2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

Complexity-Bounded Relaxed Dynamic Programming

R.M. Beumer, M.J.G. van de Molengraft, D.J. Antunes

Abstract— **The idea behind relaxed dynamic programming for optimal control problems is to settle with a suboptimal but simpler control policy that guarantees a cost within a fixed constant factor from the optimal cost. Such a policy results from parameterized approximate value functions and the complexity of these functions determines the complexity of the policy. Typically, the more stringent the constant factor from the optimal cost is, the larger the complexity. However, relaxed dynamic programming does not give any guarantees on the complexity, which might still be unpractical. To tackle this issue, we propose to rather find the best factor away from optimality for a given complexity bound. We consider a large class of problems where the value functions can be represented as the minimum of quadratic functions. For this class, we propose a modified relaxed dynamic programming algorithm that ensures bounded complexity while still providing tight cost guarantees. A crucial step in the algorithm is the search for the best cost factor for a given policy with desired complexity, shown to be an optimization problem subject to Linear Matrix Inequalities (LMIs). We provide a new subclass of problems within this class and illustrate the effectiveness of our policy in a numerical instance of this subclass.**

I. Introduction

Dynamic programming (DP) is an iterative procedure to solve optimal control problems that finds optimal policies by first computing optimal value functions. It has been successfully applied to various problems in several areas. However, the curse of dimensionality limits its applicability. In many problems, the optimal value functions can be parameterized at each algorithm iteration, bypassing the curse of dimensionality. Still, the complexity of the optimal value functions, measured by the required number of parameters, often grows fast at each algorithm iteration. Thus, even then, DP is not practical.

For these problems, one can relax the demand for optimality to reduce the complexity of the value functions. This is the idea behind relaxed dynamic programming (RDP) [1]. RDP computes, at each iteration, approximate value functions within a constant factor from the optimal value functions, expectedly (but not certainly) resulting in a smaller complexity. Based on the approximate value functions, a suboptimal control policy is then provided that guarantees a cost within ϵ from the optimal one.[1]This constant factor ϵ can be a relative and multiplicative guarantee with respect to the optimal cost J, bounding the approximate cost by $(1+\epsilon)J$, or absolute and additive, guaranteeing rather a bound $J+\epsilon$. RDP and approaches inspired by it have been successfully applied in several contexts [2]–[5].

However, RDP provides no guarantees on the complexity of the approximate value functions, which determines the complexity of the policy. Typically, the less stringent ϵ is, the smaller the complexity, but even this might not be the case; see the example in Section IV-A. The absence of complexity guarantees leads to some undesirable features. First, for a given desired ϵ, the complexity can still grow fast at each algorithm iteration, in which case the method is unpractical. Second, tuning the complexity by changing ϵ is hard. The complexity might not only be a non-monotone function of ϵ, but it might also have large discontinuities. Third, a guaranteed bound on complexity is desirable and even more important than a stringent cost guarantee in many applications, such as real-time applications with low computational budgets. Moreover, the bound on the cost determined by ϵ is often conservative and can be improved for the final policy retrieved by RDP.

To tackle this challenge, we propose to rather find the best ϵ for a given complexity bound for the value functions. We focus on problems where the value functions can be represented as the minimum of a finite set of quadratic functions that are used as a parameterization. This provides a unifying framework for several problems in different areas, including (i) switched linear systems with quadratic cost [1], [6], (ii) partially observable Markov decision processes (POMDPs) [1], (iii) optimal control problems for LTI systems with piecewise affine cost [1], (iv) optimal joint maximum a posteriori probability (JMAP) estimate of the state and mode of a Markov jump linear system (MJLS) [7], and (v) linear quadratic control problems with discretized input, a novel subclass of problems proposed in the present paper. For problems of this class, we propose a modified RDP algorithm that ensures bounded complexity while still providing cost guarantees in the same form (additive and multiplicative) as RDP. The algorithm aims to find the smallest ϵ subject to complexity guarantees. To this end, at each iteration, the algorithm relies crucially on optimizing ϵ for a given policy with desired complexity. We show this subproblem can be solved through an optimization problem subject to Linear Matrix Inequalities (LMIs).

The paper is organized as follows. Section II provides background on RDP and introduces the class of problems

The authors are with the Control Systems Technology Group, Department of Mechanical Engineering, Eindhoven University of Technology, the Netherlands. E-mails:{r.m.beumer, m.j.g.v.d.molengraft,d.antunes}@tue.nl. This research is part of the research program SYNERGIA (project number 17626), which is partly financed by the Dutch Research Council (NWO).

[1]The RDP framework proposed in [1] is very broad. Here we take some liberty to narrow it down to a special case also considered in [1] where the lower bounds on the value function approximations coincide with the value function and the upper bounds are obtained by multiplying the running cost by $\alpha = 1+\epsilon$.

considered. Section III discusses the proposed RDP procedure with complexity guarantees. Section IV applies this procedure to a numerical instance of the novel subclass of problems. Concluding remarks are provided in Section V. The proofs of the results are omitted for the sake of brevity.

