Optimal control in Markov decision processes via distributed optimization

Jie Fu, Shuo Han and Ufuk Topcu

Abstract— **Optimal control synthesis in stochastic systems with respect to quantitative temporal logic constraints can be formulated as linear programming problems. However, centralized synthesis algorithms do not scale to many practical systems. To tackle this issue, we propose a decomposition-based distributed synthesis algorithm. By decomposing a large-scale stochastic system modeled as a Markov decision process into a collection of interacting sub-systems, the original control problem is formulated as a linear programming problem with a sparse constraint matrix, which can be solved through distributed optimization methods. Additionally, we propose a decomposition algorithm which automatically exploits, if it exists, the modular structure in a given large-scale system. We illustrate the proposed methods through robotic motion planning examples.**

I. Introduction

For many systems, temporal logic formulas are used to describe desirable system properties such as safety, stability, and liveness [1]. Given a stochastic system modeled as a Markov decision process (MDP), the synthesis problem is to find a policy that achieves optimal performance under a given quantitative criterion regarding given temporal logic formulas. For instance, the objective may be to find a policy that maximizes the probability of satisfying a given temporal logic formula. In such a problem, we need to keep track of the evolution of state variables that capture system dynamics as well as predicate variables that encode properties associated with the temporal logic constraints [2], [3]. As the number of states grows exponentially in the number of variables, we often encounter large MDPs, for which the synthesis problems are impractical to solve with centralized methods. The insight for control synthesis of large-scale systems is to exploit the modular structure in a system so that we can solve the original problem by solving a set of small subproblems.

In literature, distributed control synthesis methods are proposed in the pioneering work for MDPs with discounted rewards [4], [5]. The authors formulate a two-stage distributed reinforcement learning method: The first stage constructs and solves an abstract problem derived from the original one, and the second stage iteratively computes parameters for local problems until the collection of local problems' solutions converge to one that solves the original problem. Recently, alternating direction method of multipliers (ADMM) is combined with a sub-gradient method into planning for average-reward problems in large MDPs in [6]. However, the method in [6] applies only when some special conditions are satisfied on the costs and transition kernels. Alternatively, hierarchical reinforcement learning introduces *action-aggregation* and *action-hierarchies* to address the planning problems with large MDPs [7]. In action-aggregation, a micro-action is a local policy for a subset of states and the global optimal policy maps histories of states into micro-actions. However, it is not always clear how to define the action hierarchies and how the choice of hierarchies affects the optimality in the global policy. Additionally, the aforementioned methods are in general difficult to implement and cannot handle temporal logic specifications.

For synthesis problems in MDPs with quantitative temporal logic constraints, centralized methods and tools [3], [8] are developed and applied to control design of stochastic systems and robotic motion planning [9]–[11]. Since centralized algorithms are based on either value iteration or linear programming, they inevitably hit the barrier of scalability and are not viable for large MDPs. In this paper, we develop a distributed optimization method for large MDPs subject to temporal logic contraints. We first introduce a decomposition method for large MDPs and prove a property in such a decomposition that supports the application of the proposed distributed optimization. For a subclass of MDPs whose graph structures are Planar graphs[1], we introduce an efficient decomposition algorithm that exploits the modular structure for the underlying MDP caused by loose coupling between subsets of states and its constituting components. Then, given a decomposition of the original system, we employ a distributed optimization method called *block splitting algorithm* [12] to solve the planning problem with respect to discounted-reward objectives in large MDPs and average-reward objectives in large ergodic MDPs [2]. Comparing to two-stage methods in [5], [6], [13], our method concurrently solves the set of sub-problems and penalizes solutions' mismatches in one step during each iteration, and is easy to implement. Since the distributed control synthesis is independent from the way how a large MDP is decomposed, any decomposition method can be used. Lastly, we extend

J. Fu, S. Han are with the Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA 19104, USA, jief, hanshuo@seas.upenn.edu.

U. Topcu is with the Department of Aerospace Engineering and Engineering Mechanics, the University of Texas at Austin, Austin, TX, 78712, USA, utopcu@utexas.edu.

This work is supported by AFRL # FA8650-15-C-2546, ONR # N000141310778, NSF # 1550212, NSF # 1558404. Shuo Han was supported in part by the NSF (CNS-1239224) and TerraSwarm, one of six centers of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA.

[1] As an example, gridworld MDPs have Planar graph structure, as the robot in a gridworld only transits to its adjacent cells.

[2] A Markov chain is ergodic if it is possible to eventually get from every state to every other state with positive probability. An MDP is ergodic if the Markov chain induced from any policy is ergodic.

the method to solve the synthesis problems for MDPs with two classes of quantative temporal logic objectives. Through case studies we investigate the performance and effectiveness of the proposed method.

II. Preliminaries

Let Σ be a finite set. Let Σ^*, Σ^ω be the set of finite and infinite words over Σ. $\mathsf{card}(\Sigma)$ is the cardinality of the set Σ. A probability distribution on a finite set S is a function $D : S \to [0,1]$ such that $\sum_{s\in S} D(s) = 1$. The support of D is the set $\mathsf{Supp}(D) = \{s \in S \mid D(s) > 0\}$. The set of probability distributions on a finite set S is denoted $\mathcal{D}(S)$.

Markov decision process: A *Markov decision process* (MDP) $M = \langle S, A, u_0, P\rangle$ consists of a finite set S of states, a finite set A of actions, an initial distribution $u_0 \in \mathcal{D}(S)$ of states, and a transition probability function $P : S \times A \to \mathcal{D}(S)$ that for a state $s \in S$ and an action $a \in A$ gives the probability $P(s,a)(s')$ of the next state s'. Given an MDP M we define the set of actions *enabled* at state s as $A(s) = \{a \in A \mid \exists s' \in S, P(s,a)(s') > 0\}$. The cardinality of the set $\{(s,a) \mid s \in S, a \in A(s)\}$ is the number of *state-action pairs* in the MDP.

A *path* is an infinite sequence $s_0 s_1 \ldots$ of states such that for all $i \geq 0$, there exists $a \in A$, $s_{i+1} \in \mathsf{Supp}(P(s_i,a))$. A *policy* is a function $f : S^* \to \mathcal{D}(A)$ that, given a finite state sequence representing the history, chooses a probability distribution over the set A of actions. Policy f is memoryless if it only depends on the current state, i.e., $f : S \to \mathcal{D}(A)$. Once a policy f is chosen, an MDP M is reduced to a Markov chain, denoted M^f. We denote by X_i and θ_i the random variables for the i-th state and the i-th action in this chain M^f. Given a policy f, for a measurable function ϕ that maps paths into reals, let $E^f_{u_0}[\phi]$ (resp. $E^f_s[\phi]$) be the expected value of ϕ when the policy f is used given u_0 being the initial distribution of states (resp. s being the initial state).

