Graph-based Stochastic Control with Constraints: A Unified Approach with Perfect and Imperfect Measurements

Ali-akbar Agha-mohammadi, Suman Chakravorty, Nancy M. Amato

***Abstract*— This paper is concerned with the problem of stochastic optimal control (possibly with imperfect measurements) in the presence of constraints. We propose a computationally tractable framework to address this problem. The method lends itself to sampling-based methods where we construct a graph in the state space of the problem, on which a Dynamic Programming (DP) is solved and a closed-loop feedback policy is computed. The constraints are seamlessly incorporated to the control policy selection by including their effect on the transition probabilities of the graph edges. We present a unified framework that is applicable both in the state space (with perfect measurements) and in the information space (with imperfect measurements.)**

I. Introduction

This paper proposes a computationally tractable framework to address the problem of stochastic optimal control (possibly with imperfect measurements). It is well-known that the stochastic control problem with perfect measurements can be formulated as a Markov Decision Process (MDP) problem in the state space. Similarly, the stochastic control problem with imperfect measurements can be formulated as a Markov Decision Process (MDP) defined on the space of information-states (probability distribution over all possible states), called information space. Therefore, to treat these two problems in a unified framework, we define a generic state, which can be the original state or the information-state of the problem depending on the context.

Excluding some specific cases (Linear systems with Gaussian noises and quadratic costs), it is well-known that this MDP cannot be solved analytically. It is also known that for continuous state, control, and observation spaces the computational methods are also intractable. Adding state (or control) constraints to the problem makes it more challenging. Methods such as Receding Horizon Control [1], and its variants, are the most widely used approaches in coping with such problems. RHC was originally designed for deterministic systems. Different methods attempt to generalize the RHC framework to the stochastic systems (e.g., [2], [3]) and information spaces (e.g., [4], [5]). However, issues can arise if the cost function is computationally expensive to evaluate (e.g. computing constraint violation probability), where RHC may not be computationally feasible in real time.

A. Agha-mohammadi and N. Amato are with the Dept. of Computer Science and Engineering and S. Chakravorty is with the Dept. of Aerospace Engineering, Texas A&M University, TX 77843, USA. Emails: {aliagha,schakrav,amato}@tamu.edu

The work of Agha-mohammadi and Chakravorty is supported in part by NSF award RI-1217991 and AFOSR Grant FA9550-08-1-0038 and the work of Agha-mohammadi and Amato is supported in part by NSF awards CNS-0551685, CCF-0833199, CCF-0830753, IIS-0917266, IIS-0916053, EFRI-1240483, RI-1217991, by NSF/DNDO award 2008-DN-077-ARI018-02, by NIH NCI R25 CA090301-11, by DOE awards DE-FC52-08NA28616, DE-AC02-06CH11357, B575363, B575366, by THECB NHARP award 000512-0097-2009, by Samsung, Chevron, IBM, Intel, Oracle/Sun and by Award KUS-C1-016-04, made by King Abdullah University of Science and Technology (KAUST).

In this paper, we propose an alternative framework for solving the problem of stochastic optimal control (possibly with imperfect measurements) in the presence of constraints. The method performs almost all of the computations offline. The main idea in this framework is to reduce the planning over the entire state/information space into a planning over a representative graph within the state/information space. The main challenge (in particular in the information space) then is how to generate such a graph (i.e., how to sample its nodes), and how to guarantee that the nodes of this graph are reachable. (i.e., how to connect the nodes.)

The key elements in this framework are *stabilizers* that are feedback controllers under which the induced Markov chain converges to a unique stationary behaviour. Then, according to this stationary behaviour we choose the graph nodes such that the Markov chain can hit them in a finite time. Doing so, we can form a graph in these spaces and then incorporate the problem constraints by encoding them into the transition probabilities along the graph edges. Such stabilization procedure breaks the space into *independent* smaller parts (graph edges), whose costs can be computed *offline* independent of the rest of space.

Initial results on this framework have been presented in [6]–[8] for specific robotic systems. However, in this work we provide a broader generic framework, which also provides a unified treatment in the state and information spaces. In a nutshell the contributions and highlights of this framework can be summarized as: *(i)* it presents a unified generic graph-based control framework in the state and information spaces. *(ii)* It provides a computationally tractable reduction of the original problem, while is capable of incorporating constraints into the control problem. *(iii)* It is constructed offline and *(iv)* provides a feedback solution, which is robust to large disturbances.

II. Problem formulation

In this section, we formulate the stochastic optimal control problem, in the presence of constraints.

State and Information-state: Let $x_k \in \mathbb{X}$, $u_k \in \mathbb{U}$, and $z_k \in \mathbb{Z}$ denote the system state, control, and observation at time step k, respectively. We denote the sequence of these quantities, say z, by $z_{i:j} = \{z_i, z_{i+1}, \cdots, z_j\}$. In the presence of perfect measurements, i.e., if we have access to exact x_k, the system state is used for decision making (generating control signal). However, if observations are imperfect, the decision has to be made based on the available data history $\mathcal{H}_k = \{u_{0:k-1}, z_{0:k}\}$ or equivalently, based on the information-state b_k, defined as the probability distribution of the system's state conditioned on the available data, i.e., $b_k := p(x_k|\mathcal{H}_k)$ [9]. We denote the information space by $\mathbb{B}$.

