Outer-approximating Controlled Reach-avoid Sets for Polynomial Systems

Changyuan Zhao[1,2], Chuchu Fan[3] and Bai Xue[1,2]

Abstract— **In this paper we propose a semi-definite programming method for computing outer-approximations (i.e., supersets) of controlled reach-avoid sets of discrete-time polynomial systems subject to control inputs. The controlled reach-avoid set is a set of all initial states that there exists at least one control policy which steers the system starting from each of them to enter a specified target set in finite time while avoiding a given unsafe set till the target is hit. First, a Bellman type equation, whose unique bounded solution can characterize the exact controlled reach-avoid set, is derived. By relaxing this equation, a set of quantified inequalities for outer-approximating the controlled reach-avoid set is obtained. Via comparing to a set of constraints in state-of-the-art methods on occupation measures, we find that each has its own strengths and can complement each other in outer-approximating controlled reach-avoid sets. As a consequence, we integrate them and obtain a new set of constraints, which is weaker and thus contributes to the gain of tighter outer-approximations. The resulting set of constraints can be encoded into a semi-definite program via the sum-of-squares decomposition for multivariate variables, which can be solved efficiently via interior point methods in polynomial time. Finally, several examples demonstrate the benefits of our method on gaining tighter outer-approximations of controlled reach-avoid sets over existing methods.**

I. Introduction

Computational reachability analysis, which involves the computations of reachable states, is a popular tool for formal design and verification of safety-critical systems ranging from intelligent highway systems, to aircraft management systems, to computer and communication networks, etc [20], [3]. It has been widely studied over the last three decades in several disciplines including control theory, computer science and applied mathematics. Among the many possible extensions beyond reachability analysis, reach-avoid analysis is of fundamental importance in engineering. It can formalize many important engineering problems such as collision avoidance [15], path planning [4] and target surveillance [7], [19]. The reach-avoid problem comes in the two variants of computing a reach-avoid set and of verifying reach-avoid properties for systems featuring a given initial state set [21]. In this paper, we focus our attention on computing a reach-avoid set. A reach-avoid set is the set of all initial states such that a system starting from them is guaranteed to (eventually or within a given finite horizon) reach a specified target set while avoiding a given unsafe set till hitting the target. In general, the exact computation of reach-avoid sets is impossible for dynamical and hybrid systems. Their inner- and outer-approximations are therefore studied in the existing literature. An inner-approximation is a subset of the reach-avoid set, starting from which the satisfaction of the reach-avoid specification can be guaranteed. In contrast, an outer-approximation is a super-set of the reach-avoid set, which can be used to either synthesize reach-avoid controllers (e.g.,[8]) or falsify the reach-avoid specification. If an outer-approximation does not intersect the given initial set, the system does not satisfy the reach-avoid property definitely.

In the last decades, reach-avoid problems have been widely investigated for both continuous-time systems (modeled by differential equations) and discrete-time systems (modeled by difference equations). For continuous-time systems, they have been studied in the Hamilton-Jacobi reachability framework, e.g., [16], [2], [14], [5], [6], which links reach-avoid sets with viscosity solutions to Hamilton-Jacobi equations and finally reduces the problem of computing reach-avoid sets to the problem of addressing Hamilton-Jacobi equations. However, traditional numerical methods for solving Hamilton-Jacobi equations require gridding the state space, rendering these methods only scalable on systems of special structures. Recently, via relaxing Hamilton-Jacobi equations, a more scalable method exploiting semi-definite programming for inner-approximating reach-avoid sets has been suggested in [22]. On the other hand, moment-based programming methods were proposed for outer- as well as inner-approximating reach-avoid sets in [9], [10], [11], [18], [13], [24]. In contrast, studies on reach-avoid analysis for discrete-time systems are relatively rare compared to those on continuous-time systems, although discrete-time systems are important in practice and describe the evolution of a vast class of systems such as robot, digital controllers and physical systems simulated by digital computers [1]. A convex optimization method, which is derived from a system of equations, was proposed in [23] to study inner-approximate reachability analysis for discrete-time polynomial systems free of control inputs. More recently, a moment-based method was further extended to outer-approximate controlled reach-avoid sets as well as synthesize reach-avoid controllers for discrete-time polynomial systems subject to control inputs in [8].

In this paper, we consider the computation of controlled reach-avoid sets for discrete-time polynomial systems subject to control inputs. Our method begins with the derivation of a Bellman-type equation, whose unique bounded solution is able to characterize the exact controlled reach-avoid set. Due to the impossibility in solving the derived equation

1. State Key Lab. of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, China({zhaocy,xuebai}@ios.ac.cn)

2. University of Chinese Academy of Sciences, Beijing, China

3. Department of Aeronautics and Astronautics, MIT, MA 02139, U.S. (chuchu@mit.edu)

This work has been supported through grants by NSFC under grant No. 61872341, 61836005, CAS Pioneer Hundred Talents Program, and CAS Project for Young Scientists in Basic Research(No.YSBR-040).

generally, we further relax it and construct a set of quantified inequalities whose solutions are able to characterize outer-approximations of the controlled reach-avoid set. Comparing with the set of constraints in [8] in terms of structures and performances for computing outer-approximations, we find that these two sets of inequality constraints have their own strengths and can complement each other. Therefore, we integrate them and obtain a new set of inequality constraints. This new set of constraints combines the advantages of these two sets of constraints and thus can provide less conservative outer-approximations. We encode the new set of constraints into a semi-definite program via the sum-of-squares decomposition for multivariate variables, which falls within the convex optimization framework and can be solved efficiently in polynomial time via interior point methods. Finally, several examples demonstrate the performance of our proposed method and the results show that our method indeed can provide tighter outer-approximations of the controlled reach-avoid set over the method in [8]. The main contributions of this paper are summarized as follows:

1) We derive a Bellman type equation for the first time, the strict zero super-level set of whose unique bounded solution is equal to the controlled reach-avoid set.
2) A novel set of quantified inequalities, which is built upon the set of constraints from relaxing the aforementioned Bellman type equation and the set of constraints in [8], is proposed for computing outer-approximations of the controlled reach-avoid set. This set of inequalities can be encoded into a semi-definite program. The experimental results based on several examples show that our method is able to compute tighter outer-approximations, as opposed to the method in [8].

II. Preliminaries

In this section we present discrete-time polynomial systems and controlled reach-avoid sets under consideration in this paper. Before formulating the reach-avoid problem, let us introduce some basic notions used throughout this paper: $\mathbb{N}$ stands for the set of nonnegative integers and $\mathbb{R}$ for the set of real numbers. $[0,k)$ with $k \in \mathbb{N}$ is the family of integers $\{0,\dots,k-1\}$. $\mathbb{R}[\cdot]$ denotes the ring of polynomials in variables given by the argument. Vectors are denoted by boldface letters. $\sum[\boldsymbol{x}]$ denotes the set of sum of squares polynomials, i.e.,

$$\sum[\boldsymbol{x}] = \left\{ p(\boldsymbol{x}) \in \mathbb{R}[\boldsymbol{x}] \;\middle|\; \begin{array}{l} p(\boldsymbol{x}) = \sum_{i=1}^{k} q_i^2(\boldsymbol{x}), \\ q_i(\boldsymbol{x}) \in \mathbb{R}[\boldsymbol{x}], i=1,\dots,k \end{array} \right\}.$$

Definition 1: A discrete-time polynomial system subject to control inputs (abbr. **DPSC** in the sequel) is a system being modeled by iterative nonlinear maps of the following form:

$$\begin{aligned} &\boldsymbol{x}(l+1) = \boldsymbol{f}(\boldsymbol{x}(l), \boldsymbol{u}(l)), \forall l \in \mathbb{N} \\ &\boldsymbol{x}(0) = \boldsymbol{x}_0 \in \mathbb{R}^n \end{aligned} \tag{1}$$

where $\boldsymbol{x}(\cdot): \mathbb{N} \to \mathbb{R}^n$ are state vectors, $\boldsymbol{u}(\cdot): \mathbb{N} \to U$ are control inputs with $U = \{\boldsymbol{u} \in \mathbb{R}^m \mid q(\boldsymbol{u}) \le 0\}$ being a compact set in $\mathbb{R}^n$ with $q(\boldsymbol{u}) \in \mathbb{R}[\boldsymbol{u}]$, and $\boldsymbol{f}(\cdot,\cdot): \mathbb{R}^n \times U \to \mathbb{R}^n$ with $\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u}) \in \mathbb{R}[\boldsymbol{x},\boldsymbol{u}]$.