Given an MDP M, a reward function $R : S \times A \to \mathbb{R}$ and a policy f, let γ be a discounting factor, the *discounted-reward value* is defined as $\mathsf{Val}^f_\gamma(u_0) = \mathsf{Val}^f_\gamma(s) \cdot u_0(s)$ where $\mathsf{Val}^f_\gamma(s) = E^f_s(\sum_{n=0}^\infty \gamma^n R(X_n, \theta_n))$; the *average-reward value* is defined as $\mathsf{Val}^f(u_0) = \mathsf{Val}^f(s) \cdot u_0(s)$ where $\mathsf{Val}^f(s) = \lim_{n\to\infty} \frac{1}{n} E^f_s\left[\sum_{k=0}^n R(X_k, A_k)\right]$. A discounted-reward (resp. an average-reward) problem is, for a given initial state distribution, to obtain a policy that maximizes the discounted-reward value (resp. average-reward value). For discounted-reward (average-reward) problems, the optimal value can be attained by memoryless policies [14].

A solution to the discounted-reward problem can be found by solving the linear programming (LP) problem:

$$\max_{x\in\mathbb{R}^m_+} \sum_{s\in S}\sum_{a\in A(v)} x(s,a)\cdot R(s,a) \tag{1a}$$

subject to

$$\sum_{a\in A(s)} x(s,a) - \gamma \cdot \sum_{s'\in S}\sum_{a'\in A(s')} x(s',a')\cdot P(s',a')(s) = u_0(s), \forall s \in S, \tag{1b}$$

where m is the total number of state-action pairs in the MDP, $\mathbb{R}^m_+$ is the non-negative orthant of $\mathbb{R}^m$, and variable $x(s,a)$ can be interpreted as the expected discounted time of being in state s and taking action a. Once the LP problem in (1) is solved, the optimal policy is obtained as $f(s,a) = \frac{x(s,a)}{\sum_{a'\in A(s)} x(s,a')}$ and the objective function's value is the optimal discounted-reward value under policy f given the initial distribution u_0 of states.

In an ergodic MDP, the average-reward value is a constant regardless of the initial state distribution [15]. We obtain an optimal policy for an average-reward problem by solving the LP problem

$$\max_{x\in\mathbb{R}^m_+} \sum_{s\in S}\sum_{a\in A(s)} x(s,a)\cdot R(s,a) \tag{2a}$$

subject to

$$\sum_{a\in A(s)} x(s,a) - \sum_{s'\in S}\sum_{a'\in A(s')} x(s',a')\cdot P(s',a')(s) = 0, \forall s \in S, \tag{2b}$$

$$\sum_{s\in S}\sum_{a\in A(s)} x(s,a) = 1, \tag{2c}$$

where $x(s,a)$ is understood as the long-run fraction of time that the system is at state s and the action a is taken. Once the LP problem in (2) is solved, the optimal policy is obtained as $f(s,a) = \frac{x(s,a)}{\sum_{a'\in A(s)} x(s,a')}$. The optimal objective value is the optimal average-reward value and is the same for all states.

Distributed optimization: As a prelude to the distributed synthesis method developed in section IV, now we describe the *alternating direction method of multipliers* (ADMM) [16] for the generic convex constrained minimization problem $\min_{z\in\mathbf{C}} g(z)$ where function g is closed proper convex and set $\mathbf{C}$ is closed nonempty convex. In iteration k of the ADMM algorithm the following updates are performed:

$$z^{k+1/2} := \mathbf{prox}_g(z^k - \tilde{z}^k), \tag{3a}$$

$$z^{k+1} := \Pi_{\mathbf{C}}(z^{k+1/2} + \tilde{z}^k), \tag{3b}$$

$$\tilde{z}^{k+1} := \tilde{z}^k + z^{k+1/2} - z^{k+1}, \tag{3c}$$

where $z^{k+1/2}$ and $\tilde{z}^k$ are auxiliary variables, $\Pi_{\mathbf{C}}$ is the (Euclidean) projection onto $\mathbf{C}$, and $\mathbf{prox}_g(v) = \arg\min_x \left(g(x) + (\rho/2)\|x - v\|_2^2\right)$ is the *proximal operator* of g with parameter $\rho > 0$. The algorithm handles separately the objective function g in (3a) and the constraint set $\mathbf{C}$ in (3b). In (3c) the dual update step coordinates these two steps and results in convergence.

Temporal logic: Linear temporal logic (LTL) formulas are defined by: $\phi := p \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \bigcirc\phi \mid \phi_1 \mathcal{U} \phi_2$, where $p \in \mathcal{AP}$ is an atomic proposition, and $\bigcirc$ and $\mathcal{U}$ are temporal modal operators for "next" and "until". Additional temporal logic operators are derived from basic ones: $\Diamond\varphi := \mathsf{true}\,\mathcal{U}\varphi$ (eventually) and $\Box\varphi := \neg\Diamond\neg\varphi$. Given an MDP M, let $\mathcal{AP}$ be a finite set of atomic propositions, and a function $L : S \to 2^{\mathcal{AP}}$ be a labeling function that assigns a set of atomic propositions $L(s) \subseteq \mathcal{AP}$ to each state $s \in S$ that are valid at the state s. L can be extended to paths in the usual way, i.e.,

$L(s\rho) = L(s)L(\rho)$ for $s \in S, \rho \in S^\omega$. A path $\rho = s_0 s_1 \ldots \in S^\omega$ satisfies a temporal logic formula φ if and only if $L(\rho)$ satisfies φ. In an MDP M, a policy f induces a probability distribution over paths in S^ω. The probability of satisfying an LTL formula φ is the sum of probabilities of all paths that satisfy φ in the induced Markov chain M^f.

Problem 1: Given an MDP M and an LTL formula φ, synthesize a policy that optimizes a quantitative performance measure with respect to the formula φ in the MDP M.

One quantitative performance measure we study is the probability of satisfying a temporal logic formula. We also consider the expected frequency of satisfying certain recurrent properties specified in an LTL formula.

By formulating a product MDP with the original MDP and an automaton representing the temporal logic specification (see details in Section V), it can be shown that Problem 1 with different quantitative performance measures can be formulated through pre-processing as special cases of discounted-reward and average-reward problems [3], [17]. Thus, in the following, we first introduce decomposition-based distributed synthesis methods for large MDPs with discounted-reward and average-reward criteria. Then, we show the extension for solving MDPs with quantitative temporal logic constraints.

III. Decomposition of an MDP

A. Decomposition and its property

To exploit the modular structure of a given MDP, the initial step is to decompose the state space into small subsets of states, each of which can then be related to a small problem. In this section, we introduce some terminologies in decomposition of MDPs from [5].