Unified State Space: To unify the problem formulation in the state and information spaces, we denote the generic state of the problem by s_k, which can be either x_k or b_k, and denote the generic space by $\mathbb{S}$, which can be either $\mathbb{X}$ or $\mathbb{B}$.

State Evolution Model: The process model $x_{k+1} = f_x(x_k, u_k, w_k)$ describes how the system state evolves as a function of the applied control u_k and the process noise w_k, which is drawn from the distribution $p(w_k|x_k, u_k)$. Similarly, the filtering dynamics $b_{k+1} = f_b(b_k, u_k, z_k)$ describes the evolution of the information-state of the system, where the observation z_k is assumed to be drawn from the observation model $p(z_k|x_k, u_k)$. Thus, in the unified view, we define f_s as the evolution law of the generic state $s_{k+1} = f_s(s_k, u_k, m_k)$, where m_k is either w_k or z_k. An equivalent representation of the evolution model is through the transition probability kernel $\mathbb{P}(S|s, u)$, which encodes the probability of transition from state s to set $S \subset \mathbb{S}$ under the control u.

Policy: The decision making, i.e., generating control signals at each time step is performed based on the policy $\pi_k(\cdot) : \mathbb{S} \to \mathbb{U}$, which maps the state to the control $u_k = \pi_k(s_k)$. We denote the space of policies by Π.

Cost-to-go: To find the optimal policy, we define the cost-to-go function from every state. Let the function $c(s, u) : \mathbb{S} \times \mathbb{U} \to \mathbb{R}_{\geq 0}$ defines the cost of taking action u at state s. Therefore, the cost-to-go $J^\pi : \mathbb{S} \to \mathbb{R}_{\geq 0}$ from a state s_0 under the policy π is defined as:

$$J^\pi(s_0) := \sum_{k=0}^{\infty} \mathbb{E}\left[c(s_k, \pi(s_k))\right] \tag{1}$$

Consider a goal region $S_{goal} \subset \mathbb{S}$ such that for all u, we have $c(s \in S_{goal}, u) = 0$, i.e., the goal region is cost absorbing. Then the above cost-to-go would be finite for a policy that can drive the state to the goal region in a finite time.

MDP problem: Now, we can define the stochastic control problem as the problem of finding the optimal policy that minimizes the cost-to-go function. This problem is also known as the Markov Decision Process (MDP) problem:

$$\pi^* = \arg\min_{\Pi} \sum_{k=0}^{\infty} \mathbb{E}\left[c(s_k, \pi(s_k))\right] \tag{2}$$
$$s.t. \quad s_{k+1} = f_s(s_k, \pi(s_k), m_k), \quad m_k \sim p(m_k|x_k, u_k)$$

Constrained MDP problem: Consider a failure set in the state and control space $F \subset \mathbb{X} \times \mathbb{U}$, which needed to be avoided by the system. This constraint can be added to (2) as $\Pr((x_k, u_k) \in F) < \delta$. It is well-known that the exact solution for this problem over the continuous state, control, and observations spaces is computationally intractable in general. In the next section, we discuss how we reduce this problem into a computationally tractable problem.

III. Graph-based control

In this section, we propose a graph-based controller for solving the constrained MDP problem. We start by some definitions on the feedback controllers and reachability.

A. Feedback Controllers and Reachability

One-step transition probability: Any given feedback controller $\mu(\cdot) : \mathbb{S} \to \mathbb{U}$ induces a Markov chain with the one-step transition probability $\mathbb{P}_1(S|s; \mu) := \mathbb{P}(S|s; \mu(s))$ over the space $\mathbb{S}$.

Hitting time: $\mathcal{T}(A|s, \mu) \in [0, \infty]$ denotes the hitting time on the set $A \subset \mathbb{S}$, under the controller μ starting from s:

$$\mathcal{T}(A|s, \mu) := \min\{k \geq 0, s_k \in A|s_0 = s; \mu\} \tag{3}$$

Stopping region: We call region $S \subset \mathbb{S}$ a stopping region of the controller μ if we force the controller to stop executing as the state reaches the region S, i.e., for all $s \in S$, we impose $\mathbb{P}_1(S|s, \mu) = 1$.

n-step transition probability: We define the n-step transition probability as the probability of landing in the stopping region S in at most n steps:

$$\mathbb{P}_n(S|s, \mu) = \Pr(\mathcal{T}(S|s, \mu) \leq n) \tag{4}$$

Stationary Transition Probability: Consider the controller μ that starts at state s and stops executing when the state enters region S. Thus, we define $\mathbb{P}(S|s, \mu)$ as the transition probability from s to S induced by μ, when controller stops executing, i.e., $\mathbb{P}(S|s, \mu)$ would be the probability of landing in stopping region S in a finite time:

$$\mathbb{P}(S|s, \mu) := \lim_{n \to \infty} \mathbb{P}_n(S|s, \mu) = \Pr(\mathcal{T}(S|s, \mu) < \infty) \tag{5}$$

Reachability: The stopping region S is called reachable under controller μ from s, if $\mathbb{P}(S|s, \mu) = 1$.