The trajectory of **DPSC** is driven by a control signal, which is defined as follows.

Definition 2: A control signal π for **DPSC** is a sequence $(\boldsymbol{u}(l))_{l\in\mathbb{N}}$, where $\boldsymbol{u}(\cdot): \mathbb{N} \to U$. We define $\mathcal{U}$ as the set of all control signals.

Given a control signal $\pi \in \mathcal{U}$ and an initial state $\boldsymbol{x}_0 \in \mathbb{R}^n$, we can obtain a trajectory of **DPSC**.

Definition 3: Given an initial state $\boldsymbol{x}_0 \in \mathbb{R}^n$ and a control signal $\pi = (\boldsymbol{u}(l))_{l\in\mathbb{N}}$, the trajectory of **DPSC**, induced by $\boldsymbol{x}_0$ and π, is a sequence $(\boldsymbol{\phi}_{\boldsymbol{x}_0}^{\pi}(l))_{l\in\mathbb{N}}$ satisfying $\boldsymbol{\phi}_{\boldsymbol{x}_0}^{\pi}(0) = \boldsymbol{x}_0$ and

$$\boldsymbol{\phi}_{\boldsymbol{x}_0}^{\pi}(l+1) = \boldsymbol{f}(\boldsymbol{\phi}_{\boldsymbol{x}_0}^{\pi}(l), \boldsymbol{u}(l)), \forall l \in \mathbb{N}.$$

Given an open target set $\mathcal{T}$ and a compact safe set $\mathcal{X}$, where $\mathcal{T}$ and $\mathcal{X}$ are defined by polynomial inequations as

$$\begin{aligned} &\mathcal{T} = \{\boldsymbol{x} \in \mathbb{R}^n \mid g(\boldsymbol{x}) < 0\}, \\ &\mathcal{X} = \{\boldsymbol{x} \in \mathbb{R}^n \mid h_0(\boldsymbol{x}) \le 0\} \end{aligned} \tag{2}$$

with $g(\boldsymbol{x}), h_0(\boldsymbol{x}) \in \mathbb{R}[\boldsymbol{x}]$ and $\mathcal{T} \subseteq \mathcal{X}$, the controlled reach-avoid set is defined as follows.

Definition 4: The controlled reach-avoid set $\mathcal{R}$ for **DPSC** is the set of all initial states $x_0 \in \mathcal{X}$ such that starting from x_0, there exists at least one control signal $\pi \in \mathcal{U}$ to ensure that the resulting trajectory can hit the target set $\mathcal{T}$ in a finite time $k \in \mathbb{N}$ while staying inside the safe set $\mathcal{X}$ before k, i.e.,

$$\mathcal{R} = \left\{ \boldsymbol{x}_0 \in \mathcal{X} \;\middle|\; \begin{array}{l} \exists \pi \in \mathcal{U}. \exists k \in \mathbb{N}. \\ [\boldsymbol{\phi}_{\boldsymbol{x}_0}^{\pi}(k) \in \mathcal{T} \wedge \bigwedge_{l=0}^{k} \boldsymbol{\phi}_{\boldsymbol{x}_0}^{\pi}(l) \in \mathcal{X}] \end{array} \right\}.$$

An outer-approximation is a superset of the controlled reach-avoid set $\mathcal{R}$.

III. Outer-approximating Controlled Reach-avoid Sets

In this section, we present our method to calculate outer-approximations of the controlled reach-avoid set $\mathcal{R}$. Our method originates from a Bellman-type equation, the strict zero super-level set of whose unique bounded solution is equal to the controlled reach-avoid set $\mathcal{R}$. The derivation of the Bellman type equation is detailed in Subsection III-A. Then, based on the derived Bellman-type equation and the set of constraints in [8], we obtain a novel set of constraints, which can be encoded into a semi-definite program, to outer-approximate the controlled reach-avoid set $\mathcal{R}$. The construction of this set of constraints is elucidated in Subsection III-B.

A. Bellman-type Equations

This section introduces the derivation of the Bellman equation aforementioned. Its derivation relies on an auxiliary controlled switched system, which is induced via freezing the dynamics of **DPSC** outside the safe set $\mathcal{X}$.

The controlled switched discrete-time nonlinear system (or, **CSDNS**) from **DPSC** specified by $(x_0, \mathcal{X}, f, U)$ is a quintuple $(\boldsymbol{x}_0, \widehat{\mathcal{L}}, \widehat{\mathcal{X}}, \widehat{\mathcal{Y}}, \widehat{\mathcal{F}})$ where:

- $\widehat{\mathcal{L}} = \{1, 2\}$ is a set of two locations;
- $\widehat{\mathcal{X}}$ is the state constraint set;
- $\widehat{\mathcal{Y}} = \{\widehat{\mathcal{X}}_i, i = 1, 2\}$ is a set of the safety constraints for each location;
- $\boldsymbol{x}_0 \in \widehat{\mathcal{X}}$ is the initial state;
- $\widehat{\mathcal{F}} = \{\widehat{\boldsymbol{f}}_i(\cdot, \cdot) : \widehat{\mathcal{X}}_i \times U \to \mathbb{R}^n, i = 1, 2\}$ are the dynamics;

which are constructed from **DPSC** as following:

1) $\widehat{\mathcal{X}} = \{\boldsymbol{x} \in \mathbb{R}^n \mid h(\boldsymbol{x}) \leq 0\}$ with $h(\boldsymbol{x}) \in \mathbb{R}[\boldsymbol{x}]$ and
$$\widehat{\mathcal{X}} \supseteq \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{x} = \boldsymbol{f}(\boldsymbol{y}, \boldsymbol{u}), \boldsymbol{y} \in \mathcal{X}, \boldsymbol{u} \in U\} \cup \mathcal{X},$$
which means that $\widehat{\mathcal{X}}$ has to contain the union of $\mathcal{X}$ and all reachable states starting from $\mathcal{X}$ in one step;
2) $\widehat{\mathcal{X}}_1 = \mathcal{X} = \{h_0(\boldsymbol{x}) \leq 0\}$;
3) $\widehat{\mathcal{X}}_2 = \widehat{\mathcal{X}} \setminus \mathcal{X} = \{\boldsymbol{x} \in \mathbb{R}^n \mid h(\boldsymbol{x}) \leq 0 \wedge -h_0(\boldsymbol{x}) < 0\}$;
4) $\widehat{\boldsymbol{f}}_1(\boldsymbol{x}, \boldsymbol{u}) = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$;
5) $\widehat{\boldsymbol{f}}_2(\boldsymbol{x}, \boldsymbol{u}) = \boldsymbol{x}$.