II. Background and Problem Formulation

Consider a standard deterministic optimal control problem

$$\min_{\{u_k|k\in\{0,1,\dots,h-1\}\}} \sum_{k=0}^{h-1} g_k(x_k,u_k)+g_h(x_h)$$

for the system

$$x_{k+1}=f(x_k,u_k),\quad k\in\{0,1,\dots,h-1\},$$

where $x_k\in X\subseteq\mathbb{R}^n$ and $u_k\in U$ are the state and control input at time k, respectively; U can be countable or uncountable. The DP algorithm obtains iteratively, for $k\in\{h-1,h-2,\dots,0\}$, the optimal value functions

$$J_k(x_k)=\min_{u_k\in U} g_k(x_k,u_k)+J_{k+1}(f(x_k,u_k)),$$

for all $x_k\in X$ with $J_h(x_h)=g_h(x_h)$, for all $x_h\in X$, from which an optimal policy can be obtained, for all $x_k\in X$ and all $k\in\{0,\dots,h-1\}$:

$$\mu_k(x_k)\in\operatorname*{argmin}_{u_k\in U} g_k(x_k,u_k)+J_{k+1}(f(x_k,u_k)). \tag{1}$$

For many instances of this general problem we can parameterize the optimal value functions as a pointwise minimum (or maximum) of a finite set of functions, that is,

$$J_k(x)=\min_{\beta\in\mathcal{J}_k} p(x,\beta),\quad x\in X,\ k\in\{0,\dots,h\} \tag{2}$$

(or $J_k(x)=\max_{\beta\in\mathcal{J}_k} p(x,\beta)$) where β is a set of parameters that belongs to a finite set $\mathcal{J}_k$. This is the case for several applications, including but not limited to:

1) Switched linear systems $x_{k+1}=A_{u_k}x_k$, $x_k\in\mathbb{R}^n$, $u_k\in\{1,2,\dots,m\}$, with positive semi-definite quadratic state cost $g_k(x_k,u_k)=x_k^\top Q_{u_k}x_k$, $g_h(x_h)=x_h^\top Q_h x_h$, see [1], [6]. In this case $p(x,P)=x^\top Px$ with $P\in\mathcal{J}_k$, for some finite set of positive semi-definite matrices $\mathcal{J}_k$.

2) Linear time-invariant systems $x_{k+1}=Ax_k+Bu_k$, $x_k\in\mathbb{R}^n$, $u_k\in\mathbb{R}^m$, with piecewise affine costs $g_k(x,u)=\max_{c\in\mathcal{C}} c^\top\begin{bmatrix}x^\top & u^\top & 1\end{bmatrix}^\top$, $g_h(x_h)=\max_{c\in\mathcal{C}_h} c^\top\begin{bmatrix}x_h^\top & 1\end{bmatrix}^\top$, see [1]. In this case, $J_k(x)=\max_{\beta\in\mathcal{J}_k} p(x,\beta)$, where $p(x,\beta)=\beta^\top\begin{bmatrix}x^\top & 1\end{bmatrix}^\top$ with $\beta\in\mathcal{J}_k$, for a finite set $\mathcal{J}_k$.

3) POMDPs with finite state, control and observation spaces. While this problem does not fit the problem description above, it is well-known that the optimal policy for the control input can be obtained by solving a standard DP problem where the value function takes the form (2) with $p(x,q)=q^\top x$, $q\in\mathcal{J}_k$, for a finite set $\mathcal{J}_k$, and where x_k is the probability distribution of a discrete-state at step k [1].

4) Optimal joint maximum a posteriori probability (JMAP) estimate of the state and mode of a Markov jump linear system (MJLS). Here the state corresponds to estimates of the state $\hat{x}$ and mode $\hat{\sigma}$ of an MJLS ($\hat{x}_k$ and $\hat{\sigma}_k$ at time k). The input, model and cost can be found in [7] and are omitted here. The value function takes the form (2) with $p(x,P,\hat{\xi},\gamma)=\frac{1}{2}\left(\hat{x}-\hat{\xi}\right)^\top P\left(\hat{x}-\hat{\xi}\right)+\gamma$, $x=\begin{bmatrix}\hat{x}^\top & \hat{\sigma}\end{bmatrix}^\top$, $(P,\hat{\xi},\gamma)\in\mathcal{J}_k$ for finite sets $\mathcal{J}_k$ and positive semi-definite P.

5) Linear quadratic control with a finite number of control input options. This is a new application. As shown in Section IV, here $p(x,P,q,r)=x^\top Px+q^\top x+r$ with $(P,q,r)\in\mathcal{J}_k$, for finite sets $\mathcal{J}_k$ and positive semi-definite P.

For all these applications except 2) the value functions can be written in the general form (where $\mathcal{J}_k$ is a set of tuples):

$$J_k(x)=\min_{(P,q,r)\in\mathcal{J}_k} x^\top Px+q^\top x+r. \tag{3}$$

Moreover, for application 2) the value function takes the same form but with the $\min$ operation replaced by the $\max$ operation. The results proposed here still apply to this case by exchanging these two operations. Our results can be applied to application 3) with proper adaptations along the lines of [3] since (3) holds, even though, as already mentioned, the framework is different. Hereafter we assume that $X=\mathbb{R}^n$ so that (3) is assumed to hold for every $x\in\mathbb{R}^n$.