αT-reachability: The stopping region S is called αT-reachable from s, if $\mathbb{P}_T(S|s, \mu) = \Pr(\mathcal{T}(S|s, \mu) \leq T) > \alpha$, i.e., controller can drive the system into the S in less than T steps with a probability greater than α.

Reachability Basin: The reachability basin $\breve{S}$ associated with the pair (μ, S) is the set of all states from which S is reachable under μ in the absence of constraints. Thus, the reachability and αT-reachability basins are:

$$\breve{S} := \{s \in \mathbb{S} \ : \ \mathbb{P}(S|s, \mu) = 1\}, \tag{6}$$
$$\breve{S}(\alpha, T) := \{s \in \mathbb{S} \ : \ \mathbb{P}_T(S|s, \mu) \geq \alpha\}, \tag{7}$$

Clearly, $S \subset \breve{S}$, and in practice, S is much smaller than $\breve{S}$.

B. Constraint-free Graph Construction

In this section, we reduce the problem of planning over the continuous spaces into the planning over a representative graph constructed within the state space. Doing so, we can reduce the MDP problem in (2) over the continuous space into a tractable MDP problem defined over the graph nodes.

Stabilizer sampling: The main step in the construction of the proposed framework is to sample pairs of stabilizer and reachable node, i.e., sample (μ, S) such that S is reachable under the μ, i.e., $\mathbb{P}(S|s, \mu) = 1$, with a sufficiently large $\breve{S}$. We discuss the sampling strategy and the size of $\breve{S}$ further below. Note that the reachability condition can be replaced by the αT-reachability if needed.

Connecting samples: Consider a set of N samples $\{(\mu^i, S^i)\}_{i=1}^N$, where the reachability basin of the i-th sample is shown by the $\breve{S}^i$. Defining $\{S^i\}_{i=1}^N$ as the nodes of a graph, the node S^i is connected to the node S^j if starting from any $s \in S^i$, we can reach S^j using μ^j. In other words S^i is connected to the node S^j if $S^i \subset \breve{S}^j$. Again, the reachability condition can be replaced by the αT-reachability condition.

Checking connection condition: For some controllers, the connection condition can be checked analytically. A few

examples of such controllers are given below. However, in general the Markov chain induced by the controller can be simulated numerically (e.g., using particle-based methods). Accordingly, we can approximate the reachability (or αT-reachability) probability and thus check if the condition is true or not. Since this process is done offline, the computational burden can be tolerated. However, as we will see further below, designing suitable *edge controllers* in practice increases the reachability probability such that in most cases there is no need to propagate the probability distribution. In the following, we give some examples of the controller and reachable region pairs, and discuss their corresponding reachability basin.

Example 3.1: Consider a controllable linear system with a zero-mean Gaussian process noise and consider a Stationary Linear Quadratic Regulator (SLQR) $\mu : \mathbb{X} \to \mathbb{U}$ designed to regulate the system to a point $\mathbf{v} \in \mathbb{X}$. It can be shown that the stationary distribution over the system state is a Gaussian, whose mean coincides with $\mathbf{v}$. Thus, it can be shown that a non-empty ball $X \subset \mathbb{X}$ centered at $\mathbf{v}$ is reachable under μ starting from any point in $\mathbb{X}$. If the linear system is obtained by a linearization of a nonlinear system about $\mathbf{v}$, the argument is still valid as long as the system stays in the valid linearization region.

Example 3.2: Consider a linear controllable and observable system with zero-mean Gaussian process and measurement noises, and consider a Stationary Linear Quadratic Gaussian (SLQG) controller $\mu : \mathbb{B} \to \mathbb{U}$ designed to regulate the system to a point $\mathbf{v} \in \mathbb{X}$. Let us denote the solution (which is a covariance matrix) of the stationary Riccati equation associated with the utilized stationary Kalman filter by P_s. It can be shown that the stationary distribution over the system's information-state is a Gaussian, whose mean coincides with the information-state $\grave{b} = \mathcal{N}(\mathbf{v}, P_s)$. Thus, it can be shown that a non-empty ball $B \subset \mathbb{B}$ centered at information-state $\grave{b}$ is reachable under μ starting from any Gaussian information-state. For linearized systems (originally nonlinear), the argument holds as long as the system stays in the valid linearization region.