Similar to **DPSC**, trajectories of **CSDNS** are defined as follows.

Definition 5: Given an initial state $\boldsymbol{x}_0 \in \widehat{\mathcal{X}}$ and a control signal $\pi = (\boldsymbol{u}(l))_{l \in \mathbb{N}}$, the trajectory of **CSDNS**, induced by $\boldsymbol{x}_0$ and π, is a sequence $(\widehat{\boldsymbol{\phi}}^{\pi}_{\boldsymbol{x}_0}(l))_{l \in \mathbb{N}}$ satisfying the iterative piece-wise polynomial map
$$\widehat{\boldsymbol{\phi}}^{\pi}_{\boldsymbol{x}_0}(l+1) = \widehat{\boldsymbol{f}}(\widehat{\boldsymbol{\phi}}^{\pi}_{\boldsymbol{x}_0}(l), \boldsymbol{u}(l)), \forall l \in \mathbb{N},$$
where $\widehat{\boldsymbol{\phi}}^{\pi}_{\boldsymbol{x}_0}(0) = \boldsymbol{x}_0$ and
$$\widehat{\boldsymbol{f}}(\boldsymbol{x}, \boldsymbol{u}) := 1_{\widehat{\mathcal{X}}_1}(\boldsymbol{x}) \cdot \widehat{\boldsymbol{f}}_1(\boldsymbol{x}, \boldsymbol{u}) + 1_{\widehat{\mathcal{X}}_2}(\boldsymbol{x}) \cdot \widehat{\boldsymbol{f}}_2(\boldsymbol{x}, \boldsymbol{u}) \quad (3)$$
with $\widehat{\boldsymbol{f}}(\cdot, \cdot) : S \times U \to \mathbb{R}^n, S = \widehat{\mathcal{X}}_i$ if $\boldsymbol{x} \in \widehat{\mathcal{X}}_i$, and $1_{\widehat{\mathcal{X}}_i}(\cdot) : \widehat{\mathcal{X}}_i \to \{0, 1\}$, $i = 1, 2$, representing the indicator function of the set $\widehat{\mathcal{X}}_i$, i.e.,
$$1_{\widehat{\mathcal{X}}_i}(\boldsymbol{x}) := \begin{cases} 1, & \text{if } \boldsymbol{x} \in \widehat{\mathcal{X}}_i, \\ 0, & \text{if } \boldsymbol{x} \notin \widehat{\mathcal{X}}_i. \end{cases}$$

The problem of computing the controlled reach-avoid set $\mathcal{R}$ for **DPSC** is equivalently transformed into the problem of computing the controlled reach set for **CSDNS**, i.e., the controlled reach-avoid set $\mathcal{R}$ is equal to the set of all initial states such that there exists a control signal steering **CSDNS** starting from each of them to eventually enter the target set $\mathcal{T}$. This statement is formalized in Proposition 1.

Proposition 1: The reach-avoid set $\mathcal{R}$ in Definition 4 is equal to the controlled reach set $\widehat{\mathcal{R}}$ of **CSDNS**, where
$$\widehat{\mathcal{R}} = \left\{\boldsymbol{x}_0 \in \mathcal{X} \,\middle|\, \exists \pi \in \mathcal{U}. \exists k \in \mathbb{N}. \widehat{\boldsymbol{\phi}}^{\pi}_{\boldsymbol{x}_0}(k) \in \mathcal{T}. \right\}$$

Proof: It is observed that the states in the unsafe set $\widehat{\mathcal{X}}_2$ are invariant for **CSDNS**. Therefore, trajectories of **CSDNS** is no longer able to enter the target set once they enter the unsafe set. Consequently, if they enter the target set, they can not leave the safe set before the first hitting time. Also, trajectories of **DPSC** and **CSDNS** starting from the state $\boldsymbol{x}_0 \in \mathcal{X}$ with the same signal π coincide until they go outside the safe set $\mathcal{X}$. Therefore, we have the conclusion. ■

According to Proposition 1, we can equivalently study the computation of $\widehat{\mathcal{R}}$ instead.

Next, based on **CSDNS**, we construct a discounted value function, whose strict zero super-level set is equal to the controlled reach set $\widehat{\mathcal{R}}$ as stated in Proposition 2. This discounted value function satisfies the dynamic programming principle as shown in Lemma 1, and is finally reduced to the unique bounded solution to a Bellman type equation in Theorem 1. It is formalized as below:
$$V(\boldsymbol{x}) := \sup_{\pi \in \mathcal{U}} \sup_{l \in \mathbb{N}} \alpha^l 1_{\mathcal{T}}(\widehat{\boldsymbol{\phi}}^{\pi}_{\boldsymbol{x}}(l)) \quad (4)$$
where $\alpha \in (0, 1)$ is the discount factor and $1_{\mathcal{T}}(\cdot)$ is the indicate function of the set $\mathcal{T}$.

Proposition 2: $\mathcal{R} = \{\boldsymbol{x} \in \mathcal{X} \mid V(\boldsymbol{x}) > 0\}$.

Proof: According to Proposition 1, we just need to show that $\widehat{\mathcal{R}} = \{\boldsymbol{x} \in \mathcal{X} \mid V(\boldsymbol{x}) > 0\}$.

We first show that $\widehat{\mathcal{R}} \subseteq \{\boldsymbol{x} \in \mathcal{X} \mid V(\boldsymbol{x}) > 0\}$. Let $\boldsymbol{x}_0 \in \widehat{\mathcal{R}}$. Then there exists $\pi \in \mathcal{U}$ and $k \in \mathbb{N}$ such that $\widehat{\boldsymbol{\phi}}^{\pi}_{\boldsymbol{x}_0}(k) \in \mathcal{T}$ holds. Thus, we have that
$$V(\boldsymbol{x}_0) \geq \alpha^k 1_{\mathcal{T}}(\widehat{\boldsymbol{\phi}}^{\pi}_{\boldsymbol{x}_0}(k)) = \alpha^k > 0,$$
implying that $\widehat{\mathcal{R}} \subseteq \{\boldsymbol{x} \in \mathcal{X} \mid V(\boldsymbol{x}) > 0\}$.

Next, we show that $\{\boldsymbol{x} \in \mathcal{X} \mid V(\boldsymbol{x}) > 0\} \subseteq \widehat{\mathcal{R}}$. Let $\boldsymbol{x}_0 \in \{\boldsymbol{x} \in \mathcal{X} \mid V(\boldsymbol{x}) > 0\}$, indicating that there exists $\delta > 0$ such that $V(\boldsymbol{x}_0) = \delta$. Since $\lim_{l \to \infty} \alpha^l = 0$, we have that there exists $\pi \in \mathcal{U}$ and $k \in \mathbb{N}$ such that $\widehat{\boldsymbol{\phi}}^{\pi}_{\boldsymbol{x}_0}(k) \in \mathcal{T}$. According to Proposition 1, $\boldsymbol{x}_0 \in \widehat{\mathcal{R}}$ and thus $\{\boldsymbol{x} \in \mathcal{X} \mid V(\boldsymbol{x}) > 0\} \subseteq \widehat{\mathcal{R}}$.

In summary, we have that $\mathcal{R} = \{\boldsymbol{x} \in \mathcal{X} \mid V(\boldsymbol{x}) > 0\}$. ■

Lemma 1: For $\boldsymbol{x} \in \widehat{\mathcal{X}}$ and $k \in \mathbb{N}$,
$$V(\boldsymbol{x}) = \sup_{\pi \in \mathcal{U}} \max\{\sup_{l \in [0,k)} \alpha^l 1_{\mathcal{T}}(\widehat{\boldsymbol{\phi}}^{\pi}_{\boldsymbol{x}}(l)), \alpha^k V(\widehat{\boldsymbol{\phi}}^{\pi}_{\boldsymbol{x}}(k))\}, \quad (5)$$
where $\alpha \in (0, 1)$.

