal performance value would have been $p^\star(\boldsymbol{s})$." This causal relation is represented by the blue arrow in Fig. 1. Less straightforward is the fact that (PI) can also provide counterfactual evidence for the trade-off (2) (red arrows in Fig. 1). Next, we show that this evidence comes from the dual variables of (PI), which allows us to derive an algorithm that directly solves (PI) for $\boldsymbol{s}^\dagger$.

To proceed, start by defining the Lagrangian associated with (PI) as

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{s}) = f_0(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i f_i(\boldsymbol{x}) - \boldsymbol{\lambda}^T \boldsymbol{s}, \tag{3}$$

where $\boldsymbol{\lambda} \in \mathbb{R}_+^m$ collects the dual variables $\lambda_i \geq 0$ for $i = 1, \dots, m$. Likewise, define its dual function as $g(\boldsymbol{\lambda}, \boldsymbol{s}) = \min_{\boldsymbol{x} \in \mathbb{R}^n} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{s})$. The dual function is a lower bound on $p^\star(\boldsymbol{s})$ for all $\boldsymbol{\lambda}, \boldsymbol{s} \in \mathbb{R}_+^m$. The dual value is the best of these lower bounds, namely $d^\star(\boldsymbol{s}) \triangleq \max_{\boldsymbol{\lambda} \in \mathbb{R}_+^m} g(\boldsymbol{\lambda}, \boldsymbol{s})$. Under Assumptions 1 and 2 below, this best lower bound can be shown to attain $p^\star(\boldsymbol{s})$, i.e., $d^\star(\boldsymbol{s}) = p^\star(\boldsymbol{s})$. In duality theory, this is known as *strong duality* (Boyd and Vandenberghe, 2004, Ch. 5).

Assumption 1 *The f_i and h are differentiable and convex and f_0 is differentiable and strongly convex.*

Assumption 2 *The set $\left\{(\boldsymbol{x}', \boldsymbol{s}') \in \mathbb{R}^n \times \mathbb{R}_+^m \mid f_i(\boldsymbol{x}') < \boldsymbol{s}' \text{ for } i = 1, \dots, m\right\}$ is not empty.*

Under these conditions, it is well-known that the optimal dual variable $\lambda_i^\star(\boldsymbol{s})$ of (PI) identifies how hard the i-th constraint is to satisfy. Specifically, they locally quantify how much the objective would change if the constraint were tightened or relaxed. The following theorem provides an additional counterfactual property by showing that it also uniquely identifies the compromise (2).

Theorem 2 *Let $\boldsymbol{\lambda}^\star(\boldsymbol{s})$ be the dual variables of problem (PI) with slack $\boldsymbol{s}$ and $\boldsymbol{s}^\dagger$ be the compromise in (2). Under Assumptions 1 and 2,*

$$\nabla h(\boldsymbol{s}) = \boldsymbol{\lambda}^\star(\boldsymbol{s}) \Leftrightarrow \boldsymbol{s} = \boldsymbol{s}^\dagger. \tag{4}$$

Proof We start by proving necessity ($\Rightarrow$). Note that under Assumptions 1 and 2, h and $p^\star$ are convex, differentiable functions. Indeed, $p^\star$ is the *perturbation function* of (PI), a strongly dual, convex program with a strongly convex objective (Shapiro, 2000; Bertsekas, 2009). Hence, for all $\boldsymbol{s}_0, \boldsymbol{s} \in \mathbb{R}_+^m$ such that $p^\star(\boldsymbol{s}_0) < +\infty$ and $p^\star(\boldsymbol{s}) < +\infty$ it holds that

$$p^\star(\boldsymbol{s}_0) \geq p^\star(\boldsymbol{s}) + \nabla p^\star(\boldsymbol{s})^T(\boldsymbol{s}_0 - \boldsymbol{s}) \quad \text{and} \quad h(\boldsymbol{s}_0) \geq h(\boldsymbol{s}) + \nabla h(\boldsymbol{s})^T(\boldsymbol{s}_0 - \boldsymbol{s}), \tag{5}$$

which add up to

$$p^\star(\boldsymbol{s}_0) - p^\star(\boldsymbol{s}) \geq h(\boldsymbol{s}) - h(\boldsymbol{s}_0) + \left[\nabla p^\star(\boldsymbol{s}) + \nabla h(\boldsymbol{s})\right]^T(\boldsymbol{s}_0 - \boldsymbol{s}). \tag{6}$$