Given an MDP $M = \langle S, A, u_0, P\rangle$, let Π be any partition of the state set S. That is, $\Pi = \{S_1, \ldots, S_N\} \subseteq 2^S$, $\emptyset \notin \Pi$, $S_i \cap S_j = \emptyset$ when $i \neq j$ and $\bigcup_{i=1}^N S_i = S$. A set in Π is called a *region*. The *periphery* of a region S_i is a set of states *outside* S_i, each of which can be reached with a non-zero probability by taking some action from a state *in* S_i. Formally, $\mathsf{Periphery}(S_i) = \{s' \in S \setminus S_i \mid \exists (s,a) \in S_i \times A, P(s,a)(s') > 0\}$.

Let $K_0 = \bigcup_{j=1}^N \mathsf{Periphery}(S_j)$. Given a region $S_i \in \Pi$, we call $K_i = S_i \setminus K_0$ the *kernel* of S_i. We denote m_i the number of state-action pairs restricted to K_i, for each $i = 0, \ldots, N$. That is, m_i is the cardinality of the set $\{(s,a) \mid s \in K_i, a \in A(s)\}$. We call the partition $\{K_i \mid 0 \leq i \leq N\}$ a *decomposition* of M. The following property of a decomposition is exploited in distributed optimization.

Lemma 1: Given a decomposition $\{K_i, i = 0, 1, \ldots, N\}$ obtained from partition $\Pi = \{S_1, \ldots, S_N\}$, for a state $s \in K_i$ where $i \neq 0$, if there is a state s' and an action a such that $P(s', a)(s) \neq 0$, then either $s' \in K_0$ or $s' \in K_i$.

Proof: Suppose $s' \notin K_0$ and $s' \notin K_i$, then it must be the case that $s' \in K_j$ for some $j \neq 0$ and $j \neq i$. Since from state $s' \in S_j$, after taking action a, the probability of reaching $s \in S_i$ is non-zero, we can conclude that $s \in \mathsf{Periphery}(S_j)$, which implies $s \in K_0$. The implication contradicts the fact that $s \in K_i$ since $K_0 \cap K_i = \emptyset$. Hence, either $s' \in K_i$ or $s' \in K_0$. ■

Example: Consider the MDP in Figure 1, which is taken from [18]. The shaded region shows a partition $\Pi = \{S_1 = \{s_4, s_5, s_6\}, S_2 = \{s_0, s_1, s_2, s_3, s_7\}\}$ of the state space. Then, $\mathsf{Periphery}(S_1) = \{s_2, s_7\}$ and $\mathsf{Periphery}(S_2) = \{s_4\}$. We obtain a decomposition of M as $K_0 = \cup_{i=1}^2 \mathsf{Periphery}(S_i) = \{s_2, s_4, s_7\}$, $K_1 = S_1 \setminus K_0 = \{s_5, s_6\}$, and $K_2 = S_2 \setminus K_0 = \{s_0, s_1, s_3\}$. It is observed that state s_5 can only be reached with non-zero probabilities by actions taken from states s_4 and s_6.

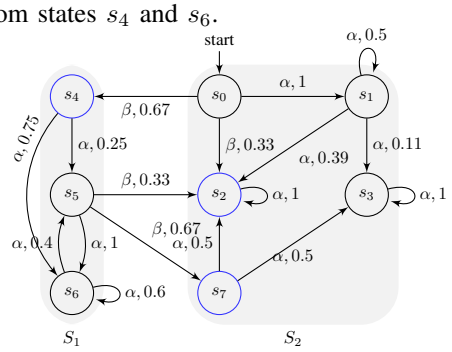


Fig. 1: Example of an MDP with states $Q = \{s_i, i = 0, \ldots, 7\}$, actions $A = \{\alpha, \beta\}$, and transition probability function P as indicated.

B. A decomposition method for a subclass of MDPs

Various methods have been developed to derive a decomposition of an MDP, for example, decompositions based on partitioning the state space of an MDP according to the communicating classes in the induced graph (defined in the following) of that MDP (see a survey in [13]). For the distributed synthesis method developed in this paper, it will be shown later in Section IV that the number of state-action pairs and the number of states in K_i are the number of variables and the number of constraints in a sub-problem, respectively. Thus, we prefer a decomposition that meets one simple desirable property: For each $i = 0, 1, \ldots, N$, the number m_i of state-action pairs in K_i is small in the sense that the classical linear programming algorithm can solve an MDP with state-action pairs of this size in a reasonable amount of time given the computational capacity of a general computer. Next, we propose a method that generates decompositions which meet the aforementioned desirable property for a subclass of MDPs. For an MDP in this subclass, its induced graph is *Planar* [3]. It can be shown that MDPs derived from classical gridworld examples, which have many practical applications in robotic motion planning, are in this subclass.

We start by relating an MDP with a directed graph.

Definition 1: The *labeled digraph* induced from an MDP $M = \langle S, A, u_0, P\rangle$ is a tuple $G = \langle S, E\rangle$ where S is a set of nodes, and $E \subseteq S \times A \times S$ is a set of labeled edges such that $(s, a, s') \in E$ if and only if $P(s,a)(s') > 0$.

[3] A graph is planar if it can be drawn in the plane in such a way that no two edges meet each other except at a vertex to which they are incident.

Let $n = \mathsf{card}(S)$ be the total number of nodes in the graph. A partition of states in the MDP gives rise to a partition of nodes in the graph. Given a partition Π and a region $S_i \in \Pi$, a node is said to be *contained* in S_i if some edge of the region is incident to the node [20]. A node contained in more than one regions is called a *boundary* node. That is, $s \in S_i$ is a boundary node if and only if there exists $(s, a, s') \in E$ or $(s', a, s) \in E$ with $s' \notin S_i$. Formally, the boundary nodes of S_i are $B_i = \mathsf{In}_i \cup \mathsf{Out}_i$ where $\mathsf{In}_i = \{s \in S_i \mid \exists j \neq i, s' \in S_j, a \in A(s'),$ and $(s', a, s) \in E\}$ and $\mathsf{Out}_i = \{s \in S_i \mid \exists s' \in S \setminus S_i, a \in A(s),$ and $(s, a, s') \in E\}$. We define $B_0 = \bigcup_{i=1}^{N} B_i$. Note that since $\bigcup_{i=1}^{N} \mathsf{In}_i = K_0$, $K_0 \subseteq B_0$. We use the number of boundary nodes as an upper bound on the size of the set K_0 of states.

Definition 2: [20] An *r-division of an n-node graph* is a partition of nodes into $O(n/r)$ subsets, each of which have $O(r)$ nodes and $O(\sqrt{r})$ boundary nodes.

Reference [20] shows an algorithm that divides a planar graph of n vertices into an r-division in $O(n \log n)$ time.

Lemma 2: Given a partition Π of an MDP with n states obtained with a r-division the induced graph, the number of states in K_0 is upper bounded by $O(n/\sqrt{r})$ and the number of states in K_i is upper bounded by $O(r)$.