Proof: Since $V(\boldsymbol{x}) = \sup_{\pi \in \mathcal{U}} \sup_{l \in \mathbb{N}} \alpha^l 1_{\mathcal{T}}(\widehat{\boldsymbol{\phi}}^{\pi}_{\boldsymbol{x}}(l))$, we have that
$$V(\boldsymbol{x}) = \begin{cases} 0, & \text{if } \boldsymbol{x} \in \widehat{\mathcal{X}} \setminus \widehat{\mathcal{R}}, \\ \alpha^{l_0}, & \text{if } \boldsymbol{x} \in \widehat{\mathcal{R}}, \end{cases} \quad (6)$$
where $l_0 = \inf_{\pi \in \mathcal{U}}\{l \in \mathbb{N} \mid \widehat{\boldsymbol{\phi}}^{\pi}_{\boldsymbol{x}}(l) \in \mathcal{T}\}$, which is the first hitting time of the target set $\mathcal{T}$. Thus, we just need to prove that
$$\sup_{\pi \in \mathcal{U}} \max\{\sup_{l \in [0,k)} \alpha^l 1_{\mathcal{T}}(\widehat{\boldsymbol{\phi}}^{\pi}_{\boldsymbol{x}}(l)), \alpha^k V(\widehat{\boldsymbol{\phi}}^{\pi}_{\boldsymbol{x}}(k))\} = \begin{cases} 0, & \text{if } \boldsymbol{x} \in \widehat{\mathcal{X}} \setminus \widehat{\mathcal{R}}, \\ \alpha^{l_0}, & \text{if } \boldsymbol{x} \in \widehat{\mathcal{R}}. \end{cases}$$

Firstly, let $\boldsymbol{x}_0 \in \widehat{\mathcal{R}}$. If $l_0 < k$, we have that
$$\sup_{\pi \in \mathcal{U}} \max\{\sup_{l \in [0,k)} \alpha^l 1_{\mathcal{T}}(\widehat{\boldsymbol{\phi}}^{\pi}_{\boldsymbol{x}_0}(l)), \alpha^k V(\widehat{\boldsymbol{\phi}}^{\pi}_{\boldsymbol{x}_0}(k))\} = \alpha^{l_0}.$$
Otherwise, if $l_0 \geq k$, we have that $\sup_{l \in [0,k)} 1_{\mathcal{T}}(\widehat{\boldsymbol{\phi}}^{\pi}_{\boldsymbol{x}_0}(l)) = 0$ for $\pi \in \mathcal{U}$, implying that
$$\sup_{\pi \in \mathcal{U}} \max\{\sup_{l \in [0,k)} \alpha^l 1_{\mathcal{T}}(\widehat{\boldsymbol{\phi}}^{\pi}_{\boldsymbol{x}_0}(l)), \alpha^k V(\widehat{\boldsymbol{\phi}}^{\pi}_{\boldsymbol{x}_0}(k))\} = \sup_{\pi \in \mathcal{U}} \alpha^k V(\widehat{\boldsymbol{\phi}}^{\pi}_{\boldsymbol{x}_0}(k)) = \alpha^k \alpha^{l_0 - k} = \alpha^{l_0}.$$

Next, let $\boldsymbol{x}_0 \in \widehat{\mathcal{X}} \setminus \widehat{\mathcal{R}}$. Then, $\alpha^l 1_{\mathcal{T}}(\widehat{\boldsymbol{\phi}}^{\pi}_{\boldsymbol{x}_0}(l)) \equiv 0$ for $l \in \mathbb{N}$ and $\pi \in \mathcal{U}$ holds. Thus, we have that $\sup_{\pi\in\mathcal{U}} \max\{\sup_{l\in[0,k)} \alpha^l 1_{\mathcal{T}}(\widehat{\boldsymbol{\phi}}^{\pi}_{\boldsymbol{x}_0}(l)), \alpha^k V(\widehat{\boldsymbol{\phi}}^{\pi}_{\boldsymbol{x}_0}(k))\} = 0$.

In summary, we have the conclusion. ∎

According to Lemma 1, we deduce that the function $V(\boldsymbol{x})$ in (4) is the unique bounded solution to a Bellman type equation.

Theorem 1: The value function $V(\cdot) : \widehat{\mathcal{X}} \to \mathbb{R}$ in (4) is the unique bounded solution to the following Bellman type equation

$$\min\{V(\boldsymbol{x}) - 1_{\mathcal{T}}(\boldsymbol{x}), V(\boldsymbol{x}) - \alpha \sup_{\boldsymbol{u}\in U} V(\widehat{\boldsymbol{f}}(\boldsymbol{x}, \boldsymbol{u}))\} = 0 \quad (7)$$

with $\alpha \in (0, 1)$.

Proof: According to Lemma 1, the value function $V(\boldsymbol{x})$ in (4) satisfies equation (5) when $k \in \mathbb{N}$. Letting $k = 1$ in (5), we obtain that $V(\boldsymbol{x})$ is a solution to the Bellman type equation (7). The boundedness of the value function $V(\cdot) : \widehat{\mathcal{X}} \to \mathbb{R}$ can easily be judged from (6).

Next, we prove the uniqueness. Assume that there exists another bounded function $V'(\cdot) : \widehat{\mathcal{X}} \to \mathbb{R}$ satisfying equation (7), and there exists $\boldsymbol{y} \in \widehat{\mathcal{X}}$ such that $V(\boldsymbol{y}) \neq V'(\boldsymbol{y})$. Since both $V(\boldsymbol{y})$ and $V'(\boldsymbol{y})$ satisfy equation (7), we have that

$$\begin{aligned}
&|V(\boldsymbol{y}) - V'(\boldsymbol{y})| \\
=&|\sup_{\boldsymbol{u}\in U} \max\{1_{\mathcal{T}}(\boldsymbol{y}), \alpha V(\widehat{\boldsymbol{f}}(\boldsymbol{y}, \boldsymbol{u}))\} \\
&- \sup_{\boldsymbol{u}\in U} \max\{1_{\mathcal{T}}(\boldsymbol{y}), \alpha V'(\widehat{\boldsymbol{f}}(\boldsymbol{y}, \boldsymbol{u}))\}| \\
\leq&\alpha \sup_{\boldsymbol{u}\in U} |V(\widehat{\boldsymbol{f}}(\boldsymbol{y}, \boldsymbol{u})) - V'(\widehat{\boldsymbol{f}}(\boldsymbol{y}, \boldsymbol{u}))|
\end{aligned}$$

Therefore,

$$|V(\boldsymbol{y}) - V'(\boldsymbol{y})| \leq \alpha^k \sup_{\pi\in\mathcal{U}} |V(\widehat{\boldsymbol{\phi}}^{\pi}_{\boldsymbol{y}}(k)) - V'(\widehat{\boldsymbol{\phi}}^{\pi}_{\boldsymbol{y}}(k))|, \forall k \in \mathbb{N}.$$

Since the functions $V(\boldsymbol{x})$ and $V'(\boldsymbol{x})$ are both bounded over $\boldsymbol{x} \in \widehat{\mathcal{X}}$, we conclude that $|V(\boldsymbol{y} - V'(\boldsymbol{y})| = 0$, which contradicts the fact that $V(\boldsymbol{y}) \neq V'(\boldsymbol{y})$.