A. Relaxed Dynamic Programming

The idea behind RDP proposed in [1] is to approximate the optimal value functions by approximate value functions

$$V_k(x)=\min_{\beta\in\mathcal{V}_k} p(x,\beta),\quad x\in\mathbb{R}^n,\ k\in\{0,\dots,h\}, \tag{4}$$

where $\mathcal{V}_k\subseteq\mathcal{J}_k$ is a pruned version of $\mathcal{J}_k$. For problems with non-negative value functions (the negative case is discussed in the sequel), the pruning procedure must be such that

$$J_k(x)\le V_k(x)\le(1+\epsilon)J_k(x),\quad x\in\mathbb{R}^n, \tag{5}$$

for a predefined $\epsilon\ge 0$. By doing so, the cost of an approximate policy obtained by replacing J_{k+1} by V_{k+1} in (1) is guaranteed to be within a factor $1+\epsilon$ of the optimal policy. In [1] infinite-horizon problems ($h\to\infty$) are also addressed, but here we focus on the finite-horizon case.

To describe the RDP algorithm we define the functions

$$V_k^\epsilon(x_k)=\min_{u_k\in U}(1+\epsilon)g_k(x_k,u_k)+V_{k+1}(f(x_k,u_k)), \tag{6}$$

for all $x_k\in\mathbb{R}^n$, $k\in\{0,\dots,h-1\}$ and $V_h(x_h)=J_h(x_h)$. This can still be written in the following form in all the mentioned applications:

$$V_k^\epsilon(x_k)=\min_{(P(\epsilon),q(\epsilon),r(\epsilon))\in\mathcal{V}_k^\epsilon} x_k^\top P(\epsilon)x_k+q(\epsilon)^\top x_k+r(\epsilon). \tag{7}$$

This implicitly defines the sets $\mathcal{V}_k^\epsilon$; see how these sets are constructed for application 5) in Section IV. In all the applications, each member $(P(\epsilon),q(\epsilon),r(\epsilon))\in\mathcal{V}_k^\epsilon$ is parameterized by ϵ in an affine manner. Therefore, we make the following assumption, which is crucial to obtain numerical efficient tests for the proposed method:

$$P(\epsilon)=\bar{P}_0+\bar{P}_1\epsilon,\quad q(\epsilon)=\bar{q}_0+\bar{q}_1\epsilon,\quad r(\epsilon)=\bar{r}_0+\bar{r}_1\epsilon. \tag{8}$$

Algorithm 1 Original RDP

Given: $\epsilon \geq 0$ and heuristic H.
Set $\mathcal{V}_h = \mathcal{J}_h$.
for $k = h-1 : -1 : 0$ **do**
  Set $\mathcal{V}_k = \emptyset$ and compute $\mathcal{V}_k^\epsilon$.
  while $\mathcal{V}_k^\epsilon \neq \emptyset$ **do**
    Remove $(\underline{P}(\epsilon), \underline{q}(\epsilon), \underline{r}(\epsilon)) \in \mathcal{V}_k^\epsilon$ with smallest H.
    if for all $x \in \mathbb{R}^n$

$$\min_{(P,q,r)\in\mathcal{V}_k} x^\top P x + q^\top x + r \leq x^\top \underline{P}(\epsilon) x + \underline{q}^\top(\epsilon) x + \underline{r}(\epsilon) \tag{11}$$

    then
      Continue $((\underline{P}(\epsilon), \underline{q}(\epsilon), \underline{r}(\epsilon))$ can be pruned).
    else
      Add $(\underline{P}(0), \underline{q}(0), \underline{r}(0))$ to $\mathcal{V}_k$.
    end if
  end while
end for

While $\mathcal{V}_k^\epsilon$ is not an ordered set, to each $(P(\epsilon), q(\epsilon), r(\epsilon)) \in \mathcal{V}_k^\epsilon$ there is an associated $(P(0), q(0), r(0)) = (\bar{P}_0, \bar{q}_0, \bar{r}_0) \in \mathcal{V}_k^0$, which will often be referred to in the sequel.

A general and recursive procedure for pruning $\mathcal{J}_k$ and obtaining $\mathcal{V}_k$ is described in Algorithm 1. It is a special case of the broad framework of RDP proposed in [1] that is tailored to the problems that admit an optimal value function taking the form (3). Only this special case is considered hereafter and referred to as RDP, although RDP as defined in [1] is broader. It relies on a heuristic H which assigns a real value to each tuple $(\underline{P}(\epsilon), \underline{q}(\epsilon), \underline{r}(\epsilon))$; H should be large when a tuple does not contribute to the minimum in (7).

An additive factor ϵ can be used by defining

$$V_k^\epsilon(x_k) = \min_{u_k \in U} g_k(x_k, u_k) + \epsilon + V_{k+1}(f(x_k, u_k)), \tag{9}$$

for all $x_k \in \mathbb{R}^n$, instead of (6), see [3]. In this case, (7) still defines $\mathcal{V}_k^\epsilon$ but $P(\epsilon)$ and $q(\epsilon)$ do not depend on ϵ; only $r(\epsilon)$ does again depend on ϵ in an affine manner. Additive factors are convenient when the value functions can be negative, see [3]. The cost guarantee is then different:

$$J_k(x) \leq V_k(x) \leq J_k(x) + (h-k)\epsilon, \quad x \in \mathbb{R}^n. \tag{10}$$

The results discussed next consider both cases.