Example 3.3: Consider a linear controllable and observable system with zero-mean Gaussian process and measurement noises, and suppose the state of the system contains both position and velocity. Thus, it cannot be regulated to a sampled state, whose velocity dimension is non-zero. In this case, we can consider a Periodic Linear Quadratic Gaussian (PLQG) controller $\mu_k : \mathbb{B} \to \mathbb{U}$ designed to track a T-periodic trajectory $\mathbf{v}_k$, i.e., $\mathbf{v}_{k+T} = \mathbf{v}_k$. Let us denote the solution of the periodic Riccati equation associated with the utilized periodic Kalman filter by P_k, where $P_{k+T} = P_k$. It can be shown that a non-empty tube $B \subset \mathbb{B}$ around the periodic information-state curve $\grave{b}_k = \mathcal{N}(\mathbf{v}_k, P_k)$ is reachable under μ_k starting from any Gaussian information-state. Again, for linearized systems, the argument holds as long as the system stays in the valid linearization region. This example can be reduced to the perfect measurement case by replacing the PLQG controller with a PLQR.

Simplified sampling strategy: As is the case in the above examples, in many cases we can establish a unique correspondence between the state space sample $\mathbf{v}$ and the (μ, S) sample. Thus, we are able to sample points (or periodic trajectories) in the state space, and then leverage them to the (μ, S) samples based on this unique correspondence.

Stopping region: By definition, the graph node S associated with the controller μ acts as the stopping region of the controller. However, in general, if the process under the stabilizer hits another graph node before its corresponding graph node, we can stop the controller and pick the best controller from this intermediate node. Therefore, we can extend the stopping region for all controllers to the union of all nodes $\Psi := \cup_{l=1}^{N} S^l$. As a result, we will not necessarily have $\mathbb{P}(S^i|s, \mu^i) = 1$ since the process may hit some other node before S^i. But, we will have $\mathbb{P}(\Psi|s, \mu^i) = 1$ for all i in the absence of constraints.

Local controllers (Simplified connection strategy): To ease the connection step, and to have more distant nodes, we can precede each stabilizer by a time-varying controller (referred to as the edge-controller). To illustrate this idea, consider two nodes S^i and S^j, where $S^i \not\subseteq \breve{S}^j$, i.e., S^i cannot be connected to S^j through μ^j. In this case, we can connect the underlying state nodes $\mathbf{v}^i$ and $\mathbf{v}^j$ in the state space by a finite trajectory $\mathbf{e}^{ij}$ (say with length ι), and then design a time varying controller $\overline{\mu}_k^{ij}$, for $k = 0, 1, \cdots, \iota$ to track this finite trajectory. Therefore, if the node S^i is in the basin of reachability of the pair $(\overline{\mu}_k^{ij}, \breve{S}^j)$, then obviously S^i would be in the basin of reachability of the pair (μ^{ij}, S^j), where controller $\mu^{ij} = \{\overline{\mu}_{0:\iota}^{ij}, \mu^j\}$. We call μ^{ij} the (i, j)-th local controller, as it connects the node S^i to the node S^j.

Example 3.4: Consider the system, nodes, and stabilizers, described in Exmp. 3.2. Now, assume that the linear system is obtained by linearizing a nonlinear system. Therefore, the reachability basin will not be the entire space of Gaussian distributions and therefore, if the underlying nodes are sparse then the node connections may not be established by relying only on the stabilizers. Suppose the closest node to $\mathbf{v}^j$ is $\mathbf{v}^i$, and they cannot be connected to each other using μ^j as $S^i \not\subseteq \breve{S}^j$. Therefore, ignoring the noises, we design an open-loop trajectory $\mathbf{e}^{ij}$ of some length ι that connects the $\mathbf{v}^i$ to $\mathbf{v}^j$. Then, we design a time-varying LQG $\overline{\mu}_{0:\iota}^{ij}$ controller to track this finite trajectory, which hands over the system to the stabilizer μ^j after the ι-th step. By this construct, we can shift the reachability basin in a desired way. Again, since these steps are done offline, the reachabilities under this controller concatenation can be examined through probability distribution propagation techniques. However, as verified in our experiments, this construct is quite powerful in practice and in all of our experiments the concatenated local controllers are able to satisfy the reachability condition.

Graph: We define the constructed graph with the set of nodes $\mathbb{V} = \{S^i\}_{i=1}^{N}$ and the set of edges (or local controllers) $\mathbb{M} = \{\mu^{ij}\}$. The set of controllers available at S^i is denoted by $\mathbb{M}(i)$, i.e., the set of edges starting from S^i.

C. Constraint-free Graph MDP

Having a representative graph of the space $\mathbb{S}$, we can reduce the planning over $\mathbb{S}$ into the planning over the graph.

Graph transition cost and probabilities: We generalize the transition costs $c(s, u)$ and probabilities to the cost of taking a local controller (instead of a single control) in a graph node

and its corresponding transition probabilities along the graph edges:

$$C^g(S^i,\mu^{ij}) := \sum_{k=0}^{\mathcal{T}^{ij}} c(s_k,\mu^{ij}(s_k)|s_0=\grave{s}^i), \quad \grave{s}^i \in S^i$$
$$\approx \sum_{k=0}^{\mathcal{T}^{ij}} c(s_k,\mu^{ij}(s_k)|s_0=s), \quad \forall s \in S^i \quad (8a)$$
$$\mathbb{P}^g(S^l|S^i,\mu^{ij}) := \mathbb{P}(S^l|\grave{s}^i,\mu^{ij}), \quad \grave{s}^i \in S^i$$
$$\approx \mathbb{P}(S^l|s,\mu^{ij}), \quad \forall s \in S^i \quad (8b)$$

where, $\mathcal{T}^{ij} := \mathcal{T}(\Psi|\grave{s}^i,\mu^{ij})$, and $\grave{s}^i$ is a representative point in S^i (e.g. its center, if S^i is ball). However, obviously for $\grave{s}^i$ to be a good representative for all points in S^i, i.e., for approximations in (8) to be valid, we need the S^i to be small enough and the cost function and transition probabilities to be smooth enough.

