Automated Synthesis of Low-rank Control Systems from sc-LTL Specifications using Tensor-Train Decompositions

John Irvin Alora, Alex Gorodetsky, Sertac Karaman, Youssef Marzouk, Nathan Lowry

***Abstract*— Correct-by-design automated construction of control systems has attracted a tremendous amount of attention. However, most existing algorithms for automated construction suffer from the *curse of dimensionality*, *i.e.*, their run time scales exponentially with increasing dimensionality of the state space. As a result, typically, systems with only a few degrees of freedom are considered. In this paper, we propose a novel algorithm based on the tensor-train decomposition that solves stochastic optimal control problems with syntactically co-safe linear temporal logic specifications. We show that, under certain conditions, the run time of the proposed algorithm scales polynomially with the dimensionality of the state space and the rank of the optimal cost-to-go function. We demonstrate the algorithm in a six-dimensional problem instance involving a simple airplane model. In this example, the proposed algorithm provides up to four orders of computational savings when compared to the standard value iteration algorithm.**

I. INTRODUCTION

Correct-by-design automated construction of control systems have attracted a tremendous amount of attention in recent years, leading to the development of algorithms that allow for systematic and provably-correct control design. In particular, this line of work has had a tremendous impact in motion and task planning problems, where automatic construction of robot control policies subject to high-level task specifications is of primary interest. These problems require reasoning with both (discrete) task specifications and continuous robot motions, which poses significant computational challenges. We refer the reader to surveys [5], [6], [7] for more information on existing work in this direction.

The seminal papers by Tabuada and Pappas [2] as well as Kloetzer and Belta [1] analyze linear systems and synthesize controllers by constructing a discrete abstraction of the state space that captures all essential properties that can be represented in the specification language. More recent work has focused on the application of similar controller synthesis methods in robot motion planning problems where the tasks are specified using the Linear Temporal Logic (LTL) language [3], [4]. In the motion planning context, a transition graph (or tree) that abstracts the system dynamics is generated incrementally using a sampling based algorithm and is then augmented with the finite state automata (which encodes the LTL specifications). The resulting product transition graph then represents possible poses of the robotic system and has the additional property that the sequence of pose transitions eventually satisfy the specification. Stochastic systems have also been considered in this context. For instance, Markov Decision Process (MDP) abstractions of systems (generated through simulation) have been considered [8], [9]. Recently, stochastic optimal control problems with temporal logic specifications have also been considered [11]. Synthesis of controllers for a certain class of continuous-time stochastic dynamical systems have been proposed [10].

Unfortunately, existing algorithms are either restricted to simple systems, e.g. linear dynamical systems, or they are intractable, e.g., the running time scales exponentially with increasing dimensionality of the state space.

The primary contribution of this paper is an algorithm for automated synthesis of control systems for stochastic dynamical systems from their syntactically co-safe linear temporal logic specifications. From a technical perspective, we extend our prior work [14] to automated synthesis problems. In our prior work, an algorithm was described for solving an MDP resulting from the discretization of a stochastic optimal control problem. This algorithm worked on the basis of replacing the cost-to-go function within dynamic programming algorithm with a compressed version. In the present paper, we apply the same technique to stochastic optimal control problems with sc-LTL specifications. The key idea is to first construct a discrete product structure that represents both the discrete abstraction of the high-level task and the continuous dynamics of the system, and then represent the stochastic cost function in the tensor-train format and execute value iteration in this form. Doing so allows us to exploit the *low rank* structure commonly found in separable functions, and construct an algorithm that has *polynomial complexity* in the dimensionality of the state space and in tensor rank. Our work leverages effective tensor decomposition algorithms, which were proposed only very recently. Specifically, the tensor-train decomposition method has been shown to be effective in several applications domains [23], [26].

We demonstrate our tensor-based value iteration algorithm on two numerical experiments. First, we run the proposed algorithm on a 3-dimensional Dubin's car and compare its performance with naïve value iteration. We then demonstrate the algorithm on an airplane model with a six-dimensional state space. In this example, the proposed algorithm provides up to four orders of computational savings when compared to a naïve implementation of the value iteration algorithm.

This paper is organized as follows. In Section II, we present a formal description of the problem. In Section III we introduce background material on consistent discretizations

This work was supported in part by the Draper Laboratory and in part by the National Science Foundation through grant #1452019.

J.I. Alora, A. Gorodetsy and S. Karaman, Youssef Marzouk are with Department of Aeronautics and Astronautics, Massachusetts Institute of Technology. {jalora,goroda,sertac,ymarz}@mit.edu. N. Lowry is with Draper Laboratory, Cambridge, MA. nlowry@draper.com.

for stochastic optimal control problems and tensor decomposition methods. In Section IV, we describe the proposed algorithm. We present the results of our simulations in Section V. Finally, we conclude with remarks in Section VI.

II. PROBLEM DEFINITION

This section is devoted to a formal problem definition. First, we describe stochastic control systems and the syntactically co-safe linear temporal logics (sc-LTL) in Section II-A. Subsequently, we provide a formal problem definition of the stochastic optimal control problem with sc-LTL specifications in Section II-B.

A. System Model and Specifications

The sets of integers and reals are denoted by $\mathbb{Z}$ and $\mathbb{R}$. Their non-negative counterparts are denoted by $\mathbb{Z}_+$ and $\mathbb{R}_+$. A probability space is denoted by $(\Omega, \mathscr{F}, \mathbb{P})$, where Ω is a sample space, $\mathscr{F}$ is a σ-algebra, and $\mathbb{P}$ is a probability measure. The expectation operator is denoted by $\mathbb{E}[\cdot]$.

1) System dynamics: Let d, d_w, $d_u \in \mathbb{Z}_+$. Consider continuous-time continuous-space stochastic dynamical systems of the following differential form:

$$dx(t) = b(x(t), u(t))dt + F(x(t), u(t))dw(t), \qquad (1)$$

where $b : X \times U \to \mathbb{R}^d$ denotes the drift vector, $F : X \times U \to \mathbb{R}^{d \times d_w}$ denotes the diffusion matrix, and $X \subset \mathbb{R}^d$ and $U \subset \mathbb{R}^{d_u}$ are compact sets with smooth boundaries and non-empty interiors. The functions b and F are assumed to be measurable, continuous, and bounded functions. The stochastic process $\{w(t) : t \geq 0\}$ is the d_w-dimensional Brownian motion defined on the probability space $(\Omega, \mathscr{F}, \mathbb{P})$.