B. Problem statement

Let $|\mathcal{A}|$ denote the cardinality (number of elements) of a discrete set $\mathcal{A}$ and note that $|\mathcal{V}_k|$ is a direct measure of the complexity of the approximate policy that results from replacing J_{k+1} by V_{k+1} in (1). Typically, in both cases of RDP considered in the previous section (with additive and multiplicative factor guarantees) the larger ϵ is, the smaller $|\mathcal{V}_k|$ is. However, while $|\mathcal{V}_k|$ is expected to reduce with respect to the non-pruned version, this procedure does not give any guarantee on the complexity, e.g., a complexity bound. The achieved complexity will depend on the specific problem at hand and can still become computationally intractable. This motivates the problem considered here:

Algorithm 2 Proposed RDP with complexity guarantees

Given: complexity bound $C \in \mathbb{N}$, heuristic H and selection procedure `selectM` with $1 \leq M \leq C$.
Set $\epsilon_h = 0$ and $\mathcal{V}_h = \mathcal{J}_h$.
for $k = h-1 : -1 : 0$ **do**
  Set $\mathcal{V}_k = \emptyset$ and compute $\mathcal{V}_k^{\epsilon_{k+1}}$.
  Remove M members of $\mathcal{V}_k^{\epsilon_{k+1}}$ and add the corresponding $(P(0), q(0), r(0))$ to $\mathcal{V}_k$ using `selectM`.
  while $\mathcal{V}_k^{\epsilon_{k+1}} \neq \emptyset$ and $|\mathcal{V}_k| < C$ **do**
    Remove $(\underline{P}(\epsilon_{k+1}), \underline{q}(\epsilon_{k+1}), \underline{r}(\epsilon_{k+1})) \in \mathcal{V}_k^{\epsilon_{k+1}}$ with smallest H.
    if there exists $\alpha \in \mathsf{P}_{|\mathcal{V}_k|}$ such that for all $x \in \mathbb{R}^n$

$$\sum_{(P,q,r)\in\mathcal{V}_k, \alpha\in\mathsf{P}_{|\mathcal{V}_k|}} \alpha_i (x^\top P x + q^\top x + r) \leq x^\top \underline{P}(\epsilon_{k+1}) x + \underline{q}^\top(\epsilon_{k+1}) x + \underline{r}(\epsilon_{k+1}) \tag{12}$$

    then
      Continue $((\underline{P}(\epsilon_{k+1}), \underline{q}(\epsilon_{k+1}), \underline{r}(\epsilon_{k+1}))$ can be pruned).
    else
      Add $(\underline{P}(0), \underline{q}(0), \underline{r}(0))$ to $\mathcal{V}_k$.
    end if
  end while
  if $\mathcal{V}_k^{\epsilon_{k+1}} \neq \emptyset$ **then**
    Compute the smallest ϵ_k^* such that all remaining $(P(\epsilon_{k+1}), q(\epsilon_{k+1}), r(\epsilon_{k+1})) \in \mathcal{V}_k^{\epsilon_{k+1}}$ satisfy (12) (can be pruned) if $\epsilon_{k+1} \geq \epsilon_k^*$.
    Set $\epsilon_k = \epsilon_k^*$. (13a)
  else
    Set $\epsilon_k = \epsilon_{k+1}$. (13b)
  end if
end for

Given a desired bound $C \in \mathbb{N}$ on the complexity of the approximate value functions, i.e.,

$$|\mathcal{V}_k| \leq C, \text{ for all } k \in \{0, 1, \ldots, h-1\},$$

provide a numerical method that still provides a tight bound on performance ϵ and ensures the complexity bound is met.

III. Relaxed Dynamic Programming with Complexity Guarantees

Algorithm 2 describes the proposed RDP method with complexity guarantees and multiplicative cost guarantees. It is explained and analyzed in Section III-A, where its cost guarantees are formally established. A crucial step of this RDP method is to determine the smallest ϵ guarantee for a policy resulting from a cost-to-go approximation with fixed complexity. We show that this can be obtained by an LMI optimization problem in Section III-B. Another important feature, which is shared with the original RDP, is the heuristic H. This is discussed in Section III-C.

A. RDP algorithm and cost guarantees

Let us first consider that the optimal value functions (10) are non-negative ($J_k(x_k) \geq 0$) and remove this restriction in

the sequel (see Theorem 2). Let $\mathsf{P}_n := \{\alpha = \begin{bmatrix}\alpha_1 \cdots \alpha_n\end{bmatrix}^\top \in \mathbb{R}^n | \sum_{i=1}^n \alpha_i = 1, \alpha_i \geq 0\}$.

Algorithm 2 starts (line 4) by adding M members from $\mathcal{V}_k^{\epsilon_{k+1}}$ to the pruned set $\mathcal{V}_k$, according to a selection procedure `selectM`. This procedure aims at finding tuples (P, q, r) that have strong chances of remaining in the final set $\mathcal{V}_k$. An example selection procedure is to pick M different minimizers of

$$\underset{(P,q,r)\in\mathcal{V}_k^{\epsilon_{k+1}}}{\operatorname{argmin}} \quad x^{i\top} P x^i + q^\top x^i + r \tag{14}$$

for M different state values x^i, $i \in \{1, \ldots, M\}$, e.g., randomly chosen. These tuples cannot be pruned according to (11), which corresponds to (12) in Algorithm 2, when $\epsilon = 0$. If M different x^i values cannot be found, the remaining tuples can themselves be picked randomly, although in such a case picking a smaller M is more suitable. Another selection procedure is to simply pick the tuples that correspond to the smallest M values of heuristic H. Section III-C discusses possible heuristics. Note that Algorithm 1 in fact uses a special case of this selection procedure with $M = 1$ and based on H. If a strong procedure to select M members is present for a given problem, then $M = C$ might make sense in which case the algorithm moves directly to the last step (line 14).