Proof: Since each boundary node is contained in at most three regions and at least one region by the property of an r-division [20], the total number of boundary nodes is $O(\sqrt{r} \cdot \frac{n}{r}) = O(n/\sqrt{r})$. The number of states in K_i is upper bounded by the size of S_i, which is $O(r)$. ∎

To obtain a decomposition, the user specifies an approximately upper bound on the number of variables for all subproblems. Then, the algorithm decides whether there is an r-division for some r that gives rise to a decomposition that has the desirable property.

Remark 1: A decomposition may be given or obtained straight-forwardly by exploiting the existing modular structure of the system. One of the future direction is to develop heuristic for decomposing graphs which are not planar. Note that a decomposition obtained from the system structure or by heuristics may not meet the desirable property for the distributed synthesis method, the proposed method still applies as long as each subproblem derived from that decomposition (see Section IV-C) can be solved given the limitation in memory and computational capacities.

IV. Distributed synthesis: Discounted-reward and average-reward problems

In this section, we show that under a decomposition, the original LP problem for a discounted-reward or average-reward can be formulated into one with a sparse constraint matrix. Then, we employ block-splitting algorithm in [12] for solving the LP problem in a distributed manner.

A. Discounted-reward case

Given a decomposition $\{K_i \mid i = 0, 1, \ldots, N\}$ of an MDP, let x_i be a vector consisting of variables $x(s, a)$ for all $s \in K_i$ with all actions enabled from s. Let $\iota_i : K_i \to \{1, \ldots, \mathsf{card}(K_i)\}$ be an index function. The constraints in (1b) can be written as: For each $s \in K_0$,

$$\sum_{a \in A(s)} x(s,a) = u_0(s) + \gamma \cdot \sum_{i=0}^{N} \sum_{s' \in K_i} \sum_{a' \in A(s')} x(s',a') \cdot P(s',a')(s), \quad (4)$$

and for each $s \in K_i$, $i = 1, \ldots, N$,

$$\sum_{a \in A(s)} x(s,a) = u_0(s) + \gamma \cdot \Big(\sum_{s' \in K_0} \sum_{a' \in A(s')} x(s',a') \cdot P(s',a')(s) + \sum_{s' \in K_i} \sum_{a' \in A(s')} x(s',a') \cdot P(s',a')(s) \Big). \quad (5)$$

Recall that, in Lemma 1, we have proven that each $s \in K_i$ with $i \neq 0$ can only be reached with non-zero probabilities from states in K_0 and K_i. As a result, for each state s in K_i with $i \neq 0$ and each action $a \in A(s)$, the constraint on variable $x(s, a)$ is only related with variables in x_i and x_0. Let $x = (x_0, x_1, \ldots, x_N)$. We denote the number of variables in x_i by m_i and the number of states in the set K_i by n_i. Let $m = \sum_{i=0}^{N} m_i$. The LP problem in (1) is then

$$\min_{x \in \mathbb{R}_+^m} \sum_{j=0}^{N} c_j^T x_j, \quad \text{subject to } Ax = b, \quad (6)$$

where $c_j^T x_j = \sum_{s \in K_j} \sum_{a \in A(s)} -R(s,a)x(s,a)$,

$$A = \begin{bmatrix} A_{00} & A_{01} & A_{02} & \cdots & A_{0N} \\ A_{10} & A_{11} & & & \\ A_{20} & & A_{22} & & \mathbf{0} \\ \vdots & \mathbf{0} & & \ddots & \\ A_{N0} & & & & A_{NN} \end{bmatrix}, \quad b = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix},$$

and $b_i \in \mathbb{R}^{n_i}$ where $b_i(k) = u_0(s)$ if $\iota_i(s) = k$. The transformation from (4) and (5) to (6) and the derivation of A_{ij} from (5) is straightforward by rewriting the constraints and we omit the detail.

B. Average-reward case

For an ergodic MDP, the constraints in the LP problem of maximizing the average reward, described by (2b), can be rewritten in the way just as how (1b) is rewritten into (4) and (5) for the discounted-reward problem. The difference is that for the average-reward case, we let $\gamma = 1$ and replace $u_0(s)$ with 0, for all $s \in S$, in (4) and (5). An additional constraint for the average-reward case is that $\sum_{s \in S} \sum_{a \in A(s)} x(s, a) = 1$. Hence, for an average-reward problem in an ergodic MDP, the corresponding LP problem in (2) is formulated as

$$\min_{x \in \mathbb{R}_+^m} \sum_{j=0}^{N} c_j^T x_j, \quad \text{subject to } \begin{bmatrix} \mathbf{1}^T \\ A \end{bmatrix} x = \begin{bmatrix} 1 \\ \mathbf{0} \end{bmatrix} \quad (7)$$

where $\mathbf{1}^T$ is a row vector of m ones, $c_j^T x_j = \sum_{s \in K_j} \sum_{a \in A(s)} -R(s,a)x(s,a)$ and the block-matrix A

has the same structure as of that in the discounted case with $\gamma = 1$. We can compactly write the constraint in (7) as $A'x = b$ where A' is a sparse constraint matrix similar in structure to the matrix A in the discounted-reward case.

For an average-reward case, we need to satisfy the constraint $\mathbf{1}^T x = 1$ in (7). This constraint leads to slow convergence and policies with large infeasibility measures in distributed optimization. To handle this issue, we approximate average reward with discounted reward [23]: For ergodic MDPs, the discounted accumulated reward, scaled by $1-\gamma$, is approximately the average reward. Further, if $\frac{1}{1-\gamma}$ is large compared to the mixing time [15] of the Markov chain, then the policy that optimizes the discounted accumulated reward with the discounting factor γ can achieve an approximately optimal average reward.

C. Distributed optimization algorithm

We solve the LP problems in (6) and (7) by employing the block splitting algorithm based on ADMM in [12]. We only present the algorithm for the discounted-reward case in (6). The extension to the average-reward case is straight-forward.

First, we introduce new variables y_i and let $f_i(y_i) = I_{\{b_i\}}(y_i)$, where for a convex set C, I_C is a function defined by $I_C(x) = 0$ for $x \in C$, $I_C(x) = \infty$ for $x \notin C$. Then, adding the term $f_i(y_i)$ into the objective function enforces $y_i = b_i$. Let $g_i(x_i) = c_i^T x_i + I_{\mathbb{R}_+^{m_i}}(x_i)$. The term $I_{\mathbb{R}_+^{m_i}}(x_i)$ enforces that x_i is a non-negative vector. We rewrite the LP problem in (6) as follows.

$$\begin{aligned} &\min_{x,y} \sum_{i=0}^{N} f_i(y_i) + \sum_{i=0}^{N} g_i(x_i) \\ &\text{subject to } y_0 = \sum_{i=0}^{N} A_{0i}x_i \text{ and for } i = 1,\dots,N, \\ &\qquad y_i = A_{i0}x_0 + A_{ii}x_i. \end{aligned} \tag{8}$$

With this formulation, we modify the block splitting algorithm in [12] to solve (8) in a parallel and distributed manner (see the Appendix for the details). The algorithm takes parameters ρ, ϵ^{rel} and ϵ^{abs}: $\rho > 0$ is a penalty parameter to ensure the constraints are satisfied, $\epsilon^{rel} > 0$ is a relative tolerance and $\epsilon^{abs} > 0$ is an absolute tolerance. The choice of ϵ^{rel} and ϵ^{abs} depends on the scale of variable values. In synthesis of MDPs, ϵ^{rel} and ϵ^{abs} may be chosen in the range of 10^{-2} to 10^{-6}. The algorithm is ensured to converge with any choice of ρ and the value of ρ may affect the convergence rate. Reference [16] proposes a method that updates parameter ρ for each iteration, for improving the convergence in practice. The readers are referred to [16] for more details on the choice of ρ, ϵ^{abs} and ϵ^{rel} and how these values affect the bound on the objective suboptimality and the convergence rate.