Graph Policy: Graph policy $\pi^g : \mathbb{V} \to \mathbb{M}$ is a function that returns a local controller for any given node of the graph. We denote the space of all graph policies by Π^g.

Graph Cost-to-go: To choose the best graph policy in Π^g, we define the graph cost-to-go J^g from every graph node. Let $S_{\overline{k}}$ denotes the graph node at the end of the $\overline{k}$-th graph transition. Then, we can formally define the cost-to-go from any node S_0 as:

$$J^g(S_0;\pi^g) = \sum_{\overline{k}=0}^{\infty} \mathbb{E}\left[C^g(S_{\overline{k}},\pi^g(S_{\overline{k}}))\right] \quad (9)$$

where, the cost-to-go from the goal node is zero $J^g(S_{goal};\pi^g)=0$ for all π^g.

Constraint-free graph MDP: The graph MDP aims at finding the best graph policy and is defined on the graph nodes as follows:

$$\pi^{g^*} = \arg\min_{\Pi^g} \sum_{\overline{k}=0}^{\infty} \mathbb{E}\left[C^g(S_{\overline{k}},\pi^g(S_{\overline{k}}))\right]$$
$$s.t. \quad S_{\overline{k+1}} \sim \mathbb{P}^g(S_{\overline{k+1}}|S_{\overline{k}},\pi^g(S_{\overline{k}})) \quad (10)$$

Obstacle-free graph DP: Since the graph MDP is defined on a finite number of nodes, we can form a tractable Dynamic Programming (DP) to find the optimal cost-to-go and graph policy:

$$J^g(S^i) = \min_{\mu^{ij}\in\mathbb{M}(i)} C^g(S^i,\mu^{ij}) + \sum_{l=1}^{N} J^g(S^l)\mathbb{P}^g(S^l|S^i,\mu^{ij}), \ \forall i$$

where $J^g(\cdot) := \min_{\pi^g} J(\cdot;\pi^g)$ is the optimal cost-to-go.

D. Incorporating constraints into the control problem

In the presence of constraints, we cannot assure that the local controller $\mu^{ij}(\cdot)$ can drive the system from S^i into its stopping region with probability one. Instead, we have to specify the failure probabilities (i.e., the probability of violating the constraints). As mentioned the failure set $F \subset \mathbb{X}\times\mathbb{U}$ encodes both state and control constraints. Let $\mathbb{P}^g(F|S^i,\mu^{ij}) := \mathbb{P}(F|\grave{s}^i,\mu^{ij})$ denote the probability that under local controller μ^{ij} the system enters the failure set F before it enters the stopping region of the controller, starting from S^i. Similarly, we generalize the cost-to-go function by defining $J^g(F)$ as a user-defined suitably high cost for hitting obstacles.

Graph MDP with constraints: Therefore, we can modify the graph MDP to incorporate constraints as follows:

$$J^g(S^i) = \min_{\mu^{ij}\in\mathbb{M}(i)} C^g(S^i,\mu^{ij}) + J^g(F)\mathbb{P}^g(F|S^i,\mu^{ij}) + \sum_{l=1}^{N} J^g(S^l)\mathbb{P}^g(S^l|S^i,\mu^{ij}), \quad \forall i \quad (12a)$$

$$\pi^g(S^i) = \arg\min_{\mu^{ij}\in\mathbb{M}(i)} C^g(S^i,\mu^{ij}) + J^g(F)\mathbb{P}^g(F|S^i,\mu^{ij}) + \sum_{l=1}^{N} J^g(S^l)\mathbb{P}^g(S^l|S^i,\mu^{ij}), \quad \forall i \quad (12b)$$

Thus, all that is required to solve the above DP equation are the values of the costs $C^g(S^i,\mu^{ij})$ and transition probabilities $\mathbb{P}^g(\cdot|S^i,\mu^{ij})$, which both can be computed *offline*, either analytically or by simulating the controller behaviour along the graph edges and storing the costs and transition probabilities.

E. Overall policy π and Success Probability

The overall feedback $\pi : \mathbb{S} \to \mathbb{U}$ is generated by combining the graph policy π^g and the local controllers μ^{ij}s. Basically, at every graph node, π^g chooses the next best local controller μ and then μ keeps generating controls until the state reaches another graph node, where again π^g picks the next best local controller. This process continues until the goal node is reached.

Initial controller: If the start state s_0 is not in any of graph nodes, we design a μ that drives it to a graph node, and from there we use π^g to guide the state towards the goal node.