Using the fact that $\boldsymbol{\lambda}^\star(\boldsymbol{s}) = -\nabla p^\star(\boldsymbol{s})$ for all $\boldsymbol{s} \in \mathbb{R}_+^m$ so that (PI) is feasible (Boyd and Vandenberghe, 2004, Section 5.6.3), the hypothesis from (4) yields $\nabla p^\star(\boldsymbol{s}) + \nabla h(\boldsymbol{s}) = \mathbf{0}$. Hence, (6) imply (2) and we obtain that $\boldsymbol{s} = \boldsymbol{s}^\dagger$.

Sufficiency in (4) ($\Leftarrow$) stems from the fact that (2) can be rearranged into $p^\star(\boldsymbol{s}^\dagger) + h(\boldsymbol{s}^\dagger) \leq p^\star(\boldsymbol{s}_0) + h(\boldsymbol{s}_0)$. Hence, $\boldsymbol{s}^\dagger$ must be the unique minimizer of the strongly convex function $q(\boldsymbol{s}) = p^\star(\boldsymbol{s}) + h(\boldsymbol{s})$, so that $\nabla p^\star(\boldsymbol{s}^\dagger) + \nabla h(\boldsymbol{s}^\dagger) = \mathbf{0}$. Once again using the fact that $\boldsymbol{\lambda}^\star(\boldsymbol{s}) = -\nabla p^\star(\boldsymbol{s})$ for all $\boldsymbol{s}$ (Boyd and Vandenberghe, 2004, Section 5.6.3) concludes the proof. ■

Theorem 2 replaces the global compromise between $p^\star$ and h in (2) by the local relation between $\boldsymbol{\lambda}^\star(\boldsymbol{s})$ and $h(\boldsymbol{s})$ in (4) (red arrows in Fig. 1). Effectively, it states that the dual variables $\boldsymbol{\lambda}^\star(\boldsymbol{s})$ of (PI) can be used to determine $\boldsymbol{s}^\dagger$ without solving it for different specifications. Indeed, (4) provides counterfactual of the form "if $\nabla h(\boldsymbol{s})$ had been a dual variable of (PI), then $p^\star(\boldsymbol{s})$ and $h(\boldsymbol{s})$ would have obeyed (2)." For Pearl (2009), $\nabla h(\boldsymbol{s}) = \boldsymbol{\lambda}^\star(\boldsymbol{s})$ is the "surgical intervention" used to modify the causal model in Fig. 1 to evaluate this counterfactual. We could then state that $\nabla h(\boldsymbol{s}) = \boldsymbol{\lambda}^\star(\boldsymbol{s})$ causes (2). Theorem 2 also provides a backtracking counterfactual of the form "if $p^\star(\boldsymbol{s})$ and $h(\boldsymbol{s})$ were to obey (2), then $\nabla h(\boldsymbol{s}) = \boldsymbol{\lambda}^\star(\boldsymbol{s})$." By enforcing (4) as we solve (PI), we therefore simultaneously obtain $\boldsymbol{s}^\dagger$ and $p^\star(\boldsymbol{s}^\dagger)$, i.e., solve (PI) for the compromise (2). In the sequel, we derive a method to do so based on a modified Arrow-Hurwicz algorithm (Arrow et al., 1958).

3.1. A modified Arrow-Hurwicz algorithm

Theorem 2 suggests a way to exploit the counterfactual information in the dual variables $\boldsymbol{\lambda}$ to directly obtain a solution of (PI) for the optimal slack $\boldsymbol{s}^\star$ without ever solving (PI). Indeed, observe that due to Assumptions 1 and 2, it holds that the primal-dual solution $(\boldsymbol{x}^\star(\boldsymbol{s}), \boldsymbol{\lambda}^\star(\boldsymbol{s}))$ is a saddle point of the Lagrangian (3) (Boyd and Vandenberghe, 2004, Section 5.4.2), i.e.,

$$\mathcal{L}(\boldsymbol{x}^\star(\boldsymbol{s}), \boldsymbol{\lambda}, \boldsymbol{s}) \leq \mathcal{L}(\boldsymbol{x}^\star(\boldsymbol{s}), \boldsymbol{\lambda}^\star(\boldsymbol{s}), \boldsymbol{s}) \leq \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}^\star(\boldsymbol{s}), \boldsymbol{s}) \tag{7}$$

for all $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{\lambda} \in \mathbb{R}_+^m$, and $\boldsymbol{s}$ such that Assumption 2 holds. In the sequel, we put forward a procedure to find points that satisfy both (4) and (7).