The evolution of the state is an X-valued stochastic process $\{x(t) : t \geq 0\}$. We denote a realization of this state process for a given sample path ω, by $x[\omega]$, which itself is a mapping from time into X, *i.e.*, $x[\omega] : \mathbb{R}_+ \to X$.

We use atomic propositions to describe various properties of the states of the system. Let AP be a finite set of atomic propositions. Let $L : X \to 2^{AP}$ be a labeling function that maps each state to the atomic propositions that hold for that state. When an atomic proposition p holds for state x, *i.e.*, $p \in L(z)$, where $z \in X$, the atomic proposition p is said to be `True` at state z; otherwise, p is said to be `False` at z. Finally, let $[[p]]$ denote the set of all $z \in X$ for which p holds.

2) Syntactically co-safe linear temporal logic: In this paper, syntactically co-safe linear temporal logic (sc-LTL) [13] is used to specify desired system behavior. The set of sc-LTL formulas reason only about finite runs. That is, properties of infinite runs cannot be represented in sc-LTL.

An sc-LTL formula is composed from the boolean operators $\neg$ (negation), $\vee$ (disjunction), $\wedge$ (conjunction) and the temporal operators $\mathcal{U}$ (until), and $\Diamond$ (eventually).

Definition 1: The *syntax of syntactically co-safe LTL (sc-LTL) formulas* over a finite set of propositions is defined inductively in the Backus-Naur form as

$$\varphi ::= p \,|\, \neg p \,|\, \varphi_1 \vee \varphi_2 \,|\, \varphi_1 \wedge \varphi_2 \,|\, \varphi_1 \mathcal{U} \varphi_2 \,|\, \Diamond \varphi,$$

where p is an atomic proposition, *i.e.*, $p \in AP$.

The sc-LTL language differs from classical logic with temporal operators, namely 'until' ($\mathcal{U}$) and 'eventually' ($\Diamond$). The formula $\varphi_1 \mathcal{U} \varphi_2$ states that φ_1 is true, until φ_2 becomes true. The formula $\Diamond \varphi$ states that φ eventually becomes true.

The sc-LTL language formulas can be represented by deterministic finite automata. Let us formalize this connection.

Definition 2: A *Deterministic Finite Automaton (DFA)* is a 5-tuple $\mathcal{A} = (Q, \Sigma, \delta, q_0, \mathbb{F})$ where Q is a finite set of states, Σ is the input alphabet, $\delta : Q \times \Sigma \to Q$ is the transition function, $q_0 \in Q$ is the initial state, and $\mathbb{F} \subseteq Q$ is the set of accepting states.

A *word* over an alphabet Σ is a sequence $w = (\pi_0, \pi_1, \ldots, \pi_{k-1})$ such that $\pi_i \in \Sigma$ for all $i \in \{0, 1, \ldots, k-1\}$. For each word $w = (\pi_0, \pi_1, \ldots, \pi_{k-1})$ over the input alphabet Σ, the corresponding *run* on automata $\mathcal{A} = (Q, \Sigma, \delta, q_0, \mathbb{F})$ is a sequence $\sigma = (q_0, q_1, \ldots, q_k)$ of states, *i.e.*, $q_i \in Q$ for all $i \in \{0, 1, \ldots, k\}$, such that *(i)* q_0 is the initial state, *(ii)* $q_{i+1} = \delta(q_i, \pi_i)$ for all $i \in \{0, 1, \ldots, k-1\}$. A run $\sigma = (q_0, q_1, \ldots, q_k)$ is said to be an *accepting run* of automaton $\mathcal{A}$, if it ends in an accepting state, *i.e.*, $q_k \in \mathbb{F}$. The set of all words that correspond to accepting runs is called the *language* generated by the automaton $\mathcal{A}$.

For any sc-LTL formula φ defined on the set AP of atomic propositions, there exists a deterministic finite automaton $\mathcal{A}_\varphi$ with input alphabet $\Sigma = 2^{AP}$ such that the language generated by $\mathcal{A}$ is precisely the language generated by φ [13].

B. Stochastic Optimal Control with sc-LTL Specifications

We are interested in reasoning about the trajectories of a stochastic system using the sc-LTL language. For this purpose, below, we define a *product system* of a stochastic dynamical system (see Equation (1)) with state space X and a finite automaton $\mathcal{A}$ (see Definition 2) with states Q. Let us define the set of all product states as $X_\times = X \times Q$.

1) The product system: For a given $x[\omega]$, let $(t_0, t_1, t_2, \ldots, t_k)$ denote the increasing sequence of time instances for which $L(x[\omega](t_i)) \neq L(x[\omega](t_i - \epsilon))$ for all small enough $\epsilon > 0$. That is, t_i is the ith time instance that the trajectory $x[\omega](t)$ is about to cross into a region where a different set of atomic propositions hold. Define $t_0 := 0$, and define $\pi_i = L(x[\omega](t_i))$ for all $i \in \{0, 1, \ldots, k-1\}$. The sequence $w_{x[\omega]} = (\pi_0, \pi_1, \ldots, \pi_{k-1})$ is called the *word* generated by the state trajectory $x[\omega]$. Let $\sigma_{x[\omega]} = (q_0, q_1, \ldots, q_k)$ denote the run generated by $w_{x[\omega]}$ on automaton $\mathcal{A}$. Finally, the *state process of the product system* is denoted by $\{s(t) : t \geq 0\}$, where $s(t) \in S$, and defined as follows: For any given sample path $\omega \in \Omega$,

$$s[\omega](t) = \big(\, x[\omega](t), q_i \,\big),$$

for all $t \in [t_{i-1}, t_i)$ and all $i \in \{1, 2, \ldots, k\}$.

In a nutshell, the state process $\{s(t) : t \geq 0\}$ of the product system encompasses the state process of the continuous-time continuous-state stochastic dynamic system along with the state evolution of the discrete automaton. Notice that the

states of the automaton evolve according to the atomic propositions that are satisfied along the trajectories of the stochastic dynamic system. For any given sc-LTL formula, we can use the corresponding automaton $\mathcal{A}_\varphi = (Q, \Sigma, \delta, q_0, \mathbb{F})$ in the product system, and be able to understand whether the trajectories of the continuous system satisfy the formula φ.

2) Feedback control policies: A *control policy* is a mapping $\mu : X_\times \to U$ that assigns a control input to each product state. The *product process under the influence of policy* μ, denoted by $\{s_\mu(t) : t \geq 0\}$, is obtained by setting the input $u(t) = \mu(s_\mu(t))$ for all $t \in \mathbb{R}_+$ in Equation (1).