Consequently, $V(\cdot) : \widehat{\mathcal{X}} \to \mathbb{R}$ with $\alpha \in (0, 1)$ is the unique bounded solution to equation (7). ∎

B. Outer-approximating Controlled Reach-avoid Sets

In this subsection we present our semi-definite programming (abbr. SDP) method for computing outer-approximations of the controlled reach-avoid set.

Based on the system of equations in Theorem 1, we derive a set of inequalities for outer-approximating the controlled reach-avoid set $\mathcal{R}$.

Corollary 1: If there exists a bounded function $v(\cdot) : \widehat{\mathcal{X}} \to \mathbb{R}$ satisfying the following constraints

$$v(\boldsymbol{x}) \geq \alpha v(\widehat{\boldsymbol{f}}(\boldsymbol{x}, \boldsymbol{u})), \forall \boldsymbol{u} \in U, \forall \boldsymbol{x} \in \mathcal{X} \setminus \mathcal{T} \quad (8)$$

$$v(\boldsymbol{x}) \geq 1, \forall \boldsymbol{x} \in \mathcal{T} \quad (9)$$

where $\alpha \in (0, 1)$. Then the strict zero super-level set of the function $v(\cdot) : \widehat{\mathcal{X}} \to \mathbb{R}$ is an outer-approximation of the reach-avoid set $\mathcal{R}$, i.e.

$$\mathcal{R} \subseteq \{\boldsymbol{x} \in \mathcal{X} \mid v(\boldsymbol{x}) > 0\}.$$

Proof: For $\boldsymbol{x}_0 \in \mathcal{R}$, it holds that $\exists \pi \in \mathcal{U}.\exists k \in \mathbb{N}.\widehat{\boldsymbol{\phi}}^{\pi}_{\boldsymbol{x}_0}(k) \in \mathcal{T}$. From (8) and (9), we have that

$$v(\boldsymbol{x}) \geq \alpha v(\widehat{\boldsymbol{\phi}}^{\pi}_{\boldsymbol{x}_0}(1)) \geq \cdots \geq \alpha^k v(\widehat{\boldsymbol{\phi}}^{\pi}_{\boldsymbol{x}_0}(k)) \geq \alpha^k > 0 \quad (10)$$

Therefore, $\boldsymbol{x}_0 \in \{\boldsymbol{x} \in \mathcal{X} \mid v(\boldsymbol{x}) > 0\}$, i.e. $\mathcal{R} \subseteq \{\boldsymbol{x} \in \mathcal{X} \mid v(\boldsymbol{x}) > 0\}$. ∎

The system of inequalities (8)-(9) can be equivalently transformed into the following inequalities (11)-(12) respectively via removing the indicator function:

$$v(\boldsymbol{x}) \geq \alpha v(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})), \forall \boldsymbol{u} \in U, \forall \boldsymbol{x} \in \mathcal{X} \setminus \mathcal{T} \quad (11)$$

$$v(\boldsymbol{x}) \geq 1, \forall \boldsymbol{x} \in \mathcal{T} \quad (12)$$

where $\alpha \in (0, 1)$. It is observed that when $\alpha = 1$, we can also calculate an outer-approximation by solving inequalities (11)-(12). It is concluded from (10) that the computed outer-approximation is $\{\boldsymbol{x} \in \widehat{\mathcal{X}} \mid v(\boldsymbol{x}) - 1 \geq 0\}$.

Therefore, we can solve the following optimization

$$\begin{aligned}
&\inf \int_{\mathcal{X}} v(\boldsymbol{x}) dx \\
&\text{s.t.}(11) - (12),
\end{aligned} \quad (13)$$

where $\alpha \in (0, 1]$ is a user-defined discount factor.

The problem of addressing optimization (13) can be relaxed into an SDP problem as shown in (14) via sum-of-squares decomposition for multivariate polynomials, which can be solved efficiently in polynomial time via interior point methods.

$$\begin{aligned}
&\min \boldsymbol{c} \cdot \hat{\boldsymbol{w}} \\
&\text{s.t.} \\
&v(\boldsymbol{x}) - \alpha v(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})) + s_0(\boldsymbol{x}, \boldsymbol{u}) h_0(\boldsymbol{x}) \\
&\qquad - s_1(\boldsymbol{x}, \boldsymbol{u}) g(\boldsymbol{x}) + s_2(\boldsymbol{x}, \boldsymbol{u}) q(\boldsymbol{u}) \in \sum[\boldsymbol{x}, \boldsymbol{u}], \\
&v(\boldsymbol{x}) - 1 + s_3(\boldsymbol{x}) g(\boldsymbol{x}) \in \sum[\boldsymbol{x}],
\end{aligned} \quad (14)$$

where $\alpha \in (0, 1]$ is a user-defined discount factor, $\boldsymbol{c} \cdot \widehat{\boldsymbol{w}} = \int_{\mathcal{X}} v(\boldsymbol{x}) d\boldsymbol{x}$, $\hat{\boldsymbol{w}}$ is the constant vector computed by integrating the monomial in $v(\boldsymbol{x}) \in \mathbb{R}[\boldsymbol{x}]$ over $\mathcal{X}$, $\boldsymbol{c}$ is the vector composed of unknown coefficients in $v(\boldsymbol{x}) \in \mathbb{R}[\boldsymbol{x}]$, and $s_i(\boldsymbol{x}, \boldsymbol{u}) \in \sum[\boldsymbol{x}]$, $i = 0, \ldots, 2$, and $s_3(\boldsymbol{x}) \in \sum[\boldsymbol{x}]$.

Also, we notice that the problem of calculating outer-approximations of the controlled reach-avoid set was reduced to an optimization problem in [8]. For convenient reference, the optimization problem is presented below.

$$\begin{aligned}
&\inf \int_{\mathcal{X}} w(\boldsymbol{x}) d\boldsymbol{x} \\
s.t.\ & v(\boldsymbol{x}) - v(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})) \geq 0, \forall \boldsymbol{x} \in \mathcal{X}, \forall \boldsymbol{u} \in U, \\
& w(\boldsymbol{x}) - v(\boldsymbol{x}) - 1 \geq 0, \forall \boldsymbol{x} \in \mathcal{X}, \\
& w(\boldsymbol{x}) \geq 0, \forall \boldsymbol{x} \in \mathcal{X}, \\
& v(\boldsymbol{x}) \geq 0, \forall \boldsymbol{x} \in \mathcal{T}.
\end{aligned} \quad (15)$$

Via solving it, which is encoded into SDP (9) in [8], we can obtain an outer-approximation $\{\boldsymbol{x} \in \mathcal{X} \mid v(\boldsymbol{x}) \geq 0\}$. Due to space limitation, we do not provide SDP (9) in [8] here and use SDP $\mathbb{O}$ to denote in the sequel.

Comparing constraints in optimizations (13) and (15), we find that constraints in optimizations (13) are weaker and thus have more solutions than ones in (15). First, unlike the constraint $v(\boldsymbol{x}) - v(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})) \geq 0, \forall \boldsymbol{x} \in \mathcal{X}, \forall \boldsymbol{u} \in U$ in (15), the term $v(\boldsymbol{x}) - \alpha v(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}))$ in optimization (13) is not required to be non-negative over the target set $\mathcal{T}$. Second, if there exists $v(\boldsymbol{x})$ satisfying $v(\boldsymbol{x}) - v(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})) \geq 0, \forall \boldsymbol{x} \in \mathcal{X} \setminus \mathcal{T}, \forall \boldsymbol{u} \in U$, it also satisfies $v(\boldsymbol{x}) - \alpha v(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})) \geq 0, \forall \boldsymbol{x} \in \mathcal{X}, \forall \boldsymbol{u} \in U$, where $\alpha = 1$. This observation leads to the conclusion that constraints in optimizations (13) have more solutions than ones in (15), which is also formally justified in Proposition 3.