Success Probability: Constructing the control graph and solving the DP on it, we end up with a Markov chain whose states are the set of graph nodes plus the failure node (constraint set). Except the goal node, the rest of graph nodes in this Markov chain are transient states. Goal node and failure node are absorbing states of this Markov chain. As a result, we can form the transition probability matrix of this Markov chain and compute the success probability $\mathbb{P}(S^{goal}|S^i,\pi^g)$ of this Markov chain from any given node S^i under the policy π^g in a closed-form [10]. Then, if the success probability does not exceed a minimum acceptable success probability p_{min}, the number of nodes in the graph has to increase until the condition is satisfied. If, from a given node S^i a successful policy in the class of admissible policies exists, then this procedure will eventually find a successful policy due to the probabilistic completeness of the method, which is established in [11].

Algorithm: The generic algorithm for offline construction of the graph is presented in Algorithm 1.

Algorithm 1: Offline Construction of the control graph

1. Construct a graph with nodes $\mathcal{V} = \{\mathbf{v}^j\}$ and edges $\mathcal{E} = \{\mathbf{e}^{ij}\}$ in the state space $\mathbb{X}$;
2. For each edge $\mathbf{e}^{ij}$, design a controller $\mu^{ij} : \mathbb{S} \to \mathbb{U}$ and compute its corresponding reachable node $S^j \subset \mathbb{S}$;
3. For each S^i and $\mu^{ij} \in \mathbb{M}(i)$, compute transition costs $C^g(S^i,\mu^{ij})$ and probabilities $\mathbb{P}^g(S^l|S^i,\mu^{ij})$, $\mathbb{P}^g(F|S^i,\mu^{ij})$;
4. Solve the graph DP, Eq. (12), to compute feedback π^g;

IV. EXPERIMENTS

In the following experiment, we verify the effectiveness of the method in handling high-dimensional systems, through a simple example of an 8-arm manipulator. The initial state of the manipulator is shown in black in Fig. 1 and the goal is to take the manipulator tip (estimation mean) into the goal region, shown by the purple circle in Fig. 1. We consider the more general case of $\mathbb{S} = \mathbb{B}$, i.e., the system is partially observable and we can only measure the system state through noisy sensors. The initial state is assumed to be uncertain. It is assumed that the initial estimation mean coincides with the true initial state. The covariance of the tip location is shown by its 3σ ellipse (in black) in Fig. 1. Obstacles (state-constraint) in this environment are shown by brown rectangles. We utilize the Stationary LQG controllers as stabilizers. To the best of our knowledge, this is the first information space planner that can provide a plan over an entire graph for such a high-dimensional system, while incorporating expensive costs in planning such as computing collision probabilities. This experiment shows that graph-based control can be used as a practical tool in many real-world problems.

A. Process Model

We consider a planar manipulator with 8 revolute joints. The state of the system is described by angles of joints and their velocities $X = (\theta_1, \cdots, \theta_8, \dot{\theta}_1, \cdots, \dot{\theta}_8)^T$, and the available control is considered to be angular accelerations of joints $u = (\alpha_1, \cdots, \alpha_8)^T$. The process noise $w = (w_1, \cdots, w_8)^T$ is modelled as a zero-mean Gaussian noise on angular accelerations. Therefore, the continuous motion model is $\dot{X}_t = AX_t + Bu_t + Gw_t$, whose discrete version can be written as $X_{k+1} = AX_k + Bu_k\delta t + Gw_k\sqrt{\delta t}$, where

$$A = \begin{pmatrix} 0_8 & I_8 \\ 0_8 & 0_8 \end{pmatrix}, \quad B = \begin{pmatrix} 0_8 \\ I_8 \end{pmatrix}, \quad G = \begin{pmatrix} 0_8 \\ I_8 \end{pmatrix} \tag{13}$$

and δt is the time interval between consecutive time steps.

B. Observation Model

We use the light-dark environment setting as the observation model, introduced in [12]. In the light-dark environment the accuracy of sensory readings are encoded by a grey level, in which the regions that have access to more accurate sensory readings are lighter than the regions that do not have access to such informative sensory readings. In this experiment, we assume that we measure the state of system, but this measurement is more accurate as we get closer to the left wall on which our sensor is mounted (The model is adopted from [12]). Thus, we have $z = [z_1, \cdots, z_8]^T = h(x, v)$, where

$$z_i = \theta_i + v_i, \qquad v_i \sim \mathcal{N}(0, (\eta|\mathtt{x}_i - l| + \sigma_b)^2) \tag{14}$$

where, $\mathtt{x}_i$ is the x-coordinate of the i-th joint location, and l is the location of the vertical wall. η defines the dependency of the noise standard deviation on the distance from wall, and σ_b is the bias standard deviation. Figure 1 shows an example of such an environment, in which $l = -1.5$, $\eta = .1$, and $\sigma_b = 10^{-4}$.