The *first entry time* for policy μ is defined as the the first time that the product process hits the boundary of $X_\times$, *i.e.*,

$$T_\mu := \inf\{t : s_\mu(t) \in \partial X_\times\},$$

where $\partial X_\times$ is the boundary of $X_\times$, *i.e.*, $\partial X_\times = \partial X \times \mathbb{F}$.

The *expected cost-to-go function under policy* μ is a mapping $J_\mu : X_\times \to \mathbb{R}$:

$$J_\mu(s_0) = \mathbb{E}\left[\int_0^{T_\mu} g(s_\mu(t))dt + h(s_\mu(T_\mu)) \,\middle|\, s_\mu(0) = s_0\right], \tag{2}$$

where $g : X_\times \to \mathbb{R}_+$ and $h : \partial X_\times \to \mathbb{R}$ are the stage cost function and terminal cost function, respectively. The *optimal cost-to-go function* maps each $s \in X_\times$ to the minimum cost-to-go at s over the set of all proper policies, *i.e.*,

$$J^*(s_0) = \inf_\mu J_\mu(s_0), \qquad \text{for all } s_0 \in X_\times.$$

An *optimal policy* μ^* is one that achieves the optimal cost to go function, *i.e.*, $J_{\mu^*}(s) = J^*(s)$ for all $s \in X_\times$.

We are interested in the following problem:

Problem 1: Given the following:

- (X, U, b, F): a continuous-time continuous-space stochastic dynamical system,
- AP: a set of atomic propositions;
- L: a labeling function that maps each state $z \in X$ to the set of all atomic propositions that hold at z;
- φ: an sc-LTL formula over AP,
- (g, h): a pair of stage cost and terminal cost functions.

compute an optimal policy μ^*.

III. PRELIMINARIES

In this section, we first introduce the Markov chain approximation (MCA) method for discretizing a continuous-time continuous-space stochastic optimal control problem into a discrete MDP. Then, we introduce low-rank tensor decompositions that compress multidimensional arrays in a computationally feasible manner. The discrete state MDP resulting from the MCA method can then be solved using the low-rank, tensor-based, dynamic programming techniques [14].

A. Consistent discretizations in stochastic optimal control

Recall, from Section II-A, that a continuous-time stochastic dynamical system is described by the 4-tuple (X, U, b, F). We now describe consistency conditions that enable a discrete state MDP to approximate this dynamical system with an error proportional to the discretization level and allow a sequence of grid refinements to allow the discrete MDP dynamics to converge to the continuous stochastic dynamics.

Let $h^l > 0$ be a sequence of real numbers, such that $\lim_{l\to\infty} h^l = 0$. Let $X^l \subset X$ denote the finite set of states contained in a regular grid with node spacing h^l. Consider a sequence of MDPs $M^l = (X^l, U, P^l, \mathcal{F}^l, G^l, H^l)$ indexed by $l \in \mathbb{N}$, where X^l is a set of states, U is a set of control actions, $P^l : X^l \times U \times X^l \to [0, 1]$ is the transition probability function, $\mathcal{F}^l \subseteq X^l$ is a set of terminal states, G^l is the stage cost, and H^l is the terminal cost. Let $\{\xi_i^l : i \in \mathbb{N}\}$ denote states encountered through the evolution the MDP dynamics. We define a *sequence of holding times* as a sequence of functions $\{\Delta t^l : l \in \mathbb{R}_+\}$, where $\Delta t^l : X^l \times U \to \mathbb{R}_+$ for all $l \in \mathbb{N}$. The sequence of holding times allow us to generate continuous-time state evolution from the discrete-time state evolution of the MDPs. More precisely, given the state evolution of a sequence of MDPs, *i.e.*, $\{\xi_i^l : i \in \mathbb{N}\}$, and a sequence of holding times, *i.e.*, $\{\Delta t^l : l \in \mathbb{N}\}$, we construct a sequence of continuous-time trajectories, denoted by $\{\xi^l : l \in \mathbb{N}\}$, as follows:

$$\xi^l(\tau) = \xi_n^l, \text{ for all } \tau \in [t_n^l, t_n^l + \Delta t^l(\xi_n^l, u(t_n^l))),$$

where $t_n^l = \sum_{i=1}^{n-1} \Delta t^l(\xi_i^l, u(t_i^l))$. Notice that, for each $l \in \mathbb{N}$, we have that $\xi^l(t) \in X^l$ and $\{\xi^l(t) : t \geq 0\}$ is a stochastic process on its own right.

A classical result by Kushner and coworkers guarantees the consistency of discretizations, if the a set of conditions, often called the local consistency conditions, are satisfied.

Definition 3 (Local Consistency Conditions): A sequence of MDPs $M^l = (X^l, U, P^l, \mathcal{F}^l, G^l, H^l)$ and a sequence of holding times $\{\Delta t^l : l \in \mathbb{N}\}$, both indexed by $l \in \mathbb{N}$, are said to be *locally consistent* with the continuous-time stochastic dynamics of Equation (1) if the following are satisfied: (i) $\lim_{l\to\infty} \sup_{i\in\mathbb{N},\, \omega\in\Omega} \|\xi_{i+1}^l - \xi_i^l\|_2 = 0$, and (ii) for all $z \in X^l, u \in U$:

$$\lim_{l\to\infty} \triangle t^l(z, u) = 0.$$

$$\lim_{l\to\infty} \frac{\mathbb{E}_{P^l}[\xi_{i+1}^l - \xi_i^l \,|\, \xi_i^l = z, u_i^l = u]}{\triangle t^l(z, u)} = b(z, u),$$

$$\lim_{l\to\infty} \frac{\mathrm{Cov}_{P^l}[\xi_{i+1}^l - \xi_i^l \,|\, \xi_i^l = z, u_i^l = u]}{\triangle t^l(z, u)} = F(z, u)F(z, u)^T,$$

A sequence of MDPs and holding times are said to be *consistent*, if they satisfy the local consistency conditions. Once a consistent discretization is obtained, we solve the the discrete MDPs by minimizing:

$$J^l(z) = \min_\mu \mathbb{E}\left[\sum_{i=1}^{N^l} G(\xi_i^l, u(t_i^l)) + H^l(\xi_{N^l}^l)\right], \tag{3}$$

where N^l is the first time the state hits $\mathcal{F}^l$, *i.e.*, $N^l = \inf\{i : \xi_i^l \in \mathcal{F}^l\}$. Then, a seminal result by Kushner et al. guarantees that these solutions converge to the solution of Problem 1.