Proposition 3: If $v(\boldsymbol{x})$ is a solution to optimization (15), $v'(\boldsymbol{x}) := v(\boldsymbol{x}) + 1$ is a solution to optimization (13) with $\alpha = 1$ as well.

Proof: If $v(\boldsymbol{x})$ is a solution to optimization (15), then

$$\begin{aligned}&v(\boldsymbol{x}) - v(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})) \geq 0, \forall \boldsymbol{x} \in \mathcal{X}, \forall \boldsymbol{u} \in U,\\ &v(\boldsymbol{x}) \geq 0, \forall \boldsymbol{x} \in \mathcal{T}.\end{aligned}$$

Thus,

$$\begin{aligned}&v'(\boldsymbol{x}) - v'(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})) \geq 0, \forall \boldsymbol{x} \in \mathcal{X} \setminus \mathcal{T}, \forall \boldsymbol{u} \in U,\\ &v'(\boldsymbol{x}) \geq 1, \forall \boldsymbol{x} \in \mathcal{T}.\end{aligned}$$

Consequently, the conclusion holds. ■

Although constraints in optimization (13) with $\alpha = 1$ may have more solutions than ones in (15), in practical computations optimization (13) with $\alpha = 1$ may not perform better than optimization (15). On the other hand, the existence of discount factor α in optimization (13) would increase its flexibility and contribute to the gain of less conservative outer-approximations. Based on SDP (14) and SDP $\mathbb{O}$, we use an example to illustrate these.

Example 1 (Van der Pol oscillator): Consider the Van der Pol oscillartor in [8] with a discrete time $\delta t = 0.01$, which is free of control inputs,

$$\begin{cases} x(l+1) = x(l) + 0.01(-2x(l)) \\ y(l+1) = y(l) + 0.01(0.8x(l) + 10(x^2(l) - 0.21)y(l))) \end{cases}$$

with $\mathcal{X} = \{(x, y)^\top \mid x^2 + y^2 - 1.5 \leq 0\}$ and $\mathcal{T} = \{(x, y)^\top \mid x^2 + y^2 - 0.01 < 0\}$.

We respectively solve SDP (14) with $\alpha = 0.99$ and $\alpha = 1$, and SDP $\mathbb{O}$. In all computations, polynomials $v(\boldsymbol{x})$ of degree 6 are used to approximate $\mathcal{R}$. The approximate results from SDP (14) with $\alpha = 0.99$ and SDP $\mathbb{O}$ are visualized in Fig. 1. We observe that the outer-approximation computed from SDP $\mathbb{O}$ is more conservative than the one from SDP (14) with $\alpha = 0.99$. However, we encountered some difficulties when solving SDP (14) with $\alpha = 1$, the only solution $v(\boldsymbol{x})$ found by the solver Mosek is one such that $v(\boldsymbol{x}) - 1$, which is used to define an outer-approximation, is very close to the zero polynomial: the coefficients are of the order 10^{-4}. Therefore, the plot is irrelevant (i.e., the solution is not reliable) and thus is not shown in Fig. 1.

Comparing SDP $\mathbb{O}$ and (14), it is observed that they have different objectives, which may be the reason leading to the better performance of SDP $\mathbb{O}$ over SDP (14) with $\alpha = 1$. As shown in [8] the objective in SDP $\mathbb{O}$ is larger than zero,

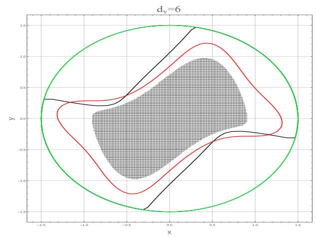


Fig. 1. An illustration of outer-approximating $\mathcal{R}$ for Example 1. Green curve denotes the boundary of $\mathcal{X}$. Black and red curves denote the boundaries of computed outer-approximations by solving SDP $\mathbb{O}$ and (14) with $\alpha = 0.99$, respectively. Gray region corresponds to $\mathcal{R}$ estimated by the Monte-Carlo simulation method.

i.e., bounded from below, and may converge to the volume of the controlled reach-avoid set from above as the degree of the polynomial $w(\boldsymbol{x})$ approaches infinity. However, the objective in SDP (14) can take on negative values and is not pertinent to the size of computed outer-approximations. Therefore, the introduction of such function $w(\boldsymbol{x})$ may enhance the performance of SDP (14) in computing outer-approximations. Based on this, we combine the strengths of optimizations (13) and (15) to obtain a new optimization, as shown in (16). In optimization (16), we retain the objective of optimization (15) and take the constraint

$$v(\boldsymbol{x}) - \alpha v(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})) \geq 0, \forall \boldsymbol{x} \in \mathcal{X} \setminus \mathcal{T}, \forall \boldsymbol{u} \in U$$

in optimization (13) rather than the constraint $v(\boldsymbol{x}) - v(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})) \geq 0, \forall \boldsymbol{x} \in \mathcal{X}, \forall \boldsymbol{u} \in U$ in optimization (15).

$$\begin{aligned}&\inf \int_{\mathcal{X}} w(\boldsymbol{x}) dx\\ \text{s.t. } &v(\boldsymbol{x}) - \alpha v(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})) \geq 0, \forall \boldsymbol{x} \in \mathcal{X} \setminus \mathcal{T}, \forall \boldsymbol{u} \in U,\\ &w(\boldsymbol{x}) - v(\boldsymbol{x}) - 1 \geq 0, \forall \boldsymbol{x} \in \mathcal{X},\\ &w(\boldsymbol{x}) \geq 0, \forall x \in \mathcal{X},\\ &v(\boldsymbol{x}) \geq 0, \forall x \in \mathcal{T}.\end{aligned} \tag{16}$$

where $\alpha \in (0, 1]$ is an user-defined discount factor.

Lemma 2: If there exist bounded functions $v(\boldsymbol{x}) : \mathcal{X} \to \mathbb{R}$ and $w(\boldsymbol{x}) : \mathcal{X} \to \mathbb{R}$ satisfying the constraints in optimization (16), $\{\boldsymbol{x} \in \mathcal{X} \mid v(\boldsymbol{x}) \geq 0\}$ is an outer-approximation of the controlled reach-avoid set $\mathcal{R}$.

Proof: By following the proof of Corollary 1, we can obtain the conclusion. ■

Similarly, optimization (16) can be relaxed into an SDP, which is presented below.

We find that solving SDP (17) is able to provide tighter outer-approximations than SDP $\mathbb{O}$ and (14). In the following we continue using the system in Example 1 to illustrate this.