Start by considering the classic Arrow-Hurwicz algorithm for solving (PI) when the slacks $\boldsymbol{s}$ are constant (Arrow et al., 1958). This method seeks a saddle point as in (7) by updating the primal and dual variables using gradients of the Lagrangian (3). Explicitly, the primal variables $\boldsymbol{x}$ are updated by descending along the negative gradient of the Lagrangian

$$\dot{\boldsymbol{x}} = -\nabla_{\boldsymbol{x}}\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{s}) = -\left(\nabla f_0(\boldsymbol{x}) + \sum_{i=1}^m \lambda_i \nabla f_i(\boldsymbol{x})\right), \tag{8}$$

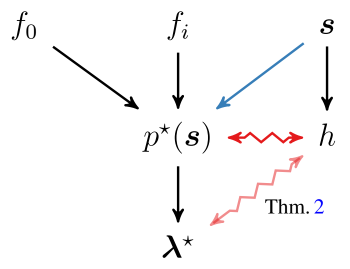
and the dual variables $\boldsymbol{\lambda}$ are updated by ascending along the gradient of the Lagrangian as in

$$\dot{\boldsymbol{\lambda}} = \Pi_{\mathbb{R}_+^m}\left[\boldsymbol{\lambda}, \nabla_{\boldsymbol{\lambda}}\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{s})\right] = \Pi_{\mathbb{R}_+^m}\left[\boldsymbol{\lambda}, f_i(\boldsymbol{x}) - s_i\right], \tag{9}$$

where $\Pi_{\mathbb{R}_+^m}$ refers to a projected dynamical system over the positive orthant of $\mathbb{R}^m$ (Nagurney and Zhang, 1996). This projection is introduced to ensure that the Lagrange multipliers are non-negative.

```
Let $\boldsymbol{x}^{(0)} = \mathbf{0}$, $\boldsymbol{\lambda}^{(0)} = \mathbb{1}$, $\boldsymbol{s}^{(0)} = \mathbb{1}$, and $0 < \eta \ll 1$.
for $t = 1, 2, \dots$ do
    $\boldsymbol{g}_x^{(t)} = \nabla f_0\left(\boldsymbol{x}^{(t-1)}\right) + \sum_{i=1}^{m} \lambda_i^{(t-1)} \nabla f_i\left(\boldsymbol{x}^{(t-1)}\right)$
    $\boldsymbol{x}^{(t)} = \boldsymbol{x}^{(t-1)} - \eta \boldsymbol{g}_x^{(t)}$
    $\lambda_i^{(t)} = \left[\lambda_i^{(t-1)} + \eta\left(f_i\left(\boldsymbol{x}^{(t-1)}\right) - s_i^{(t-1)}\right)\right]_+$
end
```

Algorithm 1: Counterfactual optimization algorithm



$$\begin{tikzcd}
f_0 \arrow[dr] & f_i \arrow[d] & \boldsymbol{s} \arrow[dl] \arrow[d] \\
& p^\star(\boldsymbol{s}) \arrow[r, leftrightsquigarrow] \arrow[d] & h \\
& \boldsymbol{\lambda}^\star \arrow[ur, leftrightsquigarrow, "\text{Thm. } 2"'] &
\end{tikzcd}$$

Figure 1: Causal diagram

To understand the intuition behind this algorithm, observe that the primal variable is updated in (8) by descending along a weighted combination of gradients from the objective and the constraints so as to reduce the value of all functions. The value of the weight of each constraint is given by its respective dual variable λ_i. If constraint i is satisfied, its Lagrangian multiplier is zero, so that its influence on the primal update is decreased by (9). On the other hand, if this constraint is violated, then $f_i(\boldsymbol{x}) - s_i > 0$ and the value of the corresponding multiplier is increased. The relative strength of each gradient in the primal update (8) is therefore related to the history of violation of each constraint.