Theorem 1 (Kushner et al. [12]): Suppose the sequence of MDPs M^l and the sequence of holding times $\{\Delta t^l : l \in \mathbb{N}\}$, both indexed by $l \in \mathbb{N}$, are locally consistent with the

continuous-time stochastic dynamics of Equation (1). Then, the stochastic process $\{\xi^l(t) : t \geq 0\}$ converges to the state process $\{x(t) : t \geq 0\}$ as l tends to infinity. Furthermore, the optimal cost-to-go function J^l for the MDPs M^l converges to the optimal-cost-to go function J^* of the continuous-time stochastic optimal control problem, *i.e.*,

$$\lim_{l\to\infty} \|J^l - J^*\| = 0.$$

In Section IV, we provide a specific consistent discretization for our product system. Next, we describe the compressed representation of the discretized value function J^l that enables us to solve high-dimensional problems.

B. Tensor decompositions

The complexity of representing the cost-to-go function J^l in Equation (3) grows exponentially with the dimensionality d of the state space. This complexity gives rise to the *curse of dimensionality* when employing conventional solution methods based on value iteration or policy iteration. In this paper, we mitigate this curse of dimensionality through low-rank compression. In particular, we only perform operations on cost functions represented in a low-rank compressed format, removing the need for storing the entire grid of cost-to-go values. Since the cost-to-go function can be considered as a tensor, defined as a multidimensional array, we can use the recent developments for tensor decompositions for compression. These tensor decomposition algorithms allow us to compress a low-rank tensor using an interpolation scheme, which has complexity scaling polynomially with the rank of the tensor and linearly with the dimensionality of the state space. In this section, we describe the tensor representations and tensor decomposition algorithms that are central to the control synthesis algorithms proposed in this paper.

Separation of variables is the underlying property that yields low-rank representation of tensors. Specifically, tensor decompositions represent a multidimensional function by sums of separable functions. For example, the representation of a multidimensional function $f : \mathbb{R}^d \to \mathbb{R}$ that has rank R in the canonical decomposition format is

$$f(x_1, \ldots, x_d) = \sum_{r=1}^{R} f_r^{(1)}(x_1) \cdot f_r^{(2)}(x_2) \cdots f_r^{(d)}(x_d).$$

where $f_r^{(k)}$ are one dimensional functions of the kth variable.

Although the canonical decomposition maintains linear complexity with dimensionality, determining the canonical rank R is an NP-hard problem [21], and finding a best rank R approximation of f is ill-posed [22]. Another representation, known as the Tucker decomposition, allows for easy computation of a low rank approximation, but has exponential complexity with dimension. The tensor-train (TT) decomposition [23], on the other hand, maintains the advantages of both representations and the algorithms for its computation have strong guarantees. Hence, we leverage the TT decomposition and present algorithms in the context of this tensor representation.

Let us consider a tensor with each dimension i discretized into the set $\mathcal{X}_i = \{x_i[1], \ldots, x_i[n_i]\}$ of $n_i \in \mathbb{Z}_+$ points. Let $\mathbf{F} : \mathcal{X}_1 \times \ldots \times \mathcal{X}_d \to \mathbb{R}$ be a tensor whose elements are function evaluations of the function f on the tensor product grid, *i.e.*, $\mathbf{F}[i_1, \ldots, i_d] = f(x_1[i_1], \ldots, x_d[i_d])$. The elements of a tensor in TT format can be computed as

$$\mathbf{F}[i_1, i_2, \ldots, i_d] = \sum_{\alpha_0=1}^{r_0} \sum_{\alpha_1=1}^{r_0} \cdots \sum_{\alpha_d=1}^{r_d} \mathbf{F}_1[\alpha_0, i_1, \alpha_1] \cdots \mathbf{F}_d[\alpha_{d-1}, i_d, \alpha_d],$$

or as a sequence of vector-matrix products

$$\mathbf{F}[i_1, i_2, \ldots, i_d] = \mathbf{F}_1[i_1, :]\mathbf{F}_1[i_2, :] \cdots \mathbf{F}_d[:, i_d], \quad (4)$$

where the three dimensional arrays $\mathbf{F}_k \in \mathbb{R}^{r_{k-1} \times n_k \times r_k}$ are called the TT-cores and r_k are called the TT-ranks with $r_0 = r_d = 1$. Thus, to store a tensor in TT-format only requires storing each of its cores, and if we assume constant ranks, the storage cost becomes $\mathcal{O}(dnr^2)$.

The TT-ranks are bounded above by the ranks of each unfolding matrix of the tensor, *i.e.* r_k is guaranteed to not be higher than the rank of the unfolding matrix $\mathbf{F}^k[i_1, \ldots, i_k; i_{k+1}, \ldots, i_d]$. The proof of this statement is constructive [23], and provides an algorithm for decomposing a tensor into its TT representation through a sequence of singular value decompositions (SVD). The algorithm allows for the computation of an approximation $\tilde{\mathbf{F}}$ to $\mathbf{F}$ with an accuracy ϵ such that $\|\tilde{\mathbf{F}} - \mathbf{F}\|_F \leq \epsilon \|\mathbf{F}\|_F$.

However, the SVD based algorithm requires evaluating all of the elements in $\mathbf{F}$, making it computationally impractical for large-scale tensors with many dimensions [23]. Thus, we will use the interpolation algorithm [24] that replaces the SVD with the CUR/Skeleton decomposition to restrict the number of elements of $\mathbf{F}$ that are evaluated.

In two dimensions, this algorithm can be realized as seeking the skeleton decomposition of a matrix A, denoted as $A = A[:, C]A[I, C]^\dagger A[C, :]$, where I with $|I| \geq r$ is a set of rows and C with $|C| \geq r$ is a set of columns in A. This decomposition requires access to only certain rows and columns of the matrix A; similarly, the higher dimensional analogue requires access to only certain fibers of the tensor.

The multidimensional interpolation method, called `TT-cross`, achieves ϵ-level accuracy using the Maxvol algorithm [25] to choose the necessary fibers for evaluation. However, this algorithm requires specification of the upper bounds to each r_k. If r_k is set too low, excessive approximation errors may result; if it is too high, the computational effort increases. We utilize a rank-adaptive version of the algorithm [26]. We denote this rank-adaptive version as `TT-cross` throughout the rest of this paper.

C. Low-rank dynamic programming algorithms

In this section, we present the tensor-based value iteration algorithm proposed in [14] for solving the discrete MDP resulting from the MCA algorithm.