When $M < C$, the algorithm finds a non-decreasing sequence of ϵ_k that will lead to guarantees on the approximate value function $V_k(x)$, see Theorem 1 below. At time k, the algorithm runs the steps of a slightly modified version of RDP with ϵ_{k+1} (using (12) rather than (11), as explained in Section III-B), compare lines 5-12 of the proposed RDP algorithm with lines 4-11 of the original RDP algorithm. However, according to line 5 of the proposed RDP algorithm, these steps are only run until either budget C is reached (meaning that ϵ_{k+1} is stringent) or not (ϵ_{k+1} is not stringent as the complexity at time k is smaller than the bound C). In the former case, in the last step (line 14), $\epsilon_k^* \geq \epsilon_{k+1}$ is computed that guarantees that the yet unpruned members in $\mathcal{V}_k^{\epsilon_{k+1}}$ would be pruned if $\epsilon_{k+1} \geq \epsilon_k^*$ and ϵ_k is set to ϵ_k^*.

As will be discussed in Section III-B, we can test (12) through a feasibility test of LMIs and we can use LMIs to compute ϵ_k^* in the last step (line 14) of Algorithm 2.

We will now provide a cost guarantee similar to the original RDP but with ϵ replaced by ϵ_0.

Theorem 1: Suppose that the optimal value functions (10) are non-negative, that is, $J_k(x_k) \geq 0$. Let

$$V_k(x) = \min_{\beta\in\mathcal{V}_k} p(x, \beta), \quad x \in \mathbb{R}^n, \tag{15}$$

where $\mathcal{V}_k \subseteq \mathcal{J}_k$ is obtained with Algorithm 2 based on $\mathcal{V}_k^\epsilon$ computed according to (6), (7), which ensures that $|\mathcal{V}_k| \leq C$, for all $k \in \{0, 1, \ldots, h-1\}$. Then

$$J_k(x) \leq V_k(x) \leq (1+\epsilon_k) J_k(x), \forall x \in \mathbb{R}^n, k \in \{0, \ldots, h-1\}.$$

□

Note that since $\epsilon_k \geq \epsilon_{k+1}$, this theorem indeed implies

$$J_k(x) \leq V_k(x) \leq (1+\epsilon_0) J_k(x), \forall x \in \mathbb{R}^n, k \in \{0, \ldots, h-1\},$$

which is a similar guarantee to the one provided by RDP (5). However, instead of fixing ϵ a priori, a complexity bound C is fixed a priori and Algorithm 2 aims at finding the smallest ϵ that still guarantees (5) with this complexity bound.

Algorithm 2 applies with almost no changes to the additive case. There are two main differences. First, the sets $\mathcal{V}_k^\epsilon$ are computed based on (9) and (7) rather than on (6) and (7). Second, (13a) and (13b) are both replaced by $\epsilon_k = \underline{\epsilon}$, as well as the initial $\epsilon_h = \underline{\epsilon}$ for some constant $\underline{\epsilon} \geq 0$, which is an extra parameter of the algorithm. This follows from the fact that now the additive cost guarantee associated with ϵ_k at each step does not need to depend on the previous step. Moreover, in this case we do not need the non-negative assumption on the cost. Then, we also obtain similar guarantees to the ones of the standard RDP algorithm as stated next.

Theorem 2: Let

$$V_k(x) = \min_{\beta\in\mathcal{V}_k} p(x, \beta), \quad x \in \mathbb{R}^n, \tag{16}$$

where $\mathcal{V}_k \subseteq \mathcal{J}_k$ are obtained with Algorithm 2 based on $\mathcal{V}_k^\epsilon$ computed according to (9), (7), and with (13a) and (13b) replaced by $\epsilon_k = \underline{\epsilon}$ for a given $\underline{\epsilon} \geq 0$ and $\epsilon_h = \underline{\epsilon}$, which ensures that

$$|\mathcal{V}_k| \leq C, \text{ for all } k \in \{0, 1, \ldots, h-1\}.$$

For times k for which (13b) is run instead of (13a), let $\epsilon_\ell^* = \underline{\epsilon}$. Then

$$J_k(x) \leq V_k(x) \leq J_k(x) + \sum_{\ell=k}^{h-1} \epsilon_\ell^*, \text{ for all } x \in \mathbb{R}^n.$$

□

This theorem implies that

$$J_k(x) \leq V_k(x) \leq J_k(x) + (h-k)\epsilon \text{ for all } x \in \mathbb{R}^n,$$

with $\epsilon = \max\{\epsilon_\ell^* | \ell \in \{k, k+1, \ldots, h-1\}\}$, which is similar to the guarantee provided by RDP.