Example 2: Consider the system in Example 1 again. Polynomials $v(\boldsymbol{x})$ of degree 6 are also used to approximate $\mathcal{R}$ in solving SDP (17). Also, two cases of $\alpha = 1$ and $\alpha = 0.99$ are considered. The results are shown in Fig. 2. Compared with the results in Example 1, it is easy to conclude that SDP (17) improves the performance of SDP (14) for both cases (i.e., $\alpha = 1$ and 0.99). Also, although the computed outer-approximation from SDP (17) with $\alpha = 1$ almost coincides

$$
\begin{aligned}
&\min \boldsymbol{c} \cdot \hat{\boldsymbol{w}} \\
&\text{s.t.} \\
&v(\boldsymbol{x}) - \alpha v(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})) + s_0(\boldsymbol{x}, \boldsymbol{u}) h_0(\boldsymbol{x}) \\
&\qquad - s_1(\boldsymbol{x}, \boldsymbol{u}) g(\boldsymbol{x}) + s_2(\boldsymbol{x}, \boldsymbol{u}) q(\boldsymbol{u}) \in \sum[\boldsymbol{x}, \boldsymbol{u}], \\
&v(\boldsymbol{x}) + s_3(\boldsymbol{x}) g(\boldsymbol{x}) \in \sum[\boldsymbol{x}], \\
&w(\boldsymbol{x}) - v(\boldsymbol{x}) - 1 + s_4(\boldsymbol{x}) h_0(\boldsymbol{x}) \in \sum[\boldsymbol{x}], \\
&w(\boldsymbol{x}) + s_5(\boldsymbol{x}) h_0(\boldsymbol{x}) \in \sum[\boldsymbol{x}],
\end{aligned}
\tag{17}
$$

where $\alpha \in (0, 1]$, $\boldsymbol{c} \cdot \widehat{\boldsymbol{w}} = \int_{\mathcal{X}} w(\boldsymbol{x}) d\boldsymbol{x}$, $\hat{\boldsymbol{w}}$ is the constant vector computed by integrating the monomial in $w(\boldsymbol{x}) \in \mathbb{R}[\boldsymbol{x}]$ over $\mathcal{X}$, $\boldsymbol{c}$ is the vector composed of unknown coefficients in $v(\boldsymbol{x}) \in \mathbb{R}[\boldsymbol{x}]$, and $w(\boldsymbol{x}) \in \mathbb{R}[\boldsymbol{x}]$ and $s_i(\boldsymbol{x}, \boldsymbol{u}) \in \sum[\boldsymbol{x}, \boldsymbol{u}]$, $i = 0, \ldots, 2$, and $s_i(\boldsymbol{x}) \in \sum[\boldsymbol{x}]$, $i = 3, \ldots, 5$.

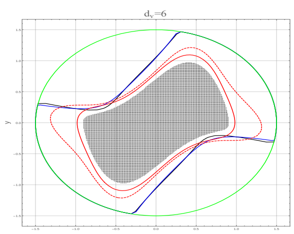


Fig. 2. An illustration of outer-approximating $\mathcal{R}$ for Example 2. Green curve denotes the boundary of $\mathcal{X}$. Black, blue and red curves denote the boundaries of computed outer-approximations from SDP $\mathbb{O}$ and (17) with $\alpha = 1$ and $\alpha = 0.99$, respectively. Red dashed curve denotes the boundary of computed outer-approximation from SDP (14) with $\alpha = 0.99$. Gray region corresponds to $\mathcal{R}$ estimated by the Monte-Carlo simulation method.

with the one from SDP $\mathbb{O}$, the former with $\alpha = 0.99$ computes much less conservative outer-approximations than the latter. Thus, SDP (17) also improves the performance of SDP $\mathbb{O}$.

IV. Experiment

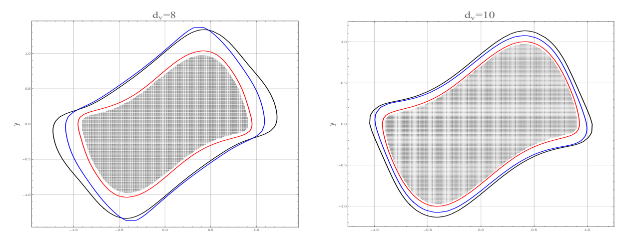
In this section we demonstrate our approach based on SDP (17) on several examples and compare it with SDP $\mathbb{O}$. All computations were performed on an i7-10875H 2.30GHz CPU with 16GB RAM running Windows 10. The sum-of-squares module in YALMIP[12] and semi-definite programming solver Mosek [17] are used to solve (17) and SDP $\mathbb{O}$. The related parameters are presented in Table I.

A. Van der Pol oscillator

Consider the system in Example 1 again, where polynomials $v(\boldsymbol{x})$ of degree 8 and 10 are employed to further demonstrate advantages of our method based on solving SDP (17). All the computed results with $\alpha = 1$ and 0.99 are visualized in Fig. 3. The results show that all outer-approximations from our method are less conservative. Also, the computed outer-approximations from SDP (17) with $\alpha = 0.99$ match $\widehat{\mathcal{R}}$ estimated via the Monte-Carlo simuation method well, especially when polynomials of degree 10 are used. However, the results from SDP $\mathbb{O}$ do not.

Ex.	α	d_v	d_w	d_s	d_{s_1}
1	0.99	8	8	14	16
1	1	8	8	14	16
1	0.99	10	10	16	20
1	1	10	10	16	20
2	0.99	10	10	10	10
2	1	10	10	10	10
3	0.99	6	6	12	14
3	1	6	6	12	14
3	0.99	8	8	14	16
3	1	8	8	14	16

TABLE I
PARAMETERS OF THE IMPLEMENTATIONS ON SOLVING (17) FOR EXAMPLES IV-A-IV-C.
α : THE FACTOR IN (17); d_v AND d_w: DEGREE OF POLYNOMIALS v AND w IN (17); d_s : DEGREE OF POLYNOMIALS s_i IN (17), $i = 0, 2, 3, 4, 5$; d_{s_1}: DEGREE OF THE POLYNOMIAL s_1 IN (17).



(a) (b)

Fig. 3. An illustration of outer-approximating $\mathcal{R}$ for Example IV-A. Black, blue and red curves denote the boundaries of outer-approximations from SDP $\mathbb{O}$ and (17) with $\alpha = 1$ and $\alpha = 0.99$, respectively. Gray region corresponds to $\mathcal{R}$ estimated by the Monte-Carlo simulation method.

B. Double integrator

Consider a double integrator from [8] subject to control inputs with a discrete time $\delta t = 0.1$,

$$
\begin{cases} x(l+1) = x(l) + 0.1 y(l) \\ y(l+1) = y(l) + 0.1 u(l) \end{cases}
$$

with $\mathcal{X} = \{(x, y)^\top \mid x^2 + y^2 - 1^2 \leq 0\}$, $\mathcal{T} = \{(x, y)^\top \mid x^2 + y^2 - 0.05^2 < 0\}$ and $u \in [-0.1, 0.1]$.

The computed results are showcased in Fig. 4, which also show that the outer-approximations from SDP (17) are less conservative than SDP $\mathbb{O}$.

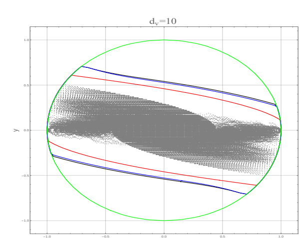
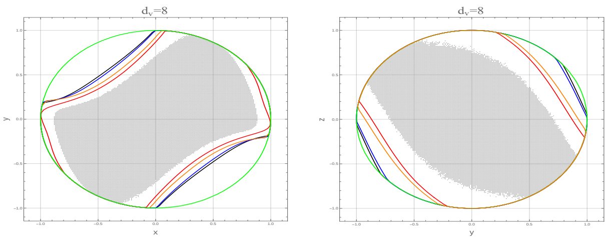


Fig. 4. An illustration of outer-approximating $\mathcal{R}$ for Example IV-B. Green curve denotes the boundary of $\mathcal{X}$. Black, blue and red curves denote the boundaries of outer-approximations computed from SDP $\mathbb{O}$ and SDP (17) with $\alpha = 1$ and $\alpha = 0.99$, respectively. Gray region corresponds to $\mathcal{R}$ estimated by the Monte-Carlo simulation method.