Value iteration is a fixed-point algorithm that starts with an initial cost-to-go function J_0^l and generates a sequence of

cost-to-go functions $\{J_k^l\}$ for $k = 1, 2, \ldots$ according to the Bellman equation:

$$J_{k+1}^l(z) \leftarrow \min_u \Big[G^l(z,u) + \sum_{z' \in X^l} P^l(z' \mid z, u) J_k^l(z) \Big], \quad (5)$$

for $z, z' \in X^l$, where $P^l(z' \mid z, u)$ denotes the probability of transitioning from state z to state z'. Under certain technical conditions [12], [15], this sequence converges $J_k^l \to J^l$ to the optimal cost-to-go function in Equation (3). We assume these conditions hold.

The tensor-based value iteration algorithm interpolates the right-hand-side of Equation (5) using the `TT-cross` algorithm. This algorithm generates a sequence of approximate cost-to-go functions $\{\hat{J}_k^l\}$, and is given in Algorithm 1.

Algorithm 1 Tensor-based Value Iteration (TVI) (See Algorithm 1 in [14])

Require: Termination criteria δ_{max}; `TT-cross` accuracy ϵ; Initial cost-to-go function $\hat{J}_0^l$
Ensure: Residual : $\delta \leftarrow \|\hat{J}_{k+1}^l - \hat{J}_k^l\|_2 < \delta_{max}$
1: $\delta \leftarrow \delta_{max} + 1$
2: $k \leftarrow 0$
3: **while** $\delta > \delta_{max}$ **do**
4: $\quad \hat{J}_{k+1}^l \leftarrow$ `TT-cross`(Equation (5), ϵ)
5: $\quad k \leftarrow k + 1$
6: $\quad \delta \leftarrow \|\hat{J}_k^l - \hat{J}_{k-1}^l\|_2$
7: **end while**

If this algorithm can guarantee an approximation with relative accuracy of ϵ during each iteration, and if the norm of each cost-to-go function $\hat{J}_k^l$ is bounded by ρ, then one can guarantee the following error

$$\lim_{k \to \infty} \|\hat{J}_k^l - J^l\| \leq \frac{\epsilon \rho}{1 - \gamma}.$$

IV. Proposed algorithm and analysis

In this section, we propose an algorithm to solve the stochastic optimal control problem with sc-LTL specifications. We first describe how to generate a consistent discretization of the product system. Then, we describe how the tensor construction of this discretized product system is used within the value iteration algorithm.

A. Discretized MDP for the product system

The discretization of the product system involves two steps: first, the stochastic dynamics in Equation (1) are discretized using an upwind discretization [14], [12]; second, the resulting discretized state space is augmented with the automaton modes.

The upwind discretization proceeds as follows. Let $e_1, \ldots, e_d$ be unit vectors in $\mathbb{R}^d$, assume F is a diagonal matrix, and let the discretization be a uniform grid. A locally consistent discretization with $h = h^l$ is defined by

$$b_i^{\pm}(z,u) = \frac{1}{2}|b_i(z,u)| \pm \frac{1}{2} b_i(z,u),$$
$$Q_n(z,u) = 2\mathrm{Tr}(F^2(z,u)) + h \sum_i |b_i(z,u)|,$$
$$P^l(z \pm e_i h | z, u) = \frac{F_{ii}^2/2 + h b_i^{\pm}(z,u)}{Q_n(z,u)},$$
$$\triangle t_h = \frac{h^2}{Q_n(z,u)},\ G(z,u) = \triangle t_h \cdot g(z,u),$$

for $z \in X^l$, where the probability transitions $P^l(z \pm e_i h | z, u)$ denote transitioning from state z to a neighboring state in the uniform grid $z \pm e_i h$ under the influence of control u.

Next, we create the MDP for the product system. In this paper, we are interested in solving for the *minimum expected satisfaction time* of the specification, φ. In the stochastic shortest path (SSP) framework, we are equivalently computing the *minimum reachability time* [16] to the terminal absorbing set $\mathcal{F}_\times^l = \mathcal{F}^l \times \mathbb{F}$, which is the product of the MDP terminal states and the automaton accepting states.

The goal is to compute a policy μ^* that minimizes the expected time of reaching the target set, $\mathcal{F}_\times^l$ from a product state $s \in X_\times^l$, where $X_\times^l = X^l \times Q$ is the product state space. Hence (with a slight abuse of notation), Problem 1 equates to computing a policy that minimizes a cost function, $J^l : X_\times^l \to \mathbb{R}$, that corresponds to the expected time. The product stage cost that corresponds to the minimum time problem is then $G_\times^l : X_\times^l \to \{0, 1\}$, with $G_\times(s) = 1$ when the system is not in a terminal state and $G_\times(s) = 0$ otherwise. Furthermore, in the minimum time problem the terminal cost for the product MDP is set to $H_\times^l(s_t, u) = 0$. Under these conditions the expected reach time is

$$J_\mu(s_0) = \mathbb{E}\Big[\int_0^{T_\mu} G_\times(s_\mu(t)) dt \mid s(0) = s_0 \Big].$$

To complete the specification of the product MDP $M_\times^l = (X_\times^l, U, P_\times^l, \mathcal{F}_\times^l, G_\times^l, H_\times^l)$, we specify the transition probability for a control $u \in U$ according to

$$P_\times^l((z', q') | (z, q), u) = \begin{cases} P^l(z' | z, u) & \text{if } q' = \delta(q, \pi) \\ 0 & \text{otherwise} \end{cases},$$

for $z, z' \in X^l$ and $q, q' \in Q$, where $\pi = L(z)$.

Once the discretization of the product MDP is obtained, the resulting value function J^l is represented in tensor-train format. In particular, this value function is a tensor $J^l \in \mathbb{R}^{n_1 \times \cdots \times n_d \times |Q|}$, where $n_i \in \mathbb{Z}_+$ denotes the number of nodes that the ith state is discretized into and $|Q|$ is the number of automaton states. This means that the element $J^l[i_1, \ldots, i_d, i_A]$refers to value of the cost-to-go function at the the discretized node $x^l = (x_1[i_1], \ldots, x_d[i_d])$, and the i_Ath automaton state.