C. Sampling (μ, S) pairs

As discussed in Section III-B, we simplify sampling (μ, S) pairs, by first sampling in the state space and then leveraging these samples to the (μ, S) samples. Thus, we first choose a set of samples in the state space $\mathcal{V} = \{\mathbf{v}^1, \cdots, \mathbf{v}^N\}$. Each sample $\mathbf{v}^i$ is a 16-dimensional vector consisting of a randomly sampled configuration (8DOF) augmented by zero velocities (to have a stabilizable sample). A drawn sample is valid (i.e., is added to $\mathcal{V}$) only if it is collision-free, i.e., if the manipulator joints associated with this sample does not collide with the obstacles (joint-obstacle collision) and if they do not collide with each other (joint-joint collision).

As explained in Exmp. 3.2, we leverage every selected (valid) sample $\mathbf{v}^j$ to a (μ^j, S^j) sample. To do so, we design a stationary LQG controller μ^j to drive the system towards the state $\mathbf{v}^j$. The stationary LQG is realized by designing a stationary Kalman filter and an stationary LQR that acts on the estimation mean. Let us denote the output of the Kalman filter (estimation) by $b_k \equiv (\widehat{x}_k^+, P_k)$ at the k-th time step, where $\widehat{x}_k^+$ is the estimation mean and P_k is the estimation covariance. It can be shown that under the SLQG controller P_k evolves deterministically and converges to P_s^j, which can be computed by solving the corresponding Riccati equation. While the estimation mean $\widehat{x}_k^+$ evolves randomly, it can be shown that its distribution under the SLQG converges to a Gaussian distribution with mean $\mathbf{v}^j$. As a result, a region $S^j \subset \mathbb{B}$ in the information space is reachable under the μ^j, if S^j has a non-empty interior containing point $b^j \equiv (\mathbf{v}^j, P_s^j)$. Thus, formally we define:

$$S^j = \{b \equiv (x, P) : \|x - \mathbf{v}^j\| < \delta_1, \|P - P_s^j\|_m < \delta_2\}, \tag{15}$$

where $\|\cdot\|$ and $\|\cdot\|_m$ denote suitable vector and matrix norms, respectively. The size of FIRM nodes are determined by δ_1 and δ_2 that have to be sufficiently small to satisfy the approximation in (8).

D. Connecting nodes

As explained in Section III-B, we simplify the connection phase by first making the connections in the state space and then leveraging them to design local controllers. To do so, we first connect each $\mathbf{v}^i$ to its n-nearest neighbors in the set $\mathcal{V} = \{\mathbf{v}^j\}_{j=1}^N$. Let us denote the edge from $\mathbf{v}^i$ to $\mathbf{v}^j$ as $\mathbf{e}^{ij}$. Then, we design the edge-controllers $\overline{\mu}_k^{ij}$ as the time-varying LQG controllers to track the finite trajectory $\mathbf{e}^{ij}$. After tracking this edge to the last point, the edge-controller hands over the system to the stabilizer μ^j, which in turn drives the system to the node S^i. The concatenation of the edge controller $\overline{\mu}_k^{ij}$ and the stabilizer μ^j is called the local controller and denoted by μ^{ij} as explained in Section III-B.

E. Transition costs and probabilities, and controller law

To compute the edge transition cost $C^g(S^i, \mu^{ij})$ and transition probabilities $\mathbb{P}^g(S^l|S^i, \mu^{ij})$ and $\mathbb{P}^g(F|S^i, \mu^{ij})$, we run the controller μ^{ij} invoked from the node S^i on an ensemble. Accordingly, we approximate the collision probability and the landing probability in S^l by counting the number of particles which collided with the obstacles (violated the constraints) and the number of particles landed in S^l, respectively. Similarly, we approximate the transition cost $C^g(S^i, \mu^{ij})$ by computing the cost of each particle

and averaging them. In this experiments we consider the underlying cost as follows:

$$C^g(S^i, \mu^{ij}) = \alpha_1 \sum_{k=1}^{\mathcal{T}^{ij}} \text{tr}(P_k) + \alpha_2 \mathcal{T}^{ij} \tag{16}$$

where P_k is the estimation covariance at the k-th time step under the local controller μ^{ij} and $\mathcal{T}^{ij}$ is the stopping time of the controller μ^{ij}. α_1 and α_2 are user-defined linear coefficients.

Controller construction: Having edge transition costs and probabilities, we form the DP in (12), and we solve it using value iteration method. Solving DP results in the π^g, which is stored as a look up table, i.e., it encodes the best next edge (local controller) that has to be taken at every node S^i.

F. Control execution with the graph

Finally, we execute the control law, where at each node we pick the best next local controller based on π^g until we reach the goal node. We illustrate an execution result in Fig. 1. The information-state at every time step is composed of a 16-dim estimation vector and a 16-by-16 estimation covariance matrix. To visualize this information-state, we map it to the information-state corresponding to the manipulator tip using the forward kinematics. The information-state of the manipulator tip at every step is shown by an ellipse (in red) which depicts the 3σ ellipse corresponding to the estimation covariance at that time step, and centered at the estimation mean. The propagation of this information-state over time is shown in Fig. 1 until the manipulator tip hits the goal region.