V. Extension to quantitative temporal logic constraints

We now extend the distributed control synthesis methods for MDPs with discounted-reward and average-reward criteria to solve Problem 1 in which quantitative temporal logic constraints are enforced.

A. Maximizing the probability of satisfying an LTL specification

Preliminaries Given an LTL formula φ as the system specification, one can always represent it by a deterministic Rabin automaton (DRA) $\mathcal{A}_\varphi = \langle Q, 2^{\mathcal{AP}}, T, I, \mathsf{Acc}\rangle$ where Q is a finite state set, $2^{\mathcal{AP}}$ is the alphabet, $I \in Q$ is the initial state, and $T : Q \times 2^{\mathcal{AP}} \to Q$ the transition function. The acceptance condition Acc is a set of tuples $\{(J_i, H_i) \in 2^Q \times 2^Q \mid i = 1,\dots,\ell\}$. The *run* for an infinite word $w = \sigma_0\sigma_1 \ldots \in (2^{\mathcal{AP}})^\omega$ is the infinite sequence of states $q_0q_1 \ldots \in Q^\omega$ where $q_0 = I$ and $q_{i+1} = T(q_i, \sigma_i)$ for $i \geq 0$. A run $\rho = q_0q_1 \ldots$ is accepted in $\mathcal{A}_\varphi$ if there exists at least one pair $(J_i, H_i) \in \mathsf{Acc}$ such that $\mathsf{Inf}(\rho) \cap J_i = \emptyset$ and $\mathsf{Inf}(\rho) \cap H_i \neq \emptyset$ where $\mathsf{Inf}(\rho)$ is the set of states that appear infinitely often in ρ.

Given an MDP $M = \langle S, A, u_0, P\rangle$ augmented with a set $\mathcal{AP}$ of atomic propositions and a labeling function $L : S \to 2^{\mathcal{AP}}$, one can compute the product MDP $\mathcal{M} = M \ltimes \mathcal{A}_\varphi = \langle V, A, \Delta, v_0, \mathsf{Acc}\rangle$ with the components defined as follows: $V = S \times Q$ is the set of states. A is the set of actions. The initial probability distribution of states is $\mu_0 : V \to [0,1]$ such that given $v = (s,q)$ with $q = T(I, L(s))$, it is that $\mu_0(v) = u_0(s)$. $\Delta : V \times A \to \mathcal{D}(V)$ is the transition probability function. Given $v = (s,q)$, σ, $v' = (s',q')$ and $q' = T(q, L(s'))$, let $\Delta(v,\sigma)(v') = P(s,\sigma)(s')$. The Rabin acceptance condition is $\mathsf{Acc} = \{(\hat{J}_i, \hat{H}_i) \mid \hat{J}_i = S \times J_i, \hat{H}_i = S \times H_i, i = 1,\dots,\ell\}$.

By construction, a path $\rho = v_0v_1 \ldots \in V^\omega$ satisfies the LTL formula φ if and only if there exists $i \in \{1,\dots,\ell\}$, $\mathsf{Inf}(\rho) \cap \hat{J}_i = \emptyset$ and $\mathsf{Inf}(\rho) \cap \hat{H}_i \neq \emptyset$. To maximize the probability of satisfying φ, the first step is to compute the set of *end components* in $\mathcal{M}$, each of which is a pair (W, f) where $W \subseteq V$ is non-empty and $f : W \to 2^A$ is a function such that for any $v \in W$, for any $a \in f(v)$, $\sum_{v' \in W} \Delta(v,a)(v') = 1$ and the induced directed graph $(W, \to_f)$ is strongly connected. Here, $v \to_f v'$ is an edge in the graph if there exists $a \in f(v)$, $\Delta(v,a)(v') > 0$. An end component (W, f) is *accepting* if $W \cap \hat{J}_i = \emptyset$ and $W \cap \hat{H}_i \neq \emptyset$ for some $i \in \{1,\dots,\ell\}$.

Let the set of accepting end components (AEC)s in $\mathcal{M}$ be $\mathsf{AEC}(\mathcal{M})$ and the set of *accepting end states* be $\mathcal{C} = \{v \mid \exists (W,f) \in \mathsf{AEC}(\mathcal{M}), v \in W\}$. Once we enter some state $v \in \mathcal{C}$, we can find an AEC (W, f) such that $v \in W$, and initiate the policy f such that for some $i \in \{1,\dots,\ell\}$, states in $\hat{J}_i$ will be visited a finite number of times and some state in $\hat{H}_i$ will be visited infinitely often.

Formulating the LP problem An optimal policy that maximizes the probability of satisfying the specification also maximizes the probability of hitting the set of accepting end states $\mathcal{C}$. Reference [21] develops GPU-based parallel algorithms which significantly speed up the computation of end components for large MDPs. After computing the set of AECs, we formulate the following LP problem to compute

the optimal policy using the proposed decomposition and distributed synthesis method for discounted-reward cases.

Given a product MDP $\mathcal{M} = \langle V, A, \Delta, \mu_0, \mathsf{Acc}\rangle$ and the set $\mathcal{C}$ of accepting end states, the modified product MDP is $\tilde{\mathcal{M}} = \langle (V \setminus \mathcal{C}) \cup \{\mathsf{sink}\}, A, \tilde{\Delta}, \tilde{\mu}_0, R\rangle$ where $(V \setminus \mathcal{C}) \cup \{\mathsf{sink}\}$ is the set of states obtained by grouping states in $\mathcal{C}$ as a single state sink. For all $a \in A$, $\tilde{\Delta}(\mathsf{sink}, a)(\mathsf{sink}) = 1$ and $\tilde{\Delta}(v, a)(\mathsf{sink}) = \sum_{v' \in \mathcal{C}} \Delta(v, a)(v')$. The initial distribution $\tilde{\mu}_0$ of states is defined as follows: For $v \in V \setminus \mathcal{C}$, $\tilde{\mu}_0(v) = \mu_0(v)$, and $\tilde{\mu}_0(\mathsf{sink}) = \sum_{v \in \mathcal{C}} \mu_0(v)$. The reward function $R : ((V \setminus \mathcal{C}) \cup \{\mathsf{sink}\}) \times A \to \mathbb{R}$ is defined such that for all v that is not sink, $R(v, a) = \sum_{v' \in (V \setminus \mathcal{C}) \cup \{\mathsf{sink}\}} \tilde{\Delta}(v, a)(v') \cdot \mathbf{1}_{\{\mathsf{sink}\}}(v')$ where $\mathbf{1}_X(v)$ is the indicator function that outputs 1 if and only if $v \in X$ and 0 otherwise. For any action $a \in A(\mathsf{sink})$, $R(\mathsf{sink}, a) = 0$.