From Equation (4) we infer that in tensor-train format this element can be computed as

$$J^l[i_1, i_2, \ldots, i_d, i_A] = \mathbf{F}_1[i_1, :]\mathbf{F}_1[i_2, :] \cdots \mathbf{F}_d[:, i_d, :]\mathbf{F}_A[:, i_d, :],$$

where we now have a $d+1$ dimensional tensor with $r_{d+1} = 1$, and the new core $\mathbf{F}_A \in \mathbb{R}^{r_d \times r_{d+1}}$. In this case r_d now

intuitively conveys the degree of separability between the automaton states and the states of the dynamical system. With this setup, we can now use Algorithm 1 to solve the product MDP. The storage cost for the value function now becomes $\mathcal{O}(dnr^2)$, where $n = n_i = |Q|$ for $i = 1, \ldots, d$.

B. Analysis

Fu and Topcu [11] analyzed the convergence of dynamic programming algorithms for stochastic optimal control problems with MTL specifications. In this section, we show the convergence of the discretized product MDP obtained through the MCA algorithm to the continuous stochastic optimal control problem with sc-LTL specifications. The following theorem is an analogue to Theorem 1 for the discrete, non-product, MDP.

Theorem 2: Given the sequence of product MDPs $M_\times^l$ and a sampling interval dt that satisfy the local consistency conditions, the discrete approximation of the cost-to-go $J^l(s)$ converges to the continuous-time cost-to-go $J(s)$ for all $s \in S$, *i.e.*, $\lim_{l\to\infty} J^l(s) = J(s)$.

Proof: (Sketch) We provide a sketch of the proof, and leave the full proof to a full version of the paper. Since the discretization of the state space is locally consistent, then by Theorem 1 the state space portion of the product state in the trajectory $\{s_i^l : i \in \mathbb{N}\} = \{(\xi_i^l, q_i) : i \in \mathbb{N}\}$ converges in distribution, *i.e.*, $\{\xi_i^l\}$ converges in distribution, and thus solves the SDE in Equation (1). The key observation is that, since $\mathcal{A}_\varphi$ is deterministic, then the chain $\{s_i^l : i \in \mathbb{N}\}$ also converges to $\{s(t) : t \geq 0\}$. The same argument is made for the product MDP value function. As $l \to \infty$, $\{\xi_i^l\} \to x(t))$ implies $\{s_i^l\} \to s(t)$ and therefore, J^l converges to J. ■

Using this proof, a straightforward conclusion can be made that the approximation results for Algorithm 1 also hold for the product system. Therefore, we can use the same algorithm and maintain the same convergence guarantees.

V. SIMULATIONS

This section demonstrates the proposed algorithm on two problem instances. The first problem instance involves a three-dimensional Dubin's vehicle, in which we compare the solutions obtained from standard value iteration and the proposed algorithm. The second problem instance involves a six-dimensional extension to model an airplane.

All computations were run on a desktop computer with a 3.6 GHz Intel Core i7 processor and 12 GB of RAM.

A. Three-dimensional Dubin's Vehicle

Consider a stochastic Dubin's vehicle with dynamics:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} v\cos\theta \\ v\sin\theta \\ u \end{pmatrix} dt + \mathbf{D}dw(t), \tag{6}$$

where the states $\mathbf{x} = (x, y, \theta)$ are the x, y position and heading angle, respectively, $u \in [-1, 1]$ is the set of inputs, $v = 1$ is the fixed velocity, $\mathbf{D} = 10^{-3} I_3$ is the diffusion diagonal matrix, and $w(\cdot)$ is a 3-D Brownian motion. The state space is bounded such that $x, y \in [0, 10]$ and $\theta \in [0, 2\pi]$.

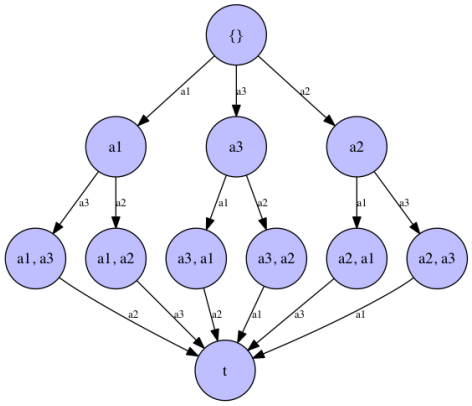


Fig. 1: DFA for eventually visiting three regions, characterized by the formula $\varphi = \Diamond A_1 \wedge \Diamond A_2 \wedge \Diamond A_3$.

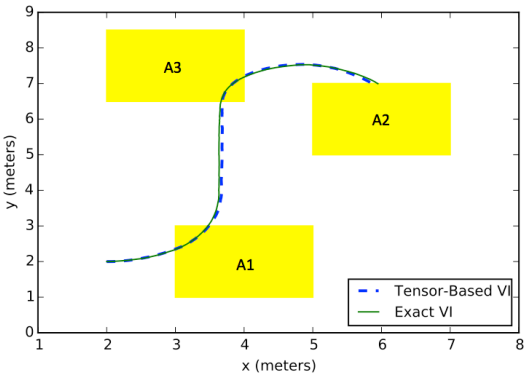
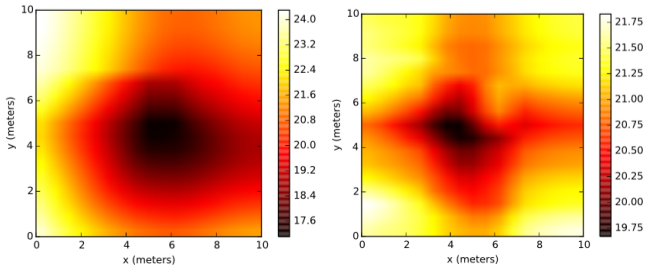


Fig. 2: Comparison of a simulated sample trajectory, $x[\omega]$ using interpolated control policy found from tensor-based value iteration and exact value iteration for a three dimensional Dubin's Car. The parameters used for the tensor-based approach were $h = 0.125$ and $\epsilon = 10^{-2}$.