The main drawback of the Arrow-Hurwicz dynamics as they stand is that they solve (PI) for a fixed slack $\boldsymbol{s}$. However, the compromise $\boldsymbol{s}^\dagger$ in (2) is unknown *a priori*. To overcome this limitation, we use (4) to replace (9) by

$$\dot{\boldsymbol{\lambda}} = \Pi_{\mathbb{R}_+^m}\left[\boldsymbol{\lambda}, f_i(\boldsymbol{x}) - (\nabla h)^{-1}(\boldsymbol{\lambda})\right]. \tag{10}$$

The inverse of ∇h exists since h is strongly convex (Assumption 1). Note that (10) takes $\boldsymbol{s} = (\nabla h)^{-1}(\boldsymbol{\lambda})$, i.e., it enforces that the specifications of (PI) satisfy (4). Hence, (8)–(10) update the primal/dual variables and specifications $\boldsymbol{s}$ such as to solve (PI) directly for $\boldsymbol{s}^\dagger$ in (2). A discretized version of the counterfactual optimization method is shown in Algorithm 1.

The dynamics (8)–(10) can be shown to converge to a point that satisfies the saddle point relation (7) as well as the left-hand side of (4) using an argument similar to (Cherukuri et al., 2016). From Theorem 2, they therefore simultaneously obtain the specification $\boldsymbol{s}^\dagger$ that satisfies (2) and the solution $\boldsymbol{z}^\star(\boldsymbol{s}^\star)$ of (PI) that achieve $p^\star(\boldsymbol{s}^\dagger)$. Due to space constraints, details of this proof are left for a future version of this manuscript.

4. SPECIFYING CONTROLLERS FOR UNKNOWN DYNAMICAL SYSTEMS

Control systems must often trade off control input cost (or energy) and regulation (Anderson and Moore, 2007; Bertsekas, 2017). The compromise between these objectives depends on the underlying dynamical system, operating conditions, and goals of the agent. What is more, this compromise may need to be revised to accommodate changes in the operating conditions, due to non-stationary environment and model uncertainty. Typically, this trade off is adjusted using domain expert knowledge of the problem. Autonomous systems, however, must be able to automatically tune these specifications without human intervention, a feature especially critical in online applications [e.g., MPC (Borrelli et al., 2017; Rawlings et al., 2017)].

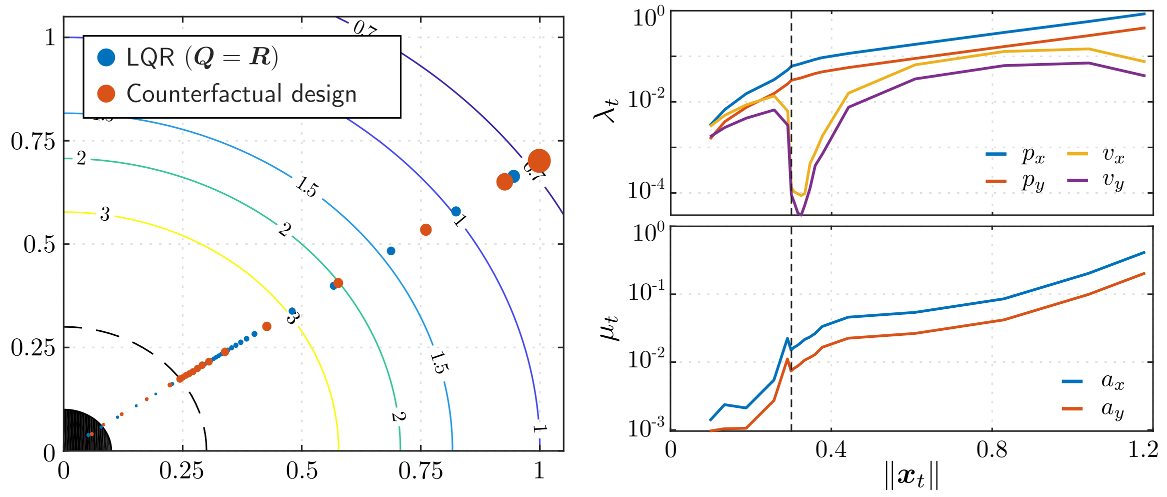


Figure 2: (a) Trajectory of agents in an environment with position-dependent friction (markers size are proportional to the input energy used at each step): LQR (PIII) (blue curve) and counterfactual controller (PII) (yellow curve). (b) Counterfactual specifications of (PII).