The discounted reward with $\gamma = 1$ from state v in the modified product MDP $\tilde{\mathcal{M}}$ is the probability of reaching a state in $\mathcal{C}$ from v under policy f in the product MDP $\mathcal{M}$. Hence, with a decomposition of $\mathcal{M}$, the proposed distributed synthesis method for discounted-reward problems can be used to compute the policy that maximizes the probability of satisfying a given LTL specification.

B. Average reward under Büchi acceptance conditions

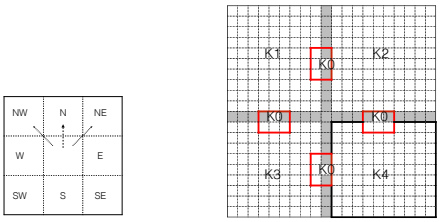
Preliminaries Consider a temporal logic formula φ that can be expressed as a deterministic Büchi automaton (DBA) $\mathcal{A}_\varphi = \langle Q, 2^{\mathcal{AP}}, T, I, F_\varphi\rangle$ where $Q, 2^{\mathcal{AP}}, T, I$ are defined similar to a DRA and $F_\varphi \subseteq Q$ is a set of *accepting states*. A run ρ is accepted in $\mathcal{A}_\varphi$ if and only if $\mathsf{Inf}(\rho) \cap F_\varphi \neq \emptyset$. Given an MDP $M = \langle S, A, u_0, P, \mathcal{AP}, L\rangle$ and a DBA $\mathcal{A}_\varphi = \langle Q, 2^{\mathcal{AP}}, T, q_0, F_\varphi\rangle$, the *product MDP with Büchi objective* is $\mathcal{M} = M \ltimes \mathcal{A}_\varphi = \langle V, A, \Delta, \mu_0, F\rangle$ where components V, A, Δ, μ_0 are obtained similarly as in the product MDP with Rabin objective. The difference is that $F \subseteq S \times F_\varphi$ is the set of accepting states. A path $\rho = v_0 v_1 \ldots \in V^\omega$ satisfies the LTL formula φ if and only if $\mathsf{Inf}(\rho) \cap F \neq \emptyset$.

Formulating the LP problem For a product MDP $\mathcal{M}$ with Büchi objective, we aim to synthesize a policy that maximizes the expected frequency of visiting an accepting state in the product MDP $\mathcal{M} = M \ltimes \mathcal{A}_\varphi$. This type of objectives ensures some recurrent properties in the temporal logic formula are satisfied as frequently as possible. For example, one such objective can be requiring a mobile robot to maximize the frequency of visiting some critical regions.

This type of objectives can be formulated as an average-reward problem in the following way: Let the reward function $R : V \times A \to \mathbb{R}$ be defined by $R(v, a) = \sum_{v' \in V} \Delta(v, a)(v') \cdot \mathbf{1}_F(v')$. By definition of the reward function, the optimal policy with respect to the average-reward criterion is the one that maximizes the frequency of visiting a state in F. If the product MDP is ergodic, we can then solve the resulting average-reward problem by the distributed optimization algorithm with a decomposition of product MDP $\mathcal{M}$.

VI. Case studies

We demonstrate the method with three robot motion planning examples. All the experiments were run on a machine with Intel Xeon 4 GHz, 8-core CPU and 64 GB RAM running Linux. The distributed optimization algorithm is implemented in MATLAB. The decomposition and other operations are implemented in Python.



(a) (b)

Fig. 2: (a) A fraction of a $m \times n$ gridworld. The dash arrow represents that if the robot takes action 'N', there are non-zero probabilities for it to arrive at NW, N, and NE cells. (b) A 20×20 gridworld. A natural partition of state space using the walls gives rise to K_0, K_1, K_2, K_3, K_4 subsets of states. States in K_0 are enclosed using the squares.

Figure 2a shows a fraction of a gridworld. A robot moves in this gridworld with uncertainty in different terrains ('grass', 'sand', 'gravel' and 'pavement'). In each terrain and for robot's different action (heading north ('N'), south ('S'), west ('W') and east ('E')), the probability of arriving at the correct cell is 0.9 for pavement, 0.85 for grass, 0.8 for gravel and 0.75 for sand. With a relatively small probability, the robot will arrive at the cell adjacent to the intended one. Figure 2b displays a 20×20 gridworld. The grey area and the boundary are walls. If the robot runs into the wall, it will be bounce back to its original cell. The walls give rise to a natural partition of the state space, as demonstrated in this figure. If no explicit modular structure in the system can be found, one can compute a decomposition using the method in section III-B. In the following example, the wall pattern is the same as in the 20×20 gridworld.

A. Discounted-reward case

We select a subset W of cells as "restricted area" and a subset G of cells as "targets". The reward function is given: For $s \in S$, $S \notin G \cup W$, $R(s, a) = -1$ counts for the amount of time the robot takes action a. For $s \in W$, for all $a \in A(s)$, $R(s, a) = -1000$. For $s \in G$, $R(s, a) = 100$ for all $a \in A(s)$. Intuitively, this reward function will encourage the robot to reach the target with as fewer expected number of steps as possible, while avoiding running into a cell in the restricted area. We select $\gamma = 0.9$.

Case 1: To show the convergence and correctness of the distributed optimization algorithm, we first consider a 100×100 gridworld example that can be solved directly with a centralized algorithm. Since at each cell there are four actions for the robot to pick, the total number of variables is 4×8220 for the 100×100 gridworld (the wall cells are excluded from the set of states). In this gridworld, there is only 1 target cell. The restricted area include 50 cells. The resulting LP problem (1) can be solved using CVX, a package for specifying and solving convex programs [22]. The problem

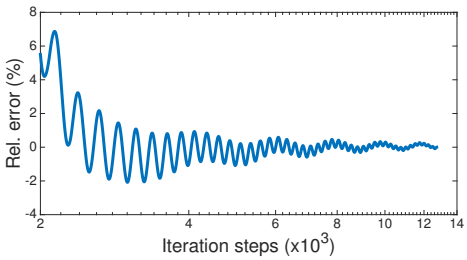


Fig. 3: Relative error in the objective value vesus iterations in 100×100 gridworld with discounted reward, under $\rho = 1000$. For clarity, we did not draw the relative error for the initial 2000 steps, which are comparatively large.

is solved in 4.77 seconds, and the optimal objective value under the optimal policy given by CVX is 10.