B. LMI-based ϵ optimization for complexity-bounded policy

This section provides numerical methods based on LMIs to check (12) and the last step (line 14) of Algorithm 2.

Proposition 3 below states that (11) holds if (12) holds with $\epsilon_{k+1} = \epsilon$. Moreover, it states that (12) is equivalent to

$$\exists \alpha \in \mathsf{P}_{|\mathcal{V}_k|} \quad \text{s.t.} \sum_{(P,q,r)\in\mathcal{V}_k} \alpha_i D_i \preceq D(\epsilon) \tag{17}$$

where

$$D_i = \begin{bmatrix} P_i & \frac{1}{2}q_i \\ \frac{1}{2}q_i^\top & r_i \end{bmatrix} \text{ and } D(\epsilon) = \begin{bmatrix} P(\epsilon) & \frac{1}{2}q(\epsilon) \\ \frac{1}{2}q(\epsilon)^\top & r(\epsilon) \end{bmatrix}. \tag{18}$$

Proposition 3: If (12) holds with ϵ_{k+1} replaced by ϵ, then (11) holds. Moreover, (12) with ϵ_{k+1} replaced by ϵ is equivalent to the LMI condition (17). □

The crucial fact that $P(\epsilon)$, $q(\epsilon)$ and $r(\epsilon)$ depend on ϵ in an affine manner (8) allows us to write the last step (line 14) of Algorithm 2 as an optimization subject to LMI constraints.

Lemma 4: Consider the disjoint sets $\mathcal{V}_k^{\epsilon_{k+1}}$ and $\mathcal{V}_k$ at a given arbitrary step of Algorithm 1. Then the smallest ϵ^* such that for $\epsilon_{k+1} \geq \epsilon^*$ (12) holds for all $(P(\epsilon_{k+1}), q(\epsilon_{k+1}), r(\epsilon_{k+1})) \in \mathcal{V}_k^{\epsilon_{k+1}}$ is given by the optimal value of the following optimization problem

$$\min \epsilon^*$$

such that $\exists \alpha \in \mathsf{P}_{|\mathcal{V}_k|}$ for which

$$\sum_{(P,q,r)\in\mathcal{V}_k} \alpha_i D_i \preceq \bar{D}_0 + \epsilon^* \bar{D}_1, \quad \forall (\bar{D}_0, \bar{D}_1) \in \mathcal{D} \tag{19}$$

where

$$\bar{D}_0 = \begin{bmatrix} \bar{P}_0 & \frac{1}{2}\bar{q}_0 \\ \frac{1}{2}\bar{q}_0^\top & \bar{r}_0 \end{bmatrix}, \quad \bar{D}_1 = \begin{bmatrix} \bar{P}_1 & \frac{1}{2}\bar{q}_1 \\ \frac{1}{2}\bar{q}_1^\top & \bar{r}_1 \end{bmatrix}$$

and

$$\mathcal{D} := \left\{ (\bar{D}_0, \bar{D}_1) | (\bar{P}_0, \bar{q}_0, \bar{r}_0) + \epsilon_{k+1}(\bar{P}_1, \bar{q}_1, \bar{r}_1) \in \mathcal{V}_k^{\epsilon_{k+1}} \right\}$$

□

Proof: Follows directly from Proposition 3, and the affine dependency of $P(\epsilon)$, $q(\epsilon)$, $r(\epsilon)$ on ϵ (see (8)). ■

This theorem applies both for the case of multiplicative cost guarantees, in which case ϵ_k^* is an increasing sequence, and the additive case, in which case ϵ_k^* is free.

Note that we can solve jointly the LMI optimization problem posed in Lemma 4 or sequentially by considering the optimization problems with restriction (19) for each member of $\mathcal{D}$, taking the maximum of the solutions of each problem, denoted here by ϵ_j^*, $j \in \{1, \ldots, |\mathcal{D}|\}$ and finally computing $\epsilon^* = \max\{\epsilon_j^* \mid j \in \{1, \ldots, |\mathcal{D}|\}\}$. The constraints on α imposed by $\alpha \in \mathsf{P}_{|\mathcal{V}_k|}$ can be written as a set of inequality constraints that can be incorporated into an extended version of the LMI.

C. Heuristic H

Algorithm 2 relies on a heuristic H, which assigns a positive real number to each tuple (P, q, r). The heuristic aims to assign small values to tuples with large chances of remaining in $\mathcal{V}_k$ at step k and large values to tuples with little chances. In general, the tightness of the provided ϵ_k guarantee at each step k strongly depends on H.

The heuristic proposed in [1] for a given application (switching linear systems) is $H = \text{trace}(P)$. Another heuristic for a different application (POMDPs) in [3] is to simply consider $H = r$. These are also valid heuristics for the problem at hand.