This problem can be tackled using counterfactual optimization by formulating it in the language of compromise from Section 2. To do so, we note that the ideal, yet infeasible, controller specification is one that regulates the system in one step without acting on it. Explicitly, it is the nominal specification of

$$\begin{aligned} \text{find} \quad & \boldsymbol{u}_1, \dots, \boldsymbol{u}_T \\ \text{such that} \quad & ([\boldsymbol{x}_t]_i)^2 \leq s_{x,i}, && i = 1, \dots, \ell \\ & ([\boldsymbol{u}_t]_j)^2 \leq s_{u,j}, && j = 1, \dots, p \\ & \boldsymbol{x}_t = \boldsymbol{A}\boldsymbol{x}_{t-1} + \boldsymbol{B}\boldsymbol{u}_t, && t = 1, \dots, T \end{aligned} \tag{PII}$$

where $\boldsymbol{x}_t \in \mathbb{R}^\ell$ is the state vector, $\boldsymbol{u}_t \in \mathbb{R}^p$ are the control actions, and g_t describes the dynamics of the system at time t. We write $[\boldsymbol{x}]_i$ to denote the i-th entry of the vector $\boldsymbol{x}$. The initial state $\boldsymbol{x}_0 \in \mathbb{R}^n$ is assumed to be given. When g_t describes linear dynamics, this problem is convex and can be written as in (PI) by taking $f_0(\boldsymbol{x}) = 0$. Notice that the nominal specification of (PII) for which $\boldsymbol{s}_x = \boldsymbol{s}_u = \boldsymbol{0}$ is such that $\|\boldsymbol{x}_t\| = \|\boldsymbol{u}_t\| = 0$. Thus, unless $\boldsymbol{x}_0 = \boldsymbol{0}$, (PII) is infeasible for these specifications. Algorithm 1 can then be used to tune $(\boldsymbol{s}_x, \boldsymbol{s}_u)$ to trade-off regulation and input energy specifications. In the sequel, we illustrate this procedure in the context of navigating an unknown terrain.

Consider an agent navigating an unknown terrain, modeled as a position-dependent friction coefficient. The states of the underlying dynamical system are $\boldsymbol{x} = [\ \boldsymbol{p}^T \quad \boldsymbol{v}^T\ ]^T$, describing the position $\boldsymbol{p} = [\ p_x \quad p_y\ ]^T$ and velocities $\boldsymbol{v} = [\ v_x \quad v_y\ ]^T$ of the agent, and its control inputs are accelerations collected in $\boldsymbol{u} = [\ a_x \quad a_y\ ]^T$. Its state-space representation is $\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}$ for

$$\boldsymbol{A} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -\gamma(\boldsymbol{p}) & 0 \\ 0 & 0 & 0 & -\gamma(\boldsymbol{p}) \end{bmatrix}, \boldsymbol{B} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, \text{ and } \gamma(\boldsymbol{p}) = \begin{cases} \|\boldsymbol{p}\|^{-2}, & \|\boldsymbol{p}\| > 0.3 \\ 0, & \|\boldsymbol{p}\| \leq 0.3 \end{cases}, \tag{11}$$

where $\gamma(\boldsymbol{p})$ is the gyroscopic friction coefficient at coordinates $\boldsymbol{p}$. In other words, the terrain becomes harder to navigate as the agent approaches the origin (the goal), until we reach a slippery region in which there is no friction. We assume that γ can be measured by the agent at its current position only, i.e., no prior information is available about its value in the environment. In the sequel, we consider a discretized version of (11) with sampling time $T_s = 0.5$ s.

To deal with changes in the environment, an MPC controller is used by planning for a horizon $T = 3$ iterations, but applying only the first action. Since the agent only has access to local information on the environment, the actions are planned assuming that the state transition matrix $\boldsymbol{A}$ is constant and that the gyroscopic friction is that of the current position. Hence, (PII) is a strongly convex program. For comparison, Figure 2a (blue curve) displays the trajectory obtained for the classical LQR