Next, we solve the same problem by decomposing the state space of the MDP along the walls into 25 regions, each of which is a 20×20 gridworld. This partition of state space yields 75 states for each $K_i, i > 0$ and 720 states for K_0. In which follows, we select $\rho = 80, 100, 200, 500, 1000$ to show the convergence of the distributed optimization algorithm. Irrespective of the choices for ρ, the average time for each iteration is about 0.16 sec. The solution accuracy relative to CVX is summarized in Table I. The 'rel. error in objval' is the relative error in objective value attained, treating the CVX solution as the accurate one, and the infeasibility is the relative primal infeasibility of the solution, measured by $\frac{\|Ax^*-b\|_2}{1+\|b\|_1}$. Figure 3 shows the convergence of the algorithm.

Case 2: Since a centralized method does not scale, for a 1000×100 gridworld, the centralized method in CVX fails to produce a solution. Thus, we solve it using the decomposition and distributed synthesis method. In this example, we partition the gridworld such that each region has 50×50 cells, which results in 40 regions. There are 1160 states in K_0 and about 2005 states in each K_i, for $i = 1, \ldots, 40$. In this example, we randomly select 40 cells to be the targets and 40 cells to be the restricted areas. By choosing $\rho = 1000$, $\epsilon^{rel} = 10^{-6}$, $\epsilon^{abs} = 10^{-6}$, the optimal policy is solved within 25342 seconds and it takes about 2.6 seconds for one iteration. The total number of iterations is 9747. Under the (approximately)-optimal policy obtained by distributed optimization, the objective value is 138.73. The relative primal infeasibility of the solution is 0.29×10^{-4}. Figure 4a shows the convergence of distributed optimization algorithm. A solution with a small infeasibility in this case can be that for some state, the probabilities of selecting all actions do not sum to 1. During planning, through normalization one can compute a randomized policy.

B. Average-reward case with quantative LTL objectives

We consider a 50×50 gridworld with no obstacles and 4 critical regions labeled "R_1" , "R_2", "R_3" and "R_4". The system is given a temporal logic specification $\varphi := \square\lozenge(R_1 \wedge \lozenge R_2) \wedge \square\lozenge(R_3 \wedge \lozenge R_4)$, i.e., the robot has to always eventually visit region R_1 and then R_2, and also always eventually visit region R_3 and then R_4. The number of states in the corresponding DBA is 14 after trimming the unreachable states, due to the fact that the robot cannot be at two cells simultaneously. The quantitative objective is to maximize the frequency of visiting all four regions (an accepting state in the DBA). The formulated MDP is ergodic and therefore our method for average-reward problems applies.

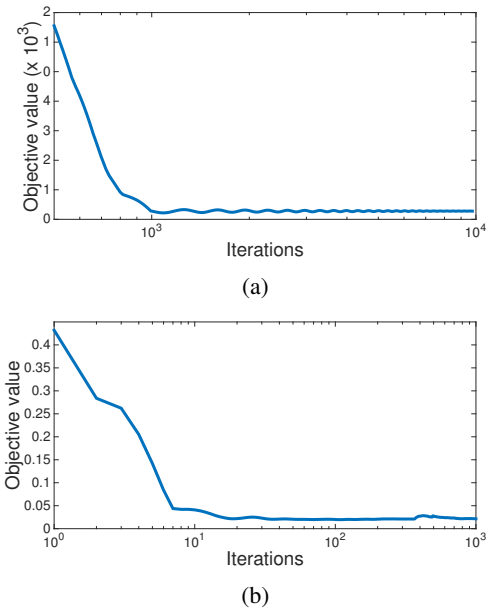


Fig. 4: Objective value vesus iterations in (a) 1000×100 gridworld (the initial 500 steps are omitted). (b) Objective value vesus iterations in 50×50 gridworld with a Büchi objective. Here we only show the first 1000 iterations as the objective value converges to the optimal one after 1000 steps.

Given $\rho = 1000$, $\epsilon^{ref} = 10^{-5}$ and $\epsilon^{abs} = 10^{-5}$, $\gamma = 0.98$, the distributed synthesis algorithm terminates in 14284 iteration steps and the optimal discounted reward is 0.9998. Scaling by $1 - \gamma = 0.02$, we obtain the average reward $0.9998 \times 0.02 = 0.02$, which is the approximately optimal value for this average reward under the obtained policy. The convergence result is shown in Figure 4b and the infeasibility measure of the obtained solution is 0.016.

VII. Conclusion

For solving large Markov decision process models of stochastic systems with temporal logic specifications, we developed a decomposition algorithm and a distributed synthesis method. This decomposition exploits the modularity in the system structure and deals with sub-problems of smaller sizes. We employed the block splitting algorithm in distributed optimization to cope with the difficulty of combining the solutions of sub-problems into a solution to the original problem. Moreover, the formal decomposition-based distributed control synthesis framework established in this paper facilitates the application of other distributed and parallel large-scale optimization algorithms [24] to further improve the rate of convergence and the feasibility of solutions for control synthesis in large MDPs. Recall that although the decomposition algorithm applies to MDPs with Planar graph

TABLE I: Relative resolution accuracy for the 100×100 gridworld with discounted reward ($\epsilon^{abs} = 10^{-5}$, $\epsilon^{rel} = 10^{-4}$).

ρ	80	100	200	500	1000
Iterations	12001	11014	13868	11866	12733
objval	9.54	9.90	9.89	9.96	9.96
rel. error (%)	4.6	1.0	1.1	0.38	0.38
infeasibility	2.7×10^{-3}	2×10^{-3}	0.885×10^{-3}	0.45×10^{-3}	0.23×10^{-3}

structure, for more general MDPs, decompositions can also be generated with heuristics and from the modular structure of the system. In the future, we will develop an interface to PRISM toolbox [8] with an implementation of the proposed decomposition and distributed synthesis algorithms.