We seek to compute a feedback policy μ that minimizes the expected time of visiting the regions $A_1, A_2, A_3 \subset X$ once in any order, for any initial state $\mathbf{x}_0 \in X$. This high level behavior can be written as an sc-LTL formula, $\varphi = \Diamond A_1 \wedge \Diamond A_2 \wedge \Diamond A_3$. The regions A_1, A_2, and A_3 are defined as atomic propositions, such that they evaluate to `True` whenever the car is within the respective region. In this example, the target regions are 2×2 squares in (x, y), where $[[A_1]] = [3, 5] \times [1, 3]$, $[[A_2]] = [5, 7] \times [5, 7]$, and $[[A_3]] = [2, 4] \times [6.5, 8.5]$. A uniform spatial step of h is chosen in order to obtain a grid discretization of X. The discretization scheme given in Section III-A is used to construct a locally consistent MDP. The sc-LTL specification, φ, is translated into a DFA, $\mathcal{A}_\varphi$, which is shown in Figure 1 using a standard model checking tool [13]. Given M^l and $\mathcal{A}_\varphi$, an approximating product MDP $M_\times^l$ is constructed. In this case, each dimension is discretized into $n = \mathcal{O}(\frac{1}{h})$ nodes along with $m = |Q|$ modes in $\mathcal{A}_\varphi$, thus the total number of discretized states is $mn^3 = \mathcal{O}(\frac{m}{h^3})$. For this example,



(a) Cost Function obtained from Exact Value Iteration (b) Cost Function obtained from Tensor-Based Value Iteration

Fig. 3: Comparison of the cost functions obtained from (a) exact VI and (b) tensor-based VI at the initial state $x_0 = (x, y, 0)^T$, at initial state of $\mathcal{A}_\varphi$.

$m = 11$.

We solved this problem instance using both the standard value iteration algorithm and the proposed tensor-based value iteration algorithm (see Algorithm 1). Tensor-based value iteration is run based on the TT-cross interpolation scheme for various discretization levels h and error tolerances ϵ.

First, we compare the trajectories generated by the two methods for the starting point $\mathbf{x}_0 = (2, 2, 0)$. Figure 2 shows the trajectories obtained from the controller obtained using tensor-based value iteration algorithm with error tolerance $\epsilon = 10^{-2}$ and that obtained using standard value iteration with stopping criteria $\|\hat{J}^l_{k+1} - \hat{J}^l_k\| \leq 10^{-5}$, both are discretized with $h = 0.125$. We obtain a maximum average error of $\max_t[\max[[x - x^*], [y - y^*], [\theta - \theta^*]]] = 9.3 \times 10^{-2}$ across all variables, where x, y, θ denote the mean position and heading computed with the tensor-based value iteration algorithm, and x^*, y^*, θ^* denote the mean states of the standard value iteration solution, for the same resolution.

Figure 3 compares the cost functions obtained from the standard value iteration algorithm and the proposed tensor-based value iteration algorithm. While tensor-based value iteration evaluates only a fraction of the state space, it is able to converge to a close approximation of J^l. Indeed, we obtain a relative difference of 7.2×10^{-6} between the exact and approximate solution. On average the number of states visited by tensor-based value iteration was approximately $16,000$ states, compared to the $54,000$ number of states required for standard value iteration.

For $h = .06125$ (and $m = 11$), the number of states evaluated by standard value iteration is approximately $390,000$, whereas tensor-based value iteration with $\epsilon = 10^{-2}$ evaluates on approximately $39,000$ states on average per iteration.

B. Six Dimensional Dubin's Plane

We now demonstrate tensor-based value iteration on a six-dimensional stochastic optimal control problem involving a simple model of an airplane considered in [20]. We consider a manuever that requires the aircraft to visit two target regions in minimum expected time.

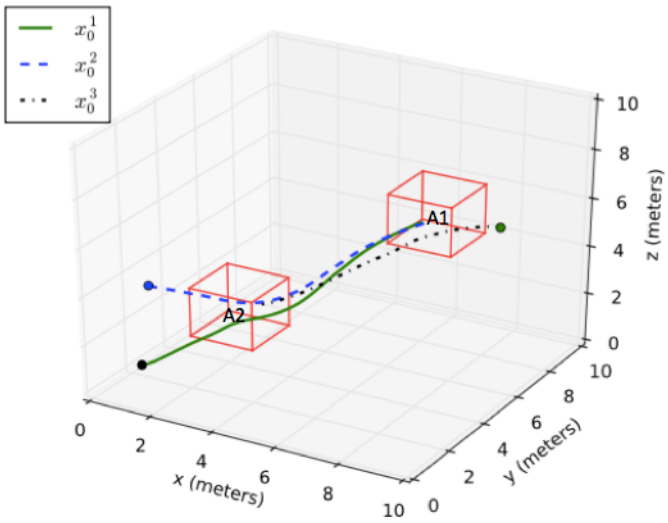


Fig. 4: Simulated sample trajectory, $x[\omega]$, starting at $\mathbf{x}_0^1 = (1, 1, 1, 0, 0, 0)$, $\mathbf{x}_0^2 = (0, 3, 3, 0, 0, 1)$, $\mathbf{x}_0^3 = (9, 7, 6, \pi, 0, 1)$ under the feedback policy found from tensor-based value iteration with $n = 32$, $m = 4$, and $\epsilon = 10^{-2}$.

Consider the following extension of the Dubin's vehicle:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\theta} \\ \dot{\gamma} \\ \dot{v} \end{pmatrix} = \begin{pmatrix} v\cos\theta\cos\gamma \\ v\sin\theta\cos\gamma \\ v\sin\gamma \\ \frac{u_1}{R_{min}} \\ u_2 \\ u_3 \end{pmatrix} dt + \mathbf{D}dw(t) \tag{7}$$

The plane moves in three dimensional space with the six state variables $(x, y, z, \theta, \gamma, v)$ specifying x-position, y-position, and z-position, heading, flight path angle, and velocity. u_1 controls the rate of change of θ whereas $u_2 = \dot{\gamma}$ and $u_3 = \dot{v}$ control the rate of change of the flight path angle and vehicle acceleration, respectively. We introduce uncertainty with the diffusion diagonal matrix $\mathbf{D} = 10^{-3} I_6$, where $w(\cdot)$ is a six-dimensional Wiener process. The state space is bounded such that $x, y, z \in [0, 10]$, $\theta \in [0, 2\pi]$, $\gamma \in [-\pi/4, \pi/4]$, $\gamma \in [0, 4]$.

The problem is to compute a feedback policy μ for any state such that the plane visits regions $[[A_1]] = [6, 6, 5] \times [8, 8, 7]$ and $[[A_2]] = [2, 2, 2] \times [4, 4, 4]$ in minimum expected time. This planning problem is expressed by the sc-LTL formula $\varphi = \Diamond A_1 \wedge \Diamond A_2$. Again, we obtain a consistent discretization of the dynamics using Algorithm 1 and uniformly discretize each dimension into 32 states. Our specification φ is translated into a DFA with 4 modes, resulting in an overall discrete product state space of 4.3×10^9 number of states.