$$\begin{aligned} &\text{minimize} && \sum_{t=1}^{T} \left(\boldsymbol{x}_t^T \boldsymbol{Q}_t \boldsymbol{x}_t + \boldsymbol{u}_t^T \boldsymbol{R}_t \boldsymbol{u}_t \right) \\ &\text{subject to} && \boldsymbol{x}_t = \boldsymbol{A}\boldsymbol{x}_{t-1} + \boldsymbol{B}\boldsymbol{u}_t, \quad t = 1, \dots, T, \end{aligned} \tag{PIII}$$

with $\boldsymbol{Q}_t = \boldsymbol{I}$ and $\boldsymbol{R}_t = \boldsymbol{I}$. Notice that as the agent approaches the origin while outside the slippery region, it begins to move slowly, i.e., taking small steps. This occurs because the relative importance between regulation and input energy is set independently of the local friction coefficient in (PIII). The agent is therefore unwilling to spend the extra energy needed to overcome the high friction region faster. Thus, it takes 53 iterations (approximately 26 s) to reach $\|\boldsymbol{x}_t\| \leq 0.1$. Though in certain applications this is the desired behavior, i.e., there is no room for trading off input energy and regulation, it is clear that agents operating in dynamic conditions can benefit from being allowed to autonomously tune their specifications. Indeed, the trajectory obtained by counterfactually solving (PII) using $h(\boldsymbol{s}) = \|\boldsymbol{s}\|^2$ (Figure 2a, yellow curve) attains $\|\boldsymbol{x}_t\| \leq 0.1$ in 19 iterations spending $\sum_t \|\boldsymbol{u}_t\|^2 = 1.5$. By modifying the relative costs in (PIII) using $\boldsymbol{Q}_t = \gamma \boldsymbol{I}$, the same completion time can be achieved using $\sum_t \|\boldsymbol{u}_t\|^2 = 1.81$ ($\gamma = 3.3$), showing the price of using a fixed trade-off in a dynamic scenario.

Interestingly, it turns out that $\boldsymbol{Q}_t$ and $\boldsymbol{R}_t$ can be tuned so as to achieve the same performance as the controller obtained from (PII) due to the fact that (PII) is related to (PIII) by Lagrangian duality. Indeed, the Lagrangian of (PII) is given by

$$\begin{aligned} \mathcal{L}(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\lambda}_t, \boldsymbol{\mu}_t) &= \sum_{t=1}^{T} \left[\sum_{i=1}^{\ell} \lambda_{t,i} \left(([\boldsymbol{x}_t]_i)^2 - s_{x,i} \right) + \sum_{j=1}^{p} \mu_{t,j} \left(([\boldsymbol{u}_t]_j)^2 - s_{u,j} \right) \right] \\ &= \sum_{t=1}^{T} \left(\boldsymbol{x}_t^T \boldsymbol{\Lambda}_t \boldsymbol{x}_t + \boldsymbol{u}_t^T \boldsymbol{M}_t \boldsymbol{u}_t^T - \boldsymbol{\lambda}_t^T \boldsymbol{s}_x - \boldsymbol{\mu}_t^T \boldsymbol{s}_u \right), \end{aligned} \tag{12}$$

where $\boldsymbol{\Lambda}_t = \operatorname{diag}(\lambda_{t,i})$ and $\boldsymbol{M}_t = \operatorname{diag}(\mu_{t,j})$. It is straightforward from (12) that if $(\boldsymbol{x}_t^\star, \boldsymbol{u}_t^\star)$ is a solution of (PIII) with $\boldsymbol{Q}_t = \boldsymbol{\Lambda}_t$ and $\boldsymbol{R}_t = \boldsymbol{M}_t$, then $(\boldsymbol{x}_t^\star, \boldsymbol{u}_t^\star) = \operatorname{argmin}_{(\boldsymbol{x}_t, \boldsymbol{u}_t)} \mathcal{L}(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\lambda}_t, \boldsymbol{\mu}_t)$ for the Lagrangian in (12). Using an appropriate sequence of $\boldsymbol{Q}_t$ and $\boldsymbol{R}_t$ in (PIII) therefore yields the same sequence of control actions as counterfactually solving (PII). Nevertheless, even with complete knowledge of the environment, manually selecting this sequence is a challenging task as the weights do not have a straightforward behavior (Figure 2b).

References

B.D.O. Anderson and J.B. Moore. *Optimal Control: Linear Quadratic Methods*. Dover, 2007.

K.J. Arrow, L. Hurwicz, and H. Uzawa. *Studies in linear and non-linear programming*. Stanford University Press, 1958.