APPENDIX

At the k-th iteration, for $i, j = 0, \ldots, N$,

$$
\begin{aligned}
y_i^{k+1/2} &:= \mathbf{prox}_{f_i}(y_i^k - \tilde{y}_i^k) = b_i, \\
x_j^{k+1/2} &:= \mathbf{prox}_{g_j}(x_j^k - \tilde{x}_j^k) \\
&= \mathbf{proj}_{\mathbb{R}_+^{m_j}}(x_j^k - \tilde{x}_j^k - c_j/\rho), \\
(x_{ij}^{k+1/2}, y_{ij}^{k+1/2}) &:= \mathbf{proj}_{ij}(x_j^k - \tilde{x}_{ij}^k, y_{ij}^k + \tilde{y}_i^k), \\
x_j^{k+1} &:= \mathbf{avg}(x_j^{k+1/2}, \{x_{ij}^{k+1/2}\}_{i=0}^N), \\
(y_i^{k+1}, \{y_{ij}^{k+1}\}_{j=0}^N) &:= \mathbf{exch}(y_i^{k+1/2}, \{y_{ij}^{k+1/2}\}_{j=0}^N), \\
&\quad \text{if } i = 0, \\
(y_i^{k+1}, \{y_{i0}^{k+1}, y_{ii}^{k+1}\}) &:= \mathbf{exch}(y_i^{k+1/2}, \{y_{i0}^{k+1/2}, y_{ii}^{k+1/2}\}), \\
&\quad \text{if } i = 1, \ldots, N, \\
\tilde{x}_j^{k+1} &:= \tilde{x}_j^k + x_j^{k+1/2} - x_j^{k+1}, \\
\tilde{y}_i^{k+1} &:= \tilde{y}_i^k + y_i^{k+1/2} - y_i^{k+1}, \\
\tilde{x}_{ij}^{k+1} &:= \tilde{x}_{ij}^k + x_{ij}^{k+1/2} - x_j^{k+1},
\end{aligned}
$$

where $\mathbf{proj}_{R_+^{m_i}}$ denotes the projection to the nonnegative orthant, $\mathbf{proj}_{ij}$ denotes projection onto $\{(x, y) \mid y = A_{ij}x\}$. $\mathbf{avg}$ is the elementwise averaging [4]; and $\mathbf{exch}$ is the exchange operator, defined as below. $\mathbf{exch}(c, \{c_j\}_{j=1}^N)$ is given by $y_{ij} := c_j + (c - \sum_{j=1}^N c_j)/(N-1)$ and $y_i := c - (c - \sum_{j=1}^N c_j)/N - 1$. The variables can be initialized to 0 at $k = 0$. Note that the computation in each iteration can be parallelized. The iteration terminates when the stopping criterion for the block splitting algorithm is met (See [12] for more details). The solution can be obtained $x^* = (x_0^{k+1/2}, \ldots, x_N^{k+1/2})$.

REFERENCES

[1] Z. Manna and A. Pnueli, *The Temporal Logic of Reactive and Concurrent Systems: Specifications*. springer, 1992, vol. 1.

[2] S. Thiébaux, C. Gretton, J. K. Slaney, D. Price, F. Kabanza, and Others, "Decision-Theoretic Planning with non-Markovian Rewards." *Journal of Artificial Intelligence Research*, vol. 25, pp. 17–74, 2006.

[3] C. Baier and J.-P. Katoen, *Principles of Model Checking*. MIT press Cambridge, 2008.

[4] H. J. Kushner and C.-H. Chen, "Decomposition of systems governed by markov chains," *IEEE Transactions on Automatic Control*, vol. 19, no. 5, pp. 501–507, 1974.

[4]Since for some i, j, $x_{ij}^{k+1/2} = 0$, in the elementwise averaging, these $x_{ij}^{k+1/2}$ will not be included.

[5] T. Dean and S.-H. Lin, "Decomposition techniques for planning in stochastic domains," in *Proceedings of the 14th international joint conference on Artificial intelligence-Volume 2*. Morgan Kaufmann Publishers Inc., 1995, pp. 1121–1127.

[6] V. Krishnamurthy, C. Rojas, and B. Wahlberg, "Computing monotone policies for markov decision processes by exploiting sparsity," in *Australian Control Conference*, Nov 2013, pp. 1–6.

[7] A. G. Barto and S. Mahadevan, "Recent advances in hierarchical reinforcement learning," *Discrete Event Dynamic Systems*, vol. 13, no. 4, pp. 341–379, 2003.

[8] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of probabilistic real-time systems," in *Proceedings of International Conference on Computer Aided Verification*, ser. LNCS, G. Gopalakrishnan and S. Qadeer, Eds., vol. 6806. Springer, 2011, pp. 585–591.

[9] X. C. Ding, S. L. Smith, C. Belta, and D. Rus, "MDP optimal control under temporal logic constraints," in *IEEE Conference on Decision and Control and European Control Conference*, 2011, pp. 532–538.

[10] E. M. Wolff, U. Topcu, and R. M. Murray, "Optimal control with weighted average costs and temporal logic specifications." in *Robotics: Science and Systems*, 2012.

[11] M. Lahijanian, S. Andersson, and C. Belta, "Temporal logic motion planning and control with probabilistic satisfaction guarantees," *IEEE Transactions on Robotics*, vol. 28, no. 2, pp. 396–409, April 2012.

[12] N. Parikh and S. Boyd, "Block splitting for distributed optimization," *Mathematical Programming Computation*, vol. 6, no. 1, pp. 77–102, Oct. 2013.

[13] C. Daoui, M. Abbad, and M. Tkiouat, "Exact decomposition approaches for Markov decision processes: A survey," *Advances in Operations Research*, vol. 2010, pp. 1–19, 2010.

[14] D. P. Bertsekas, *Dynamic Programming and Optimal Control, Vol. II*, 3rd ed. Athena Scientific, 2007.

[15] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 2009, vol. 414.

[16] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[17] T. Brázdil, V. Brozek, K. Chatterjee, V. Forejt, and A. Kucera, "Two views on multiple mean-payoff objectives in markov decision processes," in *Annual IEEE Symposium on Logic in Computer Science*, 2011, pp. 33–42.

[18] C. Baier, M. Größ er, M. Leucker, B. Bollig, and F. Ciesinski, "Controller Synthesis for Probabilistic Systems (Extended Abstract)," in *Exploring New Frontiers of Theoretical Informatics*, ser. IFIP International Federation for Information Processing, J.-J. Levy, E. Mayr, and J. Mitchell, Eds. Springer US, 2004, vol. 155, pp. 493–506.

[19] H. J. Kushner and P. Dupuis, *Numerical Methods for Stochastic Control Problems in Continuous Time*. Springer, 2001, vol. 24.

[20] G. N. Federickson, "Fast algorithms for shortest paths in planar graphs, with applications," *SIAM Journal on Computing*, vol. 16, no. 6, pp. 1004–1022, 1987.

[21] A. Wijs, J.-P. Katoen, and D. Bošnački, "Gpu-based graph decomposition into strongly connected and maximal end components," in *Computer Aided Verification*, ser. Lecture Notes in Computer Science, A. Biere and R. Bloem, Eds. Springer International Publishing, 2014, vol. 8559, pp. 310–326.

[22] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 2.1," Mar. 2014.

[23] S. Kakade, "Optimizing average reward using discounted rewards," in *Computational Learning Theory*. Springer, 2001, pp. 605–615.

[24] G. Scutari, F. Facchinei, L. Lampariello, and P. Song, "Parallel and distributed methods for nonconvex optimization," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2014, pp. 840–844.