(a) (b)

Fig. 5. An illustration of outer-approximating $\mathcal{R}$ on the planes $z = 0$ (a) and $x = 0$ (b) for Example IV-C. Green curve denotes the boundary of $\mathcal{X}$. Black, blue and red curves respectively denote the boundaries of outer-approximations computed from SDP $\mathbb{O}$ and SDP (17) with $\alpha = 1$ and $\alpha = 0.99$, when $d_v = 8$; orange curve is the boundary of computed outer-approximation from SDP (17) with $\alpha = 0.99$ when $d_v = 6$. Gray region corresponds to $\mathcal{R}$ estimated by the Monte-Carlo simulation method.

C. *Controlled 3D Van der Pol oscillator*

Consider a controlled 3D Van der Pol oscillator from [11] with a discrete time $\delta t = 0.01$,

$$\begin{cases} x(l+1) = & x(l) + 0.01(-2y(l)) \\ y(l+1) = & y(l) + 0.01(0.8x(l) - 2.1y(l) \\ & +z(l) + 10x^2(l)y(l)) \\ z(l+1) = & y(l) + 0.01(-z(l) + z^3(l) + 0.5u(l)) \end{cases}$$

with $\mathcal{X} = \{(x, y, z)^\top \mid x^2 + y^2 + z^2 - 1^2 \leq 0\}$, $\mathcal{T} = \{(x, y, z)^\top \mid x^2 + y^2 + z^2 - 0.1^2 < 0\}$ and $u \in [-1, 1]$.

We first used polynomials $v(\boldsymbol{x})$ of degree 6 to solve all SDPs. However, we didn't obtain outer-approximations when solving SDP $\mathbb{O}$ and SDP (17) with $\alpha = 1$. An outer-approximation can be obtained by solving SDP (17) with $\alpha = 0.99$, which is shown in Fig. 5. Then, we used polynomials $v(\boldsymbol{x})$ of degree 8 to solve them again. The computed results are also demonstrated in Fig. 5, which show that all outer-approximations from our method are less conservative than the one from the method in [8]. Especially, it is observed that the outer-approximation computed via solving SDP (17) with $\alpha = 0.99$ and $d_v = 6$ is less conservative than the one from SDP $\mathbb{O}$ with $d_v = 8$, which shows that our method is able to obtain tighter outer-approximations with polynomials of lower degree than the method in [8] for some systems.

V. Conclusion

In this paper we proposed a semi-definite programming method for outer-approximating controlled reach-avoid sets of discrete-time polynomial systems. With a discounted function, a Bellman type equation was derived for the first time such that the strict zero super-level set of its unique bounded solution is equal to the exact controlled reach-avoid set. The proposed method was built upon the combination of a set of inequalities, which is constructed by relaxing the Bellman type equation, and a set of inequalities from [8]. Compared with the semi-definite programming approach in [8], the semi-definite program in the present work is subject to weaker constraints with more solutions and thus can provide tighter outer-approximations for some systems, which are further justified on several examples.

References

[1] K. J. Åström and B. Wittenmark. *Computer-controlled systems: theory and design*. Courier Corporation, 2013.

[2] O. Bokanowski, N. Forcadel, and H. Zidani. Reachability and minimal times for state constrained nonlinear problems without any controllability assumption. *SIAM J. Control Optim.*, 48(7):4292–4316, 2010.

[3] T. Bräunl. *Embedded Robotics: Mobile Robot Design and Applications with Embedded systems*. Springer Science & Business Media, 2008.

[4] C. Fan, Z. Qin, U. Mathur, Q. Ning, S. Mitra, and M. Viswanathan. Controller synthesis for linear system with reach-avoid specifications. *IEEE Trans. Automat. Contr.*, 2021.

[5] J. F. Fisac, M. Chen, C. J. Tomlin, and S. S. Sastry. Reach-avoid problems with time-varying dynamics, targets and constraints. In *HSCC'15*, pages 11–20, 2015.

[6] J. F. Fisac and S. S. Sastry. The pursuit-evasion-defense differential game in dynamic constrained environments. In *CDC'15*, pages 4549–4556. IEEE, 2015.

[7] J. H. Gillula and C. J. Tomlin. Guaranteed safe online learning via reachability: tracking a ground target using a quadrotor. In *ICRA'12*, pages 2723–2730. IEEE, 2012.

[8] W. Han and R. Tedrake. Controller synthesis for discrete-time polynomial systems via occupation measures. In *IROS'18*, pages 6911–6918. IEEE, 2018.

[9] D. Henrion and M. Korda. Convex computation of the region of attraction of polynomial control systems. *IEEE Trans. Automat. Contr.*, 59(2):297–312, 2013.

[10] M. Korda, D. Henrion, and C. N. Jones. Inner approximations of the region of attraction for polynomial dynamical systems. *IFAC Proceedings Volumes*, 46(23):534–539, 2013.

[11] M. Korda, D. Henrion, and C. N. Jones. Controller design and region of attraction estimation for nonlinear dynamical systems. *IFAC Proceedings Volumes*, 47(3):2310–2316, 2014.

[12] J. Lofberg. Yalmip: A toolbox for modeling and optimization in matlab. In *ICRA'04*, pages 284–289. IEEE, 2004.

[13] A. Majumdar, R. Vasudevan, M. M. Tobenkin, and R. Tedrake. Convex optimization of nonlinear feedback controllers via occupation measures. *The International Journal of Robotics Research*, 33(9):1209–1230, 2014.

[14] K. Margellos and J. Lygeros. Hamilton–jacobi formulation for reach–avoid differential games. *IEEE Trans. Automat. Contr.*, 56(8):1849–1861, 2011.

[15] K. Margellos and J. Lygeros. Toward 4-d trajectory management in air traffic control: a study based on monte carlo simulation and reachability analysis. *IEEE Trans. Control Syst. Technol.*, 21(5):1820–1833, 2012.

[16] I. Mitchell and C. J. Tomlin. Level set methods for computation in hybrid systems. In *HSCC'00*, pages 310–323. Springer, 2000.

[17] A. Mosek. The mosek optimization toolbox for matlab manual, 2015.

[18] V. Shia, R. Vasudevan, R. Bajcsy, and R. Tedrake. Convex computation of the reachable set for controlled polynomial hybrid systems. In *CDC'14*, pages 1499–1506. IEEE, 2014.

[19] Y. Wang, C. Huang, Z. Wang, Z. Wang, and Q. Zhu. Verification in the loop: Correct-by-construction control learning with reach-avoid guarantees. *arXiv preprint arXiv:2106.03245*, 2021.

[20] B. C. Williams, M. D. Ingham, S. H. Chung, and P. H. Elliott. Model-based programming of intelligent embedded systems and robotic space explorers. *Proceedings of the IEEE*, 91(1):212–237, 2003.

[21] B. Xue, A. Easwaran, N.-J. Cho, and M. Fränzle. Reach-avoid verification for nonlinear systems based on boundary analysis. *IEEE Trans. Automat. Contr.*, 62(7):3518–3523, 2016.

[22] B. Xue, M. Fränzle, and N. Zhan. Inner-approximating reachable sets for polynomial systems with time-varying uncertainties. *IEEE Trans. Automat. Contr.*, 65(4):1468–1483, 2020.

[23] B. Xue, N. Zhan, and M. Fränzle. Inner-approximating reach-avoid sets for discrete-time polynomial systems. In *CDC'20*, pages 867–873. IEEE, 2020.

[24] P. Zhao, S. Mohan, and R. Vasudevan. Control synthesis for nonlinear optimal control via convex relaxations. In *ACC'17*, pages 2654–2661. IEEE, 2017.