Here, we propose an additional heuristic that not only depends on the tuple in question but also on the values of a given set of tuples. Since this heuristic is to be applied at each step k in line 6 of the proposed RDP algorithm, the tuple in mind is $(\underline{P}(\epsilon_{k+1}), \underline{q}(\epsilon_{k+1}), \underline{r}(\epsilon_{k+1}))$ and the set of tuples in mind is $\mathcal{V}_k$. Note that this form of the heuristic does not pose any problem neither from an implementation point of view, nor for the formal results provided, Theorems 1, 2. The proposed heuristic is obtained by applying the same approach as to determine the performance bound ϵ^* explained at the end of Section III-B pertaining to the last step of the algorithm (line 14):

$$H((P(\epsilon), q(\epsilon), r(\epsilon)), \mathcal{V}) = \begin{cases} \frac{1}{\bar{\epsilon}} & \text{if } \bar{\epsilon} > 0 \\ \infty & \text{otherwise} \end{cases} \tag{20}$$

where

$$\min \bar{\epsilon} \geq 0$$

such that for all $x \in \mathbb{R}^n$ there exists $\alpha \in \mathsf{P}_{|\mathcal{V}|}$ for which

$$\sum_{(P,q,r)\in\mathcal{V}} \alpha_i (x^\top P x + q^\top x + r) \leq x^\top P(\bar{\epsilon}) x + q^\top(\bar{\epsilon}) x + r(\bar{\epsilon}). \tag{21}$$

The rationale behind this heuristic is that a tuple that should be added to $\mathcal{V}$ by satisfying

$$x^\top P(\epsilon) x + q^\top(\epsilon) x + r(\epsilon) < \sum_{(P,q,r)\in\mathcal{V}, \alpha\in\mathsf{P}_{|\mathcal{V}|}} \alpha_i (x^\top P x + q^\top x + r)$$

which is a proxy to

$$x^\top P(0) x + q^\top(0) x + r(0) < \sum_{(P,q,r)\in\mathcal{V}, \alpha\in\mathsf{P}_{|\mathcal{V}|}} \alpha_i (x^\top P x + q^\top x + r)$$

will require a large $\bar{\epsilon}$ to meet (21). This corresponds to a small H. In turn, a tuple for which (21) is met even with $\epsilon = 0$ can simply be discarded, leading to a value of H equal to infinity.

It should be noted that by selecting the tuples in this way in line 6 of the proposed RDP algorithm, the selection made is not necessarily the optimal one. This selection procedure can be regarded as a greedy approach, as it only adds the tuples one by one without already considering the next additions. However, since the added tuple in every step is based on a heuristic that is directly linked to the performance measure, it can in general be expected to yield reasonable results.

IV. Application to DT-LTI System with Discretized Inputs

We provide here a novel application of RDP. Consider a discrete-time linear time-invariant system $x_{k+1} = Ax_k + Bu_k$, with $x_k \in \mathbb{R}^n$ and $u_k \in U \subseteq \mathbb{R}^m$ and a quadratic cost

$$\min_{\{u_k | k\in\{0,\ldots,h-1\}\}} \sum_{k=0}^{h-1} x_k^\top Q x_k + u_k^\top R u_k + x_h^\top Q_h x_h \tag{22}$$

restricted here to be over a finite time horizon h. This would be the standard linear quadratic control if $U = \mathbb{R}^m$, but here we assume $U := \{u_i \mid i \in \{1, \ldots, n_u\}\}$ is a finite set.

This class of problem falls under the framework considered since: (i) the optimal value functions take the form (3) and (ii) the sets (6) can be written as (7). In fact, using induction, (i) is established by noticing that (3) holds at $k = h$ with $\mathcal{J}_h = \mathcal{V}_h = \{(Q_h, 0, 0)\}$ and, assuming that it holds when k is replaced by $k+1$ in (3) we also get (3) at time $k \in \{h-1, \ldots, 0\}$ since, for all $x \in \mathbb{R}^n$,

$$\begin{aligned} J_k(x) = &\min_{u\in U} x^\top Q x + u^\top R u + \\ &\min_{(\bar{P},\bar{q},\bar{r})\in\mathcal{J}_{k+1}} (Ax+Bu)^\top \bar{P}(Ax+Bu) + \bar{q}^\top(Ax+Bu) + \bar{r} \\ = &\min_{(P,q,r)\in\mathcal{J}_k} x^\top P x + q^\top x + r \end{aligned}$$

with

$$\mathcal{J}_k = \{(P,q,r) = (Q + A^\top \bar{P} A, A^\top \bar{q} + 2A^\top \bar{P} B u, \\ u^\top (R + B^\top \bar{P} B) u + \bar{q}^\top B u + \bar{r}) | u \in U, (\bar{P}, \bar{q}, \bar{r}) \in \mathcal{J}_{k+1}\}.$$

In a similar fashion we can conclude that the sets (6) can be written as (7) with

$$\mathcal{V}_k^\epsilon = \{(P,q,r) = ((1+\epsilon)Q + A^\top \bar{P} A, A^\top \bar{q} + 2A^\top \bar{P} B u, \\ u^\top ((1+\epsilon)R + B^\top \bar{P} B) u + \bar{q}^\top B u + \bar{r}) | u \in U, (\bar{P}, \bar{q}, \bar{r}) \in \mathcal{V}_{k+1}\}.$$

A. Simulation results

Consider a point mass in 2D with time step size Δt with state $x_k = \begin{bmatrix} \bar{x} & v_x & \bar{y} & v_y \end{bmatrix}_k^\top$ representing the position and velocity in 2D. The matrices of the linear system are

$$A = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} \frac{1}{2}\Delta t^2 & 0 \\ \Delta t & 0 \\ 0 & \frac{1}{2}\Delta t^2 \\ 0 & \Delta t \end{bmatrix}. \quad (23)$$