The proposed algorithm successfully computes a feedback control for this problem. Figure 4 shows the state trajectories for multiple simulations, starting at various initial states. Figure 5 shows the average fraction of states evaluated per value iteration. This figure also reveals that the cost function J_h is indeed "low rank," since only a small fraction of the states are evaluated for the various discretization levels. For example, when $n = 8$, roughly 5×10^4 of the 10^6 states

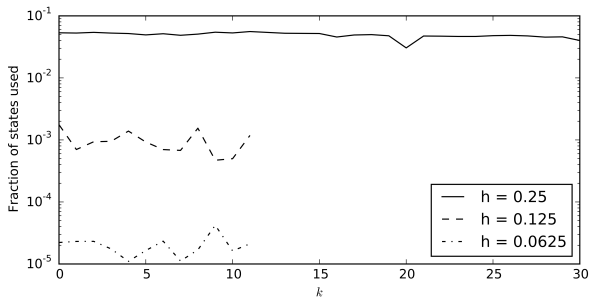


Fig. 5: Fraction of states evaluated by Tensor-based VI for various discretizations h at $\epsilon = 10^{-2}$.

are evaluated. When $n = 32$, roughly 10^5 of the 4.3×10^9 states are used. Thus, we observe computational savings of 2 to 4 orders of magnitudes, when the proposed algorithm is compared to standard value iteration algorithm.

VI. CONCLUSIONS

In this paper, we considered the problem of stochastic optimal control with syntactically co-safe linear temporal logic specifications. While all existing algorithms that guarantee optimality run in time that scales *exponentially* with increasing dimensionality of the state space, we presented an algorithm with running time that scales *linearly* with increasing dimensionality of the state space and polynomially with the rank of the optimal cost-to-go function. We demonstrated our algorithm on a challenging example involving an airplane with a six-dimensional state space. Future work will consider experimentally evaluating our approach on aerial robots and the application of the proposed algorithm to networked control systems such as power grids.

REFERENCES

[1] Kloetzer, Marius, and Calin Belta. "A fully automated framework for control of linear systems from LTL specifications." HSCC. Vol. 3927. 2006.

[2] Tabuada, Paulo, and George J. Pappas. "Linear time logic control of discrete-time linear systems." Automatic Control, IEEE Transactions on 51.12 (2006): 1862-1877.

[3] Varricchio, Valerio, Pratik Chaudhari, and Emilio Frazzoli. "Sampling-based algorithms for optimal motion planning using process algebra specifications." Robotics and Automation (ICRA), 2014 IEEE International Conference on. IEEE, 2014.

[4] Karaman, Sertac, and Emilio Frazzoli, "Sampling-based motion planning with deterministic mu-calculus specifications." Decision and Control (CDC), 2009 IEEE Conference on. IEEE, 2009.

[5] Belta, Calin, et al. "Symbolic Control and Planning of Robotic Motion [Grand Challenges of Robotics]." (2007).

[6] Kress-Gazit, Hadas, Tichakorn Wongpiromsarn, and Ufuk Topcu. "Mitigating the State Explosion Problem of Temporal Logic Synthesis." IEEE Robotics and Automation Magazine (2011).

[7] Plaku, Erion and Sertac Karaman. "Motion planning with temporal-logic specifications: Progress and challenges." AI Communications, vol. 29, no. 1, pp. 151-162, 2015.

[8] Ding, Xuchu, et al. "Optimal control of Markov decision processes with linear temporal logic constraints." Automatic Control, IEEE Transactions on 59.5 (2014): 1244-1257.

[9] Lahijanian, Morteza, et al. "Motion planning and control from temporal logic specifications with probabilistic satisfaction guarantees." Robotics and Automation (ICRA), 2010 IEEE International Conference on. IEEE, 2010.

[10] Horowitz, Matanya B., Eric M. Wolff, and Richard M. Murray. "A compositional approach to stochastic optimal control with co-safe temporal logic specifications." Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on. IEEE, 2014.

[11] Fu, Jie, and Ufuk Topcu. "Computational methods for stochastic control with metric interval temporal logic specifications." arXiv preprint arXiv:1503.07193 (2015).

[12] Kushner, Harold, and Paul G. Dupuis. Numerical methods for stochastic control problems in continuous time. Vol. 24. Springer Science and Business Media, 2013.

[13] Latvala, Timo. "Efficient model checking of safety properties." Model Checking Software. Springer Berlin Heidelberg, 2003. 74-88.

[14] Gorodetsky, Alex, Sertac Karaman, and Youssef Marzouk. "Efficient high-dimensional stochastic optimal motion control using tensor-train decomposition." Robotics: Science and Systems, Rome, Italy (2015).

[15] Bertsekas, Dimitri P. "Dynamic programming and optimal control 3rd edition, volume II." Belmont, MA: Athena Scientific (2011).

[16] De Alfaro, Luca. Computing minimum and maximum reachability times in probabilistic systems. Springer Berlin Heidelberg, 1999.

[17] Horowitz, Matanya B., Anil Damle, and Joel W. Burdick. "Linear Hamilton Jacobi Bellman equations in high dimensions." Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on. IEEE, 2014.

[18] Hansen, Eric A. "Suboptimality bounds for stochastic shortest path problems." arXiv preprint arXiv:1202.3729 (2012).

[19] Zhang, Nevin Lianwen, and Weihong Zhang. "Speeding up the convergence of value iteration in partially observable Markov decision processes." Journal of Artificial Intelligence Research (2001): 29-51.

[20] Bakolas, Efstathios, Yiming Zhao, and Panagiotis Tsiotras. "Initial guess generation for aircraft landing trajectory optimization." AIAA Guidance, Navigation, and Control Conference. Reston, VA, 2011.

[21] De Silva, Vin, and Lek-Heng Lim. "Tensor rank and the ill-posedness of the best low-rank approximation problem." SIAM Journal on Matrix Analysis and Applications 30.3 (2008): 1084-1127.

[22] Hastad, Johan. "Tensor rank is NP-complete." Journal of Algorithms 11.4 (1990): 644-654.

[23] Oseledets, Ivan V. "Tensor-train decomposition." SIAM Journal on Scientific Computing 33.5 (2011): 2295-2317.

[24] Oseledets, Ivan, and Eugene Tyrtyshnikov. "TT-cross approximation for multidimensional arrays." Linear Algebra and its Applications 432.1 (2010): 70-88.

[25] Goreinov, S. A., et al. "How to find a good submatrix." Matrix methods: theory, algorithms and applications (2010): 247-256.

[26] Savostyanov, Dmitry, and Ivan Oseledets. "Fast adaptive interpolation of multi-dimensional arrays in tensor train format." Multidimensional (nD) Systems (nDs), 2011 7th International Workshop on. IEEE, 2011.