As it can be seen in Fig. 1, there are two passages through the obstacles to reach the goal region. Although the path through the right passage is shorter, the manipulator detours to a longer path through the left passage, to gain more accurate sensory information. As a result it leads to less collision probability and a safer path.

The reduction to graph-based control also preserves the robustness to large disturbances. In other words, suppose the system is executing the local controller μ^{ij} and the system state s_k deviates to some s'_k that is significantly far from the edge $\mathbf{e}^{ij}$ (in the partially-observable case we use the estimation mean to check the deviation from $\mathbf{e}^{ij}$). In this case, we just need to design a local controller from the deviated state $s_0 = s'_k$ that takes the deviated state/information-state to the best neighbouring graph node, in real-time, and then we can continue executing the pre-computed plan from thereon.

V. Conclusion

In this paper we have proposed a graph-based framework to reduce the computationally intractable problem of constrained stochastic control into a computationally tractable problem defined over a representative graph in the space. Many features of the original solution such as (i) robustness, (ii) cheap real-time evaluations, and (iii) being far-sighted (considering the costs all the way to the goal state) are preserved in this reduction, and as a result it can be a superior alternative for the RHC variants in stochastic environments in many problems. The proposed method also unifies the state and information space problems, while providing a way of incorporating the constraints into the control law design.

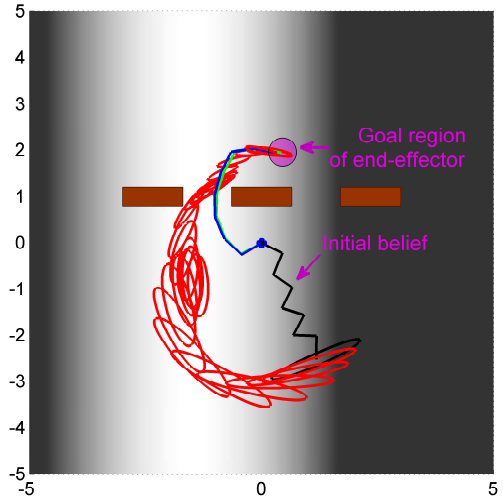

Fig. 1. This figure shows a result of executing the graph-based control for an 8-arm manipulator in a light-dark (sensing) environment. The manipulator is attached to the origin $(0, 0)$ and the purple region is the goal region for the manipulator tip. To unclutter the figure, we only show the uncertainty of the manipulator tip (end-effector). The initial mean and covariance is shown by black, and the evolution of the tip covariance during the plan execution is shown in red. The final estimation mean and the true configuration of the manipulator are shown in blue and green, respectively. The obstacles are shown in brown. The manipulator follows a longer but safer path to the goal region through the left passage, compared to the shorter but risky (with high collision probability) path through the right passage.

References

[1] D. Mayne and H. Michalska, "Receding horizon control of nonlinear systems," *IEEE Trans. Aut. Control*, vol. 35, no. 7, pp. 814–824, 1990.

[2] D. Li, F. Qian, and P. Fu, "Variance minimization approach for a class of dual control problems," *IEEE Trans. Aut. Control*, vol. 47, no. 12, pp. 2010–2020, 2002.

[3] D. van Hessem and O. Bosgra, "A full solution to the constrained stochastic closed-loop MPC problem via state and innovations feedback and its receding horizon implementation," in *Proceedings of the the Conference on Decision and Control (CDC)*, Maui, HI, December 2003, pp. 929–934.

[4] N. D. Toit and J. W. Burdick, "Robot motion planning in dynamic, uncertain environments," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 101–115, Feb 2012.

[5] S. Chakravorty and R. S. Erwin, "Information space receding horizon control," in *IEEE Symposium on Adaptive Dynamic Programming And Reinforcement Learning (ADPRL)*, April 2011.

[6] S. Chakravorty and S. Kumar, "Generalized sampling-based motion planners," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 41, no. 3, pp. 855–866, 2011.

[7] A. Agha-mohammadi, S. Chakravorty, and N. Amato, "FIRM: Feedback controller-based Information-state RoadMap -a framework for motion planning under uncertainty-," in *International Conference on Intelligent Robots and Systems (IROS)*, 2011.

[8] ——, "Nonholonomic motion planning in belief space via dynamic feedback linearization-based FIRM," in *International Conference on Intelligent Robots and Systems (IROS)*, 2012.

[9] D. Bertsekas, *Dynamic Programming and Optimal Control: 3rd Ed.* Athena Scientific, 2007.

[10] J. R. Norris, *Markov Chains*. Cambridge Univeristy Press, 1997.

[11] A. Agha-mohammadi, S. Chakravorty, and N. Amato, "On the probabilistic completeness of the sampling-based feedback motion planners in belief space," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2012.

[12] R. Platt, R. Tedrake, L. Kaelbling, and T. Lozano-Perez, "Belief space planning assuming maximum likelihood observatoins," in *Proceedings of Robotics: Science and Systems (RSS)*, June 2010.