D.P. Bertsekas. *Convex Optimization Theory*. Athena Scientific, 2009.

D.P. Bertsekas. *Dynamic Programming and Optimal Control – Vol. I*. Athena Scientific, 2017.

F. Borrelli, A. Bemporad, and M. Morari. *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press, 2017.

S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

J-B Caillau, Max Cerf, Achille Sassi, Emmanuel Trélat, and Hasnaa Zidani. Solving chance constrained optimal control problems in aerospace via kernel density estimation. *Optimal Control Applications and Methods*, 39(5):1833–1858, 2018.

A. Cherukuri, E. Mallada, and J. Cortés. Asymptotic convergence of constrained primal–dual dynamics. *Systems & Control Letters*, 87:10–15, 2016.

I. Das and J.E. Dennis. Normal-boundary intersection: A new method for generating the Pareto surface in nonlinear multicriteria optimization problems. *SIAM Journal on Optimization*, 8[3]: 631–657, 1998.

M. Ehrgott. *Multicriteria Optimization*. Springer, 2005.

Peter Geibel and Fritz Wysotzki. Risk-sensitive reinforcement learning applied to control under constraints. *Journal of Artificial Intelligence Research*, 24:81–108, 2005.

Ammar Gharaibeh, Mohammad A Salahuddin, Sayed Jahed Hussini, Abdallah Khreishah, Issa Khalil, Mohsen Guizani, and Ala Al-Fuqaha. Smart cities: A survey on data management, security, and enabling technologies. *IEEE Communications Surveys & Tutorials*, 19(4):2456–2501, 2017.

Ronald A Howard and James E Matheson. Risk-sensitive Markov decision processes. *Management Science*, 18(7):356–369, 1972.

Rolf Johansson. *System modeling and identification*, volume 1. Prentice Hall Englewood Cliffs, NJ, 1993.

Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

Petar V Kokotović. *Foundations of adaptive control*. Springer, 1991.

D. Lewis. Causation. *The Journal of Philosophy*, 70(17):556–567, 1974.

P. Li, M. Wendt, and G. Wozny. Robust model predictive control under chance constraints. *Computers & Chemical Engineering*, 24[2-7]:829–834, 2000.

A. Messac, A. Ismail-Yahaya, and C.A. Mattson. The normalized normal constraint method for generating the Pareto frontier. *Structural and Multidisciplinary Optimization*, 25[2]:86–98, 2003.

K. Miettinen. *Nonlinear Multiobjective Optimization*. Springer, 1998.

D. Mueller-Gritschneder, H. Graeb, and U. Schlichtmann. A successive approach to compute the bounded Pareto front of practical multiobjective optimization problems. *SIAM Journal on Optimization*, 20[2]:915–934, 2009.

A. Nagurney and D. Zhang. *Projected Dynamical Systems and Variational Inequalities with Applications*. Springer, 1996.

Masahiro Ono, Lars Blackmore, and Brian C Williams. Chance constrained finite horizon optimal control with nonconvex constraints. In *Proceedings of the 2010 American Control Conference*, pages 1145–1152. IEEE, 2010.

Santiago Paternain, Miguel Calvo-Fullana, Luiz FO Chamon, and Alejandro Ribeiro. Safe policies for reinforcement learning via primal-dual methods. *arXiv preprint arXiv:1911.09101*, 2019a.

Santiago Paternain, Luiz F.O. Chamon, Miguel Calvo-Fullana, and Alejandro Ribeiro. Constrained reinforcement learning has zero duality gap. In *Advances in Neural Information Processing Systems*, pages 7553–7563, 2019b.

J. Pearl. *Causality: Models, Reasoning and Inference*. Cambridge University Press, 2009.

J.B. Rawlings, D.Q. Mayne, and M.M. Diehl. *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, 2017.

A.T. Schwarm and M. Nikolaou. Chance-constrained model predictive control. *AIChE Journal*, 45[8]:1743–1752, 1999.

A. Shapiro. Duality, optimality conditions, and perturbation analysis. In H. Wolkowicz, R. Saigal, and L. Vandenberghe, editors, *Handbook of Semidefinite Programming: Theory, Algorithms, and Applications*, pages 67–91. Springer, 2000.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

J. Woodward. *Making things happen: A theory of causal explanation*. Oxford University Press, 2005.