The input set is considered to be

$$U = \{u_1, u_2, u_3, u_4, u_5\} = \left\{ \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix} \right\}$$

in which the last element represents no input. The state costs are given by (22) with

$$Q = \mathrm{diag}([10 \;\; 5 \;\; 10 \;\; 5]) \quad R = 0_{2\times 2}, \quad Q_h = 0_{4\times 4}.$$

where $\mathrm{diag}([a_1 \cdots a_n])$ is a diagonal matrix with diagonal entries $a_1, \ldots, a_n$. Furthermore, we set $\Delta t = 0.25$s and $h = 5$. The results after the pruning procedure in the final iteration step ($k = 0$) for the original (performance-based) RDP algorithm and our proposed complexity-bounded RDP algorithm are compared in Fig. 1. The performance bound ϵ corresponds to the cost guarantee with the multiplicative factor given in (5) and the complexity C represents $|\mathcal{V}_0|$. The performance-based RDP algorithm here applies Algorithm 1 with $H(P,q,r) = r$. For the complexity-bounded RDP algorithm we apply Algorithm 2 with H given by (20), i.e., the greedy selection procedure, and with the `selectM` procedure relying on the minimizer of (14) when $x^1 = 0$; thus the first tuple $(P,q,r) \in \mathcal{V}_k^{\epsilon_{k+1}}(0)$ that is added is the one with the minimum r.

The figure shows that complexity-bounded pruning using the greedy selection procedure gives similar, in fact even better, results compared to performance-based pruning. Besides, a significantly stricter performance bound could be provided in some cases of performance-based pruning if it would be computed a posteriori as is done with complexity-bounded pruning, even if the pruned set is the same, as can be seen in the results for several values of ϵ that all result in $C = 10$. As already mentioned in Section I, the complexity resulting from the original performance-based RDP algorithm is not guaranteed to be a monotone function of the performance bound. Similarly, the performance bound resulting from the complexity-bounded RDP algorithm is not guaranteed to be a monotone function of the complexity, but the advantage is that the complexity is bounded in advance and, even if the model or cost function would change, will not suddenly lead to an intractable problem.

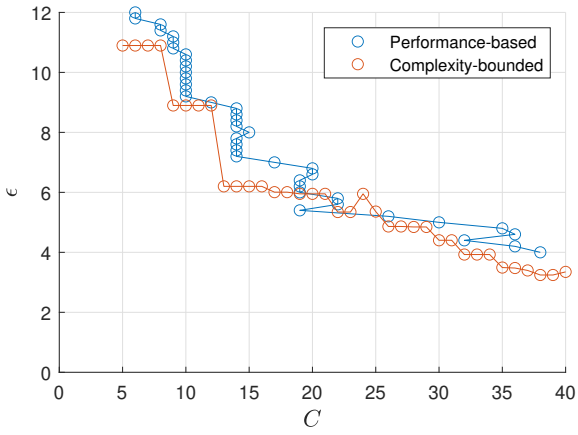

Fig. 1: Complexity and performance bounds resulting from performance-based pruning and complexity-bounded pruning for the 2D point mass example with $\Delta t = 0.25$s and $h = 5$.

V. Conclusions

We presented a modified relaxed dynamic programming algorithm with a guaranteed complexity bound. Our results apply when the value functions can be represented as minima of quadratic functions. This unifies several problems in different areas, including the LQR problems with discretized input, proposed here. The new approach removes a key limitation of the original RDP algorithm, namely the lack of complexity bounds. Instead of fixing the performance bound a priori and getting a complexity bound a posteriori, we rather find the best factor away from optimality for a preset complexity bound.

References

[1] B. Lincoln and A. Rantzer, "Relaxing dynamic programming," *IEEE Trans. on Automatic Control*, vol. 51, no. 8, pp. 1249–1260, Aug 2006.

[2] D. Görges, M. Izák, and S. Liu, "Optimal control and scheduling of switched systems," *IEEE Transactions on Automatic Control*, vol. 56, no. 1, pp. 135–140, 2011.

[3] D. J. Antunes, R. M. Beumer, M. J. G. van de Molengraft, and W. P. M. H. Heemels, "Adaptive sampling and actuation for pomdps: Application to precision agriculture," in *2022 IEEE 61st Conference on Decision and Control (CDC)*, 2022, pp. 2399–2404.

[4] Q. Wei, D. Liu, and Y. Xu, "Neuro-optimal tracking control for a class of discrete-time nonlinear systems via generalized value iteration adaptive dynamic programming approach," *Soft Computing*, vol. 20, pp. 697–706, 2016.

[5] L. Lu and J. M. Maciejowski, "Self-triggered mpc with performance guarantee using relaxed dynamic programming," *Automatica*, vol. 114, 2020.

[6] D. J. Antunes and W. Heemels, "Linear quadratic regulation of switched systems using informed policies," *IEEE Transactions on Automatic Control*, vol. 62, no. 6, pp. 2675–2688, 2017.

[7] A. R. P. Andriën and D. J. Antunes, "Near-optimal map estimation for markov jump linear systems using relaxed dynamic programming," *IEEE Control Systems Letters*, vol. 4, no. 4, pp. 815–820, 2020.